
sportsreference Documentation

Release 0.1.0

Author

Nov 25, 2019

Contents:

1	Examples	3
1.1	Get instances of all NHL teams for the 2018 season	3
1.2	Print every NBA team's name and abbreviation	3
1.3	Get a specific NFL team's season information	3
1.4	Print the date of every game for a NCAA Men's Basketball team	4
1.5	Print the number of interceptions by the away team in a NCAA Football game	4
1.6	Get a Pandas DataFrame of all stats for a MLB game	4
1.6.1	API Documentation	4
1.6.1.1	MLB Package	4
1.6.1.2	NBA Package	28
1.6.1.3	NCAAB Package	49
1.6.1.4	NCAAF Package	72
1.6.1.5	NFL Package	93
1.6.1.6	NHL Package	115
1.6.2	Examples	133
1.6.2.1	Finding Tallest Players	133
1.6.2.2	Writing To CSV and Pickle	134
1.6.2.3	Finding Top Win Percentage By Year	134
1.6.3	Installation	134
1.6.4	Testing	135
2	Indices and tables	137
	Python Module Index	139
	Index	141

Sportsreference is a free python API that pulls the stats from www.sports-reference.com and allows them to be easily be used in python-based applications, especially ones involving data analytics and machine learning.

Sportsreference exposes a plethora of sports information from major sports leagues in North America, such as the MLB, NBA, College Football and Basketball, NFL, and NHL. Every sport has its own set of valid API queries ranging from the list of teams in a league, to the date and time of a game, to the total number of wins a team has secured during the season, and many, many more metrics that paint a more detailed picture of how a team has performed during a game or throughout a season.

CHAPTER 1

Examples

The following are a few examples showcasing how easy it can be to collect an abundance of metrics and information from all of the tracked leagues. The examples below are only a miniscule subset of the total number of statistics that can be pulled using sportsreference. Visit the documentation on [Read The Docs](#) for a complete list of all information exposed by the API.

1.1 Get instances of all NHL teams for the 2018 season

```
from sportsreference.nhl.teams import Teams

teams = Teams(2018)
```

1.2 Print every NBA team's name and abbreviation

```
from sportsreference.nba.teams import Teams

teams = Teams()
for team in teams:
    print(team.name, team.abbreviation)
```

1.3 Get a specific NFL team's season information

```
from sportsreference.nfl.teams import Teams

teams = Teams()
lions = teams('DET')
```

1.4 Print the date of every game for a NCAA Men's Basketball team

```
from sportsreference.ncaab.schedule import Schedule

purdue_schedule = Schedule('purdue')
for game in purdue_schedule:
    print(game.date)
```

1.5 Print the number of interceptions by the away team in a NCAA Football game

```
from sportsreference.ncaaf.boxscore import Boxscore

championship_game = Boxscore('2018-01-08-georgia')
print(championship_game.away_interceptions)
```

1.6 Get a Pandas DataFrame of all stats for a MLB game

```
from sportsreference.mlb.boxscore import Boxscore

game = Boxscore('BOS201806070')
df = game.dataframe
```

1.6.1 API Documentation

1.6.1.1 MLB Package

The MLB package offers multiple modules which can be used to retrieve information and statistics for Major League Baseball, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of runs scored to the number of sacrifice flies, to the slugging percentage and much more. The Boxscore can be easily queried by passing a boxscore's URI on sports-reference.com which can be retrieved from the Schedule class (see Schedule module below for more information on retrieving game-specific information).

```
from sportsreference.mlb.boxscore import Boxscore

game_data = Boxscore('BOS/BOS201808020')
print(game_data.home_runs)    # Prints 15
print(game_data.away_runs)    # Prints 7
df = game_data.dataframe      # Returns a Pandas DataFrame of game metrics
```

The Boxscore module also contains a Boxscores class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.


```
from datetime import datetime
from sportsreference.mlb.boxscore import Boxscores

games_today = Boxscores(datetime.today())
print(games_today.games)  # Prints a dictionary of all matchups for today
```

The Boxscores class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```
from datetime import datetime
from sportsreference.mlb.boxscore import Boxscores

# Pulls all games between and including July 17, 2017 and July 20, 2017
games = Boxscores(datetime(2017, 7, 17), datetime(2017, 7, 20))
# Prints a dictionary of all results from July 17, 2017 and July 20, 2017
print(games.games)
```

```
class sportsreference.mlb.boxscore.Boxscore(uri)
```

Bases: object

Detailed information about the final statistics for a game.

Stores all relevant information for a game such as the date, time, location, result, and more advanced metrics such as the number of strikes, a pitcher's influence on the game, the number of putouts and much more.

Parameters `uri` (*string*) – The relative link to the boxscore HTML page, such as 'BOS/BOS201806070'.

attendance

Returns an `int` of the game's listed attendance.

away_assists

Returns an `int` of the number of assists the away team registered.

away_at_bats

Returns an `int` of the number of at bats the away team had.

away_average_leverage_index

Returns a `float` of the amount of pressure the away team's pitcher faced during the game. 1.0 denotes average pressure while numbers less than 0 denote lighter pressure.

away_base_out_runs_added

Returns a `float` of the number of base out runs added by the away team.

away_base_out_runs_saved

Returns a `float` of the number of runs saved by the away pitcher based on the number of players on bases. 0.0 denotes an average value.

away_bases_on_balls

Returns an `int` of the number of bases the away team registered as a result of balls.

away_batting_average

Returns a `float` of the batting average for the away team.

away_earned_runs

Returns a `float` of the number of runs the away team earned.

away_fly_balls

Returns an `int` of the number of fly balls the away team allowed.

away_game_score

Returns an `int` of the starting away pitcher's score determine by many factors, such as number of runs scored against, number of strikes, etc.

away_grounded_balls

Returns an `int` of the number of grounded balls the away team allowed.

away_hits

Returns an `int` of the number of hits the away team had.

away_home_runs

Returns an `int` of the number of times the away team gave up a home run.

away_inherited_runners

Returns an `int` of the number of runners a pitcher inherited when he entered the game.

away_inherited_score

Returns an `int` of the number of scorers a pitcher inherited when he entered the game.

away_innings_pitched

Returns a `float` of the number of innings the away team pitched.

away_line_drives

Returns an `int` of the number of line drives the away team allowed.

away_on_base_percentage

Returns a `float` of the percentage of at bats that result in the batter getting on base.

away_on_base_plus

Returns a `float` of the on base percentage plus the slugging percentage. Percentage ranges from 0-1.

away_pitches

Returns an `int` of the number of pitches the away team faced.

away_plate_appearances

Returns an `int` of the number of plate appearances the away team made.

away_players

Returns a list of `BoxscorePlayer` class instances for each player on the away team.

away_putouts

Returns an `int` of the number of putouts the away team registered.

away_rbi

Returns an `int` of the number of runs batted in the away team registered.

away_runs

Returns an `int` of the number of runs the away team scored.

away_slugging_percentage

Returns a `float` of the slugging percentage for the away team based on the number of bases gained per at-bat with bigger plays getting more weight.

away_strikeouts

Returns an `int` of the number of times the away team was struck out.

away_strikes

Returns an `int` of the number of times a strike was called against the away team.

away_strikes_by_contact

Returns an `int` of the number of times the away team struck out a batter who made contact with the pitch.

away_strikes_looking

Returns an `int` of the number of times the away team struck out a batter who was looking.

away_strikes_swinging

Returns an `int` of the number of times the away team struck out a batter who was swinging.

away_unknown_bat_type

Returns an `int` of the number of away at bats that were not properly tracked and therefore cannot be safely placed in another statistical category.

away_win_probability_added

Returns a `float` of the total positive influence the away team's offense had on the outcome of the game.

away_win_probability_by_pitcher

Returns a `float` of the amount of influence the away pitcher had on the game's result with 0.0 denoting zero influence and 1.0 denoting he was solely responsible for the team's win.

away_win_probability_for_offensive_player

Returns a `float` of the overall influence the away team's offense had on the outcome of the game where 0.0 denotes no influence and 1.0 denotes the offense was solely responsible for the outcome.

away_win_probability_subtracted

Returns a `float` of the total negative influence the away team's offense had on the outcome of the game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string URI that is used to instantiate the class, such as 'BOS201806070'.

date

Returns a `string` of the date the game took place.

duration

MM'.

Type Returns a `string` of the game's duration in the format 'H

home_assists

Returns an `int` of the number of assists the home team registered.

home_at_bats

Returns an `int` of the number of at bats the home team had.

home_average_leverage_index

Returns a `float` of the amount of pressure the home team's pitcher faced during the game. 1.0 denotes average pressure while numbers less than 0 denote lighter pressure.

home_base_out_runs_added

Returns a `float` of the number of base out runs added by the home team.

home_base_out_runs_saved

Returns a `float` of the number of runs saved by the home pitcher based on the number of players on bases. 0.0 denotes an average value.

home_bases_on_balls

Returns an `int` of the number of bases the home team registered as a result of balls.

home_batting_average

Returns a `float` of the batting average for the home team.

home_earned_runs

Returns a `float` of the number of runs the home team earned.

home_fly_balls

Returns an `int` of the number of fly balls the home team allowed.

home_game_score

Returns an `int` of the starting home pitcher's score determine by many factors, such as number of runs scored against, number of strikes, etc.

home_grounded_balls

Returns an `int` of the number of grounded balls the home team allowed.

home_hits

Returns an `int` of the number of hits the home team had.

home_home_runs

Returns an `int` of the number of times the home team gave up a home run.

home_inherited_runners

Returns an `int` of the number of runners a pitcher inherited when he entered the game.

home_inherited_score

Returns an `int` of the number of scorers a pitcher inherited when he entered the game.

home_innings_pitched

Returns a `float` of the number of innings the home team pitched.

home_line_drives

Returns an `int` of the number of line drives the home team allowed.

home_on_base_percentage

Returns a `float` of the percentage of at bats that result in the batter getting on base.

home_on_base_plus

Returns a `float` of the on base percentage plus the slugging percentage. Percentage ranges from 0-1.

home_pitches

Returns an `int` of the number of pitches the home team faced.

home_plate_appearances

Returns an `int` of the number of plate appearances the home team made.

home_players

Returns a list of `BoxscorePlayer` class instances for each player on the home team.

home_putouts

Returns an `int` of the number of putouts the home team registered.

home_rbi

Returns an `int` of the number of runs batted in the home team registered.

home_runs

Returns an `int` of the number of runs the home team scored.

home_slugging_percentage

Returns a `float` of the slugging percentage for the home team based on the number of bases gained per at-bat with bigger plays getting more weight.

home_strikeouts

Returns an `int` of the number of times the home team was struck out.

home_strikes

Returns an `int` of the number of times a strike was called against the home team.

home_strikes_by_contact

Returns an `int` of the number of times the home team struck out a batter who made contact with the pitch.

home_strikes_looking

Returns an `int` of the number of times the home team struck out a batter who was looking.

home_strikes_swinging

Returns an `int` of the number of times the home team struck out a batter who was swinging.

home_unknown_bat_type

Returns an `int` of the number of home at bats that were not properly tracked and therefore cannot be safely placed in another statistical category.

home_win_probability_added

Returns a `float` of the total positive influence the home team's offense had on the outcome of the game.

home_win_probability_by_pitcher

Returns a `float` of the amount of influence the home pitcher had on the game's result with 0.0 denoting zero influence and 1.0 denoting he was solely responsible for the team's win.

home_win_probability_for_offensive_player

Returns a `float` of the overall influence the home team's offense had on the outcome of the game where 0.0 denotes no influence and 1.0 denotes the offense was solely responsible for the outcome.

home_win_probability_subtracted

Returns a `float` of the total negative influence the home team's offense had on the outcome of the game.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'LAD' for the Los Angeles Dodgers.

losing_name

Returns a `string` of the losing team's name, such as 'Los Angeles Dodgers'.

time

Returns a `string` of the time the game started.

time_of_day

Returns a `string` constant indicated whether the game was played during the day or at night.

venue

Returns a `string` of the name of the ballpark where the game was played.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'HOU' for the Houston Astros.

winning_name

Returns a `string` of the winning team's name, such as 'Houston Astros'.

class sportsreference.mlb.boxscore.BoxscorePlayer(*player_id*, *player_name*,
player_data)

Bases: `sportsreference.mlb.player.AbstractPlayer`

Get player stats for an individual game.

Given a player ID, such as 'altuvjo01' for Jose Altuve, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id**(*string*) – A player’s ID according to baseball-reference.com, such as ‘altuvjo01’ for Jose Altuve. The player ID can be found by navigating to the player’s stats page and getting the string between the final slash and the ‘.html’ in the URL. In general, the ID is in the format ‘LLLLLFFNN’ where ‘LLLLL’ are the first 5 letters in the player’s last name, ‘FF’, are the first 2 letters in the player’s first name, and ‘NN’ is a number starting at ‘01’ for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name**(*string*) – A string representing the player’s first and last name, such as ‘Jose Altuve’.
- **player_data**(*string*) – A string representation of the player’s HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

average_leverage_index

Returns a `float` of the amount of pressure the player faced during the game. 1.0 denotes average pressure while numbers less than 0 denote lighter pressure.

average_leverage_index_pitcher

Returns a `float` of the amount of pressure the pitcher faced during the game. 1.0 denotes average pressure while numbers less than 0 denote lighter pressure.

base_out_runs_added

Returns a `float` of the number of base out runs added by the player.

base_out_runs_saved

Returns a `float` of the number of runs saved by the pitcher based on the number of players on bases. 0.0 denotes an average value.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values for the specified game.

earned_runs_against

Returns a `float` of the player’s overall Earned Runs Against average as calculated by $9 * \text{earned_runs} / \text{innings_pitched}$.

fly_balls

Returns an `int` of the number of fly balls the player allowed.

game_score

Returns an `int` of the pitcher’s score determine by many factors, such as number of runs scored against, number of strikes, etc.

grounded_balls

Returns an `int` of the number of grounded balls the player allowed.

home_runs_thrown

Returns an `int` of the number of home runs the player threw.

inherited_runners

Returns an `int` of the number of runners a relief pitcher inherited.

inherited_score

Returns an `int` of the number of runners on base when a relief pitcher entered the game that ended up scoring.

innings_pitched

Returns an `int` of the number of innings the player pitched in.

line_drives

Returns an `int` of the number of line drives the player allowed.

pitches_thrown

Returns an `int` of the number of pitches the player threw.

strikes

Returns an `int` of the number of times a strike was called against the player.

strikes_contact

Returns an `int` of the number of times the player threw a strike when the player made contact with the ball.

strikes_looking

Returns an `int` of the number of times the player threw a strike with the player looking.

strikes_swinging

Returns an `int` of the number of times the player threw a strike with the batter swinging.

strikes_thrown

Returns an `int` of the number of times a strikes the player threw.

unknown_bat_types

Returns an `int` of the number of line drives the player allowed.

win_probability_added

Returns a `float` of the total positive influence the player's offense had on the outcome of the game.

win_probability_added_pitcher

Returns a `float` of the total positive influence the pitcher's offense had on the outcome of the game.

win_probability_for_offensive_player

Returns a `float` of the overall influence the player's offense had on the outcome of the game where 0.0 denotes no influence and 1.0 denotes the offense was solely responsible for the outcome.

win_probability_subtracted

Returns a `float` of the total negative influence the player's offense had on the outcome of the game.

class sportsreference.mlb.boxscore.Boxscores (*date*, *end_date=None*)

Bases: `object`

Search for MLB games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **date** (*datetime object*) – The date to search for any matches. The month, day, and year are required for the search, but time is not factored into the search.
- **end_date** (*datetime object (optional)*) – Optionally specify an end date to iterate until. All boxscores starting from the date specified in the 'date' parameter up to and including the boxscores specified in the 'end_date' parameter will be pulled. If left empty, or if 'end_date' is prior to 'date', only the games from the day specified in the 'date' parameter will be saved.

games

Returns a `dictionary` object representing all of the games played on the requested day. Dictionary is in the following format:

```
{
  'date': [ # 'date' is the string date in format 'MM-DD-YYYY'
    {
```

(continues on next page)

(continued from previous page)

```
        'home_name': Name of the home team, such as 'New York
                      Yankees' (`str`),
        'home_abbr': Abbreviation for the home team, such as
                      'NYY' (`str`),
        'away_name': Name of the away team, such as 'Houston
                      Astros' (`str`),
        'away_abbr': Abbreviation for the away team, such as
                      'HOU' (`str`),
        'boxscore': String representing the boxscore URI, such
                      as 'SLN/SLN201807280' (`str`),
        'winning_name': Full name of the winning team, such as
                      'New York Yankees' (`str`),
        'winning_abbr': Abbreviation for the winning team, such
                      as 'NYY' (`str`),
        'losing_name': Full name of the losing team, such as
                      'Houston Astros' (`str`),
        'losing_abbr': Abbreviation for the losing team, such
                      as 'HOU' (`str`),
        'home_score': Integer score for the home team (`int`),
        'away_score': Integer score for the away team (`int`)
    },
    { ... },
    ...
]
```

If no games were played on ‘date’, the list for [‘date’] will be empty.

Player

The Player module contains an abstract base class that can be inherited by both the `BoxscorePlayer` and `Player` classes in the `Boxscore` and `Roster` modules, respectively. All of the properties that appear in the `AbstractPlayer` class can be read from either of the two child classes mentioned above.

class `sportsreference.mlb.player.AbstractPlayer` (*player_id*, *player_name*, *player_data*)

Bases: `object`

Get player information and stats for all seasons.

Given a player ID, such as ‘altuvjo01’ for Jose Altuve, capture all relevant stats and information like name, nationality, height/weight, career home runs, last season’s batting average, salary, contract amount, and much more.

By default, the class instance will return the player’s career stats, but single-season stats can be found by calling the instance with the requested season as denoted on baseball-reference.com.

Parameters `player_id` (*string*) – A player’s ID according to basketball-reference.com, such as ‘altuvjo01’ for Jose Altuve. The player ID can be found by navigating to the player’s stats page and getting the string between the final slash and the ‘.html’ in the URL. In general, the ID is in the format ‘LLLLLFFNN’ where ‘LLLLL’ are the first 5 letters in the player’s last name, ‘FF’, are the first 2 letters in the player’s first name, and ‘NN’ is a number starting at ‘01’ for the first time that player ID has been used and increments by 1 for every successive player.

assists

Returns an `int` of the number of assists the player had.

at_bats

Returns an `int` of the number of at bats the player had.

bases_on_balls

Returns an `int` of the number of bases the player registered as a result of balls.

bases_on_balls_given

Returns an `int` of the number of bases on balls the player has given as a pitcher.

batters_faced

Returns an `int` of the number of batters the pitcher has faced.

batting_average

Returns a `float` of the batting average for the player.

earned_runs_allowed

Returns an `int` of the number of earned runs the player allowed as a pitcher.

hits

Returns an `int` of the number of hits the player had.

hits_allowed

Returns an `int` of the number of hits the player allowed as a pitcher.

name

Returns a `string` of the player's name, such as 'Jose Altuve'.

on_base_percentage

Returns a `float` of the percentage of at bats that result in the batter getting on base.

on_base_plus_slugging_percentage

Returns a `float` of the on base percentage plus the slugging percentage. Percentage ranges from 0-1.

plate_appearances

Returns an `int` of the number of plate appearances the player had.

player_id

Returns a `string` of the player's ID on sports-reference, such as 'altuvjo01' for Jose Altuve.

putouts

Returns an `int` of the number of putouts the player had.

runs

Returns an `int` of the number of runs the player scored.

runs_allowed

Returns an `int` of the number of runs the player allowed as a pitcher.

runs_batted_in

Returns an `int` of the number of runs batted in the player registered.

slugging_percentage

Returns a `float` of the slugging percentage for the player based on the number of bases gained per at-bat with bigger plays getting more weight.

strikeouts

Returns an `int` of the number of strikeouts the player threw as a pitcher.

times_struck_out

Returns an `int` of the number of times the player was struck out.

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the `Player` class which has detailed information ranging from career home runs to single-season stats and player height, weight, and nationality. The following is an example on collecting career information for José Altuve.

```
from sportsreference.mlb.roster import Player

altuve = Player('altuvjo01')
print(altuve.name)    # Prints 'José Altuve'
print(altuve.hits)    # Prints Altuve's career hits total
# Prints a Pandas DataFrame of all relevant stats per season for Altuve
print(altuve.dataframe)
```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific season, call the class instance with the season string. All future property requests will return the season-specific stats.

```
from sportsreference.mlb.roster import Player

altuve = Player('altuvjo01') # Currently pulling career stats
print(altuve.hits)          # Prints Altuve's CAREER hits total
# Prints Altuve's hits total only for the 2017 season
print(altuve('2017').hits)
# Prints Altuve's home runs total for the 2017 season only
print(altuve.home_runs)
```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.mlb.roster import Player

altuve = Player('altuvjo01') # Currently pulling career stats
# Prints Altuve's hits total only for the 2017 season
print(altuve('2017').hits)
print(altuve('Career').hits) # Prints Altuve's career hits total
```

In addition, the Roster module also contains the `Roster` class which can be used to pull all players on a team's roster during a given season and creates instances of the `Player` class for each team member and adds them to a list to be easily queried.

```
from sportsreference.mlb.roster import Roster

astros = Roster('HOU')
for player in astros.players:
    # Prints the name of all players who played for the Astros in the most
    # recent season.
    print(player.name)
```

```
class sportsreference.mlb.roster.Player(player_id)
    Bases: sportsreference.mlb.player.AbstractPlayer
```

Get player information and stats for all seasons.

Given a player ID, such as 'altuvjo01' for Jose Altuve, capture all relevant stats and information like name, nationality, height/weight, career home runs, last season's batting average, salary, contract amount, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on baseball-reference.com.

Parameters `player_id` (*string*) – A player's ID according to basketball-reference.com, such as 'altuvjo01' for Jose Altuve. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'LLLLLFFNN' where 'LLLLL' are the first 5 letters in the player's last name, 'FF', are the first 2 letters in the player's first name, and 'NN' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.

assists

Returns an `int` of the number of assists the player had.

at_bats

Returns an `int` of the number of at bats the player had.

balks

Returns an `int` of the number of times the pitcher balked.

bases_on_balls

Returns an `int` of the number of bases the player registered as a result of balls.

bases_on_balls_given

Returns an `int` of the number of bases on balls the player has given as a pitcher.

bases_on_balls_given_per_nine_innings

Returns a `float` of the number of bases on balls the pitcher has given per nine innings played.

batters_struckout_per_nine_innings

Returns a `float` of the number of batters the pitcher has struck out per nine innings played.

batting_average

Returns a `float` of the batting average for the player.

birth_date

Returns a `datetime` object of the day and year the player was born.

complete_games

Returns an `int` of the number of complete games the player has participated in.

contract

Returns a `dictionary` of the player's contract where each key is a `string` of the year, such as '2017' and each value is a `dictionary` with the `string` key-value pairs of the player's age, team name, and salary.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values where each index is a different season plus the career stats.

defensive_chances

Returns an `int` of the number of defensive chances (equal to the number of putouts + assists + errors) the player had.

defensive_runs_saved_above_average

Returns an `int` of the number of defensive runs the player saved compared to an average player.

defensive_runs_saved_above_average_per_innings

Returns an `int` of the number of defensive runs the player was worth per 1,200 innings compared to an average player.

double_plays_turned

Returns an `int` of the number of double plays the player was involved in.

doubles

Returns an `int` of the number of doubles the player hit.

earned_runs_allowed

Returns an `int` of the number of earned runs the player allowed as a pitcher.

era

Returns a `float` of the pitcher's Earned Runs Average.

era_plus

Returns a `float` of the pitcher's ERA while adjusted for the ballpark.

errors

Returns an `int` of the number of errors the player made.

fielding_independent_pitching

Returns a `float` of the pitcher's effectiveness at preventing home runs, bases on balls, and hitting players with pitches, while causing strikeouts.

fielding_percentage

Returns a `float` of the players fielding percentage, equivalent to $(\text{putouts} + \text{assists}) / (\text{putouts} + \text{assists} + \text{errors})$. Percentage ranges from 0-1.

games

Returns an `int` of the number of games the player participated in.

games_catcher

Returns an `int` of the number of games the player was in the lineup as a catcher.

games_center_fielder

Returns an `int` of the number of games the player was in the lineup as a center fielder.

games_designated_hitter

Returns an `int` of the number of games the player was in the lineup as a designated hitter.

games_finished

Returns an `int` of the number of games the player finished as a pitcher.

games_first_baseman

Returns an `int` of the number of games the player was in the lineup as a first baseman.

games_in_batting_order

Returns an `int` of the number of games the player was in the batting lineup.

games_in_defensive_lineup

Returns an `int` of the number of games the player was in the defensive lineup.

games_left_fielder

Returns an `int` of the number of games the player was in the lineup as a left fielder.

games_outfielder

Returns an `int` of the number of games the player was in the lineup as an outfielder.

games_pinch_hitter

Returns an `int` of the number of games the player was in the lineup as a pinch hitter.

games_pinch_runner

Returns an `int` of the number of games the player was in the lineup as a pinch runner.

games_pitcher

Returns an `int` of the number of games the player was in the lineup as a pitcher.

games_right_fielder

Returns an `int` of the number of games the player was in the lineup as a right fielder.

games_second_baseman

Returns an `int` of the number of games the player was in the lineup as a second baseman.

games_shortstop

Returns an `int` of the number of games the player was in the lineup as a shortstop.

games_started

Returns an `int` of the number of games the player started.

games_third_baseman

Returns an `int` of the number of games the player was in the lineup as a third baseman.

grounded_into_double_plays

Returns an `int` of the number of double plays the player grounded into.

height

Returns a `string` of the players height in the format “feet-inches”.

hits

Returns an `int` of the number of hits the player had.

hits_against_per_nine_innings

Returns a `float` of the number of hits the player has given per nine innings played.

hits_allowed

Returns an `int` of the number of hits the player allowed as a pitcher.

home_runs

Returns an `int` of the number of home runs the player hit.

home_runs_against_per_nine_innings

Returns a `float` of the number of home runs the pitcher has given per nine innings played.

home_runs_allowed

Returns an `int` of the number of home runs a player has allowed as a pitcher.

innings_played

Returns a `float` of the total number of innings the player has played in.

intentional_bases_on_balls

Returns an `int` of the number of times the player has been intentionally walked by the opposition.

intentional_bases_on_balls_given

Returns an `int` of the number of bases the player has intentionally given as a pitcher.

league_fielding_percentage

Returns a `float` of the average fielding percentage for the league at the player’s position. Percentage ranges from 0-1.

league_range_factor_per_game

Returns a `float` of the average range factor for the league per game, equal to (putouts + assists) / games_played.

league_range_factor_per_nine_innings

Returns a `float` of the average range factor for the league per nine innings, equal to 9 * (putouts + assists) / innings_played.

losses

Returns an `int` of the number of games the player has lost as a pitcher.

name

Returns a `string` of the player’s name, such as ‘Jose Altuve’.

nationality

Returns a `string` constant denoting which country the player originates from.

on_base_percentage

Returns a `float` of the percentage of at bats that result in the batter getting on base.

on_base_plus_slugging_percentage

Returns a `float` of the on base percentage plus the slugging percentage. Percentage ranges from 0-1.

on_base_plus_slugging_percentage_plus

Returns an `int` of the on base percentage plus the slugging percentage, adjusted to the player's ballpark.

plate_appearances

Returns an `int` of the number of plate appearances the player had.

position

Returns a `string` constant of the player's primary position.

putouts

Returns an `int` of the number of putouts the player had.

range_factor_per_game

Returns a `float` of the players range factor per game, equal to $9 * (\text{putouts} + \text{assists}) / \text{games_played}$.

range_factor_per_nine_innings

Returns a `float` of the players range factor per nine innings, equal to $9 * (\text{putouts} + \text{assists}) / \text{innings_played}$.

runs

Returns an `int` of the number of runs the player scored.

runs_allowed

Returns an `int` of the number of runs the player allowed as a pitcher.

runs_batted_in

Returns an `int` of the number of runs batted in the player registered.

sacrifice_flies

Returns an `int` of the number of sacrifice flies the player hit.

sacrifice_hits

Returns an `int` of the number of sacrifice hits or sacrifice bunts the player made.

saves

Returns an `int` of the number of saves the player made as a pitcher.

season

Returns a `string` of the season in the format 'YYYY', such as '2017'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

shutouts

Returns an `int` of the number of times the player did not allow any runs and threw a complete game as a pitcher.

slugging_percentage

Returns a `float` of the slugging percentage for the player based on the number of bases gained per at-bat with bigger plays getting more weight.

stolen_bases

Returns an `int` of the number of bases the player has stolen.

strikeouts

Returns an `int` of the number of strikeouts the player threw as a pitcher.

strikeouts_thrown_per_walk

Returns a `float` of the number of batters the pitcher has struck out per the number of walks given.

team_abbreviation

Returns a `string` of the team's abbreviation, such as 'HOU' for the Houston Astros.

times_caught_stealing

Returns an `int` of the number of times the player was caught stealing.

times_hit_by_pitch

Returns an `int` of the number of times the player has been hit by a pitch.

times_hit_player

Returns an `int` of the number of times the pitcher hit a player with a pitch.

times_struck_out

Returns an `int` of the number of times the player was struck out.

total_bases

Returns an `int` of the number of bases the player has gained.

total_fielding_runs_above_average

Returns an `int` of the number of runs the player was worth compared to an average player.

total_fielding_runs_above_average_per_innings

Returns an `int` of the number of runs the player was worth per 1,200 innings compared to an average player.

triples

Returns an `int` of the number of triples the player hit.

weight

Returns an `int` of the player's weight in pounds.

whip

Returns a `float` of the pitcher's WHIP score, equivalent to (bases on balls + hits) / innings played.

wild_pitches

Returns an `int` of the number of wild pitches the player has thrown.

win_percentage

Returns a `float` of the players winning percentage as a pitcher. Percentage ranges from 0-1.

wins

Returns an `int` of the number of games the player has won as a pitcher.

class sportsreference.mlb.roster.**Roster** (*team*, *year=None*, *slim=False*)

Bases: `object`

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the `Player` class for each player, containing a detailed list of the players statistics and information.

Parameters

- **team** (*string*) – The team's abbreviation, such as 'HOU' for the Houston Astros.
- **year** (*string (optional)*) – The 4-digit year to pull the roster from, such as '2018'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to `True` to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective

stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to False.

players

Returns a list of player instances for each player on the requested team's roster if the `slim` property is False when calling the Roster class. If the `slim` property is True, returns a dictionary where each key is a string of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The Schedule module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the `Boxscore` class which has much more detailed information on the game metrics.

```
from sportsreference.mlb.schedule import Schedule

houston_schedule = Schedule('HOU')
for game in houston_schedule:
    print(game.date) # Prints the date the game was played
    print(game.result) # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class sportsreference.mlb.schedule.**Game**(*game_data*, *year*)

Bases: object

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters

- **game_data** (*string*) – The row containing the specified game information.
- **year** (*string*) – The year of the current season.

attendance

Returns an `int` of the total listed attendance for the game.

boxscore

Returns an instance of the `Boxscore` class containing more detailed stats on the game.

boxscore_index

Returns a `string` of the URI for a boxscore which can be used to access or index a game.

dataframe

Returns a pandas `DataFrame` containing all other class properties and values. The index for the `DataFrame` is the boxscore string.

dataframe_extended

Returns a pandas `DataFrame` representing the `Boxscore` class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property. The index for the `DataFrame` is the boxscore string.

date

Returns a `string` of the date the game was played on.

datetime

Returns a `datetime` object of the month, day, year, and time the game was played.

day_or_night

Returns a `string` constant to indicate whether the game was played during the day or night.

game

Returns an `int` of the game in the season, where 1 is the first game of the season.

game_duration

MM'.

Type Returns a `string` of the game's total duration in the format 'H

game_number_for_day

Returns an `int` denoting which game is played for the team during the given day. Default value is 1 where a team plays only one game during the day, but can be higher for double headers, etc. For example, if a team has a double header one day, the first game of the day will return 1 while the second game will return 2.

games_behind

Returns a `float` of the number of games behind the leader the team is. 0.0 indicates the team is tied for first. Negative numbers indicate the number of games a team is ahead of the second place team.

innings

Returns an `int` of the total number of innings that were played.

location

Returns a `string` constant to indicate whether the game was played at home or away.

loser

Returns a `string` of the name of the losing pitcher.

opponent_abbr

Returns a `string` of the opponent's 3-letter abbreviation, such as 'NYY' for the New York Yankees.

rank

Returns an `int` of the team's rank in the league with 1 being the best team.

record

Returns a `string` of the team's record in the format 'W-L'.

result

Returns a `string` constant to indicate whether the team won or lost.

runs_allowed

Returns an `int` of the total number of runs that the team allowed.

runs_scored

Returns an `int` of the total number of runs that were scored by the team.

save

Returns a `string` of the name of the pitcher credited with the save if applicable. If no saves, returns `None`.

streak

Returns a `string` of the team's winning/losing streak at the conclusion of the requested game. A winning streak is denoted by a number of '+' signs for the number of consecutive wins and a losing streak is denoted by a '-' sign.

winner

Returns a `string` of the name of the winning pitcher.

class sportsreference.mlb.schedule.**Schedule** (*abbreviation, year=None*)

Bases: `object`

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'HOU' for the Houston Astros.
- **year** (*string* *optional*) – The requested year to pull stats from.

dataframe

Returns a pandas DataFrame where each row is a representation of the Game class. Rows are indexed by the boxscore string.

dataframe_extended

Returns a pandas DataFrame where each row is a representation of the Boxscore class for every game in the schedule. Rows are indexed by the boxscore string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The Teams module exposes information for all MLB teams including the team name and abbreviation, the number of games they won during the season, the total number of bases they've stolen, and much more.

```
from sportsreference.mlb.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    print(team.batting_average)  # Prints the team's season batting average
```

Each Team instance contains a link to the Schedule class which enables easy iteration over all games for a particular team. A Pandas DataFrame can also be queried to easily grab all stats for all games.

```
from sportsreference.mlb.teams import Teams

teams = Teams()
for team in teams:
    schedule = team.schedule  # Returns a Schedule instance for each team
    # Returns a Pandas DataFrame of all metrics for all game Boxscores for
    # a season.
    df = team.schedule.dataframe_extended
```

Lastly, each Team instance also contains a link to the Roster class which enables players from the team to be easily queried. Each Roster instance contains detailed stats and information for each player on the team.

```
from sportsreference.mlb.teams import Teams

for team in Teams():
    roster = team.roster  # Gets each team's roster
    for player in roster.players:
        print(player.name)  # Prints each players name on the roster
```

```
class sportsreference.mlb.teams.Team(team_data, rank, year=None)
```

Bases: object

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as rank, name, and abbreviation, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **rank** (*int*) – A team's position in the league based on the number of points they obtained during the season.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a `string` of the team's abbreviation, such as 'HOU' for the Houston Astros.

at_bats

Returns an `int` of the total number of at bats for the team.

average_batter_age

Returns a `float` of the average batter age weighted by their number of at bats plus the number of games participated in.

average_pitcher_age

Returns a `float` of the average pitcher age weighted by the number of games started, followed by the number of games played and saves.

away_losses

Returns an `int` of the number of away losses during the season.

away_record

Returns a `string` of the team's away record. Record is in the format 'W-L'.

away_wins

Returns an `int` of the number of away wins during the season.

balks

Returns an `int` of the total number of times a pitcher has balked.

bases_on_balls

Returns an `int` of the number of bases on walks.

bases_on_walks_given

Returns an `int` of the total number of bases from walks given up by a team during the season.

bases_on_walks_given_per_nine_innings

Returns a `float` of the average number of walks conceded per nine innings.

batters_faced

Returns an `int` of the total number of batters all pitchers have faced during a season.

batting_average

Returns a `float` of the batting average for the team. Percentage ranges from 0-1.

complete_game_shutouts

Returns an `int` of the total number of complete games where the opponent scored zero runs.

complete_games

Returns an `int` of the total number of complete games a team has accumulated during the season.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'HOU'.

doubles

Returns an `int` of the total number of doubles hit by the team.

earned_runs_against

Returns a `float` of the average number of earned runs against for a team.

earned_runs_against_plus

Returns an `int` of the team's average earned runs against, adjusted for the home ballpark.

extra_inning_losses

Returns an `int` of the number of losses the team has when the game has gone to extra innings.

extra_inning_record

Returns a `string` of the team's record when the game has gone to extra innings. Record is in the format 'W-L'.

extra_inning_wins

Returns an `int` of the number of wins the team has when the game has gone to extra innings.

fielding_independent_pitching

Returns a `float` of the team's effectiveness at preventing home runs, walks, batters being hit by pitches, and strikeouts.

games

Returns an `int` of the number of games the team has played during the season.

games_finished

Returns an `int` of the number of games finished which is equivalent to the number of games played minus the number of complete games during the season.

grounded_into_double_plays

Returns an `int` of the total number double plays grounded into by the team.

hit_pitcher

Returns an `int` of the total number of times a pitcher has hit an opposing batter.

hits

Returns an `int` of the total number of hits during the season.

hits_allowed

Returns an `int` of the total number of hits allowed during the season.

hits_per_nine_innings

Returns a `float` of the average number of hits per nine innings by the opponent.

home_losses

Returns an `int` of the number of losses at home during the season.

home_record

Returns a `string` of the team's home record. Record is in the format 'W-L'.

home_runs

Returns an `int` of the total number of home runs hit by the team.

home_runs_against

Returns an `int` of the total number of home runs given up during the season.

home_runs_per_nine_innings

Returns a `float` of the average number of home runs per nine innings by the opponent.

home_wins

Returns an `int` of the number of wins at home during the season.

innings_pitched

Returns a `float` of the total number of innings pitched by a team during the season.

intentional_bases_on_balls

Returns an `int` of the total number of times a player took a base from an intentional walk.

interleague_record

Returns a `string` of the team's interleague record. Record is in the format 'W-L'.

last_ten_games_record

Returns a `string` of the team's record over the last ten games. Record is in the format 'W-L'.

last_thirty_games_record

Returns a `string` of the team's record over the last thirty games. Record is in the format 'W-L'.

last_twenty_games_record

Returns a `string` of the team's record over the last twenty games. Record is in the format 'W-L'.

league

Returns a `string` of the two letter abbreviation of the league, such as 'AL' for the American League.

losses

Returns an `int` of the total number of games the team lost during the season.

losses_last_ten_games

Returns an `int` of the number of losses in the last 10 games.

losses_last_thirty_games

Returns an `int` of the number of losses in the last 30 games.

losses_last_twenty_games

Returns an `int` of the number of losses in the last 20 games.

losses_vs_left_handed_pitchers

Returns an `int` of number of losses against left-handed pitchers.

losses_vs_right_handed_pitchers

Returns an `int` of the number of losses against right-handed pitchers.

losses_vs_teams_over_500

Returns an `int` of the number of losses against teams over 500.

losses_vs_teams_under_500

Returns an `int` of the number of losses against teams under 500.

luck

Returns an `int` of the difference between the current wins and losses compared to the pythagorean wins and losses.

name

Returns a `string` of the team's full name, such as 'Houston Astros'.

number_of_pitchers

Returns an `int` of the total number of pitchers used during a season.

number_players_used

Returns an `int` of the number of different players used during the season.

on_base_percentage

Returns a `float` of the percentage of at bats that result in a player taking a base. Percentage ranges from 0-1.

on_base_plus_slugging_percentage

Returns a `float` of the sum of the on base percentage plus the slugging percentage.

on_base_plus_slugging_percentage_plus

Returns an `int` of the on base percentage plus the slugging percentage, adjusted to the team's home ballpark.

opposing_runners_left_on_base

Returns an `int` of the total number of opponents a team has left on bases at the end of an inning.

plate_appearances

Returns an `int` of the total number of plate appearances for the team.

pythagorean_win_loss

Returns a `string` of the team's expected win-loss record based on the runs scored and allowed. Record is in the format 'W-L'.

rank

Returns an `int` of the team's rank based on their win percentage.

record_vs_left_handed_pitchers

Returns a `string` of the team's record against left-handed pitchers. Record is in the format 'W-L'.

record_vs_right_handed_pitchers

Returns a `string` of the team's record against right-handed pitchers. Record is in the format 'W-L'.

record_vs_teams_over_500

Returns a `string` of the team's record against teams with a win percentage over 500. Record is in the format 'W-L'.

record_vs_teams_under_500

Returns a `string` of the team's record against teams with a win percentage under 500. Record is in the format 'W-L'.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

run_difference

Returns a `float` of the difference between the number of runs scored and the number of runs given up per game. Positive numbers indicate the team scores more per game than they are scored on.

runners_left_on_base

Returns an `int` of the total number of runners left on base at the end of an inning.

runs

Returns a `float` of the average number of runs scored per game by the team.

runs_against

Returns a `float` of the average number of runs scored per game by the opponent.

runs_allowed_per_game

Returns a `float` of the average number of runs a team has allowed per game.

runs_batted_in

Returns an `int` of the total number of runs batted in by the team.

sacrifice_flies

Returns an `int` of the total number of sacrifice flies the team made during the season.

sacrifice_hits

Returns an `int` of the total number of sacrifice hits the team made during the season.

saves

Returns an `int` of the total number of saves a team has accumulated during the season.

schedule

Returns an instance of the Schedule class containing the team's complete schedule for the season.

shutouts

Returns an `int` of the total number of shutouts a team has accumulated during the season.

simple_rating_system

Returns a `float` of the average number of runs per game a team scores compared to average.

single_run_losses

Returns an `int` of the number of losses the team has when only one run is scored.

single_run_record

Returns a `string` of the team's record when only one run is scored. Record is in the format 'W-L'.

single_run_wins

Returns an `int` of the number of wins the team has when only one run is scored.

slugging_percentage

Returns a `float` of the ratio of total bases gained per at bat.

stolen_bases

Returns an `int` of the total number of bases stolen by the team.

streak

Returns a `string` of the team's current winning or losing streak, such as 'W 3' for a team on a 3-game winning streak.

strength_of_schedule

Returns a `float` denoting a team's strength of schedule, based on runs scores and conceded. Higher values result in more challenging schedules while 0.0 is an average schedule.

strikeouts

Returns an `int` of the total number of times a team has struck out an opponent.

strikeouts_per_base_on_balls

Returns a `float` of the average number of strikeouts per walk thrown by a team.

strikeouts_per_nine_innings

Returns a `float` of the average number of strikeouts a team throws per nine innings.

times_caught_stealing

Returns an `int` of the number of times a player was caught stealing.

times_hit_by_pitch

Returns an `int` of the total number of times a batter was hit by an opponent's pitch.

times_struck_out

Returns an `int` of the total number of times the team struck out.

total_bases

Returns an `int` of the total number of bases a team has gained during the season.

total_runs

Returns an `int` of the total number of runs scored during the season.

triples

Returns an `int` of the total number of triples hit by the team.

whip

Returns a `float` of the average number of walks plus hits by the opponent per inning.

wild_pitches

Returns an `int` of the total number of wild pitches thrown by a team during a season.

win_percentage

Returns a `float` of the number of wins divided by the number of games played during the season. Percentage ranges from 0-1.

wins

Returns an `int` of the total number of games the team won during the season.

wins_last_ten_games

Returns an `int` of the number of wins in the last 10 games.

wins_last_thirty_games

Returns an `int` of the number of wins in the last 30 games.

wins_last_twenty_games

Returns an `int` of the number of wins in the last 20 games.

wins_vs_left_handed_pitchers

Returns an `int` of number of wins against left-handed pitchers.

wins_vs_right_handed_pitchers

Returns an `int` of the number of wins against right-handed pitchers.

wins_vs_teams_over_500

Returns an `int` of the number of wins against teams over 500.

wins_vs_teams_under_500

Returns an `int` of the number of wins against teams under 500.

class `sportsreference.mlb.teams.Teams` (*year=None*)

Bases: `object`

A list of all MLB teams and their stats in a given year.

Finds and retrieves a list of all MLB teams from www.baseball-reference.com and creates a `Team` instance for every team that participated in the league in a given year. The `Team` class comprises a list of all major stats and a few identifiers for the requested season.

Parameters `year` (*string (optional)*) – The requested year to pull stats from.

dataframes

Returns a pandas `DataFrame` where each row is a representation of the `Team` class. Rows are indexed by the team abbreviation.

`sportsreference.mlb.teams.mlb_int_property_decorator` (*func*)

1.6.1.2 NBA Package

The NBA package offers multiple modules which can be use to retrieve information and statistics for the National Basketball Association, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of points scored to the number of free throws made, to the assist rate and much more. The Boxscore can be easily queried by passing a boxscore's URI on sports-reference.com which can be retrieved from the `Schedule` class (see `Schedule` module below for more information on retrieving game-specific information).


```

from sportsreference.nba.boxscore import Boxscore

game_data = Boxscore('201806080CLE')
print(game_data.away_points)  # Prints 108
print(game_data.home_points)  # Prints 85
df = game_data.dataframe  # Returns a Pandas DataFrame of game metrics

```

The Boxscore module also contains a Boxscores class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.

```

from datetime import datetime
from sportsreference.nba.boxscore import Boxscores

games_today = Boxscores(datetime.today())
print(games_today.games)  # Prints a dictionary of all matchups for today

```

The Boxscores class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```

from datetime import datetime
from sportsreference.nba.boxscore import Boxscores

# Pulls all games between and including January 1, 2018 and January 5, 2018
games = Boxscores(datetime(2018, 1, 1), datetime(2018, 1, 5))
# Prints a dictionary of all results from January 1, 2018 and January 5,
# 2018
print(games.games)

```

class sportsreference.nba.boxscore.Boxscore (*uri*)

Bases: object

Detailed information about the final statistics for a game.

Stores all relevant metrics for a game such as the date, time, location, result, and more advanced metrics such as the effective field goal rate, the true shooting percentage, the game's pace, and much more.

Parameters *uri* (*string*) – The relative link to the boxscore HTML page, such as '201710310LAL'.

away_assist_percentage

Returns a float of the percentage of the away team's field goals that were assisted. Percentage ranges from 0-100.

away_assists

Returns an int of the total number of assists by the away team.

away_block_percentage

Returns a float of the percentage of 2-point field goals that were blocked by the away team. Percentage ranges from 0-100.

away_blocks

Returns an int of the total number of blocks by the away team.

away_defensive_rating

Returns a float of the average number of points scored per 100 possessions by the away team.

away_defensive_rebound_percentage

Returns a float of the percentage of available defensive rebounds the away team grabbed. Percentage

ranges from 0-100.

away_defensive_rebounds

Returns an `int` of the total number of defensive rebounds by the away team.

away_effective_field_goal_percentage

Returns a `float` of the away team's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

away_field_goal_attempts

Returns an `int` of the total number of field goal attempts by the away team.

away_field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts by the away team. Percentage ranges from 0-1.

away_field_goals

Returns an `int` of the total number of field goals made by the away team.

away_free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt by the away team.

away_free_throw_attempts

Returns an `int` of the total number of free throw attempts by the away team.

away_free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts by the away team.

away_free_throws

Returns an `int` of the total number of free throws made by the away team.

away_losses

Returns an `int` of the number of games the team has lost after the conclusion of the game.

away_minutes_played

Returns an `int` of the total number of minutes the team played during the game.

away_offensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the away team.

away_offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the away team grabbed. Percentage ranges from 0-100.

away_offensive_rebounds

Returns an `int` of the total number of offensive rebounds by the away team.

away_personal_fouls

Returns an `int` of the total number of personal fouls by the away team.

away_players

Returns a list of `BoxscorePlayer` class instances for each player on the away team.

away_points

Returns an `int` of the number of points the away team scored.

away_steal_percentage

Returns a `float` of the percentage of possessions that ended in a steal by the away team. Percentage ranges from 0-100.

away_steals

Returns an `int` of the total number of steals by the away team.

away_three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range by the away team. Percentage ranges from 0-1.

away_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts by the away team.

away_three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts by the away team. Percentage ranges from 0-1.

away_three_point_field_goals

Returns an `int` of the total number of three point field goals made by the away team.

away_total_rebound_percentage

Returns a `float` of the percentage of available rebounds the away team grabbed. Percentage ranges from 0-100.

away_total_rebounds

Returns an `int` of the total number of rebounds by the away team.

away_true_shooting_percentage

Returns a `float` of the away team's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

away_turnover_percentage

Returns a `float` of the number of times the away team turned the ball over per 100 possessions.

away_turnovers

Returns an `int` of the total number of turnovers by the away team.

away_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts by the away team.

away_two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts by the away team. Percentage ranges from 0-1.

away_two_point_field_goals

Returns an `int` of the total number of two point field goals made by the away team.

away_wins

Returns an `int` of the number of games the team has won after the conclusion of the game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string URI that is used to instantiate the class, such as '201710310LAL'.

date

Returns a `string` of the date the game took place.

home_assist_percentage

Returns a `float` of the percentage of the home team's field goals that were assisted. Percentage ranges from 0-100.

home_assists

Returns an `int` of the total number of assists by the home team.

home_block_percentage

Returns a `float` of the percentage of 2-point field goals that were blocked by the home team. Percentage ranges from 0-100.

home_blocks

Returns an `int` of the total number of blocks by the home team.

home_defensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the away team.

home_defensive_rebound_percentage

Returns a `float` of the percentage of available defensive rebounds the home team grabbed. Percentage ranges from 0-100.

home_defensive_rebounds

Returns an `int` of the total number of defensive rebounds by the home team.

home_effective_field_goal_percentage

Returns a `float` of the home team's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

home_field_goal_attempts

Returns an `int` of the total number of field goal attempts by the home team.

home_field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts by the home team. Percentage ranges from 0-1.

home_field_goals

Returns an `int` of the total number of field goals made by the home team.

home_free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt by the home team.

home_free_throw_attempts

Returns an `int` of the total number of free throw attempts by the home team.

home_free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts by the home team.

home_free_throws

Returns an `int` of the total number of free throws made by the home team.

home_losses

Returns an `int` of the number of games the home team lost after the conclusion of the game.

home_minutes_played

Returns an `int` of the total number of minutes the team played during the game.

home_offensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the home team.

home_offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the home team grabbed. Percentage ranges from 0-100.

home_offensive_rebounds

Returns an `int` of the total number of offensive rebounds by the home team.

home_personal_fouls

Returns an `int` of the total number of personal fouls by the home team.

home_players

Returns a list of `BoxscorePlayer` class instances for each player on the home team.

home_points

Returns an `int` of the number of points the home team scored.

home_steal_percentage

Returns a `float` of the percentage of possessions that ended in a steal by the home team. Percentage ranges from 0-100.

home_steals

Returns an `int` of the total number of steals by the home team.

home_three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range by the home team. Percentage ranges from 0-1.

home_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts by the home team.

home_three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts by the home team. Percentage ranges from 0-1.

home_three_point_field_goals

Returns an `int` of the total number of three point field goals made by the home team.

home_total_rebound_percentage

Returns a `float` of the percentage of available rebounds the home team grabbed. Percentage ranges from 0-100.

home_total_rebounds

Returns an `int` of the total number of rebounds by the home team.

home_true_shooting_percentage

Returns a `float` of the home team's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

home_turnover_percentage

Returns a `float` of the number of times the home team turned the ball over per 100 possessions.

home_turnovers

Returns an `int` of the total number of turnovers by the home team.

home_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts by the home team.

home_two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts by the home team. Percentage ranges from 0-1.

home_two_point_field_goals

Returns an `int` of the total number of two point field goals made by the home team.

home_wins

Returns an `int` of the number of games the home team won after the conclusion of the game.

location

Returns a `string` of the name of the venue where the game was played.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'PHO' for the Phoenix Suns.

losing_name

Returns a `string` of the losing team's name, such as 'Phoenix Suns'.

pace

Returns a `float` of the game's overall pace, measured by the number of possessions per 40 minutes.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'DET' for the Detroit Pistons.

winning_name

Returns a `string` of the winning team's name, such as 'Detroit Pistons'.

class `sportsreference.nba.boxscore.BoxscorePlayer` (*player_id*, *player_name*,
player_data)

Bases: `sportsreference.nba.player.AbstractPlayer`

Get player stats for an individual game.

Given a player ID, such as 'hardeja01' for James Harden, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id** (*string*) – A player's ID according to basketball-reference.com, such as 'hardeja01' for James Harden. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'LLLLLFFNN' where 'LLLLL' are the first 5 letters in the player's last name, 'FF', are the first 2 letters in the player's first name, and 'NN' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'James Harden'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values for the specified game.

defensive_rating

Returns an `int` of the player's defensive rating as measured by the points allowed per 100 possessions.

minutes_played

Returns a `float` of the number of game minutes the player was on the court for.

offensive_rating

Returns an `int` of the player's offensive rating as measured by the points produced per 100 possessions.

two_point_attempts

Returns an `int` of the total number of two point field goals the player attempted during the season.

two_point_percentage

Returns a float of the player's two point field goal percentage during the season. Percentage ranges from 0-1.

two_pointers

Returns an int of the total number of two point field goals the player made.

class sportsreference.nba.boxscore.Boxscores (*date*, *end_date=None*)

Bases: object

Search for NBA games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **date** (*datetime object*) – The date to search for any matches. The month, day, and year are required for the search, but time is not factored into the search.
- **end_date** (*datetime object (optional)*) – Optionally specify an end date to iterate until. All boxscores starting from the date specified in the 'date' parameter up to and including the boxscores specified in the 'end_date' parameter will be pulled. If left empty, or if 'end_date' is prior to 'date', only the games from the day specified in the 'date' parameter will be saved.

games

Returns a dictionary object representing all of the games played on the requested day. Dictionary is in the following format:

```
{ 'date' : [ # 'date' is the string date in format 'MM-DD-YYYY'
    {
        'home_name': Name of the home team, such as 'Phoenix Suns'
                      (`str`),
        'home_abbr': Abbreviation for the home team, such as 'PHO'
                      (`str`),
        'away_name': Name of the away team, such as 'Houston
                      Rockets' (`str`),
        'away_abbr': Abbreviation for the away team, such as 'HOU'
                      (`str`),
        'boxscore': String representing the boxscore URI, such as
                      '201702040PHO' (`str`),
        'winning_name': Full name of the winning team, such as
                      'Houston Rockets' (`str`),
        'winning_abbr': Abbreviation for the winning team, such as
                      'HOU' (`str`),
        'losing_name': Full name of the losing team, such as
                      'Phoenix Suns' (`str`),
        'losing_abbr': Abbreviation for the losing team, such as
                      'PHO' (`str`),
        'home_score': Integer score for the home team (`int`),
        'away_score': Integer score for the away team (`int`)
    },
    { ... },
    ...
  ]
}
```

If no games were played on 'date', the list for ['date'] will be empty.

Player

The Player module contains an abstract base class that can be inherited by both the `BoxscorePlayer` and `Player` classes in the `Boxscore` and `Roster` modules, respectively. All of the properties that appear in the `AbstractPlayer` class can be read from either of the two child classes mentioned above.

class `sportsreference.nba.player.AbstractPlayer` (*player_id, player_name, player_data*)
Bases: `object`

Get player information and stats for all seasons.

Given a player ID, such as 'hardeja01' for James Harden, capture all relevant stats and information like name, nationality, height/weight, career three-pointers, last season's offensive rebounds, salary, contract amount, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on basketball-reference.com.

Parameters

- **player_id** (*string*) – A player's ID according to basketball-reference.com, such as 'hardeja01' for James Harden. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'LLLLLFFNN' where 'LLLLL' are the first 5 letters in the player's last name, 'FF', are the first 2 letters in the player's first name, and 'NN' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'James Harden'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

assist_percentage

Returns a `float` of the percentage of field goals the player assisted while on the floor. Percentage ranges from 0-100.

assists

Returns an `int` of the total number of assists the player tallied during the season.

block_percentage

Returns a `float` of the percentage of opposing two-point field goal attempts that were blocked by the player while on the floor. Percentage ranges from 0-100.

blocks

Returns an `int` of the total number of shots the player blocked during the season.

box_plus_minus

Returns a `float` of the total number of points per 100 possessions the player contributed in comparison to an average player in the league.

defensive_rebound_percentage

Returns a `float` of the percentage of available defensive rebounds the player grabbed. Percentage ranges from 0-100.

defensive_rebounds

Returns an `int` of the total number of defensive rebounds the player grabbed during the season.

effective_field_goal_percentage

Returns a `float` of the player's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

field_goal_attempts

Returns an `int` of the total number of field goals the player attempted during the season.

field_goal_percentage

Returns a `float` of the player's field goal percentage during the season. Percentage ranges from 0-1.

field_goals

Returns an `int` of the total number of field goals the player scored.

free_throw_attempt_rate

Returns a `float` of the number of free throw attempts per field goal attempt.

free_throw_attempts

Returns an `int` of the total number of free throws the player attempted during the season.

free_throw_percentage

Returns a `float` of the player's free throw percentage during the season. Percentage ranges from 0-1.

free_throws

Returns an `int` of the total number of free throws the player made during the season.

minutes_played

Returns an `int` of the total number of minutes the player played.

name

Returns a `string` of the players name, such as 'James Harden'.

offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the player grabbed. Percentage ranges from 0-100.

offensive_rebounds

Returns an `int` of the total number of offensive rebounds the player grabbed during the season.

personal_fouls

Returns an `int` of the total number of personal fouls the player committed during the season.

player_id

Returns a `string` of the player's ID on sports-reference, such as 'hardeja01' for James Harden.

points

Returns an `int` of the total number of points the player scored during the season.

steal_percentage

Returns a `float` of the percentage of defensive possessions that ended with the player stealing the ball while on the floor. Percentage ranges from 0-100.

steals

Returns an `int` of the total number of steals the player tallied during the season.

three_point_attempt_rate

Returns a `float` of the percentage of field goals that are shot from beyond the 3-point arc. Percentage ranges from 0-1.

three_point_attempts

Returns an `int` of the total number of three point field goals the player attempted during the season.

three_point_percentage

Returns a `float` of the player's three point field goal percentage during the season. Percentage ranges from 0-1.

three_pointers

Returns an `int` of the total number of three point field goals the player made.

total_rebound_percentage

Returns a `float` of the percentage of available rebounds the player grabbed, both offensive and defensive. Percentage ranges from 0-100.

total_rebounds

Returns an `int` of the total number of offensive and defensive rebounds the player grabbed during the season.

true_shooting_percentage

Returns a `float` of the player's true shooting percentage which takes into account two and three pointers as well as free throws. Percentage ranges from 0-1.

turnover_percentage

Returns a `float` of the average number of turnovers per 100 possessions by the player.

turnovers

Returns an `int` of the total number of times the player turned the ball over during the season for any reason.

usage_percentage

Returns a `float` of the percentage of plays the player is involved in while on the floor. Percentage ranges from 0-100.

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the `Player` class which has detailed information ranging from career points totals to single-season stats and player height, weight, and nationality. The following is an example on collecting career information for James Harden:

```
from sportsreference.nba.roster import Player

james_harden = Player('hardeja01')
print(james_harden.name) # Prints 'James Harden'
print(james_harden.points) # Prints Harden's career points total
# Prints a Pandas DataFrame of all relevant Harden stats per season
print(james_harden.dataframe)
print(james_harden.salary) # Prints Harden's career earnings
print(james_harden.contract) # Prints Harden's contract by yearly wages
```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific season, call the class instance with the season string. All future property requests will return the season-specific stats.

```
from sportsreference.nba.roster import Player

james_harden = Player('hardeja01') # Currently pulling career stats
print(james_harden.points) # Prints Harden's CAREER points total
# Prints Harden's points total only for the 2017-18 season.
print(james_harden('2017-18').points)
# Prints the number of games Harden played in the 2017-18 season.
print(james_harden.games_played)
```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.nba.roster import Player

james_harden = Player('hardeja01') # Currently pulling career stats
# Prints Harden's points total only for the 2017-18 season.
print(james_harden('2017-18').points)
print(james_harden('Career').points) # Prints Harden's career points total
```

In addition, the Roster module also contains the Roster class which can be used to pull all players on a team's roster during a given season and creates instances of the Player class for each team member and adds them to a list to be easily queried.

```
from sportsreference.nba.roster import Roster

houston = Roster('HOU')
for player in houston.players:
    # Prints the name of all players who played for Houston in the most
    # recent season.
    print(player.name)
```

class sportsreference.nba.roster.Player(*player_id*)
Bases: *sportsreference.nba.player.AbstractPlayer*

Get player information and stats for all seasons.

Given a player ID, such as 'hardeja01' for James Harden, capture all relevant stats and information like name, nationality, height/weight, career three-pointers, last season's offensive rebounds, salary, contract amount, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on basketball-reference.com.

Parameters *player_id* (*string*) – A player's ID according to basketball-reference.com, such as 'hardeja01' for James Harden. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'LLLLLFFNN' where 'LLLLL' are the first 5 letters in the player's last name, 'FF', are the first 2 letters in the player's first name, and 'NN' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.

and_ones

Returns an *int* of the total number of times the player was fouled in the act of shooting and made the basket.

birth_date

Returns a *datetime* object of the day and year the player was born.

blocking_fouls

Returns an *int* of the total number of blocking fouls the player committed.

center_percentage

Returns an *int* of the percentage of time the player spent as a center. Percentage ranges from 0-100 and is rounded to the nearest whole number.

contract

Returns a *dictionary* of the player's contract details where the key is a *string* of the season, such as '2018-19', and the value is a *string* of the salary, such as '\$40,000,000'.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values where each index is a different season plus the career stats.

defensive_box_plus_minus

Returns a `float` of the number of defensive points per 100 possessions the player contributed in comparison to an average player in the league.

defensive_win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his defensive plays.

dunks

Returns an `int` of the total number of dunks the player made during the season.

field_goal_perc_sixteen_foot_plus_two_pointers

Returns a `float` of the player's field goal percentage for shots that are greater than sixteen feet from the basket, but in front of or on the three point arc. Percentage ranges from 0-1.

field_goal_perc_ten_to_sixteen_feet

Returns a `float` of the player's field goal percentage for shots between ten and sixteen feet from the basket. Percentage ranges from 0-1.

field_goal_perc_three_to_ten_feet

Returns a `float` of the player's field goal percentage for shots between three and ten feet from the basket. Percentage ranges from 0-1.

field_goal_perc_zero_to_three_feet

Returns a `float` of the player's field goal percentage for shots between zero and three feet from the basket. Percentage ranges from 0-1.

games_played

Returns an `int` of the number of games the player participated in.

games_started

Returns an `int` of the number of games the player started.

half_court_heaves

Returns an `int` of the number of shots the player took from beyond mid-court.

half_court_heaves_made

Returns an `int` of the number of shots the player made from beyond mid-court.

height

Returns a `string` of the player's height in the format "feet-inches".

lost_ball_turnovers

Returns an `int` of the total number of turnovers the player committed due to losing the ball.

nationality

Returns a `string` constant denoting which country the player originates from.

net_plus_minus

Returns a `float` of the net number of points the player contributes to the team per 100 possessions regardless of being on the floor or not.

offensive_box_plus_minus

Returns a `float` of the number of offensive points per 100 possessions the player contributed in comparison to an average player in the league.

offensive_fouls

Returns an `int` of the total number of offensive fouls the player committed.

offensive_win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his offensive plays.

on_court_plus_minus

Returns a `float` of the number of points the player contributes to the team while on the court per 100 possessions.

other_turnovers

Returns an `int` of the total number of all other non-passing/ dribbling turnovers the player committed.

passing_turnovers

Returns an `int` of the total number of turnovers the player committed due to a bad pass.

percentage_field_goals_as_dunks

Returns a `float` of the percentage of the player's shot attempts that are dunks. Percentage ranges from 0-1.

percentage_of_three_pointers_from_corner

Returns a `float` of the percentage of 3-point shots the player attempted from the corner. Percentage ranges from 0-1.

percentage_shots_three_pointers

Returns a `float` of the percentage of shots the player takes from beyond the three point arc. Percentage ranges from 0-1.

percentage_shots_two_pointers

Returns a `float` of the percentage of shots the player takes that are 2-pointers. Percentage ranges from 0-1.

percentage_sixteen_foot_plus_two_pointers

Returns a `float` of the percentage of shots the player takes that are greater than sixteen feet from the basket, but in front of or on the three point arc. Percentage ranges from 0-1.

percentage_ten_to_sixteen_footers

Returns a `float` of the percentage of shots the player takes from ten to sixteen feet from the basket. Percentage ranges from 0-1.

percentage_three_to_ten_footers

Returns a `float` of the percentage of shots the player takes from three to ten feet from the basket. Percentage ranges from 0-1.

percentage_zero_to_three_footers

Returns a `float` of the percentage of shots the player takes from zero to three feet from the basket. Percentage ranges from 0-1.

player_efficiency_rating

Returns a `float` of the player's efficiency rating which represents the player's relative production level. An average player in the league has an efficiency rating of 15.

point_guard_percentage

Returns an `int` of the percentage of time the player spent as a point guard. Percentage ranges from 0-100 and is rounded to the nearest whole number.

points_generated_by_assists

Returns an `int` of the total number of points the player generated as a result of him assisting the shooter.

position

Returns a `string` constant of the player's primary position.

power_forward_percentage

Returns an `int` of the percentage of time the player spent as a power forward. Percentage ranges from 0-100 and is rounded to the nearest whole number.

salary

Returns an `int` of the player's annual salary rounded down.

season

Returns a `string` of the season in the format 'YYYY-YY', such as '2017-18'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

shooting_distance

Returns a `float` of the average distance the player takes a shot from in feet.

shooting_fouls

Returns an `int` of the total number of shooting fouls the player committed.

shooting_fouls_drawn

Returns an `int` of the total number of shooting fouls the player drew during the season.

shooting_guard_percentage

Returns an `int` of the percentage of time the player spent as a shooting guard. Percentage ranges from 0-100 and is rounded to the nearest whole number.

shots_blocked

Returns an `int` of the total number of shots the player took that were blocked by an opposing player.

small_forward_percentage

Returns an `int` of the percentage of time the player spent as a small forward. Percentage ranges from 0-100 and is rounded to the nearest whole number.

take_fouls

Returns an `int` of the total number of take fouls the player committed by taking a foul before the offensive player has a chance to make a shooting motion.

team_abbreviation

Returns a `string` of the abbreviation for the team the player plays for, such as 'HOU' for James Harden.

three_point_shot_percentage_from_corner

Returns a `float` of the percentage of 3-pointers from the corner that went in. Percentage ranges from 0-1.

three_pointers_assisted_percentage

Returns a `float` of the percentage of 3-point field goals by the player that are assisted. Percentage ranges from 0-1.

two_point_attempts

Returns an `int` of the total number of two point field goals the player attempted during the season.

two_point_percentage

Returns a `float` of the player's two point field goal percentage during the season. Percentage ranges from 0-1.

two_pointers

Returns an `int` of the total number of two point field goals the player made.

two_pointers_assisted_percentage

Returns a `float` of the percentage of 2-point field goals by the player that are assisted. Percentage ranges from 0-1.

value_over_replacement_player

Returns a `float` of the total number of points per 100 team possessions the player contributed compared

to a replacement-level player (who has an average score of -2.0). This value is prorated for an 82-game season.

weight

Returns an `int` of the player's weight in pounds.

win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his offensive and defensive plays.

win_shares_per_48_minutes

Returns a `float` of the number of wins the player contributed to the team per 48 minutes of playtime. An average player has a contribution of 0.100.

class `sportsreference.nba.roster.Roster` (*team*, *year=None*, *slim=False*)

Bases: `object`

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the `Player` class for each player, containing a detailed list of the players statistics and information.

Parameters

- **team** (*string*) – The team's 3-letter abbreviation, such as 'HOU' for the Houston Rockets.
- **year** (*string (optional)*) – The 4-digit year to pull the roster from, such as '2018'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to `True` to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to `False`.

players

Returns a `list` of player instances for each player on the requested team's roster if the `slim` property is `False` when calling the `Roster` class. If the `slim` property is `True`, returns a `dictionary` where each key is a string of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The `Schedule` module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the `Boxscore` class which has much more detailed information on the game metrics.

```
from sportsreference.nba.schedule import Schedule

houston_schedule = Schedule('HOU')
for game in houston_schedule:
    print(game.date)    # Prints the date the game was played
    print(game.result)  # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class `sportsreference.nba.schedule.Game` (*game_data*, *playoffs=False*)

Bases: `object`

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters `game_data` (*string*) – The row containing the specified game information.

boxscore

Returns an instance of the Boxscore class containing more detailed stats on the game.

boxscore_index

Returns a *string* of the URI for a boxscore which can be used to access or index a game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the boxscore string.

dataframe_extended

Returns a pandas DataFrame representing the Boxscore class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property. The index for the DataFrame is the boxscore string.

date

Returns a *string* of the date the game took place at, such as 'Wed, Oct 18, 2017'.

datetime

Returns a datetime object to indicate the month, day, and year the game took place.

game

Returns an *int* to indicate which game in the season was requested. The first game of the season returns 1.

location

Returns a *string* constant to indicate whether the game was played in the team's home arena or on the road.

losses

Returns an *int* of the number of losses the team has in the season after the completion of the listed game.

opponent_abbr

Returns a *string* of the opponent's 3-letter abbreviation, such as 'CHI' for the Chicago Bulls.

opponent_name

Returns a *string* of the opponent's name, such as 'Chicago Bulls'.

playoffs

Returns a *boolean* variable which evaluates to True when the game was played in the playoffs and returns False if the game took place in the regular season.

points_allowed

Returns an *int* of the number of points the team allowed during the game.

points_scored

Returns an *int* of the number of points the team scored during the game.

result

Returns a *string* constant to indicate whether the team won or lost the game.

streak

Returns a *string* of the team's current streak after the conclusion of the listed game, such as 'W 3' for a 3-game winning streak.

time

Returns a *string* of the time the game started in Eastern Time, such as '8:01p'.

wins

Returns an int of the number of wins the team has in the season after the completion of the listed game.

class sportsreference.nba.schedule.**Schedule** (*abbreviation*, *year=None*)

Bases: object

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'PHO' for the Phoenix Suns.
- **year** (*string* (*optional*)) – The requested year to pull stats from.

dataframe

Returns a pandas DataFrame where each row is a representation of the Game class. Rows are indexed by the boxscore string.

dataframe_extended

Returns a pandas DataFrame where each row is a representation of the Boxscore class for every game in the schedule. Rows are indexed by the boxscore string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The Teams module exposes information for all NBA teams including the team name and abbreviation, the number of games they won during the season, the total number of shots they've blocked, and much more.

```
from sportsreference.nba.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    print(team.blocks)  # Prints the team's total blocked shots
```

Each Team instance contains a link to the Schedule class which enables easy iteration over all games for a particular team. A Pandas DataFrame can also be queried to easily grab all stats for all games.

```
from sportsreference.nba.teams import Teams

teams = Teams()
for team in teams:
    schedule = team.schedule  # Returns a Schedule instance for each team
    # Returns a Pandas DataFrame of all metrics for all game Boxscores for
    # a season.
    df = team.schedule.dataframe_extended
```

Lastly, each Team instance also contains a link to the Roster class which enables players from the team to be easily queried. Each Roster instance contains detailed stats and information for each player on the team.

```
from sportsreference.nba.teams import Teams

teams = Teams()
for team in teams:
    # Creates an instance of the roster class for each player on the team.
    roster = team.roster
```

(continues on next page)

(continued from previous page)

```
for player in roster.players:
    print(player.name) # Prints the name of each player on the team.
```

class sportsreference.nba.teams.**Team**(team_data, rank, year=None)

Bases: object

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as rank, name, and abbreviation, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **rank** (*int*) – A team's position in the league based on the number of points they obtained during the season.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a *string* of the team's abbreviation, such as 'DET' for the Detroit Pistons.

assists

Returns an *int* of the total number of field goals that were assisted.

blocks

Returns an *int* of the total number of times the team blocked an opponent's shot.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'DET'.

defensive_rebounds

Returns an *int* of the total number of defensive rebounds the team has grabbed.

field_goal_attempts

Returns an *int* of the total number of field goals the team has attempted during the season.

field_goal_percentage

Returns a *float* of the percentage of field goals made divided by the number of attempts. Percentage ranges from 0-1.

field_goals

Returns an *int* of the total number of field goals the team has made during the season.

free_throw_attempts

Returns an *int* of the total number of free throw attempts during the season.

free_throw_percentage

Returns a *float* of the percentage of free throws made divided by the attempts. Percentage ranges from 0-1.

free_throws

Returns an *int* of the total number of free throws made during the season.

games_played

Returns an *int* of the total number of games the team has played during the season.

minutes_played

Returns an `int` of the total number of minutes played by all players on the team during the season.

name

Returns a `string` of the team's full name, such as 'Detroit Pistons'.

offensive_rebounds

Returns an `int` of the total number of offensive rebounds the team has grabbed.

opp_assists

Returns an `int` of the total number of field goals that were assisted by the opponent.

opp_blocks

Returns an `int` of the total number of times the opponent blocked the team's shot.

opp_defensive_rebounds

Returns an `int` of the total number of defensive rebounds the opponent grabbed.

opp_field_goal_attempts

Returns an `int` of the total number of field goals the opponents attempted during the season.

opp_field_goal_percentage

Returns a `float` of the percentage of field goals made divided by the number of attempts by the opponent. Percentage ranges from 0-1.

opp_field_goals

Returns an `int` of the total number of field goals the opponents made during the season.

opp_free_throw_attempts

Returns an `int` of the total number of free throw attempts during the season by the opponent.

opp_free_throw_percentage

Returns a `float` of the percentage of free throws made divided by the attempts by the opponent. Percentage ranges from 0-1.

opp_free_throws

Returns an `int` of the total number of free throws made during the season by the opponent.

opp_offensive_rebounds

Returns an `int` of the total number of offensive rebounds the opponent grabbed.

opp_personal_fouls

Returns an `int` of the total number of times the opponent fouled the team.

opp_points

Returns an `int` of the total number of points the team has been scored on during the season.

opp_steals

Returns an `int` of the total number of times the opponent stole the ball from the team.

opp_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goals the opponent attempted during the season.

opp_three_point_field_goal_percentage

Returns a `float` of the percentage of three point field goals made divided by the number of attempts by the opponent. Percentage ranges from 0-1.

opp_three_point_field_goals

Returns an `int` of the total number of three point field goals the opponent made during the season.

opp_total_rebounds

Returns an `int` of the total number of rebounds the opponent grabbed.

opp_turnovers

Returns an `int` of the total number of times the opponent turned the ball over.

opp_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goals the opponent attempted during the season.

opp_two_point_field_goal_percentage

Returns a `float` of the percentage of two point field goals made divided by the number of attempts by the opponent. Percentage ranges from 0-1.

opp_two_point_field_goals

Returns an `int` of the total number of two point field goals the opponent made during the season.

personal_fouls

Returns an `int` of the total number of times the team has fouled an opponent.

points

Returns an `int` of the total number of points the team has scored during the season.

rank

Returns an `int` of the team's rank based on the number of points they score per game.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

schedule

Returns an instance of the Schedule class containing the team's complete schedule for the season.

steals

Returns an `int` of the total number of times the team stole the ball from the opponent.

three_point_field_goal_attempts

Returns an `int` of the total number of three point field goals the team has attempted during the season.

three_point_field_goal_percentage

Returns a `float` of the percentage of three point field goals made divided by the number of attempts. Percentage ranges from 0-1.

three_point_field_goals

Returns an `int` of the total number of three point field goals the team has made during the season.

total_rebounds

Returns an `int` of the total number of rebounds the team has grabbed.

turnovers

Returns an `int` of the total number of times the team has turned the ball over.

two_point_field_goal_attempts

Returns an `int` of the total number of two point field goals the team has attempted during the season.

two_point_field_goal_percentage

Returns a `float` of the percentage of two point field goals made divided by the number of attempts. Percentage ranges from 0-1.

two_point_field_goals

Returns an `int` of the total number of two point field goals the team has made during the season.

class sportsreference.nba.teams.Teams (year=None)

Bases: object

A list of all NBA teams and their stats in a given year.

Finds and retrieves a list of all NBA teams from www.basketball-reference.com and creates a Team instance for every team that participated in the league in a given year. The Team class comprises a list of all major stats and a few identifiers for the requested season.

Parameters `year` (*string (optional)*) – The requested year to pull stats from.

dataframes

Returns a pandas DataFrame where each row is a representation of the Team class. Rows are indexed by the team abbreviation.

1.6.1.3 NCAAB Package

The NCAAB package offers multiple modules which can be used to retrieve information and statistics for Men's Division I College Basketball, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of points scored to the number of blocked shots, to the assist percentage and much more. The Boxscore can be easily queried by passing a boxscore's URI on sports-reference.com which can be retrieved from the Schedule class (see Schedule module below for more information on retrieving game-specific information).

```
from sportsreference.ncaab.boxscore import Boxscore

game_data = Boxscore('2018-04-02-21-villanova')
print(game_data.home_points) # Prints 79
print(game_data.away_points) # Prints 62
df = game_data.dataframe # Returns a Pandas DataFrame of game metrics
```

The Boxscore module also contains a Boxscores class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.

```
from datetime import datetime
from sportsreference.ncaab.boxscore import Boxscores

games_today = Boxscores(datetime.today())
print(games_today.games) # Prints a dictionary of all matchups for today
```

The Boxscores class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```
from datetime import datetime
from sportsreference.ncaab.boxscore import Boxscores

# Pulls all games between and including November 11, 2017 and November 12,
# 2017
games = Boxscores(datetime(2017, 11, 11), datetime(2017, 11, 12))
# Prints a dictionary of all results from November 11, 2017 and November 12,
# 2017
print(games.games)
```

```
class sportsreference.ncaab.boxscore.Boxscore(uri)
    Bases: object
```

Detailed information about the final statistics for a game.

Stores all relevant metrics for a game such as the date, time, location, result, and more advanced metrics such as the effective field goal rate, the true shooting percentage, the game's pace, and much more.

Parameters `uri` (*string*) – The relative link to the boxscore HTML page, such as '2017-11-10-21-kansas'.

away_assist_percentage

Returns a `float` of the percentage of the away team's field goals that were assisted. Percentage ranges from 0-100.

away_assists

Returns an `int` of the total number of assists by the away team.

away_block_percentage

Returns a `float` of the percentage of 2-point field goals that were blocked by the away team. Percentage ranges from 0-100.

away_blocks

Returns an `int` of the total number of blocks by the away team.

away_defensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the away team.

away_defensive_rebound_percentage

Returns a `float` of the percentage of available defensive rebounds the away team grabbed. Percentage ranges from 0-100.

away_defensive_rebounds

Returns an `int` of the total number of defensive rebounds by the away team.

away_effective_field_goal_percentage

Returns a `float` of the away team's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

away_field_goal_attempts

Returns an `int` of the total number of field goal attempts by the away team.

away_field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts by the away team. Percentage ranges from 0-1.

away_field_goals

Returns an `int` of the total number of field goals made by the away team.

away_free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt by the away team.

away_free_throw_attempts

Returns an `int` of the total number of free throw attempts by the away team.

away_free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts by the away team.

away_free_throws

Returns an `int` of the total number of free throws made by the away team.

away_losses

Returns an `int` of the number of games the team has lost after the conclusion of the game.

away_minutes_played

Returns an `int` of the total number of minutes the team played during the game.

away_offensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the away team.

away_offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the away team grabbed. Percentage ranges from 0-100.

away_offensive_rebounds

Returns an `int` of the total number of offensive rebounds by the away team.

away_personal_fouls

Returns an `int` of the total number of personal fouls by the away team.

away_players

Returns a list of `BoxscorePlayer` class instances for each player on the away team.

away_points

Returns an `int` of the number of points the away team scored.

away_ranking

Returns an `int` of the away team's ranking during the week, or `None` if the team wasn't ranked.

away_steal_percentage

Returns a `float` of the percentage of possessions that ended in a steal by the away team. Percentage ranges from 0-100.

away_steals

Returns an `int` of the total number of steals by the away team.

away_three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range by the away team. Percentage ranges from 0-1.

away_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts by the away team.

away_three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts by the away team. Percentage ranges from 0-1.

away_three_point_field_goals

Returns an `int` of the total number of three point field goals made by the away team.

away_total_rebound_percentage

Returns a `float` of the percentage of available rebounds the away team grabbed. Percentage ranges from 0-100.

away_total_rebounds

Returns an `int` of the total number of rebounds by the away team.

away_true_shooting_percentage

Returns a `float` of the away team's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

away_turnover_percentage

Returns a `float` of the number of times the away team turned the ball over per 100 possessions.

away_turnovers

Returns an `int` of the total number of turnovers by the away team.

away_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts by the away team.

away_two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts by the away team. Percentage ranges from 0-1.

away_two_point_field_goals

Returns an `int` of the total number of two point field goals made by the away team.

away_win_percentage

Returns a `float` of the percentage of games the away team has won after the conclusion of the game. Percentage ranges from 0-1.

away_wins

Returns an `int` of the number of games the team has won after the conclusion of the game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string URI that is used to instantiate the class, such as '2017-11-10-21-kansas'.

date

Returns a `string` of the date the game took place.

home_assist_percentage

Returns a `float` of the percentage of the home team's field goals that were assisted. Percentage ranges from 0-100.

home_assists

Returns an `int` of the total number of assists by the home team.

home_block_percentage

Returns a `float` of the percentage of 2-point field goals that were blocked by the home team. Percentage ranges from 0-100.

home_blocks

Returns an `int` of the total number of blocks by the home team.

home_defensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the away team.

home_defensive_rebound_percentage

Returns a `float` of the percentage of available defensive rebounds the home team grabbed. Percentage ranges from 0-100.

home_defensive_rebounds

Returns an `int` of the total number of defensive rebounds by the home team.

home_effective_field_goal_percentage

Returns a `float` of the home team's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

home_field_goal_attempts

Returns an `int` of the total number of field goal attempts by the home team.

home_field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts by the home team. Percentage ranges from 0-1.

home_field_goals

Returns an `int` of the total number of field goals made by the home team.

home_free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt by the home team.

home_free_throw_attempts

Returns an `int` of the total number of free throw attempts by the home team.

home_free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts by the home team.

home_free_throws

Returns an `int` of the total number of free throws made by the home team.

home_losses

Returns an `int` of the number of games the home team lost after the conclusion of the game.

home_minutes_played

Returns an `int` of the total number of minutes the team played during the game.

home_offensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the home team.

home_offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the home team grabbed. Percentage ranges from 0-100.

home_offensive_rebounds

Returns an `int` of the total number of offensive rebounds by the home team.

home_personal_fouls

Returns an `int` of the total number of personal fouls by the home team.

home_players

Returns a list of `BoxscorePlayer` class instances for each player on the home team.

home_points

Returns an `int` of the number of points the home team scored.

home_ranking

Returns an `int` of the home team's ranking during the week, or `None` if they were not ranked.

home_steal_percentage

Returns a `float` of the percentage of possessions that ended in a steal by the home team. Percentage ranges from 0-100.

home_steals

Returns an `int` of the total number of steals by the home team.

home_three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range by the home team. Percentage ranges from 0-1.

home_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts by the home team.

home_three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts by the home team. Percentage ranges from 0-1.

home_three_point_field_goals

Returns an `int` of the total number of three point field goals made by the home team.

home_total_rebound_percentage

Returns a `float` of the percentage of available rebounds the home team grabbed. Percentage ranges from 0-100.

home_total_rebounds

Returns an `int` of the total number of rebounds by the home team.

home_true_shooting_percentage

Returns a `float` of the home team's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

home_turnover_percentage

Returns a `float` of the number of times the home team turned the ball over per 100 possessions.

home_turnovers

Returns an `int` of the total number of turnovers by the home team.

home_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts by the home team.

home_two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts by the home team. Percentage ranges from 0-1.

home_two_point_field_goals

Returns an `int` of the total number of two point field goals made by the home team.

home_win_percentage

Returns a `float` of the percentage of games the home team has won after the conclusion of the game. Percentage ranges from 0-1.

home_wins

Returns an `int` of the number of games the home team won after the conclusion of the game.

location

Returns a `string` of the name of the venue where the game was played.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'INDIANA' for the Indiana Hoosiers.

losing_name

Returns a `string` of the losing team's name, such as 'Indiana' Hoosiers'.

pace

Returns a `float` of the game's overall pace, measured by the number of possessions per 40 minutes.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'PURDUE' for the Purdue Boilermakers.

winning_name

Returns a `string` of the winning team's name, such as 'Purdue Boilermakers'.

class `sportsreference.ncaab.boxscore.BoxscorePlayer` (*player_id*, *player_name*,
player_data)

Bases: `sportsreference.ncaab.player.AbstractPlayer`

Get player stats for an individual game.

Given a player ID, such as 'carsen-edwards-1' for Carsen Edwards, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id**(*string*) – A player’s ID accorsing to sports-reference.com, such as ‘carsen-edwards-1’ for Carsen Edwards. The player ID can be found by navigating to the player’s stats page and getting the string between the final slash and the ‘.html’ in the URL. In general, the ID is in the format ‘first-last-N’ where ‘first’ is the player’s first name in lowercase, ‘last’ is the player’s last name in lowercase, and ‘N’ is a number starting at ‘1’ for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name**(*string*) – A string representing the player’s first and last name, such as ‘Carsen Edwards’.
- **player_data**(*string*) – A string representation of the player’s HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values for the specified game.

defensive_rating

Returns an `int` of the player’s defensive rating as measured by the points allowed per 100 possessions.

offensive_rating

Returns an `int` of the player’s offensive rating as measured by the points produced per 100 possessions.

class `sportsreference.ncaab.boxscore.Boxscores` (*date*, *end_date=None*)

Bases: `object`

Search for NCAAB games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, a boolean value which indicates if the game is between two Division-I teams or not, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **date** (*datetime object*) – The date to search for any matches. The month, day, and year are required for the search, but time is not factored into the search.
- **end_date** (*datetime object*) – Optionally specify an end date to iterate until. All boxscores starting from the date specified in the ‘date’ parameter up to and including the boxscores specified in the ‘end_date’ parameter will be pulled. If left empty, or if ‘end_date’ is prior to ‘date’, only the games from the day specified in the ‘date’ parameter will be saved.

games

Returns a `dictionary object` representing all of the games played on the requested day. Dictionary is in the following format:

```
{'date' : [ # 'date' is the string date in format 'MM-DD-YYYY'
  {
    'home_name': Name of the home team, such as 'Purdue
                  Boilermakers' (`str`),
    'home_abbr': Abbreviation for the home team, such as
```

(continues on next page)

(continued from previous page)

```

        'PURDUE' (`str`),
    'away_name': Name of the away team, such as 'Indiana
                  Hoosiers' (`str`),
    'away_abbr': Abbreviation for the away team, such as
                  'INDIANA' (`str`),
    'boxscore': String representing the boxscore URI, such as
                  '2018-01-28-15-indiana' (`str`),
    'non_di': Boolean value which evaluates to True when at
               least one of the teams does not compete in NCAA
               Division-I basketball (`bool`),
    'top_25': Boolean value which evaluates to True when at
              least one of the teams is ranked in the AP Top 25
              polls (`bool`),
    'winning_name': Full name of the winning team, such as
                    'Purdue Boilermakers' (`str`),
    'winning_abbr': Abbreviation for the winning team, such as
                    'PURDUE' (`str`),
    'losing_name': Full name of the losing team, such as
                   'Indiana Hoosiers' (`str`),
    'losing_abbr': Abbreviation for the losing team, such as
                   'INDIANA' (`str`),
    'home_score': Integer score for the home team (`int`),
    'home_rank': Integer representing the home team's rank
                 (`int`),
    'away_score': Integer score for the away team (`int`),
    'away_rank': Integer representing the away team's rank
                 (`int`)
    },
    { ... },
    ...
]
}

```

If no games were played on ‘date’, the list for [‘date’] will be empty.

Conferences

The Conference module allows conferences to be pulled for any season using the `Conferences` class. Accessing the class properties exposes various dictionaries containing the team and conference abbreviations as well as other information. To get a list of conference abbreviations for each team, query the `team_conference` property.

```

from sportsreference.ncaab.conferences import Conferences

conferences = Conferences()
# Prints a dictionary of the team abbreviation as a key and conference
# abbreviation as the value.
print(conferences.team_conference)

```

The `conferences` property can also be queried to provide more details on the teams in every conference.

```

from sportsreference.ncaab.conferences import Conferences

conferences = Conferences()
# Prints a dictionary where each key is the conference abbreviation and
# each value is a dictionary containing the full conference name as well as

```

(continues on next page)

(continued from previous page)

```
# another dictionary of all teams in the conference, including name and
# abbreviation for each team.
print(conferences.conferences)
```

```
class sportsreference.ncaab.conferences.Conference (conference_abbreviation,
                                                    year=None)
```

Bases: object

Find teams that participated in a particular conference.

Create a dictionary which includes the names and abbreviations for all teams that participated in a conference during a given year.

Parameters

- **conference_abbreviation** (*string*) – A string of the requested conference’s abbreviation, such as ‘big-12’.
- **year** (*string (optional)*) – A string of the requested year to pull conference information from. Defaults to the most recent season.

teams

Returns a dictionary of team names and abbreviations where each key is a string of the team abbreviation and each value is a string of the full team name.

```
class sportsreference.ncaab.conferences.Conferences (year=None)
```

Bases: object

Get all conferences and teams for a season.

Retrieve a list of all conferences and teams that participated in the conference for each team in the season. The included properties allow flexibility in queries to either get the conference abbreviation for a given team, or get more detailed information including all teams for each conference.

Parameters **year** (*string (optional)*) – A string of the requested year to pull conferences from. Defaults to the most recent season.

conferences

Returns a dictionary of conference names and abbreviations where each key is a string of the abbreviation and each value is a dictionary containing the full conference name and another dictionary with individual team information. The overall dictionary is in the following structure:

```
{
    abbreviation, ie 'big-12' (str): {
        'name': Full conference name, such as 'Big 12 Conference'
            (str),
        'teams': {
            team abbreviation, such as 'kansas' (str): Full team
                name, such as 'Kansas' (str),
            ...
        }
    },
    ...
}
```

team_conference

Returns a dictionary of conference abbreviations for each team where each key is a string of the team abbreviation and each value is a string of the conference abbreviation.

Player

The Player module contains an abstract base class that can be inherited by both the `BoxscorePlayer` and `Player` classes in the `Boxscore` and `Roster` modules, respectively. All of the properties that appear in the `AbstractPlayer` class can be read from either of the two child classes mentioned above.

```
class sportsreference.ncaab.player.AbstractPlayer(player_id, player_name,  
                                                player_data)
```

Bases: `object`

Get player information and stats for all seasons.

Given a player ID, such as 'carsen-edwards-1' for Carsen Edwards, capture all relevant stats and information like name, height/weight, career three-pointers, last season's offensive rebounds, offensive points contributed, and much more.

Parameters

- **player_id** (*string*) – A player's ID according to sports-reference.com, such as 'carsen-edwards-1' for Carsen Edwards. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'first-last-N' where 'first' is the player's first name in lowercase, 'last' is the player's last name in lowercase, and 'N' is a number starting at '1' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'Carsen Edwards'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

assist_percentage

Returns a `float` of the percentage of field goals the player assisted while on the floor. Percentage ranges from 0-100.

assists

Returns an `int` of the total number of assists the player tallied during the season.

block_percentage

Returns a `float` of the percentage of opposing two-point field goal attempts that were blocked by the player while on the floor. Percentage ranges from 0-100.

blocks

Returns an `int` of the total number of shots the player blocked during the season.

defensive_rebound_percentage

Returns a `float` of the percentage of available defensive rebounds the player grabbed. Percentage ranges from 0-100.

defensive_rebounds

Returns an `int` of the total number of defensive rebounds the player grabbed during the season.

effective_field_goal_percentage

Returns a `float` of the player's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

field_goal_attempts

Returns an `int` of the total number of field goals the player attempted during the season.

field_goal_percentage

Returns a `float` of the player's field goal percentage during the season. Percentage ranges from 0-1.

field_goals

Returns an `int` of the total number of field goals the player scored.

free_throw_attempt_rate

Returns a `float` of the number of free throw attempts per field goal attempt.

free_throw_attempts

Returns an `int` of the total number of free throws the player attempted during the season.

free_throw_percentage

Returns a `float` of the player's free throw percentage during the season. Percentage ranges from 0-1.

free_throws

Returns an `int` of the total number of free throws the player made during the season.

minutes_played

Returns an `int` of the total number of minutes the player played.

name

Returns a `string` of the players name, such as 'Carsen Edwards'.

offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the player grabbed. Percentage ranges from 0-100.

offensive_rebounds

Returns an `int` of the total number of offensive rebounds the player grabbed during the season.

personal_fouls

Returns an `int` of the total number of personal fouls the player committed during the season.

player_id

Returns a `string` of the player's ID on sports-reference, such as 'carsen-edwards-1' for Carsen Edwards.

points

Returns an `int` of the total number of points the player scored during the season.

steal_percentage

Returns a `float` of the percentage of defensive possessions that ended with the player stealing the ball while on the floor. Percentage ranges from 0-100.

steals

Returns an `int` of the total number of steals the player tallied during the season.

three_point_attempt_rate

Returns a `float` of the percentage of field goals that are shot from beyond the 3-point arc. Percentage ranges from 0-1.

three_point_attempts

Returns an `int` of the total number of three point field goals the player attempted during the season.

three_point_percentage

Returns a `float` of the player's three point field goal percentage during the season. Percentage ranges from 0-1.

three_pointers

Returns an `int` of the total number of three point field goals the player made.

total_rebound_percentage

Returns a `float` of the percentage of available rebounds the player grabbed, both offensive and defensive. Percentage ranges from 0-100.

total_rebounds

Returns an `int` of the total number of offensive and defensive rebounds the player grabbed during the season.

true_shooting_percentage

Returns a `float` of the player's true shooting percentage which takes into account two and three pointers as well as free throws. Percentage ranges from 0-1.

turnover_percentage

Returns a `float` of the average number of turnovers per 100 possessions by the player.

turnovers

Returns an `int` of the total number of times the player turned the ball over during the season for any reason.

two_point_attempts

Returns an `int` of the total number of two point field goals the player attempted during the season.

two_point_percentage

Returns a `float` of the player's two point field goal percentage during the season. Percentage ranges from 0-1.

two_pointers

Returns an `int` of the total number of two point field goals the player made.

usage_percentage

Returns a `float` of the percentage of plays the player is involved in while on the floor. Percentage ranges from 0-100.

Rankings

The Rankings module includes the `Rankings` class which can be used to easily query the NCAA Men's Division-I Basketball rankings published by the Associated Press on a week-by-week basis. Different formats can be referenced, ranging from a lightweight dictionary of the most recent rankings containing only the team abbreviation and rank, to a much larger dictionary of all rankings for an entire season with results including full team name and abbreviation, current rank, week number, previous rank, and movement.

```
from sportsreference.ncaab.rankings import Rankings

rankings = Rankings()
# Prints a dictionary of just the team abbreviation and rank for the current
# week
print(rankings.current)
# Prints more detailed information including previous rank, full name, and
# movement for all teams in current week
print(rankings.current_extended)
# Prints detailed information for all teams for all weeks where rankings
# have been published for the requested season.
print(rankings.complete)
```

class `sportsreference.ncaab.rankings.Rankings` (`year=None`)

Bases: `object`

Get all Associated Press (AP) rankings on a week-by-week basis.

Grab a list of the rankings published by the Associated Press to easily query the hierarchy of teams each week. The results expose the current and previous rankings as well as the movement for each team in the list.

Parameters `year` (*string (optional)*) – A string of the requested year to pull rankings from. Defaults to the most recent season.

complete

Returns a dictionary where each key is a week number as an `int` and each value is a list of dictionaries containing the AP rankings for each week. Within each list is a dictionary of team information such as name, abbreviation, rank, and more. Note that the list might not necessarily be in the same order as the rankings.

The overall dictionary has the following structure:

```
{
  week number, ie 19 (int): [
    {
      'abbreviation': Team's abbreviation, such as 'PURDUE'
                        (str),
      'name': Team's full name, such as 'Purdue' (str),
      'rank': Team's rank for the current week (int),
      'week': Week number for the results, such as 19 (int),
      'date': Date the rankings were released, such as
              '2017-03-01'. Can also be 'Final' for the final
              rankings or 'Preseason' for preseason rankings
              (str),
      'previous': The team's previous rank, if applicable
                  (str),
      'change': The amount the team moved up or down the
                 rankings. Moves up the ladder have a positive
                 number while drops yield a negative number
                 and teams that didn't move have 0 (int)
    },
    ...
  ],
  ...
}
```

current

Returns a dictionary of the most recent rankings from the Associated Press where each key is a string of the team's abbreviation and each value is an `int` of the team's rank for the current week.

current_extended

Returns a list of dictionaries of the most recent AP rankings. The list is ordered in terms of the ranking so the #1 team will be in the first element and the #25 team will be the last element. Each dictionary has the following structure:

```
{
  'abbreviation': Team's abbreviation, such as 'PURDUE' (str),
  'name': Team's full name, such as 'Purdue' (str),
  'rank': Team's rank for the current week (int),
  'week': Week number for the results, such as 19 (int),
  'date': Date the rankings were released, such as '2017-03-01'.
          Can also be 'Final' for the final rankings or
          'Preseason' for preseason rankings (str),
  'previous': The team's previous rank, if applicable (str),
  'change': The amount the team moved up or down the rankings.
             Moves up the ladder have a positive number while
             drops yield a negative number and teams that didn't
             move have 0 (int)
}
```

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the `Player` class which has detailed information ranging from career points totals to single-season stats and player height and weight. The following is an example on collecting career information for Carsen Edwards.

```
from sportsreference.ncaab.roster import Player

carsen_edwards = Player('carsen-edwards-1')
print(carsen_edwards.name)  # Prints 'Carsen Edwards'
print(carsen_edwards.points)  # Prints Edwards' career points total
# Prints a Pandas DataFrame of all relevant stats per season for Edwards
print(carsen_edwards.dataframe)
```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific season, call the class instance with the season string. All future property requests will return the season-specific stats.

```
from sportsreference.ncaab.roster import Player

carsen_edwards = Player('carsen-edwards-1')  # Currently pulling career stats
print(carsen_edwards.points)  # Prints Edwards' CAREER points total
# Prints Edwards' points total only for the 2017-18 season.
print(carsen_edwards('2017-18').points)
# Prints the number of games Edwards played in the 2017-18 season.
print(carsen_edwards.games_played)
```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.ncaab.roster import Player

carsen_edwards = Player('carsen-edwards-1')  # Currently pulling career stats
# Prints Edwards' points total only for the 2017-18 season.
print(carsen_edwards('2017-18').points)
print(carsen_edwards('Career').points)  # Prints Edwards' career points total
```

In addition, the Roster module also contains the `Roster` class which can be used to pull all players on a team's roster during a given season and creates instances of the `Player` class for each team member and adds them to a list to be easily queried.

```
from sportsreference.ncaab.roster import Roster

purdue = Roster('PURDUE')
for player in purdue.players:
    # Prints the name of all players who played for Purdue in the most
    # recent season.
    print(player.name)
```

class `sportsreference.ncaab.roster.Player` (*player_id*)
Bases: `sportsreference.ncaab.player.AbstractPlayer`

Get player information and stats for all seasons.

Given a player ID, such as 'carsen-edwards-1' for Carsen Edwards, capture all relevant stats and information like name, height/weight, career three-pointers, last season's offensive rebounds, offensive points contributed, and much more.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on sports-reference.com.

Parameters `player_id` (*string*) – A player's ID according to sports-reference.com, such as 'carsen-edwards-1' for Carsen Edwards. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'first-last-N' where 'first' is the player's first name in lowercase, 'last' is the player's last name in lowercase, and 'N' is a number starting at '1' for the first time that player ID has been used and increments by 1 for every successive player.

box_plus_minus

Returns a `float` of the total number of points per 100 possessions the player contributed in comparison to an average player in the league.

conference

Returns a `string` of the abbreviation for the conference the team participated in for the requested season.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values where each index is a different season plus the career stats.

defensive_box_plus_minus

Returns a `float` of the number of defensive points per 100 possessions the player contributed in comparison to an average player in the league.

defensive_win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his defensive plays.

games_played

Returns an `int` of the number of games the player participated in.

games_started

Returns an `int` of the number of games the player started.

height

Returns a `string` of the player's height in the format "feet-inches".

offensive_box_plus_minus

Returns a `float` of the number of offensive points per 100 possessions the player contributed in comparison to an average player in the league.

offensive_win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his offensive plays.

player_efficiency_rating

Returns a `float` of the player's efficiency rating which represents the player's relative production level. An average player in the league has an efficiency rating of 15.

points_produced

Returns an `int` of the number of offensive points the player produced.

position

Returns a `string` constant of the player's primary position.

season

Returns a `string` of the season in the format 'YYYY-YY', such as '2017-18'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

team_abbreviation

Returns a `string` of the abbreviation for the team the player plays for, such as 'PURDUE' for Carsen Edwards.

weight

Returns an `int` of the player's weight in pounds.

win_shares

Returns a `float` of the number of wins the player contributed to the team as a result of his offensive and defensive plays.

win_shares_per_40_minutes

Returns a `float` of the number of wins the player contributed to the team per 40 minutes of playtime. An average player has a contribution of 0.100.

class sportsreference.ncaab.roster.**Roster** (*team*, *year=None*, *slim=False*)

Bases: `object`

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the `Player` class for each player, containing a detailed list of the players statistics and information.

Parameters

- **team** (*string*) – The team's abbreviation, such as 'PURDUE' for the Purdue Boilermakers.
- **year** (*string (optional)*) – The 4-digit year to pull the roster from, such as '2018'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to `True` to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to `False`.

players

Returns a `list` of player instances for each player on the requested team's roster if the `slim` property is `False` when calling the `Roster` class. If the `slim` property is `True`, returns a `dictionary` where each key is a `string` of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The `Schedule` module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the `Boxscore` class which has much more detailed information on the game metrics.

```
from sportsreference.ncaab.schedule import Schedule

purdue_schedule = Schedule('PURDUE')
for game in purdue_schedule:
    print(game.date)    # Prints the date the game was played
    print(game.result)  # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class sportsreference.ncaab.schedule.**Game** (*game_data*)

Bases: `object`

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters `game_data` (*string*) – The row containing the specified game information.

arena

Returns a *string* of the name of the arena the game was played at.

boxscore

Returns an instance of the Boxscore class containing more detailed stats on the game.

boxscore_index

Returns a *string* of the URI for a boxscore which can be used to access or index a game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the boxscore string.

dataframe_extended

Returns a pandas DataFrame representing the Boxscore class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property. The index for the DataFrame is the boxscore string.

date

Returns a *string* of the game's date, such as 'Fri, Nov 10, 2017'.

datetime

Returns a datetime object to indicate the month, day, year, and time the requested game took place.

game

Returns an *int* of the game's position in the season. The first game of the season returns 1.

location

Returns a *string* constant to indicate whether the game was played at the team's home venue, the opponent's venue, or at a neutral site.

opponent_abbr

Returns a *string* of the opponent's abbreviation, such as 'PURDUE' for the Purdue Boilermakers.

opponent_conference

Returns a *string* of the opponent's conference, such as 'Big Ten' for a team participating in the Big Ten Conference. If the team is not a Division-I school, a string constant for non-majors is returned.

opponent_name

Returns a *string* of the opponent's name, such as the 'Purdue Boilermakers'.

opponent_rank

Returns a *string* of the opponent's rank when the game was played and None if the team was unranked.

overtimes

Returns an *int* of the number of overtimes that were played during the game and 0 if the game finished at the end of regulation time.

points_against

Returns an *int* of the number of points the team allowed during the game.

points_for

Returns an *int* of the number of points the team scored during the game.

result

Returns a *string* constant to indicate whether the team won or lost the game.

season_losses

Returns an `int` of the number of games the team has lost after the conclusion of the requested game.

season_wins

Returns an `int` of the number of games the team has won after the conclusion of the requested game.

streak

Returns a `string` of the team's win streak at the conclusion of the requested game. Streak is in the format '[W/L] #' (ie. 'W 3' indicates a 3-game winning streak while 'L 2' indicates a 2-game losing streak.

time

Returns a `string` to indicate the time the game started, such as '9:00 pm/est'.

type

Returns a `string` constant to indicate whether the game was played during the regular season or in the post season.

class `sportsreference.ncaab.schedule.Schedule` (*abbreviation*, *year=None*)

Bases: `object`

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'PURDUE' for the Purdue Boilermakers.
- **year** (*string* (*optional*)) – The requested year to pull stats from.

dataframe

Returns a pandas DataFrame where each row is a representation of the Game class. Rows are indexed by the boxscore string.

dataframe_extended

Returns a pandas DataFrame where each row is a representation of the Boxscore class for every game in the schedule. Rows are indexed by the boxscore string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The Teams module exposes information for all NCAAB teams including the team name and abbreviation, the number of games they won during the season, the total number of shots they've blocked, and much more.

```
from sportsreference.ncaab.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    print(team.blocks)  # Prints the number of shots the team blocked
```

Each Team instance contains a link to the `Schedule` class which enables easy iteration over all games for a particular team. A Pandas DataFrame can also be queried to easily grab all stats for all games.

```
from sportsreference.ncaab.teams import Teams

teams = Teams()
for team in teams:
```

(continues on next page)

(continued from previous page)

```

schedule = team.schedule # Returns a Schedule instance for each team
# Returns a Pandas DataFrame of all metrics for all game Boxscores for
# a season.
df = team.schedule.dataframe_extended

```

Lastly, each Team instance also contains a link to the Roster class which enables players from the team to be easily queried. Each Roster instance contains detailed stats and information for each player on the team.

```

from sportsreference.ncaab.teams import Teams

for team in Teams():
    roster = team.roster # Gets each team's roster
    for player in roster.players:
        print(player.name) # Prints each players name on the roster

```

```

class sportsreference.ncaab.teams.Team(team_data, team_conference=None, year=None)
    Bases: object

```

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as full and short names, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **team_conference** (*string (optional)*) – A string of the team's conference abbreviation, such as 'big-12'.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a *string* of the team's short name, such as 'PURDUE' for the Purdue Boilermakers.

assist_percentage

Returns a *float* of the percentage of field goals that were assisted. Percentage ranges from 0-100.

assists

Returns an *int* of the total number of assists during the season.

away_losses

Returns an *int* of the total number of away games the team lost during the season.

away_wins

Returns an *int* of the total number of away games the team won during the season.

block_percentage

Returns a *float* of the percentage of 2-point field goals by the opponent that were blocked. Percentage ranges from 0-100.

blocks

Returns an *int* of the total number of blocks during the season.

conference

Returns a *string* of the team's conference abbreviation, such as 'big-12' for the Big 12 Conference.

conference_losses

Returns an *int* of the total number of conference games the team lost during the season.

conference_wins

Returns an `int` of the total number of conference games the team won during the season.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'PURDUE'.

defensive_rebounds

Returns an `int` of the total number of defensive rebounds during the season.

effective_field_goal_percentage

Returns a `float` of the field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

field_goal_attempts

Returns an `int` of the total number of field goal attempts during the season.

field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts. Percentage ranges from 0-1.

field_goals

Returns an `int` of the total number of field goals made during the season.

free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt.

free_throw_attempts

Returns an `int` of the total number of free throw attempts during the season.

free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts during the season.

free_throws

Returns an `int` of the total number of free throws made during the season.

free_throws_per_field_goal_attempt

Returns a `float` of the number of free throws per field goal attempt.

games_played

Returns an `int` of the total number of games the team has played during the season.

home_losses

Returns an `int` of the total number of home games the team lost during the season.

home_wins

Returns an `int` of the total number of home games the team won during the season.

losses

Returns an `int` of the total number of games the team lost during the season.

minutes_played

Returns an `int` of the total number of minutes played by the team during the season.

name

Returns a `string` of the team's full name, such as 'Purdue Boilermakers'.

net_rating

Returns a `float` of the net team rating which is equivalent to the difference between the offensive rating and the defensive (or the opponent's offensive) rating. Positive values indicate teams that score more points than they allow per 100 possessions.

offensive_rating

Returns a `float` of the average number of points scored per 100 possessions.

offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds a team grabbed. Percentage ranges from 0-100.

offensive_rebounds

Returns an `int` of the total number of offensive rebounds during the season.

opp_assist_percentage

Returns a `float` of the percentage of the opponent's field goals that were assisted. Percentage ranges from 0-100.

opp_assists

Returns an `int` of the total number of assists during the season by opponents.

opp_block_percentage

Returns a `float` of the percentage of 2-point field goals that were blocked by the opponent. Percentage ranges from 0-100.

opp_blocks

Returns an `int` of the total number of blocks during the season by opponents.

opp_defensive_rebounds

Returns an `int` of the total number of defensive rebounds during the season by opponents.

opp_effective_field_goal_percentage

Returns a `float` of the opponent's field goal percentage while giving extra weight to 3-point field goals. Percentage ranges from 0-1.

opp_field_goal_attempts

Returns an `int` of the total number of field goal attempts during the season by opponents.

opp_field_goal_percentage

Returns a `float` of the number of field goals made divided by the total number of field goal attempts by opponents. Percentage ranges from 0-1.

opp_field_goals

Returns an `int` of the total number of field goals made during the season by opponents.

opp_free_throw_attempt_rate

Returns a `float` of the average number of free throw attempts per field goal attempt by the opponent.

opp_free_throw_attempts

Returns an `int` of the total number of free throw attempts during the season by opponents.

opp_free_throw_percentage

Returns a `float` of the number of free throws made divided by the number of free throw attempts during the season by opponents.

opp_free_throws

Returns an `int` of the total number of free throws made during the season by opponents.

opp_free_throws_per_field_goal_attempt

Returns a `float` of the number of free throws per field goal attempt by the opponent.

opp_offensive_rating

Returns a `float` of the average number of points scored per 100 possessions by the opponent. This is equivalent to the team's defensive rating as it is the number of points the team allows per 100 possessions by the opponent.

opp_offensive_rebound_percentage

Returns a `float` of the percentage of available offensive rebounds the opponent grabbed. Percentage ranges from 0-100.

opp_offensive_rebounds

Returns an `int` of the total number of offensive rebounds during the season by opponents.

opp_personal_fouls

Returns an `int` of the total number of personal fouls during the season by opponents.

opp_points

Returns an `int` of the total number of points opponents have scored during the season.

opp_steal_percentage

Returns a `float` of the percentage of possessions that ended in a steal by the opponent. Percentage ranges from 0-100.

opp_steals

Returns an `int` of the total number of steals during the season by opponents.

opp_three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range by the opponent. Percentage ranges from 0-1.

opp_three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts during the season by opponents.

opp_three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts by opponents. Percentage ranges from 0-1.

opp_three_point_field_goals

Returns an `int` of the total number of three point field goals made during the season by opponents.

opp_total_rebound_percentage

Returns a `float` of the percentage of available rebounds the opponent grabbed. Percentage ranges from 0-100.

opp_total_rebounds

Returns an `int` of the total number of rebounds during the season by opponents.

opp_true_shooting_percentage

Returns a `float` of the opponent's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

opp_turnover_percentage

Returns a `float` of the number of times the opponent turned the ball over per 100 possessions.

opp_turnovers

Returns an `int` of the total number of turnovers during the season by opponents.

opp_two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts during the season by opponents.

opp_two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts by opponents. Percentage ranges from 0-1.

opp_two_point_field_goals

Returns an `int` of the total number of two point field goals made during the season by opponents.

pace

Returns a `float` of the average number of possessions per 40 minutes.

personal_fouls

Returns an `int` of the total number of personal fouls during the season.

points

Returns an `int` of the total number of points the team scored during the season.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

schedule

Returns an instance of the Schedule class containing the team's complete schedule for the season.

simple_rating_system

Returns a `float` of the team's average point differential compared to the strength of schedule. Higher values indicate stronger teams. An average team is denoted with 0.0. Negative numbers are comparatively worse than average.

steal_percentage

Returns a `float` of the percentage of opponent possessions that ended in a steal. Percentage ranges from 0-100.

steals

Returns an `int` of the total number of steals during the season.

strength_of_schedule

Returns a `float` of the team's strength of schedule based on the points above and below average. An average strength of schedule is denoted with 0.0. Negative numbers are comparatively easier than average.

three_point_attempt_rate

Returns a `float` of the percentage of field goal attempts from 3-point range. Percentage ranges from 0-1.

three_point_field_goal_attempts

Returns an `int` of the total number of three point field goal attempts during the season.

three_point_field_goal_percentage

Returns a `float` of the number of three point field goals made divided by the number of three point field goal attempts. Percentage ranges from 0-1.

three_point_field_goals

Returns an `int` of the total number of three point field goals made during the season.

total_rebound_percentage

Returns a `float` of the percentage of available rebounds a team grabbed. Percentage ranges from 0-100.

total_rebounds

Returns an `int` of the total number of rebounds during the season.

true_shooting_percentage

Returns a `float` of the team's true shooting percentage which considers free throws, 2-point field goals, and 3-point field goals. Percentage ranges from 0-1.

turnover_percentage

Returns a `float` of the number of times the team turned the ball over per 100 possessions.

turnovers

Returns an `int` of the total number of turnovers during the season.

two_point_field_goal_attempts

Returns an `int` of the total number of two point field goal attempts during the season.

two_point_field_goal_percentage

Returns a `float` of the number of two point field goals made divided by the number of two point field goal attempts. Percentage ranges from 0-1.

two_point_field_goals

Returns an `int` of the total number of two point field goals made during the season.

win_percentage

Returns a `float` of the number of wins divided by the number of games played during the season. Percentage ranges from 0-1.

wins

Returns an `int` of the total number of games the team won during the season.

class sportsreference.ncaab.teams.Teams (year=None)

Bases: `object`

A list of all NCAA Men's Basketball teams and their stats in a given year.

Finds and retrieves a list of all NCAA Men's Basketball teams from www.sports-reference.com and creates a Team instance for every team that participated in the league in a given year. The Team class comprises a list of all major stats and a few identifiers for the requested season.

Parameters `year` (*string optional*) – The requested year to pull stats from.

dataframes

Returns a pandas DataFrame where each row is a representation of the Team class. Rows are indexed by the team abbreviation.

1.6.1.4 NCAAF Package

The NCAAF package offers multiple modules which can be used to retrieve information and statistics for Division-I College Football, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of points scored to the number of pass yards, to the yards from penalties and much more. The Boxscore can be easily queried by passing a boxscore's URI on sports-reference.com which can be retrieved from the Schedule class (see Schedule module below for more information on retrieving game-specific information).

```
from sportsreference.ncaaf.boxscore import Boxscore

game_data = Boxscore('2018-01-08-georgia')
print(game_data.home_points) # Prints 23
print(game_data.away_points) # Prints 26
df = game_data.dataframe # Returns a Pandas DataFrame of game metrics
```

The Boxscore module also contains a Boxscores class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.

```
from datetime import datetime
from sportsreference.ncaaf.boxscore import Boxscores

games_today = Boxscores(datetime.today())
print(games_today.games) # Prints a dictionary of all matchups for today
```

The `Boxscores` class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```
from datetime import datetime
from sportsreference.ncaaf.boxscore import Boxscores

# Pulls all games between and including August 30, 2017 and August 31, 2017
games = Boxscores(datetime(2017, 8, 30), datetime(2017, 8, 31))
# Prints a dictionary of all results from August 30, 2017 and August 31,
# 2017
print(games.games)
```

class `sportsreference.ncaaf.boxscore.Boxscore(uri)`

Bases: `object`

Detailed information about the final statistics for a game.

Stores all relevant information for a game such as the date, time, location, result, and more advanced metrics such as the number of fumbles from sacks, a team's passing completion, rushing touchdowns and much more.

Parameters `uri` (*string*) – The relative link to the boxscore HTML page, such as ‘2018-01-08-georgia’.

away_first_downs

Returns an `int` of the number of first downs the away team gained.

away_fumbles

Returns an `int` of the number of times the away team fumbled the ball.

away_fumbles_lost

Returns an `int` of the number of times the away team turned the ball over as the result of a fumble.

away_interceptions

Returns an `int` of the number of interceptions the away team threw.

away_pass_attempts

Returns an `int` of the number of passes that were thrown by the away team.

away_pass_completions

Returns an `int` of the number of completed passes the away team made.

away_pass_touchdowns

Returns an `int` of the number of passing touchdowns the away team scored.

away_pass_yards

Returns an `int` of the number of passing yards the away team gained.

away_penalties

Returns an `int` of the number of penalties called on the away team.

away_players

Returns a list of `BoxscorePlayer` class instances for each player on the away team.

away_points

Returns an `int` of the number of points the away team scored.

away_rush_attempts

Returns an `int` of the number of rushing plays the away team made.

away_rush_touchdowns

Returns an `int` of the number of rushing touchdowns the away team scored.

away_rush_yards

Returns an `int` of the number of rushing yards the away team gained.

away_total_yards

Returns an `int` of the total number of yards the away team gained.

away_turnovers

Returns an `int` of the number of times the away team turned the ball over.

away_yards_from_penalties

Returns an `int` of the number of yards gifted as a result of penalties called on the away team.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string URI that is used to instantiate the class, such as '2018-01-08-georgia'.

date

Returns a `string` of the date the game took place.

home_first_downs

Returns an `int` of the number of first downs the home team gained.

home_fumbles

Returns an `int` of the number of times the home team fumbled the ball.

home_fumbles_lost

Returns an `int` of the number of times the home team turned the ball over as the result of a fumble.

home_interceptions

Returns an `int` of the number of interceptions the home team threw.

home_pass_attempts

Returns an `int` of the number of passes that were thrown by the home team.

home_pass_completions

Returns an `int` of the number of completed passes the home team made.

home_pass_touchdowns

Returns an `int` of the number of passing touchdowns the home team scored.

home_pass_yards

Returns an `int` of the number of passing yards the home team gained.

home_penalties

Returns an `int` of the number of penalties called on the home team.

home_players

Returns a `list` of `BoxscorePlayer` class instances for each player on the home team.

home_points

Returns an `int` of the number of points the home team scored.

home_rush_attempts

Returns an `int` of the number of rushing plays the home team made.

home_rush_touchdowns

Returns an `int` of the number of rushing touchdowns the home team scored.

home_rush_yards

Returns an `int` of the number of rushing yards the home team gained.

home_total_yards

Returns an `int` of the total number of yards the home team gained.

home_turnovers

Returns an `int` of the number of times the home team turned the ball over.

home_yards_from_penalties

Returns an `int` of the number of yards gifted as a result of penalties called on the home team.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'GEORGIA' for the Georgia Bulldogs.

losing_name

Returns a `string` of the losing team's name, such as 'Georgia'.

stadium

Returns a `string` of the name of the stadium where the game was played.

time

Returns a `string` of the time the game started.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'ALABAMA' for the Alabama Crimson Tide.

winning_name

Returns a `string` of the winning team's name, such as 'Alabama'.

```
class sportsreference.ncaaf.boxscore.BoxscorePlayer(player_id,           player_name,
                                                    player_data)
```

Bases: `sportsreference.ncaaf.player.AbstractPlayer`

Get player stats for an individual game.

Given a player ID, such as 'david-blough-1' for David Blough, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id** (*string*) – A player's ID according to sports-reference.com, such as 'david-blough-1' for David Blough. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'first-last-n' where 'first' is the player's first name, 'last' is the player's last name, and 'n' is a number starting at 1 for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'David Blough'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

average_kickoff_return_yards

Returns a `float` of the average number of yards the player gained per attempted kickoff return.

average_punt_return_yards

Returns a `float` of the average number of yards the player gained per attempted punt return.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and value for the specified game.

extra_point_percentage

Returns a `float` of the percentage of attempted extra points the player made. Percentage ranges from 0-100.

extra_points_attempted

Returns an `int` of the total number of extra points the player attempted.

field_goal_percentage

Returns a `float` of the percentage of attempted field goals the player made. Percentage ranges from 0-100.

field_goals_attempted

Returns an `int` of the total number of field goals the player attempted.

kickoff_return_yards

Returns an `int` of the total number of yards the player gained while the player attempted to return a kickoff.

kickoff_returns

Returns an `int` of the number of kickoffs the player attempted to return.

pass_yards_per_attempt

Returns a `float` of the average number of yards the player gained per pass attempt.

points_kicking

Returns an `int` of the total number of points the player gained from kicking field goals or extra points.

punt_return_yards

Returns an `int` of the total number of yards the player gained while the player attempted to return a punt.

punt_returns

Returns an `int` of the number of punts the player attempted to return.

punting_yards

Returns an `int` of the total number of yards the player punted the ball.

punting_yards_per_attempt

Returns a `float` of the average number of yards the player punted per attempt.

punts

Returns an `int` of the number of times the player punted the ball.

class sportsreference.ncaaf.boxscore.Boxscores (*date*, *end_date=None*)

Bases: `object`

Search for NCAAAF games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, a boolean value which indicates if the game is between two Division-I teams or not, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **date** (*datetime object*) – The date to search for any matches. The month, day, and year are required for the search, but time is not factored into the search.

- **end_date** (*datetime object*) – Optionally specify an end date to iterate until. All boxscores starting from the date specified in the ‘date’ parameter up to and including the boxscores specified in the ‘end_date’ parameter will be pulled. If left empty, or if ‘end_date’ is prior to ‘date’, only the games from the day specified in the ‘date’ parameter will be saved.

games

Returns a dictionary object representing all of the games played on the requested day. Dictionary is in the following format:

```
{'date' : [ # 'date' is the string date in format 'MM-DD-YYYY'
    {
        'home_name': Name of the home team, such as 'Purdue
                        Boilermakers' (`str`),
        'home_abbr': Abbreviation for the home team, such as
                        'PURDUE' (`str`),
        'away_name': Name of the away team, such as 'Indiana
                        Hoosiers' (`str`),
        'away_abbr': Abbreviation for the away team, such as
                        'INDIANA' (`str`),
        'boxscore': String representing the boxscore URI, such as
                        '2018-01-28-15-indiana' (`str`),
        'non_di': Boolean value which evaluates to True when at
                    least one of the teams does not compete in NCAA
                    Division-I basketball (`bool`),
        'top_25': Boolean value which evaluates to True when at
                    least one of the teams is ranked in the AP Top 25
                    polls (`bool`),
        'winning_name': Full name of the winning team, such as
                        'Purdue Boilermakers' (`str`),
        'winning_abbr': Abbreviation for the winning team, such as
                        'PURDUE' (`str`),
        'losing_name': Full name of the losing team, such as
                        'Indiana Hoosiers' (`str`),
        'losing_abbr': Abbreviation for the losing team, such as
                        'INDIANA' (`str`),
        'home_score': Integer score for the home team (`int`),
        'home_rank': Integer representing the home team's rank
                        (`int`),
        'away_score': Integer score for the away team (`int`),
        'away_rank': Integer representing the away team's rank
                        (`int`)
    },
    { ... },
    ...
]
```

If no games were played on ‘date’, the list for [‘date’] will be empty.

sportsreference.ncaaf.boxscore.ncaaf_int_property_sub_index (*func*)

Conferences

The Conference module allows conferences to be pulled for any season using the `Conferences` class. Accessing the class properties exposes various dictionaries containing the team and conference abbreviations as well as other information. To get a list of conference abbreviations for each team, query the `team_conference` property.

```
from sportsreference.ncaaf.conferences import Conferences

conferences = Conferences()
# Prints a dictionary of the team abbreviation as a key and conference
# abbreviation as the value.
print(conferences.team_conference)
```

The conferences property can also be queried to provide more details on the teams in every conference.

```
from sportsreference.ncaab.conferences import Conferences

conferences = Conferences()
# Prints a dictionary where each key is the conference abbreviation and
# each value is a dictionary containing the full conference name as well as
# another dictionary of all teams in the conference, including name and
# abbreviation for each team.
print(conferences.conferences)
```

```
class sportsreference.ncaaf.conferences.Conference (conference_abbreviation,
                                                    year=None,                ig-
                                                    ignore_missing=False)
```

Bases: object

Find teams that participated in a particular conference.

Create a dictionary which includes the names and abbreviations for all teams that participated in a conference during a given year.

Parameters

- **conference_abbreviation** (*string*) – A string of the requested conference’s abbreviation, such as ‘big-12’.
- **year** (*string (optional)*) – A string of the requested year to pull conference information from. Defaults to the most recent season.
- **ignore_missing** (*boolean (optional)*) – A boolean which, when True, doesn’t throw an error if the requested conference has an incomplete or empty page on www.sportsreference.com, preventing the parsing from completing successfully.

teams

Returns a dictionary of team names and abbreviations where each key is a string of the team abbreviation and each value is a string of the full team name.

```
class sportsreference.ncaaf.conferences.Conferences (year=None,                ig-
                                                    ignore_missing=False)
```

Bases: object

Get all conferences and teams for a season.

Retrieve a list of all conferences and teams that participated in the conference for each team in the season. The included properties allow flexibility in queries to either get the conference abbreviation for a given team, or get more detailed information including all teams for each conference.

Parameters

- **year** (*string (optional)*) – A string of the requested year to pull conferences from. Defaults to the most recent season.
- **ignore_missing** (*boolean (optional)*) – A boolean which, when True, doesn’t throw an error if the any conference has an incomplete or empty page on www.sportsreference.com, preventing the parsing from completing successfully.

conferences

Returns a dictionary of conference names and abbreviations where each key is a string of the abbreviation and each value is a dictionary containing the full conference name and another dictionary with individual team information. The overall dictionary is in the following structure:

```
{
  abbreviation, ie 'big-12' (str): {
    'name': Full conference name, such as 'Big 12 Conference'
        (str),
    'teams': {
      team abbreviation, such as 'kansas' (str): Full team
        name, such as 'Kansas' (str),
      ...
    }
  },
  ...
}
```

team_conference

Returns a dictionary of conference abbreviations for each team where each key is a string of the team abbreviation and each value is a string of the conference abbreviation.

Player

The Player module contains an abstract base class that can be inherited by both the BoxscorePlayer and Player classes in the Boxscore and Roster modules, respectively. All of the properties that appear in the AbstractPlayer class can be read from either of the two child classes mentioned above.

```
class sportsreference.ncaaf.player.AbstractPlayer (player_id,           player_name,
                                                  player_data)
```

Bases: object

Get player information and stats for all seasons.

Given a player ID, such as 'david-blough-1' for David Blough, capture all relevant stats and information like name, team, height/weight, career starts, single season passing yards, sacks, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on sports-reference.com.

Parameters

- **player_id** (*string*) – A player's ID according to sports-reference.com, such as 'david-blough-1' for David Blough. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'first-last-n' where 'first' is the player's first name in lowercase, 'last' is the player's last name in lowercase, and 'n' is a number starting at '1' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'David Blough'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

adjusted_yards_per_attempt

Returns a float of the adjusted number of yards gained per passing attempt, equal to (yards + 20 * pass_touchdowns - 45 * interceptions) / pass_attempts.

assists_on_tackles

Returns an `int` of the number of assists the player made on tackles.

attempted_passes

Returns an `int` of the number of passes the player attempted.

completed_passes

Returns an `int` of the number of completed passes the player threw.

extra_points_made

Returns an `int` of the number of extra points the player made.

field_goals_made

Returns an `int` of the total number of field goals the player made from any distance.

fumbles_forced

Returns an `int` of the number of times the player forced a fumble.

fumbles_recovered

Returns an `int` of the number of fumbles the player has recovered.

fumbles_recovered_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after recovering a fumble.

interceptions

Returns an `int` of the number of times the player intercepted a pass.

interceptions_returned_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after intercepting a pass. Commonly referred to as a 'Pick-6'.

interceptions_thrown

Returns an `int` of the number of interceptions the player has thrown.

kickoff_return_touchdowns

Returns an `int` of the number of kickoffs the player returned for a touchdown.

name

Returns a `string` of the player's name, such as 'David Blough'.

pass_attempts

Returns an `int` of the number of passes the player attempted.

passes_defended

Returns an `int` of the number of passes the player has defended as a defensive player.

passing_completion

Returns a `float` of the percentage of passes that were caught by a receiver. Percentage ranges from 0-100.

passing_touchdowns

Returns an `int` of the number of touchdowns passes the player has thrown.

passing_yards

Returns an `int` of the total number of yards the player gained from passing the ball.

passing_yards_per_attempt

Returns a `float` of the number of yards gained per passing attempt.

player_id

Returns a `string` of the player's ID on sports-reference, such as 'david-blough-1' for David Blough.

plays_from_scrimmage

Returns an `int` of the combined number of rushing attempts and receptions the player had.

punt_return_touchdowns

Returns an `int` of the number of punts the player returned for a touchdown.

quarterback_rating

Returns a `float` of the player's quarterback rating.

receiving_touchdowns

Returns an `int` of the number of touchdowns the player scored after receiving a pass.

receiving_yards

Returns an `int` of the number of receiving yards the player gained.

receiving_yards_per_reception

Returns a `float` of the average number of yards the player gained per reception.

receptions

Returns an `int` of the number of receptions the player made.

rush_attempts

Returns an `int` of the number of rushing plays the player attempted.

rush_touchdowns

Returns an `int` of the number of rushing touchdowns the player scored.

rush_yards

Returns an `int` of the number of rushing yards the player gained.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing attempt.

rushing_and_receiving_touchdowns

Returns an `int` of the combined number of rushing and receiving touchdowns the player scored.

sacks

Returns a `float` of the number of times the player sacked a quarterback.

solo_tackles

Returns an `int` of the number of tackles the player made by himself.

tackles_for_loss

Returns a `float` of the number of tackles for a loss the player made.

total_tackles

Returns an `int` of the number of tackles the player made.

yards_from_scrimmage

Returns an `int` of the total number of yards gained from scrimmage for both rushing and receiving.

yards_from_scrimmage_per_play

Returns a `float` of the average number of yards gained per rushing attempt and/or reception.

yards_recovered_from_fumble

Returns an `int` of the number of yards the player gained after recovering a fumble.

yards_returned_from_interceptions

Returns an `int` of the number of yards the player returned after intercepting a pass.

yards_returned_per_interception

Returns a `float` of the average number of yards the player returns after intercepting a pass.

Rankings

The Rankings module include the `Rankings` class which can be used to easily query the NCAA Men's Division-I Football rankings published by the Associated Press on a week-by-week basis. Different formats can be referenced, ranging from a lightweight dictionary of the most recent rankings containing only the team abbreviation and rank, to a much larger dictionary of all rankings for an entire season with results including full team name and abbreviation, current rank, week number, previous rank, and movement.

```
from sportsreference.ncaaf.rankings import Rankings

rankings = Rankings()
# Prints a dictionary of just the team abbreviation and rank for the current
# week.
print(rankings.current)
# Prints more detailed information including previous rank, full name, and
# movement for all teams for the current week.
print(rankings.current_extended)
# Prints detailed information for all teams for all weeks where rankings
# have been published for the requested season.
print(rankings.complete)
```

class `sportsreference.ncaaf.rankings.Rankings` (*year=None*)

Bases: `object`

Get all Associated Press (AP) rankings on a week-by-week basis.

Grab a list of the rankings published by the Associated Press to easily query the hierarchy of teams each week. The results expose the current and previous rankings as well as the movement for each team in the list.

Parameters *year* (*string (optional)*) – A string of the requested year to pull rankings from. Defaults to the most recent season.

complete

Returns a dictionary where each key is a week number as an `int` and each value is a list of dictionaries containing the AP rankings for each week. Within each list is a dictionary of team information such as name, abbreviation, rank, and more. Note that the list might not necessarily be in the same order as the rankings.

The overall dictionary has the following structure:

```
{
    week number, ie 16 (int): [
        {
            'abbreviation': Team's abbreviation, such as 'PURDUE'
                           (str),
            'name': Team's full name, such as 'Purdue' (str),
            'rank': Team's rank for the current week (int),
            'week': Week number for the results, such as 16 (int),
            'date': Date the rankings were released, such as
                   '2017-12-03'. Can also be 'Final' for the final
                   rankings or 'Preseason' for preseason rankings
                   (str),
            'previous': The team's previous rank, if applicable
                       (str),
            'change': The amount the team moved up or down the
                      rankings. Moves up the ladder have a positive
                      number while drops yield a negative number
                      and teams that didn't move have 0 (int)
```

(continues on next page)

(continued from previous page)

```

        },
        ...
    ],
    ...
}

```

current

Returns a dictionary of the most recent rankings from the Associated Press where each key is a string of the team's abbreviation and each value is an int of the team's rank for the current week.

current_extended

Returns a list of dictionaries of the most recent AP rankings. The list is ordered in terms of the ranking so the #1 team will be in the first element and the #25 team will be the last element. Each dictionary has the following structure:

```

{
    'abbreviation': Team's abbreviation, such as 'PURDUE' (str),
    'name': Team's full name, such as 'Purdue' (str),
    'rank': Team's rank for the current week (int),
    'week': Week number for the results, such as 19 (int),
    'date': Date the rankings were released, such as '2017-03-01'.
            Can also be 'Final' for the final rankings or
            'Preseason' for preseason rankings (str),
    'previous': The team's previous rank, if applicable (str),
    'change': The amount the team moved up or down the rankings.
               Moves up the ladder have a positive number while
               drops yield a negative number and teams that didn't
               move have 0 (int)
}

```

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the Player class which has detailed information ranging from career touchdowns to single-season stats and player height, weight, and nationality. The following is an example on collecting career information for David Blough.

```

from sportsreference.ncaaf.roster import Player

blough = Player('david-blough-1')
print(blough.name)    # Prints 'David Blough'
print(blough.passing_yards)  # Prints Blough's career passing yards
# Prints a Pandas DataFrame of all relevant stats per season for Blough
print(blough.dataframe)

```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific team, call the class instance with the season string. All future property requests will return the season-specific stats.

```

from sportsreference.ncaaf.roster import Player

blough = Player('david-blough-1')  # Currently pulling career stats
print(blough.passing_yards)  # Prints Blough's CAREER passing yards total
# Prints Blough's passing yards total only for the 2017 season
print(blough('2017').passing_yards)
# Prints Blough's passing touchdowns for the 2017 season only
print(blough.passing_touchdowns)

```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.ncaaf.roster import Player

blough = Player('david-blough-1') # Currently pulling career stats
# Prints Blough's passing yards total only for the 2017 season
print(blough('2017').passing_yards)
print(blough('Career').passing_yards) # Prints Blough's career passing yards
```

In addition, the Roster module also contains the Roster class which can be used to pull all players on a team's roster during a given season and creates instances of the Player class for each team member and adds them to a list to be easily queried.

```
from sportsreference.ncaaf.roster import Roster

boilermakers = Roster('PURDUE')
for player in boilermakers.players:
    # Prints the name of all players who played for the Purdue Boilermakers
    # in the most recent season.
    print(player.name)
```

class sportsreference.ncaaf.roster.**Player** (*player_id*)
Bases: *sportsreference.ncaaf.player.AbstractPlayer*

Get player information and stats for all seasons.

Given a player ID, such as 'david-blough-1' for David Blough, capture all relevant stats and information like name, team, height/weight, career starts, single season passing yards, sacks, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on sports-reference.com.

Parameters *player_id* (*string*) – A player's ID according to sports-reference.com, such as 'david-blough-1' for David Blough. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'first-last-n' where 'first' is the player's first name in lowercase, 'last' is the player's last name in lowercase, and 'n' is a number starting at '1' for the first time that player ID has been used and increments by 1 for every successive player.

adjusted_yards_per_attempt

Returns a *float* of the adjusted number of yards gained per passing attempt, equal to $(yards + 20 * pass_touchdowns - 45 * interceptions) / pass_attempts$.

assists_on_tackles

Returns an *int* of the number of assists the player made on tackles.

attempted_passes

Returns an *int* of the number of passes the player attempted.

completed_passes

Returns an *int* of the number of completed passes the player threw.

dataframe

Returns a *pandas DataFrame* containing all other relevant class properties and values where each index is a different season plus the career stats.

extra_points_made

Returns an *int* of the number of extra points the player made.

field_goals_made

Returns an `int` of the total number of field goals the player made from any distance.

fumbles_forced

Returns an `int` of the number of times the player forced a fumble.

fumbles_recovered

Returns an `int` of the number of fumbles the player has recovered.

fumbles_recovered_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after recovering a fumble.

games

Returns an `int` of the number of games the player participated in.

height

Returns a `string` of the player's height in the format "feet-inches".

interceptions

Returns an `int` of the number of times the player intercepted a pass.

interceptions_returned_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after intercepting a pass. Commonly referred to as a 'Pick-6'.

interceptions_thrown

Returns an `int` of the number of interceptions the player has thrown.

kickoff_return_touchdowns

Returns an `int` of the number of kickoffs the player returned for a touchdown.

other_touchdowns

Returns an `int` of the total number of all other types of touchdowns the player has scored.

pass_attempts

Returns an `int` of the number of passes the player attempted.

passes_defended

Returns an `int` of the number of passes the player has defended as a defensive player.

passing_completion

Returns a `float` of the percentage of passes that were caught by a receiver. Percentage ranges from 0-100.

passing_touchdowns

Returns an `int` of the number of touchdowns passes the player has thrown.

passing_yards

Returns an `int` of the total number of yards the player gained from passing the ball.

plays_from_scrimmage

Returns an `int` of the combined number of rushing attempts and receptions the player had.

points

Returns an `int` of the number of points the player has scored.

position

Returns a `string` of the player's primary position.

punt_return_touchdowns

Returns an `int` of the number of punts the player returned for a touchdown.

quarterback_rating

Returns a `float` of the player's quarterback rating.

receiving_touchdowns

Returns an `int` of the number of touchdowns the player scored after receiving a pass.

receiving_yards

Returns an `int` of the number of receiving yards the player gained.

receiving_yards_per_reception

Returns a `float` of the average number of yards the player gained per reception.

receptions

Returns an `int` of the number of receptions the player made.

rush_attempts

Returns an `int` of the number of rushing plays the player attempted.

rush_touchdowns

Returns an `int` of the number of rushing touchdowns the player scored.

rush_yards

Returns an `int` of the number of rushing yards the player gained.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing attempt.

rushing_and_receiving_touchdowns

Returns an `int` of the combined number of rushing and receiving touchdowns the player scored.

sacks

Returns a `float` of the number of times the player sacked a quarterback.

safeties

Returns an `int` of the number of safeties the player has scored.

season

Returns a `string` of the season in the format 'YYYY', such as '2017'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

solo_tackles

Returns an `int` of the number of tackles the player made by himself.

tackles_for_loss

Returns a `float` of the number of tackles for a loss the player made.

team_abbreviation

Returns a `string` of the team's abbreviation, such as 'PURDUE' for the Purdue Boilermakers.

total_tackles

Returns an `int` of the number of tackles the player made.

total_touchdowns

Returns an `int` of the total number of touchdowns the player has scored.

two_point_conversions

Returns an `int` of the number of two point conversions the player has scored.

weight

Returns an `int` of the player's weight in pounds.

yards_from_scrimmage

Returns an `int` of the total number of yards gained from scrimmage for both rushing and receiving.

yards_from_scrimmage_per_play

Returns a `float` of the average number of yards gained per rushing attempt and/or reception.

yards_recovered_from_fumble

Returns an `int` of the number of yards the player gained after recovering a fumble.

yards_returned_from_interceptions

Returns an `int` of the number of yards the player returned after intercepting a pass.

yards_returned_per_interception

Returns a `float` of the average number of yards the player returns after intercepting a pass.

year

Returns a `string` of the player's class designation, such as 'FR' for freshmen.

class sportsreference.ncaaf.roster.Roster (team, year=None, slim=False)

Bases: object

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the Player class for each player, containing a detailed list of the player's statistics and information.

Parameters

- **team** (*string*) – The team's abbreviation, such as 'PURDUE' for the Purdue Boilermakers.
- **year** (*string (optional)*) – The 4-digit year to pull the roster from, such as '2017'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to True to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to False.

players

Returns a `list` of player instances for each player on the requested team's roster if the `slim` property is False when calling the Roster class. If the `slim` property is True, returns a `dictionary` where each key is a `string` of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The Schedule module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the Boxscore class which has much more detailed information on the game metrics.

```
from sportsreference.ncaaf.schedule import Schedule

purdue_schedule = Schedule('PURDUE')
for game in purdue_schedule:
    print(game.date)    # Prints the date the game was played
    print(game.result)  # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class sportsreference.ncaaf.schedule.Game (game_data)

Bases: object

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters `game_data` (*string*) – The row containing the specified game information.

boxscore

Returns an instance of the Boxscore class containing more detailed stats on the game.

boxscore_index

Returns a *string* of the URI for a boxscore which can be used to access or index a game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the boxscore string.

dataframe_extended

Returns a pandas DataFrame representing the Boxscore class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter ‘dataframe’ property. The index for the DataFrame is the boxscore string.

date

Returns a *string* of the date the game was played, such as ‘Sep 2, 2017’.

datetime

Returns a datetime object of the month, day, year, and time the game was played. If the game doesn’t include a time, the default value of ‘00:00’ will be used.

day_of_week

Returns a *string* of the 3-letter abbreviation of the day of the week the game was played on, such as ‘Sat’ for Saturday.

game

Returns an *int* to indicate which game in the season was requested. The first game of the season returns 1.

location

Returns a *string* constant to indicate whether the game was played at home, away, or in a neutral location.

losses

Returns an *int* of the number of games the team has lost so far in the season at the conclusion of the requested game.

opponent_abbr

Returns a *string* of the opponent’s abbreviation, such as ‘PURDUE’ for the Purdue Boilermakers.

opponent_conference

Returns a *string* of the conference the team participates in, such as ‘Big Ten’ for the Big Ten Conference. If a team does not compete in Division-I, a string constant for the non-major school will be returned.

opponent_name

Returns a *string* of the opponent’s name, such as ‘Purdue Boilermakers’ for the Purdue Boilermakers.

opponent_rank

Returns an *int* of the opponent’s rank at the time the game was played.

points_against

Returns an *int* of the number of points the team allowed during the game.

points_for

Returns an *int* of the number of points the team scored during the game.

rank

Returns an *int* of the team’s rank at the time the game was played.

result

Returns a `string` constant to indicate whether the team won or lost the game.

streak

Returns a `string` of the team's winning streak at the conclusion of the requested game. Streaks are listed in the format '[W/L] #' (ie. 'W 3' for a 3-game winning streak and 'L 2' for a 2-game losing streak).

time

00 PM'.

Type Returns a `string` of the time the game started, such as '12

wins

Returns an `int` of the number of games the team has won so far in the season at the conclusion of the requested game.

class `sportsreference.ncaaf.schedule.Schedule` (*abbreviation*, *year=None*)

Bases: `object`

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'MICHIGAN' for the Michigan Wolverines.
- **year** (*string* (*optional*)) – The requested year to pull stats from.

dataframe

Returns a `pandas DataFrame` where each row is a representation of the `Game` class. Rows are indexed by the `boxscore` string.

dataframe_extended

Returns a `pandas DataFrame` where each row is a representation of the `Boxscore` class for every game in the schedule. Rows are indexed by the `boxscore` string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The `Teams` module exposes information for all NCAAF teams including the team name and abbreviation, the number of games they won during the season, the total number of pass yards, and much more.

```
from sportsreference.ncaaf.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    print(team.pass_yards) # Prints the team's total passing yards
```

Each `Team` instance contains a link to the `Schedule` class which enables easy iteration over all games for a particular team. A `Pandas DataFrame` can also be queried to easily grab all stats for all games.

```
from sportsreference.ncaaf.teams import Teams

teams = Teams()
for team in teams:
    schedule = team.schedule # Returns a Schedule instance for each team
```

(continues on next page)

(continued from previous page)

```
# Returns a Pandas DataFrame of all metrics for all game Boxscores for
# a season.
df = team.schedule.dataframe_extended
```

Lastly, each Team instance also contains a link to the Roster class which enables players from the team to be easily queried. Each Roster instance contains detailed stats and information for each player on the team.

```
from sportsreference.ncaaf.teams import Teams

for team in Teams():
    roster = team.roster # Gets each team's roster
    for player in roster.players:
        print(player.name) # Prints each players name on the roster
```

class sportsreference.ncaaf.teams.Team(team_data, team_conference=None, year=None)
Bases: object

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as full and short names, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **team_conference** (*string (optional)*) – A string of the team's conference abbreviation, such as 'big-12'.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a *string* of the team's short name, such as 'PURDUE' for the Purdue Boilermakers.

conference

Returns a *string* of the team's conference abbreviation, such as 'big-12' for the Big 12 Conference.

conference_losses

Returns an *int* of the total number of conference games the team lost during the season.

conference_win_percentage

Returns a *float* of the percentage of conference wins divided by the number of conference games played during the season. Percentage ranges from 0-1.

conference_wins

Returns an *int* of the total number of conference games the team won during the season.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'PURDUE'.

first_downs

Returns a *float* of the total number of first downs achieved per game.

first_downs_from_penalties

Returns a *float* of the average number of first downs from an opponent's penalties per game.

fumbles_lost

Returns a *float* of the average number of fumbles per game.

games

Returns an `int` of the total number of games the team has played during the season.

interceptions

Returns a `float` of the average number of interceptions thrown per game.

losses

Returns an `int` of the total number of games the team lost during the season.

name

Returns a `string` of the team's full name, such as 'Purdue Boilermakers'.

opponents_first_downs

Returns a `float` of the opponents' total number of first downs achieved per game.

opponents_first_downs_from_penalties

Returns a `float` of the opponents' average number of first downs from an opponent's penalties per game.

opponents_fumbles_lost

Returns a `float` of the opponents' average number of fumbles per game.

opponents_interceptions

Returns a `float` of the opponents' average number of interceptions thrown per game.

opponents_pass_attempts

Returns a `float` of the opponents' average number of passes that are attempted per game.

opponents_pass_completion_percentage

Returns a `float` of the opponents' percentage of completed passes per game. Percentage ranges from 0-100.

opponents_pass_completions

Returns a `float` of the opponents' average number of completed passes per game.

opponents_pass_first_downs

Returns a `float` of the opponents' average number of first downs from passing plays per game.

opponents_pass_touchdowns

Returns a `float` of the opponents' average number of passing touchdowns scored per game.

opponents_pass_yards

Returns a `float` of the opponents' average number of yards gained from passing per game.

opponents_penalties

Returns a `float` of the opponents' average number of penalties conceded per game.

opponents_plays

Returns a `float` of the opponents' average number of offensive plays per game.

opponents_rush_attempts

Returns a `float` of the opponents' average number of rushing plays per game.

opponents_rush_first_downs

Returns a `float` of the opponents' average number of first downs from rushing plays per game.

opponents_rush_touchdowns

Returns a `float` of the opponents' average number of rushing touchdowns scored per game.

opponents_rush_yards

Returns a `float` of the opponents' average number of yards gained from rushing per game.

opponents_rush_yards_per_attempt

Returns a `float` of the opponents' average number of yards gained per rushing attempt per game.

opponents_turnovers

Returns a `float` of the opponents' average number of turnovers per game.

opponents_yards

Returns a `float` of the opponents' average number of yards gained per game.

opponents_yards_from_penalties

Returns a `float` of the opponents' average number of yards gained from an opponent's penalties per game.

opponents_yards_per_play

Returns a `float` of the opponents' average number of yards gained per play.

pass_attempts

Returns a `float` of the average number of passes that are attempted per game.

pass_completion_percentage

Returns a `float` of the percentage of completed passes per game. Percentage ranges from 0-100.

pass_completions

Returns a `float` of the average number of completed passes per game.

pass_first_downs

Returns a `float` of the average number of first downs from passing plays per game.

pass_touchdowns

Returns a `float` of the average number of passing touchdowns scored per game.

pass_yards

Returns a `float` of the average number of yards gained from passing per game.

penalties

Returns a `float` of the average number of penalties conceded per game.

plays

Returns a `float` of the average number of offensive plays per game.

points_against_per_game

Returns a `float` of the average number of points conceded per game.

points_per_game

Returns a `float` of the average number of points scored by the team per game.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

rush_attempts

Returns a `float` of the average number of rushing plays per game.

rush_first_downs

Returns a `float` of the average number of first downs from rushing plays per game.

rush_touchdowns

Returns a `float` of the average number of rushing touchdowns scored per game.

rush_yards

Returns a `float` of the average number of yards gained from rushing per game.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing attempt per game.

schedule

Returns an instance of the Schedule class containing the team's complete schedule for the season.

simple_rating_system

Returns a `float` of the team's relative strength based on the average margin of victory and the strength of schedule. An average team is denoted with 0.0 while a negative score indicates a comparatively weak team.

strength_of_schedule

Returns a `float` of the team's strength of schedule based on the number of points above or below average. An average difficulty schedule is denoted with 0.0 while a negative score indicates a comparatively easy schedule.

turnovers

Returns a `float` of the average number of turnovers per game.

win_percentage

Returns a `float` of the percentage of wins divided by the number of games played during the season. Percentage ranges from 0-1.

wins

Returns an `int` of the total number of games the team won during the season.

yards

Returns a `float` of the average number of yards gained per game.

yards_from_penalties

Returns a `float` of the average number of yards gained from an opponent's penalties per game.

yards_per_play

Returns a `float` of the average number of yards gained per play.

class sportsreference.ncaaf.teams.**Teams** (*year=None*)

Bases: `object`

A list of all NCAA Men's Football teams and their stats in a given year.

Finds and retrieves a list of all NCAA Men's Football teams from www.sports-reference.com and creates a `Team` instance for every team that participated in the league in a given year. The `Team` class comprises a list of all major stats and a few identifiers for the requested season.

Parameters *year* (*string optional*) – The requested year to pull stats from.

dataframes

Returns a pandas `DataFrame` where each row is a representation of the `Team` class. Rows are indexed by the team abbreviation.

1.6.1.5 NFL Package

The NFL package offers multiple modules which can be used to retrieve information and statistics for the National Football League, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of points scored to the number of passing yards, to the number of yards lost from sacks and much more. The Boxscore can be easily queried by passing a boxscore's URI on sports-reference.com which can be retrieved from the `Schedule` class (see `Schedule` module below for more information on retrieving game-specific information).

```
from sportsreference.nfl.boxscore import Boxscore

game_data = Boxscore('201802040nwe')
print(game_data.home_points)  # Prints 33
print(game_data.away_points)  # Prints 41
df = game_data.dataframe  # Returns a Pandas DataFrame of game metrics
```

The Boxscore module also contains a Boxscores class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.

```
from sportsreference.nfl.boxscore import Boxscores

games_today = Boxscores(1, 2017)
# Prints a dictionary of all matchups for week 1 of 2017
print(games_today.games)
```

The Boxscores class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```
from sportsreference.nfl.boxscore import Boxscores

# Pulls all games from weeks 7 and 8 in 2017
games = Boxscores(7, 2017, 8)
# Prints a dictionary of all games from weeks 7 and 8 in 2017
print(games.games)
```

class sportsreference.nfl.boxscore.Boxscore(*uri*)
Bases: object

Detailed information about the final statistics for a game.

Stores all relevant information for a game such as the date, time, location, result, and more advanced metrics such as the number of yards from sacks, a team's passing completion, rushing touchdowns and much more.

Parameters *uri* (*string*) – The relative link to the boxscore HTML page, such as '201802040nwe'.

attendance

Returns an *int* of the game's listed attendance.

away_abbreviation

Returns a *string* of the away team's abbreviation, such as 'NWE'.

away_first_downs

Returns an *int* of the number of first downs the away team gained.

away_fourth_down_attempts

Returns an *int* of the number of fourth down plays the away team attempted to convert.

away_fourth_down_conversions

Returns an *int* of the number of fourth down plays the away team successfully converted.

away_fumbles

Returns an *int* of the number of times the away team fumbled the ball.

away_fumbles_lost

Returns an *int* of the number of times the away team turned the ball over as the result of a fumble.

away_interceptions

Returns an `int` of the number of interceptions the away team threw.

away_net_pass_yards

Returns an `int` of the net pass yards gained by the away team.

away_pass_attempts

Returns an `int` of the number of passes that were thrown by the away team.

away_pass_completions

Returns an `int` of the number of completed passes the away team made.

away_pass_touchdowns

Returns an `int` of the number of passing touchdowns the away team scored.

away_pass_yards

Returns an `int` of the number of passing yards the away team gained.

away_penalties

Returns an `int` of the number of penalties called on the away team.

away_players

Returns a list of `BoxscorePlayer` class instances for each player on the away team.

away_points

Returns an `int` of the number of points the away team scored.

away_rush_attempts

Returns an `int` of the number of rushing plays the away team made.

away_rush_touchdowns

Returns an `int` of the number of rushing touchdowns the away team scored.

away_rush_yards

Returns an `int` of the number of rushing yards the away team gained.

away_third_down_attempts

Returns an `int` of the number of third down plays the away team attempted to convert.

away_third_down_conversions

Returns an `int` of the number of third down plays the away team successfully converted.

away_time_of_possession

Returns a string of the amount of time the home team had possession of the football in the format 'MM:SS'.

away_times_sacked

Returns an `int` of the number of times the away team was sacked.

away_total_yards

Returns an `int` of the total number of yards the away team gained.

away_turnovers

Returns an `int` of the number of times the away team turned the ball over.

away_yards_from_penalties

Returns an `int` of the number of yards gifted as a result of penalties called on the away team.

away_yards_lost_from_sacks

Returns an `int` of the number of yards the away team lost as the result of a sack.

dataframe

Returns a pandas `DataFrame` containing all other class properties and values. The index for the `DataFrame` is the string URI that is used to instantiate the class, such as '201802040nwe'.

date

Returns a *string* of the date the game took place.

duration

MM'.

Type Returns a *string* of the game's duration in the format 'H

home_abbreviation

Returns a *string* of the home team's abbreviation, such as 'KAN'.

home_first_downs

Returns an *int* of the number of first downs the home team gained.

home_fourth_down_attempts

Returns an *int* of the number of fourth down plays the home team attempted to convert.

home_fourth_down_conversions

Returns an *int* of the number of fourth down plays the home team successfully converted.

home_fumbles

Returns an *int* of the number of times the home team fumbled the ball.

home_fumbles_lost

Returns an *int* of the number of times the home team turned the ball over as the result of a fumble.

home_interceptions

Returns an *int* of the number of interceptions the home team threw.

home_net_pass_yards

Returns an *int* of the net pass yards gained by the home team.

home_pass_attempts

Returns an *int* of the number of passes that were thrown by the home team.

home_pass_completions

Returns an *int* of the number of completed passes the home team made.

home_pass_touchdowns

Returns an *int* of the number of passing touchdowns the home team scored.

home_pass_yards

Returns an *int* of the number of passing yards the home team gained.

home_penalties

Returns an *int* of the number of penalties called on the home team.

home_players

Returns a *list* of `BoxscorePlayer` class instances for each player on the home team.

home_points

Returns an *int* of the number of points the home team scored.

home_rush_attempts

Returns an *int* of the number of rushing plays the home team made.

home_rush_touchdowns

Returns an *int* of the number of rushing touchdowns the home team scored.

home_rush_yards

Returns an *int* of the number of rushing yards the home team gained.

home_third_down_attempts

Returns an *int* of the number of third down plays the home team attempted to convert.

home_third_down_conversions

Returns an `int` of the number of third down plays the home team successfully converted.

home_time_of_possession

Returns a `string` of the amount of time the home team had possession of the football in the format 'MM:SS'.

home_times_sacked

Returns an `int` of the number of times the home team was sacked.

home_total_yards

Returns an `int` of the total number of yards the home team gained.

home_turnovers

Returns an `int` of the number of times the home team turned the ball over.

home_yards_from_penalties

Returns an `int` of the number of yards gifted as a result of penalties called on the home team.

home_yards_lost_from_sacks

Returns an `int` of the number of yards the home team lost as the result of a sack.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'KAN' for the Kansas City Chiefs.

losing_name

Returns a `string` of the losing team's name, such as 'Kansas City Chiefs'.

stadium

Returns a `string` of the name of the stadium where the game was played.

time

Returns a `string` of the time the game started.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'NWE' for the New England Patriots.

winning_name

Returns a `string` of the winning team's name, such as 'New England Patriots'.

class sportsreference.nfl.boxscore.BoxscorePlayer(*player_id*, *player_name*,
player_data)

Bases: *sportsreference.nfl.player.AbstractPlayer*

Get player stats for an individual game.

Given a player ID, such as 'BreeDr01' for Drew Brees, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id** (*string*) – A player's ID according to pro-football-reference.com, such as 'BreeDr00' for Drew Brees. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.htm' in the URL. In general,

the ID is in the format 'LlIlFfNN' where 'Llll' are the first 4 letters in the player's last name with the first letter capitalized, 'Ff' are the first 2 letters in the player's first name where the first letter is capitalized, and 'NN' is a number starting at '00' for the first time that player ID has been used and increments by 1 for every successive player.

- **player_name** (*string*) – A string representing the player's first and last name, such as 'David Blough'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

average_kickoff_return_yards

Returns a `float` of the average number of yards the player returned a kickoff for.

combined_tackles

Returns an `int` of the number of solo and assisted tackles the player made.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values for the specified game.

fumbles_lost

Returns an `int` of the number of times the player fumbled the ball and the opponent recovered the ball.

quarterback_hits

Returns an `int` of the number of times the player hit the quarterback.

solo_tackles

Returns an `int` of the number of solo tackles the player made during the game.

tackles_for_loss

Returns an `int` of the number of times the player tackles an opponent for a loss on the play.

yards_lost_from_sacks

Returns an `int` of the total number of yards the player lost after being sacked by the opponent.

class `sportsreference.nfl.boxscore.Boxscores` (*week, year, end_week=None*)

Bases: `object`

Search for NFL games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **week** (*int*) – The week number to pull games from.
- **year** (*int*) – The 4-digit year to pull games from.
- **end_week** (*int (optional)*) – Optionally specify an end week to iterate until. All boxscores starting from the week specified in the 'week' parameter up to and including the boxscores specified in the 'end_week' parameter will be pulled. If left empty, or if 'end_week' is prior to 'week', only the games from the day specified in the 'date' parameter will be saved.

games

Returns a `dictionary` object representing all of the games played on the requested day. Dictionary is in the following format:

```
{'week' : [ # 'week' is the string week in format 'W-YYYY'
    {
        'home_name': Name of the home team, such as 'Kansas City
            Chiefs' (`str`),
        'home_abbr': Abbreviation for the home team, such as 'KAN'
            (`str`),
        'away_name': Name of the away team, such as 'Houston
            Texans' (`str`),
        'away_abbr': Abbreviation for the away team, such as 'HTX'
            (`str`),
        'boxscore': String representing the boxscore URI, such as
            'SLN/SLN201807280' (`str`),
        'winning_name': Full name of the winning team, such as
            'Kansas City Chiefs' (`str`),
        'winning_abbr': Abbreviation for the winning team, such as
            'KAN' (`str`),
        'losing_name': Full name of the losing team, such as
            'Houston Texans' (`str`),
        'losing_abbr': Abbreviation for the losing team, such as
            'HTX' (`str`),
        'home_score': Integer score for the home team (`int`),
        'away_score': Integer score for the away team (`int`)
    },
    { ... },
    ...
]
}
```

If no games were played on ‘week’, the list for [‘week’] will be empty.

sportsreference.nfl.boxscore.nfl_int_property_sub_index (*func*)

Player

The Player module contains an abstract base class that can be inherited by both the BoxscorePlayer and Player classes in the Boxscore and Roster modules, respectively. All of the properties that appear in the AbstractPlayer class can be read from either of the two child classes mentioned above.

class sportsreference.nfl.player.**AbstractPlayer** (*player_id, player_name, player_data*)

Bases: object

Get player information and stats for all seasons.

Given a player ID, such as ‘BreeDr00’ for Drew Brees, capture all relevant stats and information like name, team, height/weight, career starts, single season passing yards, sacks, and much more.

By default, the class instance will return the player’s career stats, but single-season stats can be found by calling the instance with the requested season as denoted on pro-football-reference.com.

Parameters

- **player_id** (*string*) – A player’s ID according to pro-football-reference.com, such as ‘BreeDr00’ for Drew Brees. The player ID can be found by navigating to the player’s stats page and getting the string between the final slash and the ‘.htm’ in the URL. In general, the ID is in the format ‘LIIIFNN’ where ‘LIII’ are the first 4 letters in the player’s last name with the first letter capitalized, ‘Ff’ are the first 2 letters in the player’s first name where the first letter is capitalized, and ‘NN’ is a number starting at ‘00’ for the first time that player ID has been used and increments by 1 for every successive player.

- **player_name** (*string*) – A string representing the player’s first and last name, such as ‘Drew Brees’.
- **player_data** (*string*) – A string representation of the player’s HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

assists_on_tackles

Returns an `int` of the number of assist the player made on tackles.

attempted_passes

Returns an `int` of the number of passes the player attempted.

completed_passes

Returns an `int` of the number of completed passes the player threw.

extra_points_attempted

Returns an `int` of the number of extra points the player attempted.

extra_points_made

Returns an `int` of the number of extra points the player made.

field_goals_attempted

Returns an `int` of the total number of field goals the player attempted from any distance.

field_goals_made

Returns an `int` of the total number of field goals the player made from any distance.

fumbles

Returns an `int` of the number of times the player fumbled the ball.

fumbles_forced

Returns an `int` of the number of times the player forced a fumble.

fumbles_recovered

Returns an `int` of the number of fumbles the player has recovered.

fumbles_recovered_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after recovering a fumble.

interceptions

Returns an `int` of the number of times the player intercepted a pass.

interceptions_returned_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after intercepting a pass. Commonly referred to as a ‘Pick-6’.

interceptions_thrown

Returns an `int` of the number of interceptions the player has thrown.

kickoff_return_touchdown

Returns an `int` of the number of kickoffs the player returned for a touchdown.

kickoff_return_yards

Returns an `int` of the amount of yards the player gained while returning a kickoff.

kickoff_returns

Returns an `int` of the number of kickoffs the player returned.

longest_interception_return

Returns an `int` of the most yards the player has returned after intercepting a pass.

longest_kickoff_return

Returns an `int` of the highest number of yards the player has gained while returning a kickoff.

longest_pass

Returns an `int` of the longest completed pass the player threw.

longest_punt

Returns an `int` of the longest punt the player has kicked.

longest_punt_return

Returns an `int` of the highest number of yards the player has gained while returning a punt.

longest_reception

Returns an `int` of the highest number of yards the player gained as a result of a single reception.

longest_rush

Returns an `int` of the highest number of yards the player gained during a single rushing attempt.

name

Returns a `string` of the player's name, such as 'Drew Brees'.

passes_defended

Returns an `int` of the number of passes the player has defended as a defensive player.

passing_touchdowns

Returns an `int` of the number of touchdowns passes the player has thrown.

passing_yards

Returns an `int` of the number of yards receivers have gained as a result of the player's passes.

player_id

Returns a `string` of the player's ID on pro-football-reference, such as 'BreeDr00' for Drew Brees.

punt_return_touchdown

Returns an `int` of the number of punts the player returned for a touchdown.

punt_return_yards

Returns an `int` of the amount of yards the player gained while returning a punt.

punt_returns

Returns an `int` of the number of times a player returned a punt.

punts

Returns an `int` of the number of times the player punted the ball.

quarterback_rating

Returns a `float` of the player's quarterback rating.

receiving_touchdowns

Returns an `int` of the number of touchdowns the player scored after receiving a pass.

receiving_yards

Returns an `int` of the number of receiving yards the player gained.

receptions

Returns an `int` of the number of receptions the player made.

rush_attempts

Returns an `int` of the number of rushing plays the player attempted.

rush_touchdowns

Returns an `int` of the number of rushing touchdowns the player scored.

rush_yards

Returns an `int` of the number of rushing yards the player gained.

sacks

Returns a float of the number of times the player sacked a quarterback.

times_pass_target

Returns an int of the number of times the player was the target of a pass.

times_sacked

Returns an int of the number of times the player was sacked as a quarterback.

total_punt_yards

Returns an int of the total number of yards the player has punted the ball.

yards_per_punt

Returns a float of the average distance the player punts the ball.

yards_per_punt_return

Returns a float of the average number of yards the player returned per punt.

yards_recovered_from_fumble

Returns an int of the number of yards the player gained after recovering a fumble.

yards_returned_from_interception

Returns an int of the number of yards the player returned after intercepting a pass.

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the `Player` class which has detailed information ranging from career touchdowns to single-season stats and player height, weight, and nationality. The following is an example on collecting career information for Drew Brees.

```
from sportsreference.nfl.roster import Player

brees = Player('BreeDr00')
print(brees.name) # Prints 'Drew Brees'
print(brees.passing_yards) # Prints Brees' career passing yards
# Prints a Pandas DataFrame of all relevant stats per season for Brees
print(brees.dataframe)
```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific team, call the class instance with the season string. All future property requests will return the season-specific stats.

```
from sportsreference.nfl.roster import Player

brees = Player('BreeDr00') # Currently pulling career stats
print(brees.passing_yards) # Prints Brees' CAREER passing yards total
# Prints Brees' passing yards total only for the 2017 season
print(brees('2017').passing_yards)
# Prints Brees' passing touchdowns for the 2017 season only
print(brees.passing_touchdowns)
```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.nfl.roster import Player

brees = Player('BreeDr00') # Currently pulling career stats
# Prints Brees' passing yards total only for the 2017 season
```

(continues on next page)

(continued from previous page)

```
print(brees('2017').passing_yards)
print(brees('Career').passing_yards)  # Prints Brees' career passing yards
```

In addition, the Roster module also contains the `Roster` class which can be used to pull all players on a team's roster during a given season and creates instances of the `Player` class for each team member and adds them to a list to be easily queried.

```
from sportsreference.nfl.roster import Roster

saints = Roster('NOR')
for player in saints.players:
    # Prints the name of all players who played for the New Orleans Saints
    # in the most recent season.
    print(player.name)
```

class `sportsreference.nfl.roster.Player` (*player_id*)
 Bases: `sportsreference.nfl.player.AbstractPlayer`

Get player information and stats for all seasons.

Given a player ID, such as 'BreeDr00' for Drew Brees, capture all relevant stats and information like name, team, height/weight, career starts, single season passing yards, sacks, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on pro-football-reference.com.

Parameters `player_id` (*string*) – A player's ID according to pro-football-reference.com, such as 'BreeDr00' for Drew Brees. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.htm' in the URL. In general, the ID is in the format 'LIIIFfNN' where 'LIII' are the first 4 letters in the player's last name with the first letter capitalized, 'Ff' are the first 2 letters in the player's first name where the first letter is capitalized, and 'NN' is a number starting at '00' for the first time that player ID has been used and increments by 1 for every successive player.

adjusted_net_yards_per_attempt_index

Returns an `int` comparing players by the net average adjusted yards gained per attempt where 100 denotes an average player in this category and higher numbers are better.

adjusted_net_yards_per_pass_attempt

Returns a `float` of the adjusted net yards gained per pass attempt, equal to $(\text{pass_yards} - \text{sack_yards} + (20 * \text{pass_touchdowns}) - (45 * \text{interceptions})) / (\text{pass_attempts} + \text{times_sacked})$.

adjusted_yards_per_attempt

Returns a `float` of the adjusted number of yards gained per passing attempt, equal to $(\text{yards} + 20 * \text{pass_touchdowns} - 45 * \text{interceptions}) / \text{pass_attempts}$.

adjusted_yards_per_attempt_index

Returns an `int` comparing players by the average adjusted yards gained per attempt where 100 denotes an average player in this category and higher numbers are better.

all_purpose_yards

Returns an `int` of the number of all-purpose yards the player has gained from receptions, rushes, and kickoff and punt returns.

approximate_value

Returns an `int` of the player's approximate value which is a singular number used to compare players across seasons and positions, but is only intended to be a rough estimate.

assists_on_tackles

Returns an `int` of the number of assist the player made on tackles.

attempted_passes

Returns an `int` of the number of passes the player attempted.

birth_date

Returns a `datetime` object of the day and year the player was born.

blocked_punts

Returns an `int` of the number of the player's punts that have been blocked.

catch_percentage

Returns a `float` of the percentage of passes the player caught while being the target of a pass. Percentage ranges from 0-100.

completed_passes

Returns an `int` of the number of completed passes the player threw.

completion_percentage_index

Returns an `int` comparing players by their passing completion percentage where 100 denotes an average player in this category and higher numbers are better.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values where each index is a different season plus the career stats.

espn_qbr

Returns a `float` of the player's Total Quarterback Rating according to ESPN.

extra_point_percentage

Returns a `float` of the percentage of extra points the player made. Percentage ranges from 0-100.

extra_points_attempted

Returns an `int` of the number of extra points the player attempted.

extra_points_made

Returns an `int` of the number of extra points the player made.

field_goal_percentage

Returns a `float` of the percentage of field goals the player made. Percentage ranges from 0-100.

field_goals_attempted

Returns an `int` of the total number of field goals the player attempted from any distance.

field_goals_made

Returns an `int` of the total number of field goals the player made from any distance.

fifty_plus_yard_field_goal_attempts

Returns an `int` of the number of field goals the player attempted from fifty or more yards out.

fifty_plus_yard_field_goals_made

Returns an `int` of the number of field goals the player made from fifty or more yards out.

fourth_quarter_comebacks

Returns an `int` of the number of times the player has lead a team to victory or a tie as a quarterback while the team trailed at the beginning of the fourth quarter by scoring at the end of a drive.

fourty_to_fourty_nine_yard_field_goal_attempts

Returns an `int` of the number of field goals the player attempted from fourty to fourty-nine yards out.

fourty_to_fourty_nine_yard_field_goals_made

Returns an `int` of the number of field goals the player made from fourty to fourty-nine yards out.

fumbles

Returns an `int` of the number of times the player fumbled the ball.

fumbles_forced

Returns an `int` of the number of times the player forced a fumble.

fumbles_recovered

Returns an `int` of the number of fumbles the player has recovered.

fumbles_recovered_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after recovering a fumble.

game_winning_drives

Returns an `int` of the number of times the player has lead a drive that resulted in a score in the fourth quarter while the team was trailing.

games

Returns an `int` of the number of games the player participated in.

games_started

Returns an `int` of the number of games the player started.

height

Returns a `string` of the player's height in the format "feet-inches".

interception_percentage

Returns a `float` of the percentage of passes the player throws that are intercepted. Percentage ranges from 0-100.

interception_percentage_index

Returns an `int` comparing players by the percentage of their passes that are intercepted where 100 denotes an average player in this category and higher numbers are better.

interceptions

Returns an `int` of the number of times the player intercepted a pass.

interceptions_returned_for_touchdown

Returns an `int` of the number of touchdowns the player has scored after intercepting a pass. Commonly referred to as a 'Pick-6'.

interceptions_thrown

Returns an `int` of the number of interceptions the player has thrown.

kickoff_return_touchdown

Returns an `int` of the number of kickoffs the player returned for a touchdown.

kickoff_return_yards

Returns an `int` of the amount of yards the player gained while returning a kickoff.

kickoff_returns

Returns an `int` of the number of kickoffs the player returned.

less_than_nineteen_yards_field_goal_attempts

Returns an `int` of the number of field goals the player attempted from nineteen or fewer yards out.

less_than_nineteen_yards_field_goals_made

Returns an `int` of the number of field goals the player made from nineteen or fewer yards out.

longest_field_goal_made

Returns an `int` of the longest field goal the player made.

longest_interception_return

Returns an `int` of the most yards the player has returned after intercepting a pass.

longest_kickoff_return

Returns an `int` of the highest number of yards the player has gained while returning a kickoff.

longest_pass

Returns an `int` of the longest completed pass the player threw.

longest_punt

Returns an `int` of the longest punt the player has kicked.

longest_punt_return

Returns an `int` of the highest number of yards the player has gained while returning a punt.

longest_reception

Returns an `int` of the highest number of yards the player gained as a result of a single reception.

longest_rush

Returns an `int` of the highest number of yards the player gained during a single rushing attempt.

net_yards_per_attempt_index

Returns an `int` comparing players by the net average yards gained per attempt where 100 denotes an average player in this category and higher numbers are better.

net_yards_per_pass_attempt

Returns a `float` of the net yards gained per pass attempt, equal to $(\text{pass_yards} - \text{sack_yards}) / (\text{pass_attempts} + \text{times_sacked})$.

passer_rating_index

Returns an `int` comparing players by their quarterback rating where 100 denotes an average player in this category and higher numbers are better.

passes_defended

Returns an `int` of the number of passes the player has defended as a defensive player.

passing_completion

Returns a `float` of the percentage of passes that were caught by a receiver. Percentage ranges from 0-100.

passing_touchdown_percentage

Returns a `float` of the percentage of total passes that are touchdowns. Percentage ranges from 0-100.

passing_touchdowns

Returns an `int` of the number of touchdowns passes the player has thrown.

passing_yards

Returns an `int` of the number of yards receivers have gained as a result of the player's passes.

passing_yards_per_attempt

Returns a `float` of the number of yards gained per passing attempt.

position

Returns a `string` of the player's primary position.

punt_return_touchdown

Returns an `int` of the number of punts the player returned for a touchdown.

punt_return_yards

Returns an `int` of the amount of yards the player gained while returning a punt.

punt_returns

Returns an `int` of the number of times a player returned a punt.

punts

Returns an `int` of the number of times the player punted the ball.

qb_record

Returns a `string` of the player's quarterback record as a starter in the format 'W-L-T'.

quarterback_rating

Returns a `float` of the player's quarterback rating.

receiving_touchdowns

Returns an `int` of the number of touchdowns the player scored after receiving a pass.

receiving_yards

Returns an `int` of the number of receiving yards the player gained.

receiving_yards_per_game

Returns a `float` of the average number of receiving yards the player gains per game.

receiving_yards_per_reception

Returns a `float` of the average number of yards the player gained per reception.

receptions

Returns an `int` of the number of receptions the player made.

receptions_per_game

Returns a `float` of the average number of receptions the player makes per game.

rush_attempts

Returns an `int` of the number of rushing plays the player attempted.

rush_attempts_per_game

Returns a `float` of the average number of rushing attempts the player made per game.

rush_touchdowns

Returns an `int` of the number of rushing touchdowns the player scored.

rush_yards

Returns an `int` of the number of rushing yards the player gained.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing attempt.

rush_yards_per_game

Returns a `float` of the average number of rushing yards gained per game.

rushing_and_receiving_touchdowns

Returns an `int` of the combined number of rushing and receiving touchdowns the player scored.

sack_percentage

Returns a `float` of the percentage of times sacked during a passing attempt, equal to `times_sacked / (pass_attempts + times_sacked)`. Percentage ranges from 0-100.

sack_percentage_index

Returns an `int` comparing players by the percentage of plays that end in the player being sacked where 100 denotes an average player in this category and higher numbers are better.

sacks

Returns a `float` of the number of times the player sacked a quarterback.

safeties

Returns an `int` of the number of safeties the player has scored.

season

Returns a `string` of the season in the format 'YYYY', such as '2017'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

tackles

Returns an `int` of the number of tackles the player made.

team_abbreviation

Returns a `string` of the team's abbreviation, such as 'NOR' for the New Orleans Saints.

thirty_to_thirty_nine_yard_field_goal_attempts

Returns an `int` of the number of field goals the player attempted from thirty to thirty-nine yards out.

thirty_to_thirty_nine_yard_field_goals_made

Returns an `int` of the number of field goals the player made from thirty to thirty-nine yards out.

times_pass_target

Returns an `int` of the number of times the player was the target of a pass.

times_sacked

Returns an `int` of the number of times the player was sacked as a quarterback.

total_punt_yards

Returns an `int` of the total number of yards the player has punted the ball.

touchdown_percentage_index

Returns an `int` comparing players by the percentage of their passes that result in a touchdown where 100 denotes an average player in this category and higher numbers are better.

touches

Returns an `int` of the combined number of rushing attempts and receptions the player had.

twenty_to_twenty_nine_yard_field_goal_attempts

Returns an `int` of the number of field goals the player attempted from twenty to twenty-nine yards out.

twenty_to_twenty_nine_yard_field_goals_made

Returns an `int` of the number of field goals the player made from twenty to twenty-nine yards out.

weight

Returns an `int` of the player's weight in pounds.

yards_from_scrimmage

Returns an `int` of the total number of yards gained from scrimmage for both rushing and receiving.

yards_lost_to_sacks

Returns an `int` of the number of yards lost as a result of sacks.

yards_per_attempt_index

Returns an `int` comparing players by the average number of yards gained per attempt where 100 denotes an average player in this category and higher numbers are better.

yards_per_completed_pass

Returns a `float` of the number of yards gained per completed pass.

yards_per_game_played

Returns a `float` of the number of passing yards gained per game.

yards_per_kickoff_return

Returns a `float` of the average number of yards the player returned per kickoff.

yards_per_punt_return

Returns a `float` of the average number of yards the player returned per punt.

yards_per_touch

Returns a `float` of the average number of yards gained per rushing attempt and/or reception.

yards_recovered_from_fumble

Returns an `int` of the number of yards the player gained after recovering a fumble.

yards_returned_from_interception

Returns an int of the number of yards the player returned after intercepting a pass.

class sportsreference.nfl.roster.**Roster** (*team*, *year=None*, *slim=False*)

Bases: object

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the Player class for each player, containing a detailed list of the player's statistics and information.

Parameters

- **team** (*string*) – The team's abbreviation, such as 'NOR' for the New Orleans Saints.
- **year** (*string (optional)*) – The 4-digit year to pull the roster from, such as '2017'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to True to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to False.

players

Returns a list of player instances for each player on the requested team's roster if the `slim` property is False when calling the Roster class. If the `slim` property is True, returns a dictionary where each key is a string of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The Schedule module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the `Boxscore` class which has much more detailed information on the game metrics.

```
from sportsreference.nfl.schedule import Schedule

houston_schedule = Schedule('HTX')
for game in houston_schedule:
    print(game.date)    # Prints the date the game was played
    print(game.result)  # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class sportsreference.nfl.schedule.**Game** (*game_data*, *game_type*, *year*)

Bases: object

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters

- **game_data** (*string*) – The row containing the specified game's information.
- **game_type** (*string*) – A constant to denote whether a game took place in the regular season or in the playoffs.
- **year** (*string*) – The year as a 4-digit string. Note that this is the year that the bulk of the season took place. For example the Super Bowl for the 2017 season took place in early

February 2018, but 2017 should be passed as that was the year the bulk of the season was played in.

boxscore

Returns an instance of the Boxscore class containing more detailed stats on the game.

boxscore_index

Returns a `string` of the URI for a boxscore which can be used to access or index a game.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the boxscore string.

dataframe_extended

Returns a pandas DataFrame representing the Boxscore class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property. The index for the DataFrame is the boxscore string.

date

Returns a `string` of the month and day the game was played, such as 'September 7'.

datetime

Returns a datetime object representing the date the game was played.

day

Returns a `string` of the day of the week the game was played as a 3-letter abbreviation, such as 'Sun' for Sunday.

extra_points_attempted

Returns an `int` of the number of times the team attempted to convert an extra point after scoring a touchdown.

extra_points_made

Returns an `int` of the number of extra points the team successfully converted after scoring a touchdown.

field_goals_attempted

Returns an `int` of the total number of times the team attempted a field goal.

field_goals_made

Returns an `int` of the total number of field goals the team scored.

fourth_down_attempts

Returns an `int` of the total number of fourth downs the team attempted to convert.

fourth_down_conversions

Returns an `int` of the number of fourth downs the team successfully converted.

interceptions

Returns an `int` of the number of interceptions the team threw.

location

Returns a `string` constant indicating whether the game was played at home, away, or a neutral site, such as the Super Bowl.

opponent_abbr

Returns a `string` of the opponent's 3-letter abbreviation, such as 'NWE' for the New England Patriots.

opponent_name

Returns a `string` of the opponent's full name, such as the 'New England Patriots'.

overtime

Returns a `boolean` value that evaluates to True if the game went to overtime and False if it ended in regulation.

pass_attempts

Returns an `int` of the number of passes the team attempted during the game.

pass_completion_rate

Returns a `float` of the percentage of passes that were completed by the team. Percentage ranges from 0-100.

pass_completions

Returns an `int` of the number of completed passed by the team.

pass_touchdowns

Returns an `int` of the number of touchdowns the team scored as a result of passing plays.

pass_yards

Returns an `int` of the number of yards the team gained as a result of passing plays.

pass_yards_per_attempt

Returns a `float` of the average number of yards gained per passing play.

points_allowed

Returns an `int` of the number of points allowed by the team.

points_scored

Returns an `int` of the number of points scored by the team.

punt_yards

Returns an `int` of the total number of yards the team punted the ball.

punts

Returns an `int` of the number of times the team punted the ball.

quarterback_rating

Returns a `float` of the quarterback's rating for the game.

result

Returns a `string` constant indicating whether the team won or lost the game. NFL games may end in a tie if the score is even at the end of OT. If a game has no result (canceled, yet to be played, etc.) return `None`.

rush_attempts

Returns an `int` of the total number of times the team attempted a rushing play.

rush_touchdowns

Returns an `int` of the number of touchdowns the team scored as a result of rushing plays.

rush_yards

Returns an `int` of the total number of yards the team gain as a result of rushing plays.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing play.

third_down_attempts

Returns an `int` of the total number of third downs the team attempted to convert.

third_down_conversions

Returns an `int` of the number of third downs the team successfully converted.

time_of_possession

Returns a `string` of the total time the team spent with the ball. Time is in the format 'MM:SS'.

times_sacked

Returns an `int` of the number of times the quarterback was sacked by the opponent.

type

Returns a `string` constant indicating whether the game is a regular season or playoff matchup.

week

Returns an `int` of the week number in the season, such as 1 for the first week of the regular season.

yards_lost_from_sacks

Returns an `int` of the total number of yards lost as a result of a sack.

class `sportsreference.nfl.schedule.Schedule` (*abbreviation*, *year=None*)

Bases: `object`

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'NWE' for the New England Patriots.
- **year** (*string* (*optional*)) – The requested year to pull stats from.

dataframe

Returns a `pandas DataFrame` where each row is a representation of the `Game` class. Rows are indexed by the `boxscore` string.

dataframe_extended

Returns a `pandas DataFrame` where each row is a representation of the `Boxscore` class for every game in the schedule. Rows are indexed by the `boxscore` string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The `Teams` module exposes information for all MLB teams including the team name and abbreviation, the number of games they won during the season, the average margin of victory, and much more.

```
from sportsreference.nfl.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    # Prints the team's average margin of victory
    print(team.margin_of_victory)
```

Each `Team` instance contains a link to the `Schedule` class which enables easy iteration over all games for a particular team. A `Pandas DataFrame` can also be queried to easily grab all stats for all games.

```
from sportsreference.nfl.teams import Teams

teams = Teams()
for team in teams:
    schedule = team.schedule    # Returns a Schedule instance for each team
    # Returns a Pandas DataFrame of all metrics for all game Boxscores for
    # a season.
    df = team.schedule.dataframe_extended
```

Lastly, each `Team` instance also contains a link to the `Roster` class which enables players from the team to be easily queried. Each `Roster` instance contains detailed stats and information for each player on the team.

```

from sportsreference.nfl.teams import Teams

for team in Teams():
    roster = team.roster # Gets each team's roster
    for player in roster.players:
        print(player.name) # Prints each players name on the roster

```

class sportsreference.nfl.teams.**Team**(*team_data*, *rank*, *year=None*)

Bases: object

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as rank, name, and abbreviation, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **rank** (*int*) – A team's position in the league based on the number of points they obtained during the season.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a *string* of team's abbreviation, such as 'KAN' for the Kansas City Chiefs.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'KAN'.

defensive_simple_rating_system

Returns a *float* of the team's defensive strength according to the simple rating system. An average team is denoted with 0.0 and a negative score is a comparatively weaker team.

first_downs

Returns an *int* of the total number of first downs the team achieved during the season.

first_downs_from_penalties

Returns an *int* of the total number of first downs conceded as a result of penalties called on the team.

fumbles

Returns an *int* of the total number of times the team fumbled the ball during the season.

games_played

Returns an *int* of the number of games played during the season.

interceptions

Returns an *int* of the total number of interceptions the team has thrown.

losses

Returns an *int* of the number of games the team lost during the season.

margin_of_victory

Returns a *float* of the average margin of victory per game.

name

Returns a *string* of the team's full name, such as 'Kansas City Chiefs'.

offensive_simple_rating_system

Returns a `float` of the team's offensive strength according to the simple rating system. An average team is denoted with 0.0 and a negative score is a comparatively weaker team.

pass_attempts

Returns an `int` of the total number of passes that were attempted.

pass_completions

Returns an `int` of the total number of passes that were completed.

pass_first_downs

Returns an `int` of the number of first downs the team gained from passing plays.

pass_net_yards_per_attempt

Returns a `float` of the net yards gained per passing play including sacks.

pass_touchdowns

Returns an `int` of the total number of touchdowns the team has scored from passing.

pass_yards

Returns an `int` of the total number of yards the team gained from passing.

penalties

Returns an `int` of the total number of penalties called on the team during the season.

percent_drives_with_points

Returns a `float` of the percentage of drives that result in points for the offense. Percentage ranges from 0-100.

percent_drives_with_turnovers

Returns a `float` of the percentage of drives that result in an offensive turnover. Percentage ranges from 0-100.

plays

Returns an `int` of the total number of offensive plays the team has made during the season.

points_against

Returns an `int` of the total number of points allowed during the season.

points_contributed_by_offense

Returns a `float` of the number of expected points contributed by the offense.

points_difference

Returns an `int` of the difference between the number of points scored and allowed during the season.

points_for

Returns an `int` of the total number of points scored during the season.

rank

Returns an `int` of the team's rank based on the number of points they scored during the season.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

rush_attempts

Returns an `int` of the total number of rushing plays that were attempted.

rush_first_downs

Returns an `int` of the total number of first downs gained from rushing plays.

rush_touchdowns

Returns an `int` of the total number of touchdowns from rushing plays.

rush_yards

Returns an `int` of the total number of yards that were gained from rushing plays.

rush_yards_per_attempt

Returns a `float` of the average number of yards gained per rushing play.

schedule

Returns an instance of the `Schedule` class containing the team's complete schedule for the season.

simple_rating_system

Returns a `float` of the team's relative strength based on average margin of victory plus strength of schedule. An average team is denoted with 0.0 and a negative score is a comparatively weaker team.

strength_of_schedule

Returns a `float` of the team's strength of schedule. An average difficulty schedule is denoted with a 0.0 and a negative number is comparatively easier than average.

turnovers

Returns an `int` of the total number of turnovers the team committed during the season.

win_percentage

Returns a `float` of the number of wins divided by the number of games played. Percentage ranges from 0-1.

wins

Returns an `int` of the number of games the team won during the season.

yards

Returns an `int` of the total number of yards the team has gained during the season.

yards_from_penalties

Returns an `int` of the total number of yards surrendered as a result of penalties called on the team.

yards_per_play

Returns a `float` of the average number of yards gained per play during the season.

class `sportsreference.nfl.teams.Teams` (*year=None*)

Bases: `object`

A list of all NFL teams and their stats in a given year.

Finds and retrieves a list of all NFL teams from www.pro-football-reference.com and creates a `Team` instance for every team that participated in the league in a given year. The `Team` class comprises a list of all major stats and a few identifiers for the requested season.

Parameters *year* (*string (optional)*) – The requested year to pull stats from.

dataframes

Returns a pandas `DataFrame` where each row is a representation of the `Team` class. Rows are indexed by the team abbreviation.

1.6.1.6 NHL Package

The NHL package offers multiple modules which can be used to retrieve information and statistics for the National Hockey League, such as team names, season stats, game schedules, and boxscore metrics.

Boxscore

The Boxscore module can be used to grab information from a specific game. Metrics range from number of goals scored to the number of penalty minutes, to the save percentage and much more. The Boxscore can be easily queried by

passing a boxscore's URI on sports-reference.com which can be retrieved from the `Schedule` class (see `Schedule` module below for more information on retrieving game-specific information).

```
from sportsreference.nhl.boxscore import Boxscore

game_data = Boxscore('201806070VEG')
print(game_data.home_goals)  # Prints 3
print(game_data.away_goals)  # Prints 4
df = game_data.dataframe  # Returns a Pandas DataFrame of game metrics
```

The `Boxscore` module also contains a `Boxscores` class which searches for all games played on a particular day and returns a dictionary of matchups between all teams on the requested day. The dictionary includes the names and abbreviations for each matchup as well as the boxscore link if applicable.

```
from datetime import datetime
from sportsreference.nhl.boxscore import Boxscores

games_today = Boxscores(datetime.today())
print(games_today.games)  # Prints a dictionary of all matchups for today
```

The `Boxscores` class also allows the ability to query over a range of dates using a second optional parameter during instantiation of the class. To query a range of dates, enter the start date as the first parameter and the inclusive end date as the second parameter.

```
from datetime import datetime
from sportsreference.nhl.boxscore import Boxscores

# Pulls all games between and including February 4, 2017 and February 5,
# 2017
games = Boxscores(datetime(2017, 2, 4), datetime(2017, 2, 5))
# Prints a dictionary of all results from February 4, 2017 and February 5,
# 2017
print(games.games)
```

class `sportsreference.nhl.boxscore.Boxscore` (*uri*)

Bases: `object`

Detailed information about the final statistics for a game.

Stores all relevant information for a game such as the date, time, location, result, and more advanced metrics such as the number of goals scored, the number of points for a player, the amount of power play assists and much more.

Parameters *uri* (*string*) – The relative link to the boxscore HTML page, such as '201806070VEG'.

arena

Returns a `string` of the name of the ballpark where the game was played.

attendance

Returns an `int` of the game's listed attendance.

away_assists

Returns an `int` of the number of assists the away team registered.

away_even_strength_assists

Returns an `int` of the number of assists the away team registered while at even strength.

away_even_strength_goals

Returns an `int` of the number of goals the away team scored at even strength.

away_game_winning_goals

Returns an `int` of the number of game winning goals the away team scored.

away_goals

Returns an `int` of the number of goals the away team scored.

away_penalties_in_minutes

Returns an `int` of the length of time the away team spent in the penalty box.

away_players

Returns a `list` of `BoxscorePlayer` class instances for each player on the away team.

away_points

Returns an `int` of the number of points the away team registered.

away_power_play_assists

Returns an `int` of the number of assists the away team registered while on a power play.

away_power_play_goals

Returns an `int` of the number of goals the away team scored while on a power play.

away_save_percentage

Returns a `float` of the percentage of shots the away team saved. Percentage ranges from 0-1.

away_saves

Returns an `int` of the number of saves the away team made.

away_shooting_percentage

Returns a `float` of the away team's shooting percentage. Percentage ranges from 0-100.

away_short_handed_assists

Returns an `int` of the number of assists the away team registered while short handed.

away_short_handed_goals

Returns an `int` of the number of goals the away team scored while short handed.

away_shots_on_goal

Returns an `int` of the number of shots on goal the away team registered.

away_shutout

Returns an `int` denoting whether or not the away team shutout the home team.

dataframe

Returns a `pandas DataFrame` containing all other class properties and values. The index for the `DataFrame` is the string URI that is used to instantiate the class, such as '201806070VEG'.

date

Returns a `string` of the date the game took place.

duration

MM'.

Type Returns a `string` of the game's duration in the format 'H

home_assists

Returns an `int` of the number of assists the home team registered.

home_even_strength_assists

Returns an `int` of the number of assists the home team registered while at even strength.

home_even_strength_goals

Returns an `int` of the number of goals the home team scored at even strength.

home_game_winning_goals

Returns an `int` of the number of game winning goals the home team scored.

home_goals

Returns an `int` of the number of goals the home team scored.

home_penalties_in_minutes

Returns an `int` of the length of time the home team spent in the penalty box.

home_players

Returns a `list` of `BoxscorePlayer` class instances for each player on the home team.

home_points

Returns an `int` of the number of points the home team registered.

home_power_play_assists

Returns an `int` of the number of assists the home team registered while on a power play.

home_power_play_goals

Returns an `int` of the number of goals the home team scored while on a power play.

home_save_percentage

Returns a `float` of the percentage of shots the home team saved. Percentage ranges from 0-1.

home_saves

Returns an `int` of the number of saves the home team made.

home_shooting_percentage

Returns a `float` of the home team's shooting percentage. Percentage ranges from 0-100.

home_short_handed_assists

Returns an `int` of the number of assists the home team registered while short handed.

home_short_handed_goals

Returns an `int` of the number of goals the home team scored while short handed.

home_shots_on_goal

Returns an `int` of the number of shots on goal the home team registered.

home_shutout

Returns an `int` denoting whether or not the home team shutout the home team.

losing_abbr

Returns a `string` of the losing team's abbreviation, such as 'WSH' for the Washington Capitals.

losing_name

Returns a `string` of the losing team's name, such as 'Washington Capitals'.

playoff_round

Returns a `string` denoting which round of the playoffs the game is a part of, such as 'Western First Round', or None if the game was played during the regular season.

time

Returns a `string` of the time the game started.

winner

Returns a `string` constant indicating whether the home or away team won.

winning_abbr

Returns a `string` of the winning team's abbreviation, such as 'VEG' for the Vegas Golden Knights.

winning_name

Returns a `string` of the winning team's name, such as 'Vegas Golden Knights'.

```
class sportsreference.nhl.boxscore.BoxscorePlayer(player_id, player_name,
                                                player_data)
Bases: sportsreference.nhl.player.AbstractPlayer
```

Get player stats for an individual game.

Given a player ID, such as 'zettehe01' for Henrik Zetterberg, their full name, and all associated stats from the Boxscore page in HTML format, parse the HTML and extract only the relevant stats for the specified player and assign them to readable properties.

This class inherits the `AbstractPlayer` class. As a result, all properties associated with `AbstractPlayer` can also be read directly from this class.

As this class is instantiated from within the `Boxscore` class, it should not be called directly and should instead be queried using the appropriate players properties from the `Boxscore` class.

Parameters

- **player_id** (*string*) – A player's ID according to hockey-reference.com, such as 'zettehe01' for Henrik Zetterberg. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'LLLLFFNN' where 'LLLL' are the first 5 letters in the player's last name, 'FF', are the first 2 letters in the player's first name, and 'NN' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

dataframe

Returns a `pandas DataFrame` containing all other relevant properties and values for the specified game.

decision

Returns a `string` denoting whether the goalie won or lost the game.

defensive_zone_start_percentage

Returns a `float` of the percentage of starts that took place in the player's defensive zone. Percentage ranges from 0-100.

defensive_zone_starts

Returns an `int` of the number of starts that took place in the player's defensive zone.

individual_corsi_for_events

Returns an `int` of the number of individual events that impacted the player's Corsi For score during the game.

offensive_zone_starts

Returns an `int` of the number of starts that took place in the player's offensive zone.

on_ice_shot_attempts_against

Returns an `int` of the Corsi Against shot attempts that occurred while the player was on the ice.

on_ice_shot_attempts_for

Returns an `int` of the Corsi For shot attempts that occurred while the player was on the ice.

shifts

Returns an `int` of the number of shifts the player had on the ice during the game.

time_on_ice

Returns a `string` of the total time the player has spent on ice in the format 'MM:SS'.

```
class sportsreference.nhl.boxscore.Boxscores(date, end_date=None)
```

Bases: object

Search for NHL games taking place on a particular day.

Retrieve a dictionary which contains a list of all games being played on a particular day. Output includes a link to the boxscore, and the names and abbreviations for both the home teams. If no games are played on a particular day, the list will be empty.

Parameters

- **date** (*datetime object*) – The date to search for any matches. The month, day, and year are required for the search, but time is not factored into the search.
- **end_date** (*datetime object*) – Optionally specify an end date to iterate until. All boxscores starting from the date specified in the ‘date’ parameter up to and including the boxscores specified in the ‘end_date’ parameter will be pulled. If left empty, or if ‘end_date’ is prior to ‘date’, only the games from the day specified in the ‘date’ parameter will be saved.

games

Returns a dictionary object representing all of the games played on the requested day. Dictionary is in the following format:

```
{ 'date' : [ # 'date' is the string date in format 'MM-DD-YYYY'
    {
        'home_name': Name of the home team, such as 'New York
                    Rangers' (`str`),
        'home_abbr': Abbreviation for the home team, such as
                    'NYR' (`str`),
        'away_name': Name of the away team, such as 'Boston Bruins'
                    (`str`),
        'away_abbr': Abbreviation for the away team, such as 'BOS'
                    (`str`),
        'boxscore': String representing the boxscore URI, such as
                    '201702040VAN' (`str`),
        'winning_name': Full name of the winning team, such as
                    'New York Rangers' (`str`),
        'winning_abbr': Abbreviation for the winning team, such as
                    'NYR' (`str`),
        'losing_name': Full name of the losing team, such as
                    'Boston Bruins' (`str`),
        'losing_abbr': Abbreviation for the losing team, such as
                    'BOS' (`str`),
        'home_score': Integer score for the home team (`int`),
        'away_score': Integer score for the away team (`int`)
    },
    { ... },
    ...
  ]
}
```

If no games were played on ‘date’, the list for [‘date’] will be empty.

```
sportsreference.nhl.boxscore.nhl_int_property_decorator(func)
```

Player

The Player module contains an abstract base class that can be inherited by both the BoxscorePlayer and Player classes in the Boxscore and Roster modules, respectively. All of the properties that appear in the

`AbstractPlayer` class can be read from either of the two child classes mentioned above.

class `sportsreference.nhl.player.AbstractPlayer` (*player_id*, *player_name*, *player_data*)
 Bases: `object`

Get player information and stats for all seasons.

Given a player ID, such as 'zettehe01' for Henrik Zetterberg, capture all relevant stats and information like name, team, height/weight, career goals, single-season assists, penalty minutes, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on sports-reference.com.

Parameters

- **player_id** (*string*) – A player's ID according to hockey-reference.com, such as 'zettehe01' for Henrik Zetterberg. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'llllffnn' where 'lllll' is the first five letters of the player's last name, 'ff' is the first two letters of the player's first name, and 'nn' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.
- **player_name** (*string*) – A string representing the player's first and last name, such as 'Henrik Zetterberg'.
- **player_data** (*string*) – A string representation of the player's HTML data from the Boxscore page. If the player appears in multiple tables, all of their information will appear in one single string concatenated together.

assists

Returns an `int` of the number of goals the player has assisted.

blocks_at_even_strength

Returns an `int` of the number of shots the player blocks while at even strength.

corsi_for_percentage

Returns a `float` of the 'Corsi For' percentage, equal to `corsi_for / (corsi_for + corsi_against)`. Percentage ranges from 0-100.

even_strength_assists

Returns an `int` of the number of goals the player has assisted while at even strength.

even_strength_goals

Returns an `int` of the number of goals the player has scored at even strength.

game_winning_goals

Returns an `int` of the number of game-winning goals the player has scored.

goals

Returns an `int` of the number of goals the player scored.

goals_against

Returns an `int` of the number of goals the opponent scored on the player while in goal.

hits_at_even_strength

Returns an `int` of the number of hits the player makes while at even strength.

name

Returns a `string` of the player's name, such as 'Henrik Zetterberg'.

offensive_zone_start_percentage

Returns a `float` of the percentage of faceoffs that occur in the offensive zone while the player is on ice. Percentage ranges from 0-100.

penalties_in_minutes

Returns an `int` of the number of minutes the player has served as a result of penalties.

player_id

Returns a `string` of the player's ID on hockey-reference, such as 'zettehe01' for Henrik Zetterberg.

plus_minus

Returns an `int` representing the relative presence the player has on the outcome of the game.

points

Returns an `int` of the number of points the player has gained.

power_play_assists

Returns an `int` of the number of goals the player has assisted while on a power play.

power_play_goals

Returns an `int` of the number of goals the player has scored while on a power play.

relative_corsi_for_percentage

Returns a `float` of the player's relative 'Corsi For' percentage, equal to the difference between the player's on and off-ice Corsi For percentage.

save_percentage

Returns a `float` of the percentage of shots the player has saved. Percentage ranges from 0-1.

saves

Returns an `int` of the number of shots the player has saved while in goal.

shooting_percentage

Returns a `float` of the percentage of the player's shots that go in the goal. Percentage ranges from 0-100.

short_handed_assists

Returns an `int` of the number of goals the player has assisted while short handed.

short_handed_goals

Returns an `int` of the number of goals the player has scored while short handed.

shots_against

Returns an `int` of the number of shots the opponent took while the player is in goal.

shots_on_goal

Returns an `int` of the number of shots on goal the player has made.

shutouts

Returns an `int` of the number of shutouts the player has registered while in goal.

Roster

The Roster module contains detailed player information, allowing each player to be queried by their player ID using the `Player` class which has detailed information ranging from career points totals to single-season stats and player height and weight. The following is an example on collecting career information for Henrik Zetterberg:

```
from sportsreference.nhl.roster import Player

zetterberg = Player('zettehe01')
print(zetterberg.name)  # Prints 'Henrik Zetterberg'
```

(continues on next page)

(continued from previous page)

```
print(zetterberg.points) # Prints Zetterberg's career points total
# Prints a Pandas DataFrame of all relevant Zetterberg stats per season
print(zetterberg.dataframe)
```

By default, the player's career stats are returned whenever a property is called. To get stats for a specific season, call the class instance with the season string. All future property requests will return the season-specific stats.

```
from sportsreference.nhl.roster import Player

zetterberg = Player('zettehe01') # Currently pulling career stats
print(zetterberg.points) # Prints Zetterberg's CAREER points total
# Prints Zetterberg's points total only for the 2017-18 season.
print(zetterberg('2017-18').points)
# Prints the number of games Zetterberg played in the 2017-18 season.
print(zetterberg.games_played)
```

After requesting single-season stats, the career stats can be requested again by calling the class without arguments or with the 'Career' string passed.

```
from sportsreference.nhl.roster import Player

zetterberg = Player('zettehe01') # Currently pulling career stats
# Prints Zetterberg's points total only for the 2017-18 season.
print(zetterberg('2017-18').points)
print(zetterberg('Career').points) # Prints Zetterberg's career points total
```

In addition, the Roster module also contains the Roster class which can be used to pull all players on a team's roster during a given season and creates instances of the Player class for each team member and adds them to a list to be easily queried.

```
from sportsreference.nhl.roster import Roster

detroit = Roster('DET')
for player in detroit.players:
    # Prints the name of all players who played for Houston in the most
    # recent season.
    print(player.name)
```

class sportsreference.nhl.roster.**Player** (*player_id*)
Bases: *sportsreference.nhl.player.AbstractPlayer*

Get player information and stats for all seasons.

Given a player ID, such as 'zettehe01' for Henrik Zetterberg, capture all relevant stats and information like name, team, height/weight, career goals, single-season assists, penalty minutes, and much more.

By default, the class instance will return the player's career stats, but single-season stats can be found by calling the instance with the requested season as denoted on sports-reference.com.

Parameters *player_id* (*string*) – A player's ID according to hockey-reference.com, such as 'zettehe01' for Henrik Zetterberg. The player ID can be found by navigating to the player's stats page and getting the string between the final slash and the '.html' in the URL. In general, the ID is in the format 'llllffnn' where 'llll' is the first five letters of the player's last name, 'ff' is the first two letters of the player's first name, and 'nn' is a number starting at '01' for the first time that player ID has been used and increments by 1 for every successive player.

adjusted_assists

Returns an int of the adjusted number of goals the player has assisted.

adjusted_goals

Returns an `int` of the adjusted number of goals the player has scored.

adjusted_goals_against_average

Returns a `float` of the adjusted goals against average for the player while in goal.

adjusted_goals_created

Returns an `int` of the adjusted number of goals the player created.

adjusted_points

Returns an `int` of the adjusted number of points the player has gained.

age

Returns an `int` of the player's age on February 1st of the season.

average_time_on_ice

Returns a `string` of the average time the player spends on the ice per game.

blocks_at_even_strength

Returns an `int` of the number of shots the player blocks while at even strength.

corsi_against

Returns a `float` of the player's 'Corsi Against' factor at even strength, equal to shots + blocks + misses.

corsi_for

Returns a `float` of the player's 'Corsi For' factor at even strength, equal to shots + blocks + misses.

dataframe

Returns a `pandas DataFrame` containing all other relevant class properties and values where each index is a different season plus the career stats.

defensive_point_shares

Returns a `float` of the player's defensive point share, equal to the approximate number of points the player contributed to while on defense.

defensive_zone_start_percentage

Returns a `float` of the percentage of faceoffs that occur in the defensive zone while the player is on ice. Percentage ranges from 0-100.

even_strength_goals_allowed

Returns an `int` of the number of goals the player allowed in goal while at even strength.

even_strength_save_percentage

Returns a `float` of the player's save percentage while at even strength.

even_strength_shots_faced

Returns an `int` of the number of shots the player has faced while at even strength.

faceoff_losses

Returns an `int` of the number of faceoffs the player lost.

faceoff_percentage

Returns a `float` of the percentage of faceoffs the player wins. Percentage ranges from 0-100.

faceoff_wins

Returns an `int` of the number of faceoffs the player won.

fenwick_against

Returns an `int` of the player's 'Fenwick Against' factor at even strength, equal to shots + misses.

fenwick_for

Returns an `int` of the player's 'Fenwick For' factor at even strength, equal to shots + misses.

fenwick_for_percentage

Returns a `float` of the player's 'Fenwick For' percentage, equal to `fenwick_for / (fenwick_for + fenwick_against)`. Percentage ranges from 0-100.

games_played

Returns an `int` of the number of games the player participated in.

giveaways

Returns an `int` of the number of times the player gave the puck away to an opponent.

goal_against_percentage_relative

Returns an `int` of the player's goals against average compared to the league average where 100 is an average player and 0 means the player saved every single shot.

goalie_point_shares

Returns a `float` of the player's point share while in goal.

goals_against_average

Returns a `float` of the average number of goals the opponent has scored per game while the player is in goal.

goals_against_on_ice

Returns an `int` of the number of times the team has been scored on while the player is on ice.

goals_created

Returns an `int` of the number of goals the player created, equal to $(goals + assists * 0.5) * team_goals / (team_goals + team_assists * 0.5)$.

goals_for_on_ice

Returns an `int` of the number of goals the team has scored while the player is on ice.

goals_saved_above_average

Returns a `float` of the number of goals the player saved above the league average.

height

Returns a `string` of the player's height in the format "feet-inches".

league

Returns a `string` of the league the player's team participates in.

losses

Returns an `int` of the number of times the team lost while the player is in goal.

minutes

Returns an `int` of the total number of minutes the player has spent in goal.

name

Returns a `string` of the player's name, such as 'Henrik Zetterberg'.

offensive_point_shares

Returns a `float` of the player's offensive point share, equal to the approximate number of points the player contributed to while on offense.

pdo

Returns a `float` of the team's PDO while the player is on ice at even strength, equal to the team's shooting percentage + save percentage. Percentage ranges from 0-100.

point_shares

Returns a `float` of the player's total point share, equal to the sum of the player's offensive and defensive point share.

power_play_goals_against_on_ice

Returns an `int` of the total number of power play goals against while the player was on ice.

power_play_goals_allowed

Returns an `int` of the number of goals the player allowed in goal while on a power play.

power_play_goals_for_on_ice

Returns an `int` of the total number of power play goals for while the player was on ice.

power_play_save_percentage

Returns a `float` of the player's save percentage while on a power play.

power_play_shots_faced

Returns an `int` of the number of shots the player has faced while on a power play.

quality_start_percentage

Returns a `float` of the percentage of the player's starts that are considered quality starts while in goal. Percentage ranges from 0-1.

quality_starts

Returns an `int` of the number of quality starts the player has had, equal to starting out with an in-game save percentage greater than the player's average save percentage for the year.

really_bad_starts

Returns an `int` of the number of really bad starts the player has had, equal to starting out with an in-game save percentage less than 85%.

relative_fenwick_for_percentage

Returns a `float` of the player's relative 'Fenwick For' percentage, equal to the difference between the player's on and off-ice Fenwick For percentage.

save_percentage_on_ice

Returns an `int` of the team's save percentage while the player is on ice.

season

Returns a `string` of the season in the format 'YYYY-YY', such as '2017-18'. If no season was requested, the career stats will be returned for the player and the season will default to 'Career'.

shooting_percentage_on_ice

Returns a `float` of the team's shooting percentage while the player is on ice.

shootout_attempts

Returns an `int` of the number of shootouts the player attempted.

shootout_goals

Returns an `int` of the number of shootout goals the player scored.

shootout_misses

Returns an `int` of the number of shootouts the player failed to score.

shootout_percentage

Returns a `float` of the percentage of shootouts the player scores in. Percentage ranges from 0-100.

short_handed_goals_allowed

Returns an `int` of the number of goals the player allowed in goal while short handed.

short_handed_save_percentage

Returns a `float` of the player's save percentage while short handed.

short_handed_shots_faced

Returns an `int` of the number of shots the player has faced while short handed.

takeaways

Returns an `int` of the number of times the player took the puck away from an opponent.

team_abbreviation

Returns a `string` of the team's abbreviation, such as 'DET' for the Detroit Red Wings.

ties_plus_overtime_loss

Returns an `int` of the number of times the team has either tied or lost in overtime or a shootout while the player is in goal.

time_on_ice

Returns an `int` of the total time the player has spent on ice in minutes.

time_on_ice_even_strength

Returns a `float` of the amount of time the player spent on ice in minutes while at even strength.

total_goals_against_on_ice

Returns an `int` of the total number of goals against while the player was on ice.

total_goals_for_on_ice

Returns an `int` of the total number of goals for while the player was on ice.

total_shots

Returns an `int` of the total number of shots the player took regardless of them being on goal or not.

weight

Returns an `int` of the player's weight in pounds.

wins

Returns an `int` of the number of times the team won while the player is in goal.

class `sportsreference.nhl.roster.Roster` (*team*, *year=None*, *slim=False*)

Bases: `object`

Get stats for all players on a roster.

Request a team's roster for a given season and create instances of the `Player` class for each player, containing a detailed list of the player's statistics and information.

Parameters

- **team** (*string*) – The team's abbreviation, such as 'DET' for the Detroit Red Wings.
- **year** (*string (optional)*) – The 6-digit year to pull the roster from, such as '2017-18'. If left blank, defaults to the most recent season.
- **slim** (*boolean (optional)*) – Set to `True` to return a limited subset of player information including the name and player ID for each player as opposed to all of their respective stats which greatly reduces the time to return a response if just the names and IDs are desired. Defaults to `False`.

players

Returns a `list` of player instances for each player on the requested team's roster if the `slim` property is `False` when calling the `Roster` class. If the `slim` property is `True`, returns a `dictionary` where each key is a string of the player's ID and each value is the player's first and last name as listed on the roster page.

Schedule

The `Schedule` module can be used to iterate over all games in a team's schedule to get game information such as the date, score, result, and more. Each game also has a link to the `Boxscore` class which has much more detailed information on the game metrics.

```
from sportsreference.nhl.schedule import Schedule

detroit_schedule = Schedule('DET')
for game in detroit_schedule:
    print(game.date)    # Prints the date the game was played
    print(game.result)  # Prints whether the team won or lost
    # Creates an instance of the Boxscore class for the game.
    boxscore = game.boxscore
```

class sportsreference.nhl.schedule.**Game**(*game_data*, *year*)

Bases: object

A representation of a matchup between two teams.

Stores all relevant high-level match information for a game in a team's schedule including date, time, opponent, and result.

Parameters

- **game_data** (*string*) – The row containing the specified game information.
- **year** (*string*) – The year of the current season.

boxscore

Returns an instance of the Boxscore class containing more detailed stats on the game.

boxscore_index

Returns a *string* of the URI for a boxscore which can be used to access or index a game.

corsi_against

Returns an *int* of the Corsi Against at Even Strength metric which equals the number of shots + blocks + misses by the opponent.

corsi_for

Returns an *int* of the Corsi For at Even Strength metric which equals the number of shots + blocks + misses.

corsi_for_percentage

Returns a *float* of the percentage of control a team had of the puck which is calculated by the *corsi_for* value divided by the sum of *corsi_for* and *corsi_against*. Values greater than 50.0 indicate the team had more control of the puck than their opponent. Percentage ranges from 0-100.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the boxscore string.

dataframe_extended

Returns a pandas DataFrame representing the Boxscore class for the game. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property. The index for the DataFrame is the boxscore string.

date

Returns a *string* of the date the game was played, such as '2017-10-05'.

datetime

Returns a datetime object to indicate the month, day, and year the game was played at.

faceoff_losses

Returns an *int* of the number of faceoffs the team lost at even strength.

faceoff_win_percentage

Returns a `float` of percentage of faceoffs the team won while at even strength. Percentage ranges from 0-100.

faceoff_wins

Returns an `int` of the number of faceoffs the team won at even strength.

fenwick_against

Returns an `int` of the Fenwick Against at Even Strength metric which equals the number of shots + misses by the opponent.

fenwick_for

Returns an `int` of the Fenwick For at Even Strength metric which equals the number of shots + misses.

fenwick_for_percentage

Returns a `float` of the percentage of control a team had of the puck which is calculated by the `fenwick_for` value divided by the sum of `fenwick_for` and `fenwick_against`. Values greater than 50.0 indicate the team had more control of the puck than their opponent. Percentage ranges from 0-100.

game

Returns an `int` to indicate which game in the season was requested. The first game of the season returns 1.

goals_allowed

Returns an `int` of the number of goals the team allowed during the game.

goals_scored

Returns an `int` of the number of goals the team scored during the game.

location

Returns a `string` constant to indicate whether the game was played at home or away.

offensive_zone_start_percentage

Returns a `float` of the percentage of stats that took place in the offensive half. Value is calculated by the number of offensive zone starts divided by the sum of offensive zone starts and defensive zone starts. Percentage ranges from 0-100.

opp_penalties_in_minutes

Returns an `int` of the total number of minutes the opponent served for penalties.

opp_power_play_goals

Returns an `int` of the number of power play goals the opponent scored.

opp_power_play_opportunities

Returns an `int` of the number of power play opportunities the opponent had.

opp_short_handed_goals

Returns an `int` of the number of shorthanded goals the opponent scored.

opp_shots_on_goal

Returns an `int` of the total number of shots on goal the opponent registered.

opponent_abbr

Returns a `string` of the opponent's 3-letter abbreviation, such as 'NYR' for the New York Rangers.

opponent_name

Returns a `string` of the opponent's name, such as 'New York Rangers'.

overtime

Returns an `int` of the number of overtimes that were played during the game, or an `int` constant if the game went to a shootout.

pdo

Returns a `float` of the team's PDO at Even Strength metric which is calculated by the sum of the shooting percentage and save percentage. Percentage ranges from 0-100.

penalties_in_minutes

Returns an `int` of the total number of minutes the team served for penalties.

power_play_goals

Returns an `int` of the number of power play goals the team scored.

power_play_opportunities

Returns an `int` of the number of power play opportunities the team had.

result

Returns a `string` constant to indicate whether the team lost in regulation, lost in overtime, or won.

short_handed_goals

Returns an `int` of the number of shorthanded goals the team scored.

shots_on_goal

Returns an `int` of the total number of shots on goal the team registered.

class `sportsreference.nhl.schedule.Schedule` (*abbreviation*, *year=None*)

Bases: `object`

An object of the given team's schedule.

Generates a team's schedule for the season including wins, losses, and scores if applicable.

Parameters

- **abbreviation** (*string*) – A team's short name, such as 'NYR' for the New York Rangers.
- **year** (*string (optional)*) – The requested year to pull stats from.

dataframe

Returns a pandas DataFrame where each row is a representation of the Game class. Rows are indexed by the boxscore string.

dataframe_extended

Returns a pandas DataFrame where each row is a representation of the Boxscore class for every game in the schedule. Rows are indexed by the boxscore string. This property provides much richer context for the selected game, but takes longer to process compared to the lighter 'dataframe' property.

Teams

The Teams module exposes information for all NHL teams including the team name and abbreviation, the number of games they won during the season, the total number of shots on goal, and much more.

```
from sportsreference.nhl.teams import Teams

teams = Teams()
for team in teams:
    print(team.name)    # Prints the team's name
    print(team.shots_on_goal)  # Prints the team's total shots on goal
```

Each Team instance contains a link to the `Schedule` class which enables easy iteration over all games for a particular team. A Pandas DataFrame can also be queried to easily grab all stats for all games.

```

from sportsreference.nhl.teams import Teams

teams = Teams()
for team in teams:
    schedule = team.schedule # Returns a Schedule instance for each team
    # Returns a Pandas DataFrame of all metrics for all game Boxscores for
    # a season.
    df = team.schedule.dataframe_extended

```

Lastly, each Team instance also contains a link to the Roster class which enables players from the team to be easily queried. Each Roster instance contains detailed stats and information for each player on the team.

```

from sportsreference.nhl.teams import Teams

teams = Teams()
for team in teams:
    # Creates an instance of the roster class for each player on the team.
    roster = team.roster
    for player in roster.players:
        print(player.name) # Prints the name of each player on the team.

```

```

class sportsreference.nhl.teams.Team(team_data, rank, year=None)
    Bases: object

```

An object containing all of a team's season information.

Finds and parses all team stat information and identifiers, such as rank, name, and abbreviation, and sets them as properties which can be directly read from for easy reference.

Parameters

- **team_data** (*string*) – A string containing all of the rows of stats for a given team. If multiple tables are being referenced, this will be comprised of multiple rows in a single string.
- **rank** (*int*) – A team's position in the league based on the number of points they obtained during the season.
- **year** (*string (optional)*) – The requested year to pull stats from.

abbreviation

Returns a *string* of the team's abbreviation, such as 'DET' for the Detroit Red Wings.

average_age

Returns a *float* of the average age of all players on the team, weighted by their time on ice.

dataframe

Returns a pandas DataFrame containing all other class properties and values. The index for the DataFrame is the string abbreviation of the team, such as 'DET'.

games_played

Returns an *int* of the total number of games the team has played in the season.

goals_against

Returns an *int* of the total number of goals opponents scored against the team during the season.

goals_for

Returns an *int* of the total number of goals a team scored during the season.

losses

Returns an *int* of the total number of losses the team had in the season.

name

Returns a `string` of the team's full name, such as 'Detroit Red Wings'.

overtime_losses

Returns an `int` of the total number of overtime losses the team had in the season.

pdo_at_even_strength

Returns a `float` of the PDO at even strength which equates to the shooting percentage plus the save percentage.

penalty_killing_percentage

Returns a `float` denoting the percentage of power plays that have been successfully defended without a goal being conceded. Percentage ranges from 0-100.

points

Returns an `int` of the total number of points the team gained in the season.

points_percentage

Returns a `float` denoting the percentage of points gained divided by the maximum possible points available during the season. Percentage ranges from 0-1.

power_play_goals

Returns an `int` of the total number of power play goals scored.

power_play_goals_against

Returns an `int` of the total number of power play goals conceded.

power_play_opportunities

Returns an `int` of the total number of power play opportunities for a team during the season.

power_play_opportunities_against

Returns an `int` of the total number of power play opportunities for the opponents during the season.

power_play_percentage

Returns a `float` denoting the percentage of power play opportunities where the team has scored. Percentage ranges from 0-100.

rank

Returns an `int` of the team's rank based on the number of points they obtained in the season.

roster

Returns an instance of the Roster class containing all players for the team during the season with all career stats.

save_percentage

Returns a `float` denoting the percentage of shots the team has saved during the season. Percentage ranges from 0-1.

schedule

Returns an instance of the Schedule class containing the team's complete schedule for the season.

shooting_percentage

Returns a `float` denoting the percentage of shots to goals during the season. Percentage ranges from 0-100.

short_handed_goals

Returns an `int` of the number of short handed goals the team has scored during the season.

short_handed_goals_against

Returns an `int` of the number of short handed goals the team has conceded during the season.

shots_against

Returns an `int` of the total number of shots on goal the team's opponents made during the season.

shots_on_goal

Returns an `int` of the total number of shots on goal the team made during the season.

simple_rating_system

Returns a `float` which takes into account the average goal differential vs a team's strength of schedule. The league average evaluates to 0.0. Teams which have a positive score are comparatively stronger than average while teams with a negative score are weaker.

strength_of_schedule

Returns a `float` denoting a team's strength of schedule, based on goals scores and conceded. Higher values result in more challenging schedules while 0.0 is an average schedule.

total_goals_per_game

Returns a `float` for the average number of goals scored per game.

wins

Returns an `int` of the total number of wins the team had in the season.

class `sportsreference.nhl.teams.Teams` (*year=None*)

Bases: `object`

A list of all NHL teams and their stats in a given year.

Finds and retrieves a list of all NHL teams from www.hockey-reference.com and creates a `Team` instance for every team that participated in the league in a given year. The `Team` class comprises a list of all major stats and a few identifiers for the requested season.

Parameters *year* (*string (optional)*) – The requested year to pull stats from.

dataframes

Returns a pandas `DataFrame` where each row is a representation of the `Team` class. Rows are indexed by the team abbreviation.

1.6.2 Examples

Thanks to the broad range of metrics that are pulled from sports-reference.com, there are multiple ways you can use the *sportsreference* package. This page has multiple examples beyond those listed on the home page to demonstrate some cool things you can do which leverage the tool. This page is by no means exhaustive and the examples aren't necessarily the most efficient in the hope of providing the most clarity.

In general, most examples shown for a specific sport are applicable for all sports currently supported by *sportsreference*.

1.6.2.1 Finding Tallest Players

For each team, find the tallest player on the roster and print out their name and height in inches.

```
from sportsreference.nba.teams import Teams

def get_height_in_inches(height):
    feet, inches = height.split('-')
    return int(feet) * 12 + int(inches)

def print_tallest_player(team_heights):
    tallest_player = max(team_heights, key=team_heights.get)
    print('%s: %s in.' % (tallest_player, team_heights[tallest_player]))

for team in Teams():
```

(continues on next page)

(continued from previous page)

```
print('=' * 80)
print(team.name)
team_heights = {}
for player in team.roster.players:
    height = get_height_in_inches(player.height)
    team_heights[player.name] = height
print_tallest_player(team_heights)
```

1.6.2.2 Writing To CSV and Pickle

To prevent re-pulling data from datasets that won't change, such as completed games with fixed statistics, the pandas DataFrame can be saved to the local filesystem for re-use later on. Two common file types for this are CSV files and the high-performing Pickle files. CSV files are a common file type that many tools and editors support and save an interpretation of the DataFrame, while a Pickle file is a special file that saves the DataFrame exactly as-is. Pickle files are faster to read and write compared to CSV files and don't pose a risk of missing or altered data compared to CSV files.

Save the combined stats for each team to both a CSV and Pickle file.

```
from sportsreference.ncaab.teams import Teams

for team in Teams():
    team.dataframe.to_csv('%s.csv' % team.abbreviation.lower())
    team.dataframe.to_pickle('%s.pkl' % team.abbreviation.lower())
```

1.6.2.3 Finding Top Win Percentage By Year

For each year in a range, find the team with the most wins during the season and print their name and the win total.

```
from sportsreference.mlb.teams import Teams

def print_most_wins(year, wins):
    most_wins = max(wins, key=wins.get)
    print('%s: %s - %s' % (year, wins[most_wins], most_wins))

for year in range(2000, 2019):
    wins = {}
    for team in Teams(year):
        wins[team.name] = team.wins
    print_most_wins(year, wins)
```

1.6.3 Installation

The easiest way to install *sportsreference* is by downloading the latest released binary from PyPI using PIP. For instructions on installing PIP, visit [PyPA.io](https://pyPA.io) for detailed steps on installing the package manager for your local environment.

Next, run:

```
pip install sportsreference
```

to download and install the latest official release of *sportsreference* on your machine. You now have the latest stable version of *sportsreference* installed and can begin using it following the examples!

If the bleeding-edge version of *sportsreference* is desired, clone this repository using git and install all of the package requirements with PIP:

```
git clone https://github.com/roclark/sportsreference
cd sportsreference
pip install -r requirements.txt
```

Once complete, create a Python wheel for your default version of Python by running the following command:

```
python setup.py sdist bdist_wheel
```

This will create a *.whl* file in the *dist* directory which can be installed with the following command:

```
pip install dist/*.whl
```

1.6.4 Testing

Sportsreference contains a testing suite which aims to test all major portions of code for proper functionality. To run the test suite against your environment, ensure all of the requirements are installed by running:

```
pip install -r requirements.txt
```

Next, start the tests by running `py.test` while optionally including coverage flags which identify the amount of production code covered by the testing framework:

```
py.test --cov=sportsreference --cov-report term-missing tests/
```

If the tests were successful, it will return a green line will show a message at the end of the output similar to the following:

```
===== 380 passed in 245.56 seconds =====
```

If a test failed, it will show the number of failed and what went wrong within the test output. If that's the case, ensure you have the latest version of code and are in a supported environment. Otherwise, create an issue on GitHub to attempt to get the issue resolved.

CHAPTER 2

Indices and tables

- `genindex`
- `modindex`
- `search`

S

- `sportsreference.mlb.boxscore`, 5
- `sportsreference.mlb.player`, 12
- `sportsreference.mlb.roster`, 14
- `sportsreference.mlb.schedule`, 20
- `sportsreference.mlb.teams`, 22
- `sportsreference.nba.boxscore`, 29
- `sportsreference.nba.player`, 36
- `sportsreference.nba.roster`, 39
- `sportsreference.nba.schedule`, 43
- `sportsreference.nba.teams`, 46
- `sportsreference.ncaab.boxscore`, 49
- `sportsreference.ncaab.conferences`, 57
- `sportsreference.ncaab.player`, 58
- `sportsreference.ncaab.rankings`, 60
- `sportsreference.ncaab.roster`, 62
- `sportsreference.ncaab.schedule`, 64
- `sportsreference.ncaab.teams`, 67
- `sportsreference.ncaaf.boxscore`, 73
- `sportsreference.ncaaf.conferences`, 78
- `sportsreference.ncaaf.player`, 79
- `sportsreference.ncaaf.rankings`, 82
- `sportsreference.ncaaf.roster`, 84
- `sportsreference.ncaaf.schedule`, 87
- `sportsreference.ncaaf.teams`, 90
- `sportsreference.nfl.boxscore`, 94
- `sportsreference.nfl.player`, 99
- `sportsreference.nfl.roster`, 103
- `sportsreference.nfl.schedule`, 109
- `sportsreference.nfl.teams`, 113
- `sportsreference.nhl.boxscore`, 116
- `sportsreference.nhl.player`, 121
- `sportsreference.nhl.roster`, 123
- `sportsreference.nhl.schedule`, 128
- `sportsreference.nhl.teams`, 131

A

- abbreviation (*sportsreference.mlb.teams.Team attribute*), 23
- abbreviation (*sportsreference.nba.teams.Team attribute*), 46
- abbreviation (*sportsreference.ncaab.teams.Team attribute*), 67
- abbreviation (*sportsreference.ncaaf.teams.Team attribute*), 90
- abbreviation (*sportsreference.nfl.teams.Team attribute*), 113
- abbreviation (*sportsreference.nhl.teams.Team attribute*), 131
- AbstractPlayer (class in *sportsreference.mlb.player*), 12
- AbstractPlayer (class in *sportsreference.nba.player*), 36
- AbstractPlayer (class in *sportsreference.ncaab.player*), 58
- AbstractPlayer (class in *sportsreference.ncaaf.player*), 79
- AbstractPlayer (class in *sportsreference.nfl.player*), 99
- AbstractPlayer (class in *sportsreference.nhl.player*), 121
- adjusted_assists (*sportsreference.nhl.roster.Player attribute*), 123
- adjusted_goals (*sportsreference.nhl.roster.Player attribute*), 123
- adjusted_goals_against_average (*sportsreference.nhl.roster.Player attribute*), 124
- adjusted_goals_created (*sportsreference.nhl.roster.Player attribute*), 124
- adjusted_net_yards_per_attempt_index (*sportsreference.nfl.roster.Player attribute*), 103
- adjusted_net_yards_per_pass_attempt (*sportsreference.nfl.roster.Player attribute*), 103
- adjusted_points (*sportsreference.nhl.roster.Player attribute*), 124
- adjusted_yards_per_attempt (*sportsreference.ncaaf.player.AbstractPlayer attribute*), 79
- adjusted_yards_per_attempt (*sportsreference.ncaaf.roster.Player attribute*), 84
- adjusted_yards_per_attempt (*sportsreference.nfl.roster.Player attribute*), 103
- adjusted_yards_per_attempt_index (*sportsreference.nfl.roster.Player attribute*), 103
- age (*sportsreference.nhl.roster.Player attribute*), 124
- all_purpose_yards (*sportsreference.nfl.roster.Player attribute*), 103
- and_ones (*sportsreference.nba.roster.Player attribute*), 39
- approximate_value (*sportsreference.nfl.roster.Player attribute*), 103
- arena (*sportsreference.ncaab.schedule.Game attribute*), 65
- arena (*sportsreference.nhl.boxscore.Boxscore attribute*), 116
- assist_percentage (*sportsreference.nba.player.AbstractPlayer attribute*), 36
- assist_percentage (*sportsreference.ncaab.player.AbstractPlayer attribute*), 58
- assist_percentage (*sportsreference.ncaab.teams.Team attribute*), 67
- assists (*sportsreference.mlb.player.AbstractPlayer attribute*), 12
- assists (*sportsreference.mlb.roster.Player attribute*), 15
- assists (*sportsreference.nba.player.AbstractPlayer attribute*), 36
- assists (*sportsreference.nba.teams.Team attribute*), 46
- assists (*sportsreference.ncaab.player.AbstractPlayer attribute*), 58
- assists (*sportsreference.ncaab.teams.Team attribute*),

67
assists (*sportsreference.nhl.player.AbstractPlayer attribute*), 121
assists_on_tackles (*sportsreference.ncaaf.player.AbstractPlayer attribute*), 79
assists_on_tackles (*sportsreference.ncaaf.roster.Player attribute*), 84
assists_on_tackles (*sportsreference.nfl.player.AbstractPlayer attribute*), 100
assists_on_tackles (*sportsreference.nfl.roster.Player attribute*), 103
at_bats (*sportsreference.mlb.player.AbstractPlayer attribute*), 12
at_bats (*sportsreference.mlb.roster.Player attribute*), 15
at_bats (*sportsreference.mlb.teams.Team attribute*), 23
attempted_passes (*sportsreference.ncaaf.player.AbstractPlayer attribute*), 80
attempted_passes (*sportsreference.ncaaf.roster.Player attribute*), 84
attempted_passes (*sportsreference.nfl.player.AbstractPlayer attribute*), 100
attempted_passes (*sportsreference.nfl.roster.Player attribute*), 104
attendance (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
attendance (*sportsreference.mlb.schedule.Game attribute*), 20
attendance (*sportsreference.nfl.boxscore.Boxscore attribute*), 94
attendance (*sportsreference.nhl.boxscore.Boxscore attribute*), 116
average_age (*sportsreference.nhl.teams.Team attribute*), 131
average_batter_age (*sportsreference.mlb.teams.Team attribute*), 23
average_kickoff_return_yards (*sportsreference.ncaaf.boxscore.BoxscorePlayer attribute*), 75
average_kickoff_return_yards (*sportsreference.nfl.boxscore.BoxscorePlayer attribute*), 98
average_leverage_index (*sportsreference.mlb.boxscore.BoxscorePlayer attribute*), 10
average_leverage_index_pitcher (*sportsreference.mlb.boxscore.BoxscorePlayer attribute*), 10
average_pitcher_age (*sportsreference.mlb.teams.Team attribute*), 23
average_punt_return_yards (*sportsreference.ncaaf.boxscore.BoxscorePlayer attribute*), 75
average_time_on_ice (*sportsreference.nhl.roster.Player attribute*), 124
away_abbreviation (*sportsreference.nfl.boxscore.Boxscore attribute*), 94
away_assist_percentage (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_assist_percentage (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_assists (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_assists (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_assists (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_assists (*sportsreference.nhl.boxscore.Boxscore attribute*), 116
away_at_bats (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_average_leverage_index (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_base_out_runs_added (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_base_out_runs_saved (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_bases_on_balls (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_batting_average (*sportsreference.mlb.boxscore.Boxscore attribute*), 5
away_block_percentage (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_block_percentage (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_blocks (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_blocks (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_defensive_rating (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_defensive_rating (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_defensive_rebound_percentage (*sportsreference.nba.boxscore.Boxscore attribute*), 29
away_defensive_rebound_percentage (*sportsreference.ncaab.boxscore.Boxscore attribute*), 50
away_defensive_rebounds (*sportsrefer-*

<i>ence.nba.boxscore.Boxscore attribute</i>), 30	away_free_throw_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50
away_defensive_rebounds (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_free_throws (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30
away_earned_runs (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 5	away_free_throws (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50
away_effective_field_goal_percentage (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_fumbles (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 73
away_effective_field_goal_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_fumbles (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94
away_even_strength_assists (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 116	away_fumbles_lost (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 73
away_even_strength_goals (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 116	away_fumbles_lost (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94
away_field_goal_attempts (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_game_score (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 5
away_field_goal_attempts (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_game_winning_goals (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 116
away_field_goal_percentage (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_goals (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 117
away_field_goal_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_grounded_balls (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_field_goals (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_hits (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_field_goals (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_home_runs (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_first_downs (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 73	away_inherited_runners (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_first_downs (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94	away_inherited_score (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_fly_balls (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 5	away_innings_pitched (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_fourth_down_attempts (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94	away_interceptions (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 73
away_fourth_down_conversions (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94	away_interceptions (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 94
away_free_throw_attempt_rate (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_line_drives (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 6
away_free_throw_attempt_rate (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_losses (<i>sportsreference.mlb.teams.Team attribute</i>), 23
away_free_throw_attempts (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_losses (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30
away_free_throw_attempts (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50	away_losses (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 50
away_free_throw_percentage (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30	away_losses (<i>sportsreference.ncaab.teams.Team attribute</i>), 67
	away_minutes_played (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 30

away_minutes_played	(sportsreference.ncaab.boxscore.Boxscore attribute), 50	away_minutes_played	(sportsreference.ncaab.boxscore.Boxscore attribute), 51
away_net_pass_yards	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_pitches	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_offensive_rating	(sportsreference.nba.boxscore.Boxscore attribute), 30	away_plate_appearances	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_offensive_rating	(sportsreference.ncaab.boxscore.Boxscore attribute), 51	away_players	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_offensive_rebound_percentage	(sportsreference.nba.boxscore.Boxscore attribute), 30	away_players	(sportsreference.nba.boxscore.Boxscore attribute), 30
away_offensive_rebound_percentage	(sportsreference.ncaab.boxscore.Boxscore attribute), 51	away_players	(sportsreference.ncaab.boxscore.Boxscore attribute), 51
away_offensive_rebounds	(sportsreference.nba.boxscore.Boxscore attribute), 30	away_players	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73
away_offensive_rebounds	(sportsreference.ncaab.boxscore.Boxscore attribute), 51	away_players	(sportsreference.nfl.boxscore.Boxscore attribute), 95
away_on_base_percentage	(sportsreference.mlb.boxscore.Boxscore attribute), 6	away_players	(sportsreference.nhl.boxscore.Boxscore attribute), 117
away_on_base_plus	(sportsreference.mlb.boxscore.Boxscore attribute), 6	away_points	(sportsreference.nba.boxscore.Boxscore attribute), 30
away_pass_attempts	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73	away_points	(sportsreference.ncaab.boxscore.Boxscore attribute), 51
away_pass_attempts	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_points	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73
away_pass_completions	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73	away_points	(sportsreference.nfl.boxscore.Boxscore attribute), 95
away_pass_completions	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_points	(sportsreference.nhl.boxscore.Boxscore attribute), 117
away_pass_touchdowns	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73	away_power_play_assists	(sportsreference.nhl.boxscore.Boxscore attribute), 117
away_pass_touchdowns	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_power_play_goals	(sportsreference.nhl.boxscore.Boxscore attribute), 117
away_pass_yards	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73	away_putouts	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_pass_yards	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_ranking	(sportsreference.ncaab.boxscore.Boxscore attribute), 51
away_penalties	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73	away_rbi	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_penalties	(sportsreference.nfl.boxscore.Boxscore attribute), 95	away_record	(sportsreference.mlb.teams.Team attribute), 23
away_penalties_in_minutes	(sportsreference.nhl.boxscore.Boxscore attribute), 117	away_runs	(sportsreference.mlb.boxscore.Boxscore attribute), 6
away_personal_fouls	(sportsreference.nba.boxscore.Boxscore attribute), 30	away_rush_attempts	(sportsreference.ncaaf.boxscore.Boxscore attribute), 73
away_personal_fouls	(sportsreference	away_rush_attempts	(sportsreference.nfl.boxscore.Boxscore attribute), 95
		away_rush_touchdowns	(sportsreference.ncaaf.boxscore.Boxscore attribute),

- 73
- away_rush_touchdowns (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_rush_yards (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 73
- away_rush_yards (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_save_percentage (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_saves (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_shooting_percentage (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_short_handed_assists (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_short_handed_goals (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_shots_on_goal (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_shutout (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- away_slugging_percentage (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_steal_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 30
- away_steal_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_steals (*sportsreference.nba.boxscore.Boxscore* attribute), 30
- away_steals (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_strikeouts (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_strikes (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_strikes_by_contact (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_strikes_looking (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_strikes_swinging (*sportsreference.mlb.boxscore.Boxscore* attribute), 6
- away_third_down_attempts (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_third_down_conversions (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_three_point_attempt_rate (*sportsreference.nba.boxscore.Boxscore* attribute), 30
- away_three_point_attempt_rate (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_three_point_field_goal_attempts (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_three_point_field_goal_attempts (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_three_point_field_goal_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_three_point_field_goal_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_three_point_field_goals (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_three_point_field_goals (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_time_of_possession (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_times_sacked (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_total_rebound_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_total_rebound_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_total_rebounds (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_total_rebounds (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_total_yards (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74
- away_total_yards (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_true_shooting_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_true_shooting_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_turnover_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_turnover_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_turnovers (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- away_turnovers (*sportsreference.ncaab.boxscore.Boxscore* attribute), 51
- away_turnovers (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74
- away_turnovers (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- away_two_point_field_goal_attempts (*sportsreference.nba.boxscore.Boxscore* attribute), 31

- tribute), 31
- away_two_point_field_goal_attempts (sportsreference.ncaa.bboxscore.Boxscore attribute), 51
- away_two_point_field_goal_percentage (sportsreference.nba.bboxscore.Boxscore attribute), 31
- away_two_point_field_goal_percentage (sportsreference.ncaa.bboxscore.Boxscore attribute), 52
- away_two_point_field_goals (sportsreference.nba.bboxscore.Boxscore attribute), 31
- away_two_point_field_goals (sportsreference.ncaa.bboxscore.Boxscore attribute), 52
- away_unknown_bat_type (sportsreference.mlb.bboxscore.Boxscore attribute), 7
- away_win_percentage (sportsreference.ncaa.bboxscore.Boxscore attribute), 52
- away_win_probability_added (sportsreference.mlb.bboxscore.Boxscore attribute), 7
- away_win_probability_by_pitcher (sportsreference.mlb.bboxscore.Boxscore attribute), 7
- away_win_probability_for_offensive_players (sportsreference.mlb.bboxscore.Boxscore attribute), 7
- away_win_probability_subtracted (sportsreference.mlb.bboxscore.Boxscore attribute), 7
- away_wins (sportsreference.mlb.teams.Team attribute), 23
- away_wins (sportsreference.nba.bboxscore.Boxscore attribute), 31
- away_wins (sportsreference.ncaa.bboxscore.Boxscore attribute), 52
- away_wins (sportsreference.ncaa.teams.Team attribute), 67
- away_yards_from_penalties (sportsreference.ncaaf.bboxscore.Boxscore attribute), 74
- away_yards_from_penalties (sportsreference.nfl.bboxscore.Boxscore attribute), 95
- away_yards_lost_from_sacks (sportsreference.nfl.bboxscore.Boxscore attribute), 95
- B**
- balks (sportsreference.mlb.roster.Player attribute), 15
- balks (sportsreference.mlb.teams.Team attribute), 23
- base_out_runs_added (sportsreference.mlb.bboxscore.BoxscorePlayer attribute), 10
- base_out_runs_saved (sportsreference.mlb.bboxscore.BoxscorePlayer attribute), 10
- bases_on_balls (sportsreference.mlb.player.AbstractPlayer attribute), 13
- bases_on_balls (sportsreference.mlb.roster.Player attribute), 15
- bases_on_balls (sportsreference.mlb.teams.Team attribute), 23
- bases_on_balls_given (sportsreference.mlb.player.AbstractPlayer attribute), 13
- bases_on_balls_given (sportsreference.mlb.roster.Player attribute), 15
- bases_on_balls_given_per_nine_innings (sportsreference.mlb.roster.Player attribute), 15
- bases_on_walks_given (sportsreference.mlb.teams.Team attribute), 23
- bases_on_walks_given_per_nine_innings (sportsreference.mlb.teams.Team attribute), 23
- batters_faced (sportsreference.mlb.player.AbstractPlayer attribute), 13
- batters_faced (sportsreference.mlb.teams.Team attribute), 23
- batters_struckout_per_nine_innings (sportsreference.mlb.roster.Player attribute), 15
- batting_average (sportsreference.mlb.player.AbstractPlayer attribute), 13
- batting_average (sportsreference.mlb.roster.Player attribute), 15
- batting_average (sportsreference.mlb.teams.Team attribute), 23
- birth_date (sportsreference.mlb.roster.Player attribute), 15
- birth_date (sportsreference.nba.roster.Player attribute), 39
- birth_date (sportsreference.nfl.roster.Player attribute), 104
- block_percentage (sportsreference.nba.player.AbstractPlayer attribute), 36
- block_percentage (sportsreference.ncaa.player.AbstractPlayer attribute), 58
- block_percentage (sportsreference.ncaa.teams.Team attribute), 67
- blocked_punts (sportsreference.nfl.roster.Player attribute), 104
- blocking_fouls (sportsreference.nba.roster.Player attribute), 39
- blocks (sportsreference.nba.player.AbstractPlayer attribute), 36

- blocks (*sportsreference.nba.teams.Team* attribute), 46
- blocks (*sportsreference.ncaab.player.AbstractPlayer* attribute), 58
- blocks (*sportsreference.ncaab.teams.Team* attribute), 67
- blocks_at_even_strength (*sportsreference.nhl.player.AbstractPlayer* attribute), 121
- blocks_at_even_strength (*sportsreference.nhl.roster.Player* attribute), 124
- box_plus_minus (*sportsreference.nba.player.AbstractPlayer* attribute), 36
- box_plus_minus (*sportsreference.ncaab.roster.Player* attribute), 63
- Boxscore (class in *sportsreference.mlb.boxscore*), 5
- Boxscore (class in *sportsreference.nba.boxscore*), 29
- Boxscore (class in *sportsreference.ncaab.boxscore*), 49
- Boxscore (class in *sportsreference.ncaaf.boxscore*), 73
- Boxscore (class in *sportsreference.nfl.boxscore*), 94
- Boxscore (class in *sportsreference.nhl.boxscore*), 116
- boxscore (*sportsreference.mlb.schedule.Game* attribute), 20
- boxscore (*sportsreference.nba.schedule.Game* attribute), 44
- boxscore (*sportsreference.ncaab.schedule.Game* attribute), 65
- boxscore (*sportsreference.ncaaf.schedule.Game* attribute), 88
- boxscore (*sportsreference.nfl.schedule.Game* attribute), 110
- boxscore (*sportsreference.nhl.schedule.Game* attribute), 128
- boxscore_index (*sportsreference.mlb.schedule.Game* attribute), 20
- boxscore_index (*sportsreference.nba.schedule.Game* attribute), 44
- boxscore_index (*sportsreference.ncaab.schedule.Game* attribute), 65
- boxscore_index (*sportsreference.ncaaf.schedule.Game* attribute), 88
- boxscore_index (*sportsreference.nfl.schedule.Game* attribute), 110
- boxscore_index (*sportsreference.nhl.schedule.Game* attribute), 128
- BoxscorePlayer (class in *sportsreference.mlb.boxscore*), 9
- BoxscorePlayer (class in *sportsreference.nba.boxscore*), 34
- BoxscorePlayer (class in *sportsreference.ncaab.boxscore*), 54
- BoxscorePlayer (class in *sportsreference.ncaaf.boxscore*), 75
- BoxscorePlayer (class in *sportsreference.nfl.boxscore*), 97
- BoxscorePlayer (class in *sportsreference.nhl.boxscore*), 118
- Boxscores (class in *sportsreference.mlb.boxscore*), 11
- Boxscores (class in *sportsreference.nba.boxscore*), 35
- Boxscores (class in *sportsreference.ncaab.boxscore*), 55
- Boxscores (class in *sportsreference.ncaaf.boxscore*), 76
- Boxscores (class in *sportsreference.nfl.boxscore*), 98
- Boxscores (class in *sportsreference.nhl.boxscore*), 119
- ## C
- catch_percentage (*sportsreference.nfl.roster.Player* attribute), 104
- center_percentage (*sportsreference.nba.roster.Player* attribute), 39
- combined_tackles (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98
- complete (*sportsreference.ncaab.rankings.Rankings* attribute), 61
- complete (*sportsreference.ncaaf.rankings.Rankings* attribute), 82
- complete_game_shutouts (*sportsreference.mlb.teams.Team* attribute), 23
- complete_games (*sportsreference.mlb.roster.Player* attribute), 15
- complete_games (*sportsreference.mlb.teams.Team* attribute), 23
- completed_passes (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
- completed_passes (*sportsreference.ncaaf.roster.Player* attribute), 84
- completed_passes (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
- completed_passes (*sportsreference.nfl.roster.Player* attribute), 104
- completion_percentage_index (*sportsreference.nfl.roster.Player* attribute), 104
- Conference (class in *sportsreference.ncaab.conferences*), 57
- Conference (class in *sportsreference.ncaaf.conferences*), 78
- conference (*sportsreference.ncaab.roster.Player* attribute), 63
- conference (*sportsreference.ncaab.teams.Team* attribute), 67
- conference (*sportsreference.ncaaf.teams.Team* attribute), 90
- conference_losses (*sportsreference.ncaab.teams.Team* attribute), 67

`conference_losses` (*sportsreference.ncaaf.teams.Team* attribute), 90

`conference_win_percentage` (*sportsreference.ncaaf.teams.Team* attribute), 90

`conference_wins` (*sportsreference.ncaab.teams.Team* attribute), 67

`conference_wins` (*sportsreference.ncaaf.teams.Team* attribute), 90

`Conferences` (class in *sportsreference.ncaab.conferences*), 57

`Conferences` (class in *sportsreference.ncaaf.conferences*), 78

`conferences` (*sportsreference.ncaab.conferences.Conferences* attribute), 57

`conferences` (*sportsreference.ncaaf.conferences.Conferences* attribute), 79

`contract` (*sportsreference.mlb.roster.Player* attribute), 15

`contract` (*sportsreference.nba.roster.Player* attribute), 39

`corsi_against` (*sportsreference.nhl.roster.Player* attribute), 124

`corsi_against` (*sportsreference.nhl.schedule.Game* attribute), 128

`corsi_for` (*sportsreference.nhl.roster.Player* attribute), 124

`corsi_for` (*sportsreference.nhl.schedule.Game* attribute), 128

`corsi_for_percentage` (*sportsreference.nhl.player.AbstractPlayer* attribute), 121

`corsi_for_percentage` (*sportsreference.nhl.schedule.Game* attribute), 128

`current` (*sportsreference.ncaab.rankings.Rankings* attribute), 61

`current` (*sportsreference.ncaaf.rankings.Rankings* attribute), 83

`current_extended` (*sportsreference.ncaab.rankings.Rankings* attribute), 61

`current_extended` (*sportsreference.ncaaf.rankings.Rankings* attribute), 83

D

`dataframe` (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

`dataframe` (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

`dataframe` (*sportsreference.mlb.roster.Player* attribute), 15

`dataframe` (*sportsreference.mlb.schedule.Game* attribute), 20

`dataframe` (*sportsreference.mlb.schedule.Schedule* attribute), 22

`dataframe` (*sportsreference.mlb.teams.Team* attribute), 23

`dataframe` (*sportsreference.nba.boxscore.Boxscore* attribute), 31

`dataframe` (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 34

`dataframe` (*sportsreference.nba.roster.Player* attribute), 39

`dataframe` (*sportsreference.nba.schedule.Game* attribute), 44

`dataframe` (*sportsreference.nba.schedule.Schedule* attribute), 45

`dataframe` (*sportsreference.nba.teams.Team* attribute), 46

`dataframe` (*sportsreference.ncaab.boxscore.Boxscore* attribute), 52

`dataframe` (*sportsreference.ncaab.boxscore.BoxscorePlayer* attribute), 55

`dataframe` (*sportsreference.ncaab.roster.Player* attribute), 63

`dataframe` (*sportsreference.ncaab.schedule.Game* attribute), 65

`dataframe` (*sportsreference.ncaab.schedule.Schedule* attribute), 66

`dataframe` (*sportsreference.ncaab.teams.Team* attribute), 68

`dataframe` (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74

`dataframe` (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76

`dataframe` (*sportsreference.ncaaf.roster.Player* attribute), 84

`dataframe` (*sportsreference.ncaaf.schedule.Game* attribute), 88

`dataframe` (*sportsreference.ncaaf.schedule.Schedule* attribute), 89

`dataframe` (*sportsreference.ncaaf.teams.Team* attribute), 90

`dataframe` (*sportsreference.nfl.boxscore.Boxscore* attribute), 95

`dataframe` (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98

`dataframe` (*sportsreference.nfl.roster.Player* attribute), 104

`dataframe` (*sportsreference.nfl.schedule.Game* attribute), 110

- dataframe (*sportsreference.nfl.schedule.Schedule* attribute), 112
- dataframe (*sportsreference.nfl.teams.Team* attribute), 113
- dataframe (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- dataframe (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
- dataframe (*sportsreference.nhl.roster.Player* attribute), 124
- dataframe (*sportsreference.nhl.schedule.Game* attribute), 128
- dataframe (*sportsreference.nhl.schedule.Schedule* attribute), 130
- dataframe (*sportsreference.nhl.teams.Team* attribute), 131
- dataframe_extended (*sportsreference.mlb.schedule.Game* attribute), 20
- dataframe_extended (*sportsreference.mlb.schedule.Schedule* attribute), 22
- dataframe_extended (*sportsreference.nba.schedule.Game* attribute), 44
- dataframe_extended (*sportsreference.nba.schedule.Schedule* attribute), 45
- dataframe_extended (*sportsreference.ncaab.schedule.Game* attribute), 65
- dataframe_extended (*sportsreference.ncaab.schedule.Schedule* attribute), 66
- dataframe_extended (*sportsreference.ncaaf.schedule.Game* attribute), 88
- dataframe_extended (*sportsreference.ncaaf.schedule.Schedule* attribute), 89
- dataframe_extended (*sportsreference.nfl.schedule.Game* attribute), 110
- dataframe_extended (*sportsreference.nfl.schedule.Schedule* attribute), 112
- dataframe_extended (*sportsreference.nhl.schedule.Game* attribute), 128
- dataframe_extended (*sportsreference.nhl.schedule.Schedule* attribute), 130
- dataframes (*sportsreference.mlb.teams.Teams* attribute), 28
- dataframes (*sportsreference.nba.teams.Teams* attribute), 49
- dataframes (*sportsreference.ncaab.teams.Teams* attribute), 72
- dataframes (*sportsreference.ncaaf.teams.Teams* attribute), 93
- dataframes (*sportsreference.nfl.teams.Teams* attribute), 115
- dataframes (*sportsreference.nhl.teams.Teams* attribute), 133
- date (*sportsreference.mlb.boxscore.Boxscore* attribute), 7
- date (*sportsreference.mlb.schedule.Game* attribute), 20
- date (*sportsreference.nba.boxscore.Boxscore* attribute), 31
- date (*sportsreference.nba.schedule.Game* attribute), 44
- date (*sportsreference.ncaab.boxscore.Boxscore* attribute), 52
- date (*sportsreference.ncaab.schedule.Game* attribute), 65
- date (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74
- date (*sportsreference.ncaaf.schedule.Game* attribute), 88
- date (*sportsreference.nfl.boxscore.Boxscore* attribute), 95
- date (*sportsreference.nfl.schedule.Game* attribute), 110
- date (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- date (*sportsreference.nhl.schedule.Game* attribute), 128
- datetime (*sportsreference.mlb.schedule.Game* attribute), 20
- datetime (*sportsreference.nba.schedule.Game* attribute), 44
- datetime (*sportsreference.ncaab.schedule.Game* attribute), 65
- datetime (*sportsreference.ncaaf.schedule.Game* attribute), 88
- datetime (*sportsreference.nfl.schedule.Game* attribute), 110
- datetime (*sportsreference.nhl.schedule.Game* attribute), 128
- day (*sportsreference.nfl.schedule.Game* attribute), 110
- day_of_week (*sportsreference.ncaaf.schedule.Game* attribute), 88
- day_or_night (*sportsreference.mlb.schedule.Game* attribute), 20
- decision (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
- defensive_box_plus_minus (*sportsreference.nba.roster.Player* attribute), 40
- defensive_box_plus_minus (*sportsreference.ncaab.roster.Player* attribute), 63
- defensive_chances (*sportsreference.mlb.roster.Player* attribute), 15
- defensive_point_shares (*sportsreference.nhl.roster.Player* attribute), 124
- defensive_rating (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 34
- defensive_rating (*sportsreference.ncaab.boxscore.BoxscorePlayer* attribute), 52

- tribute), 55
- defensive_rebound_percentage (*sportsreference.nba.player.AbstractPlayer* attribute), 36
- defensive_rebound_percentage (*sportsreference.ncaab.player.AbstractPlayer* attribute), 58
- defensive_rebounds (*sportsreference.nba.player.AbstractPlayer* attribute), 36
- defensive_rebounds (*sportsreference.nba.teams.Team* attribute), 46
- defensive_rebounds (*sportsreference.ncaab.player.AbstractPlayer* attribute), 58
- defensive_rebounds (*sportsreference.ncaab.teams.Team* attribute), 68
- defensive_runs_saved_above_average (*sportsreference.mlb.roster.Player* attribute), 15
- defensive_runs_saved_above_average_per_inning (*sportsreference.mlb.roster.Player* attribute), 15
- defensive_simple_rating_system (*sportsreference.nfl.teams.Team* attribute), 113
- defensive_win_shares (*sportsreference.nba.roster.Player* attribute), 40
- defensive_win_shares (*sportsreference.ncaab.roster.Player* attribute), 63
- defensive_zone_start_percentage (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
- defensive_zone_start_percentage (*sportsreference.nhl.roster.Player* attribute), 124
- defensive_zone_starts (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
- double_plays_turned (*sportsreference.mlb.roster.Player* attribute), 15
- doubles (*sportsreference.mlb.roster.Player* attribute), 15
- doubles (*sportsreference.mlb.teams.Team* attribute), 23
- dunks (*sportsreference.nba.roster.Player* attribute), 40
- duration (*sportsreference.mlb.boxscore.Boxscore* attribute), 7
- duration (*sportsreference.nfl.boxscore.Boxscore* attribute), 96
- duration (*sportsreference.nhl.boxscore.Boxscore* attribute), 117
- E**
- earned_runs_against (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10
- earned_runs_against (*sportsreference.mlb.teams.Team* attribute), 24
- earned_runs_against_plus (*sportsreference.mlb.teams.Team* attribute), 24
- earned_runs_allowed (*sportsreference.mlb.player.AbstractPlayer* attribute), 13
- earned_runs_allowed (*sportsreference.mlb.roster.Player* attribute), 16
- effective_field_goal_percentage (*sportsreference.nba.player.AbstractPlayer* attribute), 36
- effective_field_goal_percentage (*sportsreference.ncaab.player.AbstractPlayer* attribute), 58
- effective_field_goal_percentage (*sportsreference.ncaab.teams.Team* attribute), 68
- era (*sportsreference.mlb.roster.Player* attribute), 16
- era_plus (*sportsreference.mlb.roster.Player* attribute), 16
- errors (*sportsreference.mlb.roster.Player* attribute), 16
- errors_per_inning (*sportsreference.nfl.roster.Player* attribute), 104
- even_strength_assists (*sportsreference.nhl.player.AbstractPlayer* attribute), 121
- even_strength_goals (*sportsreference.nhl.player.AbstractPlayer* attribute), 121
- even_strength_goals_allowed (*sportsreference.nhl.roster.Player* attribute), 124
- even_strength_save_percentage (*sportsreference.nhl.roster.Player* attribute), 124
- even_strength_shots_faced (*sportsreference.nhl.roster.Player* attribute), 124
- extra_inning_losses (*sportsreference.mlb.teams.Team* attribute), 24
- extra_inning_record (*sportsreference.mlb.teams.Team* attribute), 24
- extra_inning_wins (*sportsreference.mlb.teams.Team* attribute), 24
- extra_point_percentage (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
- extra_point_percentage (*sportsreference.nfl.roster.Player* attribute), 104
- extra_points_attempted (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
- extra_points_attempted (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
- extra_points_attempted (*sportsreference.nfl.roster.Player* attribute), 104
- extra_points_attempted (*sportsreference*

ence.nfl.schedule.Game attribute), 110
 extra_points_made (*sportsreference.ncaaf.player.AbstractPlayer attribute*), 80
 extra_points_made (*sportsreference.ncaaf.roster.Player attribute*), 84
 extra_points_made (*sportsreference.nfl.player.AbstractPlayer attribute*), 100
 extra_points_made (*sportsreference.nfl.roster.Player attribute*), 104
 extra_points_made (*sportsreference.nfl.schedule.Game attribute*), 110

F

faceoff_losses (*sportsreference.nhl.roster.Player attribute*), 124
 faceoff_losses (*sportsreference.nhl.schedule.Game attribute*), 128
 faceoff_percentage (*sportsreference.nhl.roster.Player attribute*), 124
 faceoff_win_percentage (*sportsreference.nhl.schedule.Game attribute*), 128
 faceoff_wins (*sportsreference.nhl.roster.Player attribute*), 124
 faceoff_wins (*sportsreference.nhl.schedule.Game attribute*), 129
 fenwick_against (*sportsreference.nhl.roster.Player attribute*), 124
 fenwick_against (*sportsreference.nhl.schedule.Game attribute*), 129
 fenwick_for (*sportsreference.nhl.roster.Player attribute*), 124
 fenwick_for (*sportsreference.nhl.schedule.Game attribute*), 129
 fenwick_for_percentage (*sportsreference.nhl.roster.Player attribute*), 124
 fenwick_for_percentage (*sportsreference.nhl.schedule.Game attribute*), 129
 field_goal_attempts (*sportsreference.nba.player.AbstractPlayer attribute*), 37
 field_goal_attempts (*sportsreference.nba.teams.Team attribute*), 46
 field_goal_attempts (*sportsreference.ncaab.player.AbstractPlayer attribute*), 58
 field_goal_attempts (*sportsreference.ncaab.teams.Team attribute*), 68
 field_goal_perc_sixteen_foot_plus_two_points (*sportsreference.nba.roster.Player attribute*), 40
 field_goal_perc_ten_to_sixteen_feet (*sportsreference.nba.roster.Player attribute*),

40
 field_goal_perc_three_to_ten_feet (*sportsreference.nba.roster.Player attribute*), 40
 field_goal_perc_zero_to_three_feet (*sportsreference.nba.roster.Player attribute*), 40
 field_goal_percentage (*sportsreference.nba.player.AbstractPlayer attribute*), 37
 field_goal_percentage (*sportsreference.nba.teams.Team attribute*), 46
 field_goal_percentage (*sportsreference.ncaab.player.AbstractPlayer attribute*), 58
 field_goal_percentage (*sportsreference.ncaab.teams.Team attribute*), 68
 field_goal_percentage (*sportsreference.ncaaf.boxscore.BoxscorePlayer attribute*), 76
 field_goal_percentage (*sportsreference.nfl.roster.Player attribute*), 104
 field_goals (*sportsreference.nba.player.AbstractPlayer attribute*), 37
 field_goals (*sportsreference.nba.teams.Team attribute*), 46
 field_goals (*sportsreference.ncaab.player.AbstractPlayer attribute*), 58
 field_goals (*sportsreference.ncaab.teams.Team attribute*), 68
 field_goals_attempted (*sportsreference.ncaaf.boxscore.BoxscorePlayer attribute*), 76
 field_goals_attempted (*sportsreference.nfl.player.AbstractPlayer attribute*), 100
 field_goals_attempted (*sportsreference.nfl.roster.Player attribute*), 104
 field_goals_attempted (*sportsreference.nfl.schedule.Game attribute*), 110
 field_goals_made (*sportsreference.ncaaf.player.AbstractPlayer attribute*), 80
 field_goals_made (*sportsreference.ncaaf.roster.Player attribute*), 84
 field_goals_made (*sportsreference.nfl.player.AbstractPlayer attribute*), 100
 field_goals_made (*sportsreference.nfl.roster.Player attribute*), 104
 field_goals_made (*sportsreference.nfl.schedule.Game attribute*), 110
 fielding_independent_pitching (*sportsrefer-*

ence.mlb.roster.Player attribute), 16

fielding_independent_pitching (*sportsreference.mlb.teams.Team* attribute), 24

fielding_percentage (*sportsreference.mlb.roster.Player* attribute), 16

fifty_plus_yard_field_goal_attempts (*sportsreference.nfl.roster.Player* attribute), 104

fifty_plus_yard_field_goals_made (*sportsreference.nfl.roster.Player* attribute), 104

first_downs (*sportsreference.ncaaf.teams.Team* attribute), 90

first_downs (*sportsreference.nfl.teams.Team* attribute), 113

first_downs_from_penalties (*sportsreference.ncaaf.teams.Team* attribute), 90

first_downs_from_penalties (*sportsreference.nfl.teams.Team* attribute), 113

fly_balls (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

fourth_down_attempts (*sportsreference.nfl.schedule.Game* attribute), 110

fourth_down_conversions (*sportsreference.nfl.schedule.Game* attribute), 110

fourth_quarter_comebacks (*sportsreference.nfl.roster.Player* attribute), 104

fourty_to_fourty_nine_yard_field_goal_attempts (*sportsreference.nfl.roster.Player* attribute), 104

fourty_to_fourty_nine_yard_field_goals_made (*sportsreference.nfl.roster.Player* attribute), 104

free_throw_attempt_rate (*sportsreference.nba.player.AbstractPlayer* attribute), 37

free_throw_attempt_rate (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59

free_throw_attempt_rate (*sportsreference.ncaab.teams.Team* attribute), 68

free_throw_attempts (*sportsreference.nba.player.AbstractPlayer* attribute), 37

free_throw_attempts (*sportsreference.nba.teams.Team* attribute), 46

free_throw_attempts (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59

free_throw_attempts (*sportsreference.ncaab.teams.Team* attribute), 68

free_throw_percentage (*sportsreference.nba.teams.Team* attribute), 46

free_throw_percentage (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59

free_throw_percentage (*sportsreference.ncaab.teams.Team* attribute), 68

free_throws (*sportsreference.nba.player.AbstractPlayer* attribute), 37

free_throws (*sportsreference.nba.teams.Team* attribute), 46

free_throws (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59

free_throws (*sportsreference.ncaab.teams.Team* attribute), 68

free_throws_per_field_goal_attempt (*sportsreference.ncaab.teams.Team* attribute), 68

fumbles (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

fumbles (*sportsreference.nfl.roster.Player* attribute), 104

fumbles (*sportsreference.nfl.teams.Team* attribute), 113

fumbles_forced (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 85

fumbles_forced (*sportsreference.ncaaf.roster.Player* attribute), 85

fumbles_forced (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

fumbles_forced (*sportsreference.nfl.roster.Player* attribute), 105

fumbles_lost (*sportsreference.ncaaf.teams.Team* attribute), 90

fumbles_lost (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98

fumbles_recovered (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80

fumbles_recovered (*sportsreference.ncaaf.roster.Player* attribute), 85

fumbles_recovered (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

fumbles_recovered (*sportsreference.nfl.roster.Player* attribute), 105

fumbles_recovered_for_touchdown (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80

fumbles_recovered_for_touchdown (*sportsref-*

erence.ncaaf.roster.Player attribute), 85
fumbles_recovered_for_touchdown (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
fumbles_recovered_for_touchdown (*sportsreference.nfl.roster.Player* attribute), 105

G

Game (class in *sportsreference.mlb.schedule*), 20
Game (class in *sportsreference.nba.schedule*), 43
Game (class in *sportsreference.ncaab.schedule*), 64
Game (class in *sportsreference.ncaaf.schedule*), 87
Game (class in *sportsreference.nfl.schedule*), 109
Game (class in *sportsreference.nhl.schedule*), 128
game (*sportsreference.mlb.schedule.Game* attribute), 21
game (*sportsreference.nba.schedule.Game* attribute), 44
game (*sportsreference.ncaab.schedule.Game* attribute), 65
game (*sportsreference.ncaaf.schedule.Game* attribute), 88
game (*sportsreference.nhl.schedule.Game* attribute), 129
game_duration (*sportsreference.mlb.schedule.Game* attribute), 21
game_number_for_day (*sportsreference.mlb.schedule.Game* attribute), 21
game_score (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10
game_winning_drives (*sportsreference.nfl.roster.Player* attribute), 105
game_winning_goals (*sportsreference.nhl.player.AbstractPlayer* attribute), 121
games (*sportsreference.mlb.boxscore.Boxscores* attribute), 11
games (*sportsreference.mlb.roster.Player* attribute), 16
games (*sportsreference.mlb.teams.Team* attribute), 24
games (*sportsreference.nba.boxscore.Boxscores* attribute), 35
games (*sportsreference.ncaab.boxscore.Boxscores* attribute), 55
games (*sportsreference.ncaaf.boxscore.Boxscores* attribute), 77
games (*sportsreference.ncaaf.roster.Player* attribute), 85
games (*sportsreference.ncaaf.teams.Team* attribute), 90
games (*sportsreference.nfl.boxscore.Boxscores* attribute), 98
games (*sportsreference.nfl.roster.Player* attribute), 105
games (*sportsreference.nhl.boxscore.Boxscores* attribute), 120
games_behind (*sportsreference.mlb.schedule.Game* attribute), 21
games_catcher (*sportsreference.mlb.roster.Player* attribute), 16

games_center_fielder (*sportsreference.mlb.roster.Player* attribute), 16
games_designated_hitter (*sportsreference.mlb.roster.Player* attribute), 16
games_finished (*sportsreference.mlb.roster.Player* attribute), 16
games_finished (*sportsreference.mlb.teams.Team* attribute), 24
games_first_base (*sportsreference.mlb.roster.Player* attribute), 16
games_in_batting_order (*sportsreference.mlb.roster.Player* attribute), 16
games_in_defensive_lineup (*sportsreference.mlb.roster.Player* attribute), 16
games_left_fielder (*sportsreference.mlb.roster.Player* attribute), 16
games_outfielder (*sportsreference.mlb.roster.Player* attribute), 16
games_pinch_hitter (*sportsreference.mlb.roster.Player* attribute), 16
games_pinch_runner (*sportsreference.mlb.roster.Player* attribute), 16
games_pitcher (*sportsreference.mlb.roster.Player* attribute), 16
games_played (*sportsreference.nba.roster.Player* attribute), 40
games_played (*sportsreference.nba.teams.Team* attribute), 46
games_played (*sportsreference.ncaab.roster.Player* attribute), 63
games_played (*sportsreference.ncaab.teams.Team* attribute), 68
games_played (*sportsreference.nfl.teams.Team* attribute), 113
games_played (*sportsreference.nhl.roster.Player* attribute), 125
games_played (*sportsreference.nhl.teams.Team* attribute), 131
games_right_fielder (*sportsreference.mlb.roster.Player* attribute), 16
games_second_base (*sportsreference.mlb.roster.Player* attribute), 16
games_shortstop (*sportsreference.mlb.roster.Player* attribute), 17
games_started (*sportsreference.mlb.roster.Player* attribute), 17
games_started (*sportsreference.nba.roster.Player* attribute), 40
games_started (*sportsreference.ncaab.roster.Player* attribute), 63
games_started (*sportsreference.nfl.roster.Player* attribute), 105
games_third_base (*sportsreference.mlb.roster.Player* attribute), 17

giveaways (*sportsreference.nhl.roster.Player* attribute), 125

goal_against_percentage_relative (*sportsreference.nhl.roster.Player* attribute), 125

goalie_point_shares (*sportsreference.nhl.roster.Player* attribute), 125

goals (*sportsreference.nhl.player.AbstractPlayer* attribute), 121

goals_against (*sportsreference.nhl.player.AbstractPlayer* attribute), 121

goals_against (*sportsreference.nhl.teams.Team* attribute), 131

goals_against_average (*sportsreference.nhl.roster.Player* attribute), 125

goals_against_on_ice (*sportsreference.nhl.roster.Player* attribute), 125

goals_allowed (*sportsreference.nhl.schedule.Game* attribute), 129

goals_created (*sportsreference.nhl.roster.Player* attribute), 125

goals_for (*sportsreference.nhl.teams.Team* attribute), 131

goals_for_on_ice (*sportsreference.nhl.roster.Player* attribute), 125

goals_saved_above_average (*sportsreference.nhl.roster.Player* attribute), 125

goals_scored (*sportsreference.nhl.schedule.Game* attribute), 129

grounded_balls (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

grounded_into_double_plays (*sportsreference.mlb.roster.Player* attribute), 17

grounded_into_double_plays (*sportsreference.mlb.teams.Team* attribute), 24

H

half_court_heaves (*sportsreference.nba.roster.Player* attribute), 40

half_court_heaves_made (*sportsreference.nba.roster.Player* attribute), 40

height (*sportsreference.mlb.roster.Player* attribute), 17

height (*sportsreference.nba.roster.Player* attribute), 40

height (*sportsreference.ncaab.roster.Player* attribute), 63

height (*sportsreference.ncaaf.roster.Player* attribute), 85

height (*sportsreference.nfl.roster.Player* attribute), 105

height (*sportsreference.nhl.roster.Player* attribute), 125

hit_pitcher (*sportsreference.mlb.teams.Team* attribute), 24

hits (*sportsreference.mlb.player.AbstractPlayer* attribute), 13

hits (*sportsreference.mlb.roster.Player* attribute), 17

hits (*sportsreference.mlb.teams.Team* attribute), 24

hits_against_per_nine_innings (*sportsreference.mlb.roster.Player* attribute), 17

hits_allowed (*sportsreference.mlb.player.AbstractPlayer* attribute), 13

hits_allowed (*sportsreference.mlb.roster.Player* attribute), 17

hits_allowed (*sportsreference.mlb.teams.Team* attribute), 24

hits_at_even_strength (*sportsreference.nhl.player.AbstractPlayer* attribute), 121

hits_per_nine_innings (*sportsreference.mlb.teams.Team* attribute), 24

home_abbreviation (*sportsreference.nfl.boxscore.Boxscore* attribute), 96

home_assist_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31

home_assist_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 52

home_assists (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_assists (*sportsreference.nba.boxscore.Boxscore* attribute), 31

home_assists (*sportsreference.ncaab.boxscore.Boxscore* attribute), 52

home_assists (*sportsreference.nhl.boxscore.Boxscore* attribute), 117

home_at_bats (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_average_leverage_index (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_base_out_runs_added (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_base_out_runs_saved (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_bases_on_balls (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_batting_average (*sportsreference.mlb.boxscore.Boxscore* attribute), 7

home_block_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 31

home_block_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 52

home_blocks (*sportsreference.nba.boxscore.Boxscore* attribute), 31

home_blocks (*sportsreference*

<i>ence.ncaab.boxscore.Boxscore</i>	<i>attribute</i>), 52	<i>ence.nfl.boxscore.Boxscore</i>	<i>attribute</i>), 96
home_defensive_rating	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_free_throw_attempt_rate	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_defensive_rating	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_free_throw_attempt_rate	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52
home_defensive_rebound_percentage	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_free_throw_attempts	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_defensive_rebound_percentage	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_free_throw_attempts	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_defensive_rebounds	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_free_throw_percentage	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_defensive_rebounds	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_free_throw_percentage	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_earned_runs	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 7	home_free_throws	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_effective_field_goal_percentage	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_free_throws	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_effective_field_goal_percentage	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_fumbles	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
home_even_strength_assists	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 117	home_fumbles	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_even_strength_goals	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 117	home_fumbles_lost	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
home_field_goal_attempts	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_fumbles_lost	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_field_goal_attempts	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_game_score	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 7
home_field_goal_percentage	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_game_winning_goals	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 117
home_field_goal_percentage	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_goals	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_field_goals	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_grounded_balls	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_field_goals	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 52	home_hits	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_first_downs	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74	home_home_runs	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_first_downs	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_inherited_runners	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_fly_balls	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 7	home_inherited_score	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_fourth_down_attempts	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_innings_pitched	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_fourth_down_conversions	(<i>sportsreference</i>	home_interceptions	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
		home_interceptions	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96

home_line_drives	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8	74	
home_losses	(<i>sportsreference.mlb.teams.Team</i> attribute), 24	home_pass_yards	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_losses	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_penalties	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
home_losses	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53	home_penalties	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_losses	(<i>sportsreference.ncaab.teams.Team</i> attribute), 68	home_penalties_in_minutes	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_minutes_played	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_personal_fouls	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_minutes_played	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53	home_personal_fouls	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_net_pass_yards	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_pitches	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_offensive_rating	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_plate_appearances	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_offensive_rating	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53	home_players	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8
home_offensive_rebound_percentage	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_players	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_offensive_rebound_percentage	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53	home_players	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_offensive_rebounds	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32	home_players	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
home_offensive_rebounds	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53	home_players	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_on_base_percentage	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8	home_players	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_on_base_plus	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8	home_points	(<i>sportsreference.nba.boxscore.Boxscore</i> attribute), 32
home_pass_attempts	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74	home_points	(<i>sportsreference.ncaab.boxscore.Boxscore</i> attribute), 53
home_pass_attempts	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_points	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74
home_pass_completions	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74	home_points	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96
home_pass_completions	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_points	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_pass_touchdowns	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute), 74	home_power_play_assists	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_pass_touchdowns	(<i>sportsreference.nfl.boxscore.Boxscore</i> attribute), 96	home_power_play_goals	(<i>sportsreference.nhl.boxscore.Boxscore</i> attribute), 118
home_pass_yards	(<i>sportsreference.ncaaf.boxscore.Boxscore</i> attribute),	home_putouts	(<i>sportsreference.mlb.boxscore.Boxscore</i> attribute), 8

home_rbi (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8	ence.nba.boxscore.Boxscore attribute), 33
home_record (<i>sportsreference.mlb.teams.Team attribute</i>), 24	home_steal_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_runs (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8	home_steals (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_runs (<i>sportsreference.mlb.roster.Player attribute</i>), 17	home_steals (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_runs (<i>sportsreference.mlb.teams.Team attribute</i>), 24	home_strikeouts (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8
home_runs_against (<i>sportsreference.mlb.teams.Team attribute</i>), 24	home_strikes (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8
home_runs_against_per_nine_innings (<i>sportsreference.mlb.roster.Player attribute</i>), 17	home_strikes_by_contact (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8
home_runs_allowed (<i>sportsreference.mlb.roster.Player attribute</i>), 17	home_strikes_looking (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8
home_runs_per_nine_innings (<i>sportsreference.mlb.teams.Team attribute</i>), 24	home_strikes_swinging (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8
home_runs_thrown (<i>sportsreference.mlb.boxscore.BoxscorePlayer attribute</i>), 10	home_third_down_attempts (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 96
home_rush_attempts (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 74	home_third_down_conversions (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 96
home_rush_attempts (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 96	home_three_point_attempt_rate (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_rush_touchdowns (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 74	home_three_point_attempt_rate (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_rush_touchdowns (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 96	home_three_point_field_goal_attempts (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_rush_yards (<i>sportsreference.ncaaf.boxscore.Boxscore attribute</i>), 74	home_three_point_field_goal_attempts (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_rush_yards (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 96	home_three_point_field_goal_percentage (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_save_percentage (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_three_point_field_goal_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_saves (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_three_point_field_goals (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_shooting_percentage (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_three_point_field_goals (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_short_handed_assists (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_time_of_possession (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 97
home_short_handed_goals (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_times_sacked (<i>sportsreference.nfl.boxscore.Boxscore attribute</i>), 97
home_shots_on_goal (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_total_rebound_percentage (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_shutout (<i>sportsreference.nhl.boxscore.Boxscore attribute</i>), 118	home_total_rebound_percentage (<i>sportsreference.ncaab.boxscore.Boxscore attribute</i>), 53
home_slugging_percentage (<i>sportsreference.mlb.boxscore.Boxscore attribute</i>), 8	home_total_rebounds (<i>sportsreference.nba.boxscore.Boxscore attribute</i>), 33
home_steal_percentage (<i>sportsrefer-</i>	

home_total_rebounds (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_total_yards (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74

home_total_yards (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

home_true_shooting_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_true_shooting_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_turnover_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_turnover_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_turnovers (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_turnovers (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_turnovers (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 74

home_turnovers (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

home_two_point_field_goal_attempts (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_two_point_field_goal_attempts (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_two_point_field_goal_percentage (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_two_point_field_goal_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_two_point_field_goals (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_two_point_field_goals (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_unknown_bat_type (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

home_win_percentage (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_win_probability_added (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

home_win_probability_by_pitcher (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

home_win_probability_for_offensive_player (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

home_win_probability_subtracted (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

home_wins (*sportsreference.mlb.teams.Team* attribute), 24

home_wins (*sportsreference.nba.boxscore.Boxscore* attribute), 33

home_wins (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

home_wins (*sportsreference.ncaab.teams.Team* attribute), 68

home_yards_from_penalties (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75

home_yards_from_penalties (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

home_yards_lost_from_sacks (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

I

individual_corsi_for_events (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119

inherited_runners (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

inherited_score (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

innings (*sportsreference.mlb.schedule.Game* attribute), 21

innings_pitched (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10

innings_pitched (*sportsreference.mlb.teams.Team* attribute), 24

innings_played (*sportsreference.mlb.roster.Player* attribute), 17

intentional_bases_on_balls (*sportsreference.mlb.roster.Player* attribute), 17

intentional_bases_on_balls (*sportsreference.mlb.teams.Team* attribute), 25

intentional_bases_on_balls_given (*sportsreference.mlb.roster.Player* attribute), 17

interception_percentage (*sportsreference.nfl.roster.Player* attribute), 105

interception_percentage_index (*sportsreference.nfl.roster.Player* attribute), 105

interceptions (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80

interceptions (*sportsreference.ncaaf.roster.Player* attribute), 85

- [interceptions](#) (*sportsreference.ncaafootball.teams.Team* attribute), 91
[interceptions](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[interceptions](#) (*sportsreference.nfl.roster.Player* attribute), 105
[interceptions](#) (*sportsreference.nfl.schedule.Game* attribute), 110
[interceptions](#) (*sportsreference.nfl.teams.Team* attribute), 113
[interceptions_returned_for_touchdown](#) (*sportsreference.ncaafootball.player.AbstractPlayer* attribute), 80
[interceptions_returned_for_touchdown](#) (*sportsreference.ncaafootball.roster.Player* attribute), 85
[interceptions_returned_for_touchdown](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[interceptions_returned_for_touchdown](#) (*sportsreference.nfl.roster.Player* attribute), 105
[interceptions_thrown](#) (*sportsreference.ncaafootball.player.AbstractPlayer* attribute), 80
[interceptions_thrown](#) (*sportsreference.ncaafootball.roster.Player* attribute), 85
[interceptions_thrown](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[interceptions_thrown](#) (*sportsreference.nfl.roster.Player* attribute), 105
[interleague_record](#) (*sportsreference.mlb.teams.Team* attribute), 25
- ## K
- [kickoff_return_touchdown](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[kickoff_return_touchdown](#) (*sportsreference.nfl.roster.Player* attribute), 105
[kickoff_return_touchdowns](#) (*sportsreference.ncaafootball.player.AbstractPlayer* attribute), 80
[kickoff_return_touchdowns](#) (*sportsreference.ncaafootball.roster.Player* attribute), 85
[kickoff_return_yards](#) (*sportsreference.ncaafootball.boxscore.BoxscorePlayer* attribute), 76
[kickoff_return_yards](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[kickoff_return_yards](#) (*sportsreference.nfl.roster.Player* attribute), 105
[kickoff_returns](#) (*sportsreference.ncaafootball.boxscore.BoxscorePlayer* attribute), 76
[kickoff_returns](#) (*sportsreference.nfl.player.AbstractPlayer* attribute), 100
[kickoff_returns](#) (*sportsreference.nfl.roster.Player* attribute), 105
- ## L
- [last_ten_games_record](#) (*sportsreference.mlb.teams.Team* attribute), 25
[last_thirty_games_record](#) (*sportsreference.mlb.teams.Team* attribute), 25
[last_twenty_games_record](#) (*sportsreference.mlb.teams.Team* attribute), 25
[league](#) (*sportsreference.mlb.teams.Team* attribute), 25
[league](#) (*sportsreference.nhl.roster.Player* attribute), 125
[league_fielding_percentage](#) (*sportsreference.mlb.roster.Player* attribute), 17
[league_range_factor_per_game](#) (*sportsreference.mlb.roster.Player* attribute), 17
[league_range_factor_per_nine_innings](#) (*sportsreference.mlb.roster.Player* attribute), 17
[less_than_nineteen_yards_field_goal_attempts](#) (*sportsreference.nfl.roster.Player* attribute), 105
[less_than_nineteen_yards_field_goals_made](#) (*sportsreference.nfl.roster.Player* attribute), 105
[line_drives](#) (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 10
[location](#) (*sportsreference.mlb.schedule.Game* attribute), 21
[location](#) (*sportsreference.nba.boxscore.Boxscore* attribute), 33
[location](#) (*sportsreference.nba.schedule.Game* attribute), 44
[location](#) (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54
[location](#) (*sportsreference.ncaab.schedule.Game* attribute), 65
[location](#) (*sportsreference.ncaafootball.schedule.Game* attribute), 88
[location](#) (*sportsreference.nfl.schedule.Game* attribute), 110
[location](#) (*sportsreference.nhl.schedule.Game* attribute), 129

`longest_field_goal_made` (*sportsreference.nfl.roster.Player* attribute), 105

`longest_interception_return` (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

`longest_interception_return` (*sportsreference.nfl.roster.Player* attribute), 105

`longest_kickoff_return` (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

`longest_kickoff_return` (*sportsreference.nfl.roster.Player* attribute), 105

`longest_pass` (*sportsreference.nfl.player.AbstractPlayer* attribute), 100

`longest_pass` (*sportsreference.nfl.roster.Player* attribute), 106

`longest_punt` (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

`longest_punt` (*sportsreference.nfl.roster.Player* attribute), 106

`longest_punt_return` (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

`longest_punt_return` (*sportsreference.nfl.roster.Player* attribute), 106

`longest_reception` (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

`longest_reception` (*sportsreference.nfl.roster.Player* attribute), 106

`longest_rush` (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

`longest_rush` (*sportsreference.nfl.roster.Player* attribute), 106

`loser` (*sportsreference.mlb.schedule.Game* attribute), 21

`losing_abbr` (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

`losing_abbr` (*sportsreference.nba.boxscore.Boxscore* attribute), 33

`losing_abbr` (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

`losing_abbr` (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75

`losing_abbr` (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

`losing_abbr` (*sportsreference.nhl.boxscore.Boxscore* attribute), 118

`losing_name` (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

`losing_name` (*sportsreference.nba.boxscore.Boxscore* attribute), 33

`losing_name` (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54

`losing_name` (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75

`losing_name` (*sportsreference.nfl.boxscore.Boxscore* attribute), 97

`losing_name` (*sportsreference.nhl.boxscore.Boxscore* attribute), 118

`losses` (*sportsreference.mlb.roster.Player* attribute), 17

`losses` (*sportsreference.mlb.teams.Team* attribute), 25

`losses` (*sportsreference.nba.schedule.Game* attribute), 44

`losses` (*sportsreference.ncaab.teams.Team* attribute), 68

`losses` (*sportsreference.ncaaf.schedule.Game* attribute), 88

`losses` (*sportsreference.ncaaf.teams.Team* attribute), 91

`losses` (*sportsreference.nfl.teams.Team* attribute), 113

`losses` (*sportsreference.nhl.roster.Player* attribute), 125

`losses` (*sportsreference.nhl.teams.Team* attribute), 131

`losses_last_ten_games` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_last_thirty_games` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_last_twenty_games` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_vs_left_handed_pitchers` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_vs_right_handed_pitchers` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_vs_teams_over_500` (*sportsreference.mlb.teams.Team* attribute), 25

`losses_vs_teams_under_500` (*sportsreference.mlb.teams.Team* attribute), 25

`lost_ball_turnovers` (*sportsreference.nba.roster.Player* attribute), 40

`luck` (*sportsreference.mlb.teams.Team* attribute), 25

M

`margin_of_victory` (*sportsreference.nfl.teams.Team* attribute), 113

`minutes` (*sportsreference.nhl.roster.Player* attribute), 125

`minutes_played` (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 34

minutes_played (sportsreference.nba.player.AbstractPlayer attribute), 37

minutes_played (sportsreference.nba.teams.Team attribute), 46

minutes_played (sportsreference.ncaab.player.AbstractPlayer attribute), 59

minutes_played (sportsreference.ncaab.teams.Team attribute), 68

mlb_int_property_decorator() (in module sportsreference.mlb.teams), 28

N

name (sportsreference.mlb.player.AbstractPlayer attribute), 13

name (sportsreference.mlb.roster.Player attribute), 17

name (sportsreference.mlb.teams.Team attribute), 25

name (sportsreference.nba.player.AbstractPlayer attribute), 37

name (sportsreference.nba.teams.Team attribute), 47

name (sportsreference.ncaab.player.AbstractPlayer attribute), 59

name (sportsreference.ncaab.teams.Team attribute), 68

name (sportsreference.ncaaf.player.AbstractPlayer attribute), 80

name (sportsreference.ncaaf.teams.Team attribute), 91

name (sportsreference.nfl.player.AbstractPlayer attribute), 101

name (sportsreference.nfl.teams.Team attribute), 113

name (sportsreference.nhl.player.AbstractPlayer attribute), 121

name (sportsreference.nhl.roster.Player attribute), 125

name (sportsreference.nhl.teams.Team attribute), 131

nationality (sportsreference.mlb.roster.Player attribute), 17

nationality (sportsreference.nba.roster.Player attribute), 40

ncaaf_int_property_sub_index() (in module sportsreference.ncaaf.boxscore), 77

net_plus_minus (sportsreference.nba.roster.Player attribute), 40

net_rating (sportsreference.ncaab.teams.Team attribute), 68

net_yards_per_attempt_index (sportsreference.nfl.roster.Player attribute), 106

net_yards_per_pass_attempt (sportsreference.nfl.roster.Player attribute), 106

nfl_int_property_sub_index() (in module sportsreference.nfl.boxscore), 99

nhl_int_property_decorator() (in module sportsreference.nhl.boxscore), 120

number_of_pitchers (sportsreference.mlb.teams.Team attribute), 25

number_players_used (sportsreference.mlb.teams.Team attribute), 25

O

offensive_box_plus_minus (sportsreference.nba.roster.Player attribute), 40

offensive_box_plus_minus (sportsreference.ncaab.roster.Player attribute), 63

offensive_fouls (sportsreference.nba.roster.Player attribute), 40

offensive_point_shares (sportsreference.nhl.roster.Player attribute), 125

offensive_rating (sportsreference.nba.boxscore.BoxscorePlayer attribute), 34

offensive_rating (sportsreference.ncaab.boxscore.BoxscorePlayer attribute), 55

offensive_rating (sportsreference.ncaab.teams.Team attribute), 68

offensive_rebound_percentage (sportsreference.nba.player.AbstractPlayer attribute), 37

offensive_rebound_percentage (sportsreference.ncaab.player.AbstractPlayer attribute), 59

offensive_rebound_percentage (sportsreference.ncaab.teams.Team attribute), 69

offensive_rebounds (sportsreference.nba.player.AbstractPlayer attribute), 37

offensive_rebounds (sportsreference.nba.teams.Team attribute), 47

offensive_rebounds (sportsreference.ncaab.player.AbstractPlayer attribute), 59

offensive_rebounds (sportsreference.ncaab.teams.Team attribute), 69

offensive_simple_rating_system (sportsreference.nfl.teams.Team attribute), 113

offensive_win_shares (sportsreference.nba.roster.Player attribute), 40

offensive_win_shares (sportsreference.ncaab.roster.Player attribute), 63

offensive_zone_start_percentage (sportsreference.nhl.player.AbstractPlayer attribute), 121

offensive_zone_start_percentage (sportsreference.nhl.schedule.Game attribute), 129

offensive_zone_starts (sportsreference.nhl.boxscore.BoxscorePlayer attribute), 119

on_base_percentage (sportsreference.mlb.player.AbstractPlayer attribute),

13
on_base_percentage (*sportsreference.mlb.roster.Player* attribute), 18
on_base_percentage (*sportsreference.mlb.teams.Team* attribute), 25
on_base_plus_slugging_percentage (*sportsreference.mlb.player.AbstractPlayer* attribute), 13
on_base_plus_slugging_percentage (*sportsreference.mlb.roster.Player* attribute), 18
on_base_plus_slugging_percentage (*sportsreference.mlb.teams.Team* attribute), 25
on_base_plus_slugging_percentage_plus (*sportsreference.mlb.roster.Player* attribute), 18
on_base_plus_slugging_percentage_plus (*sportsreference.mlb.teams.Team* attribute), 25
on_court_plus_minus (*sportsreference.nba.roster.Player* attribute), 41
on_ice_shot_attempts_against (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
on_ice_shot_attempts_for (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
opp_assist_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_assists (*sportsreference.nba.teams.Team* attribute), 47
opp_assists (*sportsreference.ncaab.teams.Team* attribute), 69
opp_block_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_blocks (*sportsreference.nba.teams.Team* attribute), 47
opp_blocks (*sportsreference.ncaab.teams.Team* attribute), 69
opp_defensive_rebounds (*sportsreference.nba.teams.Team* attribute), 47
opp_defensive_rebounds (*sportsreference.ncaab.teams.Team* attribute), 69
opp_effective_field_goal_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_field_goal_attempts (*sportsreference.nba.teams.Team* attribute), 47
opp_field_goal_attempts (*sportsreference.ncaab.teams.Team* attribute), 69
opp_field_goal_percentage (*sportsreference.nba.teams.Team* attribute), 47
opp_field_goal_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_field_goals (*sportsreference.nba.teams.Team* attribute), 47
opp_field_goals (*sportsreference.ncaab.teams.Team* attribute), 69
opp_free_throw_attempt_rate (*sportsreference.ncaab.teams.Team* attribute), 69
opp_free_throw_attempts (*sportsreference.nba.teams.Team* attribute), 47
opp_free_throw_attempts (*sportsreference.ncaab.teams.Team* attribute), 69
opp_free_throw_percentage (*sportsreference.nba.teams.Team* attribute), 47
opp_free_throw_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_free_throws (*sportsreference.nba.teams.Team* attribute), 47
opp_free_throws (*sportsreference.ncaab.teams.Team* attribute), 69
opp_free_throws_per_field_goal_attempt (*sportsreference.ncaab.teams.Team* attribute), 69
opp_offensive_rating (*sportsreference.ncaab.teams.Team* attribute), 69
opp_offensive_rebound_percentage (*sportsreference.ncaab.teams.Team* attribute), 69
opp_offensive_rebounds (*sportsreference.nba.teams.Team* attribute), 47
opp_offensive_rebounds (*sportsreference.ncaab.teams.Team* attribute), 70
opp_penalties_in_minutes (*sportsreference.nhl.schedule.Game* attribute), 129
opp_personal_fouls (*sportsreference.nba.teams.Team* attribute), 47
opp_personal_fouls (*sportsreference.ncaab.teams.Team* attribute), 70
opp_points (*sportsreference.nba.teams.Team* attribute), 47
opp_points (*sportsreference.ncaab.teams.Team* attribute), 70
opp_power_play_goals (*sportsreference.nhl.schedule.Game* attribute), 129
opp_power_play_opportunities (*sportsreference.nhl.schedule.Game* attribute), 129
opp_short_handed_goals (*sportsreference.nhl.schedule.Game* attribute), 129
opp_shots_on_goal (*sportsreference.nhl.schedule.Game* attribute), 129
opp_steal_percentage (*sportsreference.ncaab.teams.Team* attribute), 70
opp_steals (*sportsreference.nba.teams.Team* attribute), 47
opp_steals (*sportsreference.ncaab.teams.Team* attribute), 70
opp_three_point_attempt_rate (*sportsreference.ncaab.teams.Team* attribute), 70
opp_three_point_field_goal_attempts

(sportsreference.nba.teams.Team attribute), 47
 opp_three_point_field_goal_attempts
(sportsreference.ncaab.teams.Team attribute), 70
 opp_three_point_field_goal_percentage
(sportsreference.nba.teams.Team attribute), 47
 opp_three_point_field_goal_percentage
(sportsreference.ncaab.teams.Team attribute), 70
 opp_three_point_field_goals *(sportsreference.nba.teams.Team attribute)*, 47
 opp_three_point_field_goals *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_total_rebound_percentage *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_total_rebounds *(sportsreference.nba.teams.Team attribute)*, 47
 opp_total_rebounds *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_true_shooting_percentage *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_turnover_percentage *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_turnovers *(sportsreference.nba.teams.Team attribute)*, 47
 opp_turnovers *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_two_point_field_goal_attempts *(sportsreference.nba.teams.Team attribute)*, 48
 opp_two_point_field_goal_attempts *(sportsreference.ncaab.teams.Team attribute)*, 70
 opp_two_point_field_goal_percentage
(sportsreference.nba.teams.Team attribute), 48
 opp_two_point_field_goal_percentage
(sportsreference.ncaab.teams.Team attribute), 70
 opp_two_point_field_goals *(sportsreference.nba.teams.Team attribute)*, 48
 opp_two_point_field_goals *(sportsreference.ncaab.teams.Team attribute)*, 70
 opponent_abbr *(sportsreference.mlb.schedule.Game attribute)*, 21
 opponent_abbr *(sportsreference.nba.schedule.Game attribute)*, 44
 opponent_abbr *(sportsreference.ncaab.schedule.Game attribute)*, 65
 opponent_abbr *(sportsreference.ncaaf.schedule.Game attribute)*, 88
 opponent_abbr *(sportsreference.nfl.schedule.Game attribute)*, 110
 opponent_abbr *(sportsreference.nhl.schedule.Game attribute)*, 129
 opponent_conference *(sportsreference.ncaab.schedule.Game attribute)*, 65
 opponent_conference *(sportsreference.ncaaf.schedule.Game attribute)*, 88
 opponent_name *(sportsreference.nba.schedule.Game attribute)*, 44
 opponent_name *(sportsreference.ncaab.schedule.Game attribute)*, 65
 opponent_name *(sportsreference.ncaaf.schedule.Game attribute)*, 88
 opponent_name *(sportsreference.nfl.schedule.Game attribute)*, 110
 opponent_name *(sportsreference.nhl.schedule.Game attribute)*, 129
 opponent_rank *(sportsreference.ncaab.schedule.Game attribute)*, 65
 opponent_rank *(sportsreference.ncaaf.schedule.Game attribute)*, 88
 opponents_first_downs *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_first_downs_from_penalties
(sportsreference.ncaaf.teams.Team attribute), 91
 opponents_fumbles_lost *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_interceptions *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_pass_attempts *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_pass_completion_percentage
(sportsreference.ncaaf.teams.Team attribute), 91
 opponents_pass_completions *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_pass_first_downs *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_pass_touchdowns *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_pass_yards *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_penalties *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_plays *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_rush_attempts *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_rush_first_downs *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_rush_touchdowns *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_rush_yards *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_rush_yards_per_attempt *(sportsreference.ncaaf.teams.Team attribute)*, 91
 opponents_turnovers *(sportsreference.ncaaf.teams.Team attribute)*, 91

opponents_yards (*sportsreference.ncaaf.teams.Team* attribute), 92
 opponents_yards_from_penalties (*sportsreference.ncaaf.teams.Team* attribute), 92
 opponents_yards_per_play (*sportsreference.ncaaf.teams.Team* attribute), 92
 opposing_runners_left_on_base (*sportsreference.mlb.teams.Team* attribute), 26
 other_touchdowns (*sportsreference.ncaaf.roster.Player* attribute), 85
 other_turnovers (*sportsreference.nba.roster.Player* attribute), 41
 overtime (*sportsreference.nfl.schedule.Game* attribute), 110
 overtime (*sportsreference.nhl.schedule.Game* attribute), 129
 overtime_losses (*sportsreference.nhl.teams.Team* attribute), 132
 overtimes (*sportsreference.ncaab.schedule.Game* attribute), 65

P

pace (*sportsreference.nba.boxscore.Boxscore* attribute), 33
 pace (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54
 pace (*sportsreference.ncaab.teams.Team* attribute), 70
 pass_attempts (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 pass_attempts (*sportsreference.ncaaf.roster.Player* attribute), 85
 pass_attempts (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_attempts (*sportsreference.nfl.schedule.Game* attribute), 110
 pass_attempts (*sportsreference.nfl.teams.Team* attribute), 114
 pass_completion_percentage (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_completion_rate (*sportsreference.nfl.schedule.Game* attribute), 111
 pass_completions (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_completions (*sportsreference.nfl.schedule.Game* attribute), 111
 pass_completions (*sportsreference.nfl.teams.Team* attribute), 114
 pass_first_downs (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_first_downs (*sportsreference.nfl.teams.Team* attribute), 114
 pass_net_yards_per_attempt (*sportsreference.nfl.teams.Team* attribute), 114
 pass_touchdowns (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_touchdowns (*sportsreference.nfl.schedule.Game* attribute), 111
 pass_touchdowns (*sportsreference.nfl.teams.Team* attribute), 114
 pass_yards (*sportsreference.ncaaf.teams.Team* attribute), 92
 pass_yards (*sportsreference.nfl.schedule.Game* attribute), 111
 pass_yards (*sportsreference.nfl.teams.Team* attribute), 114
 pass_yards_per_attempt (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 pass_yards_per_attempt (*sportsreference.nfl.schedule.Game* attribute), 111
 passer_rating_index (*sportsreference.nfl.roster.Player* attribute), 106
 passes_defended (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 passes_defended (*sportsreference.ncaaf.roster.Player* attribute), 85
 passes_defended (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 passes_defended (*sportsreference.nfl.roster.Player* attribute), 106
 passing_completion (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 passing_completion (*sportsreference.ncaaf.roster.Player* attribute), 85
 passing_completion (*sportsreference.nfl.roster.Player* attribute), 106
 passing_touchdown_percentage (*sportsreference.nfl.roster.Player* attribute), 106
 passing_touchdowns (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 passing_touchdowns (*sportsreference.ncaaf.roster.Player* attribute), 85
 passing_touchdowns (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 passing_touchdowns (*sportsreference.nfl.roster.Player* attribute), 106
 passing_turnovers (*sportsreference.nba.roster.Player* attribute), 41
 passing_yards (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 passing_yards (*sportsreference.ncaaf.roster.Player*

attribute), 85
 passing_yards (sportsreference.nfl.player.AbstractPlayer attribute), 101
 passing_yards (sportsreference.nfl.roster.Player attribute), 106
 passing_yards_per_attempt (sportsreference.ncaaf.player.AbstractPlayer attribute), 80
 passing_yards_per_attempt (sportsreference.nfl.roster.Player attribute), 106
 pdo (sportsreference.nhl.roster.Player attribute), 125
 pdo (sportsreference.nhl.schedule.Game attribute), 129
 pdo_at_even_strength (sportsreference.nhl.teams.Team attribute), 132
 penalties (sportsreference.ncaaf.teams.Team attribute), 92
 penalties (sportsreference.nfl.teams.Team attribute), 114
 penalties_in_minutes (sportsreference.nhl.player.AbstractPlayer attribute), 122
 penalties_in_minutes (sportsreference.nhl.schedule.Game attribute), 130
 penalty_killing_percentage (sportsreference.nhl.teams.Team attribute), 132
 percent_drives_with_points (sportsreference.nfl.teams.Team attribute), 114
 percent_drives_with_turnovers (sportsreference.nfl.teams.Team attribute), 114
 percentage_field_goals_as_dunks (sportsreference.nba.roster.Player attribute), 41
 percentage_of_three_pointers_from_corner (sportsreference.nba.roster.Player attribute), 41
 percentage_shots_three_pointers (sportsreference.nba.roster.Player attribute), 41
 percentage_shots_two_pointers (sportsreference.nba.roster.Player attribute), 41
 percentage_sixteen_foot_plus_two_pointers (sportsreference.nba.roster.Player attribute), 41
 percentage_ten_to_sixteen_footers (sportsreference.nba.roster.Player attribute), 41
 percentage_three_to_ten_footers (sportsreference.nba.roster.Player attribute), 41
 percentage_zero_to_three_footers (sportsreference.nba.roster.Player attribute), 41
 personal_fouls (sportsreference.nba.player.AbstractPlayer attribute), 37
 personal_fouls (sportsreference.nba.teams.Team attribute), 48
 personal_fouls (sportsreference.ncaab.player.AbstractPlayer attribute), 59
 personal_fouls (sportsreference.ncaab.teams.Team attribute), 70
 pitches_thrown (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 10
 plate_appearances (sportsreference.mlb.player.AbstractPlayer attribute), 13
 plate_appearances (sportsreference.mlb.roster.Player attribute), 18
 plate_appearances (sportsreference.mlb.teams.Team attribute), 26
 Player (class in sportsreference.mlb.roster), 14
 Player (class in sportsreference.nba.roster), 39
 Player (class in sportsreference.ncaab.roster), 62
 Player (class in sportsreference.ncaaf.roster), 84
 Player (class in sportsreference.nfl.roster), 103
 Player (class in sportsreference.nhl.roster), 123
 player_efficiency_rating (sportsreference.nba.roster.Player attribute), 41
 player_efficiency_rating (sportsreference.ncaab.roster.Player attribute), 63
 player_id (sportsreference.mlb.player.AbstractPlayer attribute), 13
 player_id (sportsreference.nba.player.AbstractPlayer attribute), 37
 player_id (sportsreference.ncaab.player.AbstractPlayer attribute), 59
 player_id (sportsreference.ncaaf.player.AbstractPlayer attribute), 80
 player_id (sportsreference.nfl.player.AbstractPlayer attribute), 101
 player_id (sportsreference.nhl.player.AbstractPlayer attribute), 122
 players (sportsreference.mlb.roster.Roster attribute), 20
 players (sportsreference.nba.roster.Roster attribute), 43
 players (sportsreference.ncaab.roster.Roster attribute), 64
 players (sportsreference.ncaaf.roster.Roster attribute), 87
 players (sportsreference.nfl.roster.Roster attribute), 109
 players (sportsreference.nhl.roster.Roster attribute), 127
 playoff_round (sportsreference.nhl.boxscore.Boxscore attribute), 118
 playoffs (sportsreference.nba.schedule.Game attribute), 44

plays (*sportsreference.ncaaf.teams.Team* attribute), 92
plays (*sportsreference.nfl.teams.Team* attribute), 114
plays_from_scrimmage (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
plays_from_scrimmage (*sportsreference.ncaaf.roster.Player* attribute), 85
plus_minus (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
point_guard_percentage (*sportsreference.nba.roster.Player* attribute), 41
point_shares (*sportsreference.nhl.roster.Player* attribute), 125
points (*sportsreference.nba.player.AbstractPlayer* attribute), 37
points (*sportsreference.nba.teams.Team* attribute), 48
points (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
points (*sportsreference.ncaab.teams.Team* attribute), 71
points (*sportsreference.ncaaf.roster.Player* attribute), 85
points (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
points (*sportsreference.nhl.teams.Team* attribute), 132
points_against (*sportsreference.ncaab.schedule.Game* attribute), 65
points_against (*sportsreference.ncaaf.schedule.Game* attribute), 88
points_against (*sportsreference.nfl.teams.Team* attribute), 114
points_against_per_game (*sportsreference.ncaaf.teams.Team* attribute), 92
points_allowed (*sportsreference.nba.schedule.Game* attribute), 44
points_allowed (*sportsreference.nfl.schedule.Game* attribute), 111
points_contributed_by_offense (*sportsreference.nfl.teams.Team* attribute), 114
points_difference (*sportsreference.nfl.teams.Team* attribute), 114
points_for (*sportsreference.ncaab.schedule.Game* attribute), 65
points_for (*sportsreference.ncaaf.schedule.Game* attribute), 88
points_for (*sportsreference.nfl.teams.Team* attribute), 114
points_generated_by_assists (*sportsreference.nba.roster.Player* attribute), 41
points_kicking (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
points_per_game (*sportsreference.ncaaf.teams.Team* attribute), 92
points_percentage (*sportsreference.nhl.teams.Team* attribute), 132
points_produced (*sportsreference.ncaab.roster.Player* attribute), 63
points_scored (*sportsreference.nba.schedule.Game* attribute), 44
points_scored (*sportsreference.nfl.schedule.Game* attribute), 111
position (*sportsreference.mlb.roster.Player* attribute), 18
position (*sportsreference.nba.roster.Player* attribute), 41
position (*sportsreference.ncaab.roster.Player* attribute), 63
position (*sportsreference.ncaaf.roster.Player* attribute), 85
position (*sportsreference.nfl.roster.Player* attribute), 106
power_forward_percentage (*sportsreference.nba.roster.Player* attribute), 41
power_play_assists (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
power_play_goals (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
power_play_goals (*sportsreference.nhl.schedule.Game* attribute), 130
power_play_goals (*sportsreference.nhl.teams.Team* attribute), 132
power_play_goals_against (*sportsreference.nhl.teams.Team* attribute), 132
power_play_goals_against_on_ice (*sportsreference.nhl.roster.Player* attribute), 125
power_play_goals_allowed (*sportsreference.nhl.roster.Player* attribute), 125
power_play_goals_for_on_ice (*sportsreference.nhl.roster.Player* attribute), 126
power_play_opportunities (*sportsreference.nhl.schedule.Game* attribute), 130
power_play_opportunities (*sportsreference.nhl.teams.Team* attribute), 132
power_play_opportunities_against (*sportsreference.nhl.teams.Team* attribute), 132
power_play_percentage (*sportsreference.nhl.teams.Team* attribute), 132
power_play_save_percentage (*sportsreference.nhl.roster.Player* attribute), 126
power_play_shots_faced (*sportsreference.nhl.roster.Player* attribute), 126
punt_return_touchdown (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

punt_return_touchdown (*sportsreference.nfl.roster.Player* attribute), 106
 punt_return_touchdowns (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 80
 punt_return_touchdowns (*sportsreference.ncaaf.roster.Player* attribute), 85
 punt_return_yards (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 punt_return_yards (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 punt_return_yards (*sportsreference.nfl.roster.Player* attribute), 106
 punt_returns (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 punt_returns (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 punt_returns (*sportsreference.nfl.roster.Player* attribute), 106
 punt_yards (*sportsreference.nfl.schedule.Game* attribute), 111
 punting_yards (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 punting_yards_per_attempt (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 punts (*sportsreference.ncaaf.boxscore.BoxscorePlayer* attribute), 76
 punts (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 punts (*sportsreference.nfl.roster.Player* attribute), 106
 punts (*sportsreference.nfl.schedule.Game* attribute), 111
 putouts (*sportsreference.mlb.player.AbstractPlayer* attribute), 13
 putouts (*sportsreference.mlb.roster.Player* attribute), 18
 pythagorean_win_loss (*sportsreference.mlb.teams.Team* attribute), 26

Q

qb_record (*sportsreference.nfl.roster.Player* attribute), 106
 quality_start_percentage (*sportsreference.nhl.roster.Player* attribute), 126
 quality_starts (*sportsreference.nhl.roster.Player* attribute), 126
 quarterback_hits (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute),

98
 quarterback_rating (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
 quarterback_rating (*sportsreference.ncaaf.roster.Player* attribute), 85
 quarterback_rating (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 quarterback_rating (*sportsreference.nfl.roster.Player* attribute), 107
 quarterback_rating (*sportsreference.nfl.schedule.Game* attribute), 111

R

range_factor_per_game (*sportsreference.mlb.roster.Player* attribute), 18
 range_factor_per_nine_innings (*sportsreference.mlb.roster.Player* attribute), 18
 rank (*sportsreference.mlb.schedule.Game* attribute), 21
 rank (*sportsreference.mlb.teams.Team* attribute), 26
 rank (*sportsreference.nba.teams.Team* attribute), 48
 rank (*sportsreference.ncaaf.schedule.Game* attribute), 88
 rank (*sportsreference.nfl.teams.Team* attribute), 114
 rank (*sportsreference.nhl.teams.Team* attribute), 132
 Rankings (class in *sportsreference.ncaab.rankings*), 60
 Rankings (class in *sportsreference.ncaaf.rankings*), 82
 really_bad_starts (*sportsreference.nhl.roster.Player* attribute), 126
 receiving_touchdowns (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
 receiving_touchdowns (*sportsreference.ncaaf.roster.Player* attribute), 85
 receiving_touchdowns (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 receiving_touchdowns (*sportsreference.nfl.roster.Player* attribute), 107
 receiving_yards (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
 receiving_yards (*sportsreference.ncaaf.roster.Player* attribute), 86
 receiving_yards (*sportsreference.nfl.player.AbstractPlayer* attribute), 101
 receiving_yards (*sportsreference.nfl.roster.Player* attribute), 107
 receiving_yards_per_game (*sportsreference.nfl.roster.Player* attribute), 107
 receiving_yards_per_reception (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

receiving_yards_per_reception (*sportsreference.ncaaf.roster.Player* attribute), 86

receiving_yards_per_reception (*sportsreference.nfl.roster.Player* attribute), 107

receptions (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

receptions (*sportsreference.ncaaf.roster.Player* attribute), 86

receptions (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

receptions (*sportsreference.nfl.roster.Player* attribute), 107

receptions_per_game (*sportsreference.nfl.roster.Player* attribute), 107

record (*sportsreference.mlb.schedule.Game* attribute), 21

record_vs_left_handed_pitchers (*sportsreference.mlb.teams.Team* attribute), 26

record_vs_right_handed_pitchers (*sportsreference.mlb.teams.Team* attribute), 26

record_vs_teams_over_500 (*sportsreference.mlb.teams.Team* attribute), 26

record_vs_teams_under_500 (*sportsreference.mlb.teams.Team* attribute), 26

relative_corsi_for_percentage (*sportsreference.nhl.player.AbstractPlayer* attribute), 122

relative_fenwick_for_percentage (*sportsreference.nhl.roster.Player* attribute), 126

result (*sportsreference.mlb.schedule.Game* attribute), 21

result (*sportsreference.nba.schedule.Game* attribute), 44

result (*sportsreference.ncaab.schedule.Game* attribute), 65

result (*sportsreference.ncaaf.schedule.Game* attribute), 88

result (*sportsreference.nfl.schedule.Game* attribute), 111

result (*sportsreference.nhl.schedule.Game* attribute), 130

Roster (class in *sportsreference.mlb.roster*), 19

Roster (class in *sportsreference.nba.roster*), 43

Roster (class in *sportsreference.ncaab.roster*), 64

Roster (class in *sportsreference.ncaaf.roster*), 87

Roster (class in *sportsreference.nfl.roster*), 109

Roster (class in *sportsreference.nhl.roster*), 127

roster (*sportsreference.mlb.teams.Team* attribute), 26

roster (*sportsreference.nba.teams.Team* attribute), 48

roster (*sportsreference.ncaab.teams.Team* attribute), 71

roster (*sportsreference.ncaaf.teams.Team* attribute), 92

roster (*sportsreference.nfl.teams.Team* attribute), 114

roster (*sportsreference.nhl.teams.Team* attribute), 132

run_difference (*sportsreference.mlb.teams.Team* attribute), 26

runners_left_on_base (*sportsreference.mlb.teams.Team* attribute), 26

runs (*sportsreference.mlb.player.AbstractPlayer* attribute), 13

runs (*sportsreference.mlb.roster.Player* attribute), 18

runs (*sportsreference.mlb.teams.Team* attribute), 26

runs_against (*sportsreference.mlb.teams.Team* attribute), 26

runs_allowed (*sportsreference.mlb.player.AbstractPlayer* attribute), 13

runs_allowed (*sportsreference.mlb.roster.Player* attribute), 18

runs_allowed (*sportsreference.mlb.schedule.Game* attribute), 21

runs_allowed_per_game (*sportsreference.mlb.teams.Team* attribute), 26

runs_batted_in (*sportsreference.mlb.player.AbstractPlayer* attribute), 13

runs_batted_in (*sportsreference.mlb.roster.Player* attribute), 18

runs_batted_in (*sportsreference.mlb.teams.Team* attribute), 26

runs_scored (*sportsreference.mlb.schedule.Game* attribute), 21

rush_attempts (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

rush_attempts (*sportsreference.ncaaf.roster.Player* attribute), 86

rush_attempts (*sportsreference.ncaaf.teams.Team* attribute), 92

rush_attempts (*sportsreference.nfl.player.AbstractPlayer* attribute), 101

rush_attempts (*sportsreference.nfl.roster.Player* attribute), 107

rush_attempts (*sportsreference.nfl.schedule.Game* attribute), 111

rush_attempts (*sportsreference.nfl.teams.Team* attribute), 114

rush_attempts_per_game (*sportsreference.nfl.roster.Player* attribute), 107

rush_first_downs (*sportsreference.ncaaf.teams.Team* attribute), 92

rush_first_downs (*sportsreference.nfl.teams.Team* attribute), 114

rush_touchdowns (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

rush_touchdowns (*sportsreference.ncaa.f.roster.Player attribute*), 86
 rush_touchdowns (*sportsreference.ncaa.f.teams.Team attribute*), 92
 rush_touchdowns (*sportsreference.nfl.player.AbstractPlayer attribute*), 101
 rush_touchdowns (*sportsreference.nfl.roster.Player attribute*), 107
 rush_touchdowns (*sportsreference.nfl.schedule.Game attribute*), 111
 rush_touchdowns (*sportsreference.nfl.teams.Team attribute*), 114
 rush_yards (*sportsreference.ncaa.f.player.AbstractPlayer attribute*), 81
 rush_yards (*sportsreference.ncaa.f.roster.Player attribute*), 86
 rush_yards (*sportsreference.ncaa.f.teams.Team attribute*), 92
 rush_yards (*sportsreference.nfl.player.AbstractPlayer attribute*), 101
 rush_yards (*sportsreference.nfl.roster.Player attribute*), 107
 rush_yards (*sportsreference.nfl.schedule.Game attribute*), 111
 rush_yards (*sportsreference.nfl.teams.Team attribute*), 114
 rush_yards_per_attempt (*sportsreference.ncaa.f.player.AbstractPlayer attribute*), 81
 rush_yards_per_attempt (*sportsreference.ncaa.f.roster.Player attribute*), 86
 rush_yards_per_attempt (*sportsreference.ncaa.f.teams.Team attribute*), 92
 rush_yards_per_attempt (*sportsreference.nfl.roster.Player attribute*), 107
 rush_yards_per_attempt (*sportsreference.nfl.schedule.Game attribute*), 111
 rush_yards_per_attempt (*sportsreference.nfl.teams.Team attribute*), 115
 rush_yards_per_game (*sportsreference.nfl.roster.Player attribute*), 107
 rushing_and_receiving_touchdowns (*sportsreference.ncaa.f.player.AbstractPlayer attribute*), 81
 rushing_and_receiving_touchdowns (*sportsreference.ncaa.f.roster.Player attribute*), 86
 rushing_and_receiving_touchdowns (*sportsreference.nfl.roster.Player attribute*), 107
 sack_percentage_index (*sportsreference.nfl.roster.Player attribute*), 107
 sacks (*sportsreference.ncaa.f.player.AbstractPlayer attribute*), 81
 sacks (*sportsreference.ncaa.f.roster.Player attribute*), 86
 sacks (*sportsreference.nfl.player.AbstractPlayer attribute*), 101
 sacks (*sportsreference.nfl.roster.Player attribute*), 107
 sacrifice_flies (*sportsreference.mlb.roster.Player attribute*), 18
 sacrifice_flies (*sportsreference.mlb.teams.Team attribute*), 26
 sacrifice_hits (*sportsreference.mlb.roster.Player attribute*), 18
 sacrifice_hits (*sportsreference.mlb.teams.Team attribute*), 26
 safeties (*sportsreference.ncaa.f.roster.Player attribute*), 86
 safeties (*sportsreference.nfl.roster.Player attribute*), 107
 salary (*sportsreference.nba.roster.Player attribute*), 42
 save (*sportsreference.mlb.schedule.Game attribute*), 21
 save_percentage (*sportsreference.nhl.player.AbstractPlayer attribute*), 122
 save_percentage (*sportsreference.nhl.teams.Team attribute*), 132
 save_percentage_on_ice (*sportsreference.nhl.roster.Player attribute*), 126
 saves (*sportsreference.mlb.roster.Player attribute*), 18
 saves (*sportsreference.mlb.teams.Team attribute*), 26
 saves (*sportsreference.nhl.player.AbstractPlayer attribute*), 122
 Schedule (*class in sportsreference.mlb.schedule*), 21
 Schedule (*class in sportsreference.nba.schedule*), 45
 Schedule (*class in sportsreference.ncaab.schedule*), 66
 Schedule (*class in sportsreference.ncaa.f.schedule*), 89
 Schedule (*class in sportsreference.nfl.schedule*), 112
 Schedule (*class in sportsreference.nhl.schedule*), 130
 schedule (*sportsreference.mlb.teams.Team attribute*), 26
 schedule (*sportsreference.nba.teams.Team attribute*), 48
 schedule (*sportsreference.ncaab.teams.Team attribute*), 71
 schedule (*sportsreference.ncaa.f.teams.Team attribute*), 92
 schedule (*sportsreference.nfl.teams.Team attribute*), 115
 schedule (*sportsreference.nhl.teams.Team attribute*), 132
 season (*sportsreference.mlb.roster.Player attribute*), 18
 season (*sportsreference.nba.roster.Player attribute*), 42
 season (*sportsreference.ncaab.roster.Player attribute*), 66

S

63
season (*sportsreference.ncaaf.roster.Player* attribute), 86
season (*sportsreference.nfl.roster.Player* attribute), 107
season (*sportsreference.nhl.roster.Player* attribute), 126
season_losses (*sportsreference.ncaab.schedule.Game* attribute), 65
season_wins (*sportsreference.ncaab.schedule.Game* attribute), 66
shifts (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
shooting_distance (*sportsreference.nba.roster.Player* attribute), 42
shooting_fouls (*sportsreference.nba.roster.Player* attribute), 42
shooting_fouls_drawn (*sportsreference.nba.roster.Player* attribute), 42
shooting_guard_percentage (*sportsreference.nba.roster.Player* attribute), 42
shooting_percentage (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
shooting_percentage (*sportsreference.nhl.teams.Team* attribute), 132
shooting_percentage_on_ice (*sportsreference.nhl.roster.Player* attribute), 126
shootout_attempts (*sportsreference.nhl.roster.Player* attribute), 126
shootout_goals (*sportsreference.nhl.roster.Player* attribute), 126
shootout_misses (*sportsreference.nhl.roster.Player* attribute), 126
shootout_percentage (*sportsreference.nhl.roster.Player* attribute), 126
short_handed_assists (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
short_handed_goals (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
short_handed_goals (*sportsreference.nhl.schedule.Game* attribute), 130
short_handed_goals (*sportsreference.nhl.teams.Team* attribute), 132
short_handed_goals_against (*sportsreference.nhl.teams.Team* attribute), 132
short_handed_goals_allowed (*sportsreference.nhl.roster.Player* attribute), 126
short_handed_save_percentage (*sportsreference.nhl.roster.Player* attribute), 126
short_handed_shots_faced (*sportsreference.nhl.roster.Player* attribute), 126
shots_against (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
shots_against (*sportsreference.nhl.teams.Team* attribute), 132
shots_blocked (*sportsreference.nba.roster.Player* attribute), 42
shots_on_goal (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
shots_on_goal (*sportsreference.nhl.schedule.Game* attribute), 130
shots_on_goal (*sportsreference.nhl.teams.Team* attribute), 132
shutouts (*sportsreference.mlb.roster.Player* attribute), 18
shutouts (*sportsreference.mlb.teams.Team* attribute), 27
shutouts (*sportsreference.nhl.player.AbstractPlayer* attribute), 122
simple_rating_system (*sportsreference.mlb.teams.Team* attribute), 27
simple_rating_system (*sportsreference.ncaab.teams.Team* attribute), 71
simple_rating_system (*sportsreference.ncaaf.teams.Team* attribute), 92
simple_rating_system (*sportsreference.nfl.teams.Team* attribute), 115
simple_rating_system (*sportsreference.nhl.teams.Team* attribute), 133
single_run_losses (*sportsreference.mlb.teams.Team* attribute), 27
single_run_record (*sportsreference.mlb.teams.Team* attribute), 27
single_run_wins (*sportsreference.mlb.teams.Team* attribute), 27
slugging_percentage (*sportsreference.mlb.player.AbstractPlayer* attribute), 13
slugging_percentage (*sportsreference.mlb.roster.Player* attribute), 18
slugging_percentage (*sportsreference.mlb.teams.Team* attribute), 27
small_forward_percentage (*sportsreference.nba.roster.Player* attribute), 42
solo_tackles (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
solo_tackles (*sportsreference.ncaaf.roster.Player* attribute), 86
solo_tackles (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98
sportsreference.mlb.boxscore (module), 5
sportsreference.mlb.player (module), 12

- sportsreference.mlb.roster (module), 14
- sportsreference.mlb.schedule (module), 20
- sportsreference.mlb.teams (module), 22
- sportsreference.nba.boxscore (module), 29
- sportsreference.nba.player (module), 36
- sportsreference.nba.roster (module), 39
- sportsreference.nba.schedule (module), 43
- sportsreference.nba.teams (module), 46
- sportsreference.ncaab.boxscore (module), 49
- sportsreference.ncaab.conferences (module), 57
- sportsreference.ncaab.player (module), 58
- sportsreference.ncaab.rankings (module), 60
- sportsreference.ncaab.roster (module), 62
- sportsreference.ncaab.schedule (module), 64
- sportsreference.ncaab.teams (module), 67
- sportsreference.ncaaf.boxscore (module), 73
- sportsreference.ncaaf.conferences (module), 78
- sportsreference.ncaaf.player (module), 79
- sportsreference.ncaaf.rankings (module), 82
- sportsreference.ncaaf.roster (module), 84
- sportsreference.ncaaf.schedule (module), 87
- sportsreference.ncaaf.teams (module), 90
- sportsreference.nfl.boxscore (module), 94
- sportsreference.nfl.player (module), 99
- sportsreference.nfl.roster (module), 103
- sportsreference.nfl.schedule (module), 109
- sportsreference.nfl.teams (module), 113
- sportsreference.nhl.boxscore (module), 116
- sportsreference.nhl.player (module), 121
- sportsreference.nhl.roster (module), 123
- sportsreference.nhl.schedule (module), 128
- sportsreference.nhl.teams (module), 131
- stadium (sportsreference.ncaaf.boxscore.Boxscore attribute), 75
- stadium (sportsreference.nfl.boxscore.Boxscore attribute), 97
- steal_percentage (sportsreference.nba.player.AbstractPlayer attribute), 37
- steal_percentage (sportsreference.ncaab.player.AbstractPlayer attribute), 59
- steal_percentage (sportsreference.ncaab.teams.Team attribute), 71
- steals (sportsreference.nba.player.AbstractPlayer attribute), 37
- steals (sportsreference.nba.teams.Team attribute), 48
- steals (sportsreference.ncaab.player.AbstractPlayer attribute), 59
- steals (sportsreference.ncaab.teams.Team attribute), 71
- stolen_bases (sportsreference.mlb.roster.Player attribute), 18
- stolen_bases (sportsreference.mlb.teams.Team attribute), 27
- streak (sportsreference.mlb.schedule.Game attribute), 21
- streak (sportsreference.mlb.teams.Team attribute), 27
- streak (sportsreference.nba.schedule.Game attribute), 44
- streak (sportsreference.ncaab.schedule.Game attribute), 66
- streak (sportsreference.ncaaf.schedule.Game attribute), 89
- strength_of_schedule (sportsreference.mlb.teams.Team attribute), 27
- strength_of_schedule (sportsreference.ncaab.teams.Team attribute), 71
- strength_of_schedule (sportsreference.ncaaf.teams.Team attribute), 93
- strength_of_schedule (sportsreference.nfl.teams.Team attribute), 115
- strength_of_schedule (sportsreference.nhl.teams.Team attribute), 133
- strikeouts (sportsreference.mlb.player.AbstractPlayer attribute), 13
- strikeouts (sportsreference.mlb.roster.Player attribute), 18
- strikeouts (sportsreference.mlb.teams.Team attribute), 27
- strikeouts_per_base_on_balls (sportsreference.mlb.teams.Team attribute), 27
- strikeouts_per_nine_innings (sportsreference.mlb.teams.Team attribute), 27
- strikeouts_thrown_per_walk (sportsreference.mlb.roster.Player attribute), 18
- strikes (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 11
- strikes_contact (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 11
- strikes_looking (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 11
- strikes_swinging (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 11
- strikes_thrown (sportsreference.mlb.boxscore.BoxscorePlayer attribute), 11

11

T

- tackles (*sportsreference.nfl.roster.Player* attribute), 107
- tackles_for_loss (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
- tackles_for_loss (*sportsreference.ncaaf.roster.Player* attribute), 86
- tackles_for_loss (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98
- take_fouls (*sportsreference.nba.roster.Player* attribute), 42
- takeaways (*sportsreference.nhl.roster.Player* attribute), 126
- Team (class in *sportsreference.mlb.teams*), 22
- Team (class in *sportsreference.nba.teams*), 46
- Team (class in *sportsreference.ncaab.teams*), 67
- Team (class in *sportsreference.ncaaf.teams*), 90
- Team (class in *sportsreference.nfl.teams*), 113
- Team (class in *sportsreference.nhl.teams*), 131
- team_abbreviation (*sportsreference.mlb.roster.Player* attribute), 19
- team_abbreviation (*sportsreference.nba.roster.Player* attribute), 42
- team_abbreviation (*sportsreference.ncaab.roster.Player* attribute), 63
- team_abbreviation (*sportsreference.ncaaf.roster.Player* attribute), 86
- team_abbreviation (*sportsreference.nfl.roster.Player* attribute), 108
- team_abbreviation (*sportsreference.nhl.roster.Player* attribute), 126
- team_conference (*sportsreference.ncaab.conferences.Conferences* attribute), 57
- team_conference (*sportsreference.ncaaf.conferences.Conferences* attribute), 79
- Teams (class in *sportsreference.mlb.teams*), 28
- Teams (class in *sportsreference.nba.teams*), 48
- Teams (class in *sportsreference.ncaab.teams*), 72
- Teams (class in *sportsreference.ncaaf.teams*), 93
- Teams (class in *sportsreference.nfl.teams*), 115
- Teams (class in *sportsreference.nhl.teams*), 133
- teams (*sportsreference.ncaab.conferences.Conference* attribute), 57
- teams (*sportsreference.ncaaf.conferences.Conference* attribute), 78
- third_down_attempts (*sportsreference.nfl.schedule.Game* attribute), 111
- third_down_conversions (*sportsreference.nfl.schedule.Game* attribute), 111
- thirty_to_thirty_nine_yard_field_goal_attempts (*sportsreference.nfl.roster.Player* attribute), 108
- thirty_to_thirty_nine_yard_field_goals_made (*sportsreference.nfl.roster.Player* attribute), 108
- three_point_attempt_rate (*sportsreference.nba.player.AbstractPlayer* attribute), 37
- three_point_attempt_rate (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
- three_point_attempt_rate (*sportsreference.ncaab.teams.Team* attribute), 71
- three_point_attempts (*sportsreference.nba.player.AbstractPlayer* attribute), 37
- three_point_attempts (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
- three_point_field_goal_attempts (*sportsreference.nba.teams.Team* attribute), 48
- three_point_field_goal_attempts (*sportsreference.ncaab.teams.Team* attribute), 71
- three_point_field_goal_percentage (*sportsreference.nba.teams.Team* attribute), 48
- three_point_field_goal_percentage (*sportsreference.ncaab.teams.Team* attribute), 71
- three_point_field_goals (*sportsreference.nba.teams.Team* attribute), 48
- three_point_field_goals (*sportsreference.ncaab.teams.Team* attribute), 71
- three_point_percentage (*sportsreference.nba.player.AbstractPlayer* attribute), 37
- three_point_percentage (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
- three_point_shot_percentage_from_corner (*sportsreference.nba.roster.Player* attribute), 42
- three_pointers (*sportsreference.nba.player.AbstractPlayer* attribute), 38
- three_pointers (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
- three_pointers_assisted_percentage (*sportsreference.nba.roster.Player* attribute), 42
- ties_plus_overtime_loss (*sportsreference.nhl.roster.Player* attribute), 127

time (*sportsreference.mlb.boxscore.Boxscore* attribute), 9
 time (*sportsreference.nba.schedule.Game* attribute), 44
 time (*sportsreference.ncaab.schedule.Game* attribute), 66
 time (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75
 time (*sportsreference.ncaaf.schedule.Game* attribute), 89
 time (*sportsreference.nfl.boxscore.Boxscore* attribute), 97
 time (*sportsreference.nhl.boxscore.Boxscore* attribute), 118
 time_of_day (*sportsreference.mlb.boxscore.Boxscore* attribute), 9
 time_of_possession (*sportsreference.nfl.schedule.Game* attribute), 111
 time_on_ice (*sportsreference.nhl.boxscore.BoxscorePlayer* attribute), 119
 time_on_ice (*sportsreference.nhl.roster.Player* attribute), 127
 time_on_ice_even_strength (*sportsreference.nhl.roster.Player* attribute), 127
 times_caught_stealing (*sportsreference.mlb.roster.Player* attribute), 19
 times_caught_stealing (*sportsreference.mlb.teams.Team* attribute), 27
 times_hit_by_pitch (*sportsreference.mlb.roster.Player* attribute), 19
 times_hit_by_pitch (*sportsreference.mlb.teams.Team* attribute), 27
 times_hit_player (*sportsreference.mlb.roster.Player* attribute), 19
 times_pass_target (*sportsreference.nfl.player.AbstractPlayer* attribute), 102
 times_pass_target (*sportsreference.nfl.roster.Player* attribute), 108
 times_sacked (*sportsreference.nfl.player.AbstractPlayer* attribute), 102
 times_sacked (*sportsreference.nfl.roster.Player* attribute), 108
 times_sacked (*sportsreference.nfl.schedule.Game* attribute), 111
 times_struck_out (*sportsreference.mlb.player.AbstractPlayer* attribute), 13
 times_struck_out (*sportsreference.mlb.roster.Player* attribute), 19
 times_struck_out (*sportsreference.mlb.teams.Team* attribute), 27
 total_bases (*sportsreference.mlb.roster.Player* attribute), 19
 total_bases (*sportsreference.mlb.teams.Team* attribute), 27
 total_fielding_runs_above_average (*sportsreference.mlb.roster.Player* attribute), 19
 total_fielding_runs_above_average_per_innings (*sportsreference.mlb.roster.Player* attribute), 19
 total_goals_against_on_ice (*sportsreference.nhl.roster.Player* attribute), 127
 total_goals_for_on_ice (*sportsreference.nhl.roster.Player* attribute), 127
 total_goals_per_game (*sportsreference.nhl.teams.Team* attribute), 133
 total_punt_yards (*sportsreference.nfl.player.AbstractPlayer* attribute), 102
 total_punt_yards (*sportsreference.nfl.roster.Player* attribute), 108
 total_rebound_percentage (*sportsreference.nba.player.AbstractPlayer* attribute), 38
 total_rebound_percentage (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
 total_rebound_percentage (*sportsreference.ncaab.teams.Team* attribute), 71
 total_rebounds (*sportsreference.nba.player.AbstractPlayer* attribute), 38
 total_rebounds (*sportsreference.nba.teams.Team* attribute), 48
 total_rebounds (*sportsreference.ncaab.player.AbstractPlayer* attribute), 59
 total_rebounds (*sportsreference.ncaab.teams.Team* attribute), 71
 total_runs (*sportsreference.mlb.teams.Team* attribute), 27
 total_shots (*sportsreference.nhl.roster.Player* attribute), 127
 total_tackles (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81
 total_tackles (*sportsreference.ncaaf.roster.Player* attribute), 86
 total_touchdowns (*sportsreference.ncaaf.roster.Player* attribute), 86
 touchdown_percentage_index (*sportsreference.nfl.roster.Player* attribute), 108
 touches (*sportsreference.nfl.roster.Player* attribute), 108
 triples (*sportsreference.mlb.roster.Player* attribute), 19

`triples` (*sportsreference.mlb.teams.Team* attribute), 27

`true_shooting_percentage` (*sportsreference.nba.player.AbstractPlayer* attribute), 38

`true_shooting_percentage` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`true_shooting_percentage` (*sportsreference.ncaab.teams.Team* attribute), 71

`turnover_percentage` (*sportsreference.nba.player.AbstractPlayer* attribute), 38

`turnover_percentage` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`turnover_percentage` (*sportsreference.ncaab.teams.Team* attribute), 71

`turnovers` (*sportsreference.nba.player.AbstractPlayer* attribute), 38

`turnovers` (*sportsreference.nba.teams.Team* attribute), 48

`turnovers` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`turnovers` (*sportsreference.ncaab.teams.Team* attribute), 71

`turnovers` (*sportsreference.ncaaf.teams.Team* attribute), 93

`turnovers` (*sportsreference.nfl.teams.Team* attribute), 115

`twenty_to_twenty_nine_yard_field_goal_attempts` (*sportsreference.nfl.roster.Player* attribute), 108

`twenty_to_twenty_nine_yard_field_goals_made` (*sportsreference.nfl.roster.Player* attribute), 108

`two_point_attempts` (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 34

`two_point_attempts` (*sportsreference.nba.roster.Player* attribute), 42

`two_point_attempts` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`two_point_conversions` (*sportsreference.ncaaf.roster.Player* attribute), 86

`two_point_field_goal_attempts` (*sportsreference.nba.teams.Team* attribute), 48

`two_point_field_goal_attempts` (*sportsreference.ncaab.teams.Team* attribute), 71

`two_point_field_goal_percentage` (*sportsreference.nba.teams.Team* attribute), 48

`two_point_field_goal_percentage` (*sportsreference.ncaab.teams.Team* attribute), 71

`two_point_field_goals` (*sportsreference.nba.teams.Team* attribute), 48

`two_point_field_goals` (*sportsreference.ncaab.teams.Team* attribute), 72

`two_point_percentage` (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 34

`two_point_percentage` (*sportsreference.nba.roster.Player* attribute), 42

`two_point_percentage` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`two_pointers` (*sportsreference.nba.boxscore.BoxscorePlayer* attribute), 35

`two_pointers` (*sportsreference.nba.roster.Player* attribute), 42

`two_pointers` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

`two_pointers_assisted_percentage` (*sportsreference.nba.roster.Player* attribute), 42

`type` (*sportsreference.ncaab.schedule.Game* attribute), 66

`type` (*sportsreference.nfl.schedule.Game* attribute), 111

U

`unknown_bat_types` (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 11

`usage_percentage` (*sportsreference.nba.player.AbstractPlayer* attribute), 38

`usage_percentage` (*sportsreference.ncaab.player.AbstractPlayer* attribute), 60

V

`value_over_replacement_player` (*sportsreference.nba.roster.Player* attribute), 42

`venue` (*sportsreference.mlb.boxscore.Boxscore* attribute), 9

W

`week` (*sportsreference.nfl.schedule.Game* attribute), 112

`weight` (*sportsreference.mlb.roster.Player* attribute), 19

`weight` (*sportsreference.nba.roster.Player* attribute), 43

`weight` (*sportsreference.ncaab.roster.Player* attribute), 64

`weight` (*sportsreference.ncaaf.roster.Player* attribute), 86

`weight` (*sportsreference.nfl.roster.Player* attribute), 108

`weight` (*sportsreference.nhl.roster.Player* attribute), 127

- whip (*sportsreference.mlb.roster.Player* attribute), 19
 whip (*sportsreference.mlb.teams.Team* attribute), 27
 wild_pitches (*sportsreference.mlb.roster.Player* attribute), 19
 wild_pitches (*sportsreference.mlb.teams.Team* attribute), 27
 win_percentage (*sportsreference.mlb.roster.Player* attribute), 19
 win_percentage (*sportsreference.mlb.teams.Team* attribute), 27
 win_percentage (*sportsreference.ncaab.teams.Team* attribute), 72
 win_percentage (*sportsreference.ncaaf.teams.Team* attribute), 93
 win_percentage (*sportsreference.nfl.teams.Team* attribute), 115
 win_probability_added (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 11
 win_probability_added_pitcher (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 11
 win_probability_for_offensive_player (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 11
 win_probability_subtracted (*sportsreference.mlb.boxscore.BoxscorePlayer* attribute), 11
 win_shares (*sportsreference.nba.roster.Player* attribute), 43
 win_shares (*sportsreference.ncaab.roster.Player* attribute), 64
 win_shares_per_40_minutes (*sportsreference.ncaab.roster.Player* attribute), 64
 win_shares_per_48_minutes (*sportsreference.nba.roster.Player* attribute), 43
 winner (*sportsreference.mlb.boxscore.Boxscore* attribute), 9
 winner (*sportsreference.mlb.schedule.Game* attribute), 21
 winner (*sportsreference.nba.boxscore.Boxscore* attribute), 34
 winner (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54
 winner (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75
 winner (*sportsreference.nfl.boxscore.Boxscore* attribute), 97
 winner (*sportsreference.nhl.boxscore.Boxscore* attribute), 118
 winning_abbr (*sportsreference.mlb.boxscore.Boxscore* attribute), 9
 winning_abbr (*sportsreference.nba.boxscore.Boxscore* attribute), 34
 winning_abbr (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54
 winning_abbr (*sportsreference.nfl.boxscore.Boxscore* attribute), 97
 winning_abbr (*sportsreference.nhl.boxscore.Boxscore* attribute), 118
 winning_name (*sportsreference.mlb.boxscore.Boxscore* attribute), 9
 winning_name (*sportsreference.nba.boxscore.Boxscore* attribute), 34
 winning_name (*sportsreference.ncaab.boxscore.Boxscore* attribute), 54
 winning_name (*sportsreference.ncaaf.boxscore.Boxscore* attribute), 75
 winning_name (*sportsreference.nfl.boxscore.Boxscore* attribute), 97
 winning_name (*sportsreference.nhl.boxscore.Boxscore* attribute), 118
 wins (*sportsreference.mlb.roster.Player* attribute), 19
 wins (*sportsreference.mlb.teams.Team* attribute), 28
 wins (*sportsreference.nba.schedule.Game* attribute), 44
 wins (*sportsreference.ncaab.teams.Team* attribute), 72
 wins (*sportsreference.ncaaf.schedule.Game* attribute), 89
 wins (*sportsreference.ncaaf.teams.Team* attribute), 93
 wins (*sportsreference.nfl.teams.Team* attribute), 115
 wins (*sportsreference.nhl.roster.Player* attribute), 127
 wins (*sportsreference.nhl.teams.Team* attribute), 133
 wins_last_ten_games (*sportsreference.mlb.teams.Team* attribute), 28
 wins_last_thirty_games (*sportsreference.mlb.teams.Team* attribute), 28
 wins_last_twenty_games (*sportsreference.mlb.teams.Team* attribute), 28
 wins_vs_left_handed_pitchers (*sportsreference.mlb.teams.Team* attribute), 28
 wins_vs_right_handed_pitchers (*sportsreference.mlb.teams.Team* attribute), 28
 wins_vs_teams_over_500 (*sportsreference.mlb.teams.Team* attribute), 28
 wins_vs_teams_under_500 (*sportsreference.mlb.teams.Team* attribute), 28
- Y**
 yards (*sportsreference.ncaaf.teams.Team* attribute), 93
 yards (*sportsreference.nfl.teams.Team* attribute), 115
 yards_from_penalties (*sportsreference.ncaaf.teams.Team* attribute), 93

`yards_from_penalties` (*sportsreference.nfl.teams.Team* attribute), 115

`yards_from_scrimmage` (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

`yards_from_scrimmage` (*sportsreference.ncaaf.roster.Player* attribute), 86

`yards_from_scrimmage` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_from_scrimmage_per_play` (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

`yards_from_scrimmage_per_play` (*sportsreference.ncaaf.roster.Player* attribute), 86

`yards_lost_from_sacks` (*sportsreference.nfl.boxscore.BoxscorePlayer* attribute), 98

`yards_lost_from_sacks` (*sportsreference.nfl.schedule.Game* attribute), 112

`yards_lost_to_sacks` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_attempt_index` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_completed_pass` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_game_played` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_kickoff_return` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_play` (*sportsreference.ncaaf.teams.Team* attribute), 93

`yards_per_play` (*sportsreference.nfl.teams.Team* attribute), 115

`yards_per_punt` (*sportsreference.nfl.player.AbstractPlayer* attribute), 102

`yards_per_punt_return` (*sportsreference.nfl.player.AbstractPlayer* attribute), 102

`yards_per_punt_return` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_per_touch` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_recovered_from_fumble` (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

`yards_recovered_from_fumble` (*sportsreference.ncaaf.roster.Player* attribute), 86

`yards_recovered_from_fumble` (*sportsreference.nfl.player.AbstractPlayer* attribute), 102

`yards_recovered_from_fumble` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_returned_from_interception` (*sportsreference.nfl.player.AbstractPlayer* attribute), 102

`yards_returned_from_interception` (*sportsreference.nfl.roster.Player* attribute), 108

`yards_returned_from_interceptions` (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

`yards_returned_from_interceptions` (*sportsreference.ncaaf.roster.Player* attribute), 87

`yards_returned_per_interception` (*sportsreference.ncaaf.player.AbstractPlayer* attribute), 81

`yards_returned_per_interception` (*sportsreference.ncaaf.roster.Player* attribute), 87

`year` (*sportsreference.ncaaf.roster.Player* attribute), 87