sportsipy Documentation

Release 0.1.0

Author

Jan 07, 2021

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
- 1.7 Find the number of goals a football team has scored . . . 4
  - 1.7.1 Examples . . . 4
    - 1.7.1.1 Finding Tallest Players . . . 4
    - 1.7.1.2 Writing To CSV and Pickle . . . 5
    - 1.7.1.3 Finding Top Win Percentage By Year . . . 5
  - 1.7.2 Installation . . . 6
  - 1.7.3 Testing . . . 6

2 Indices and tables **7**

Sportsipy 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.

Sportsipy 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 sportsipy. 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 sportsipy.nhl.teams import Teams

teams = Teams(2018)
```

1.2 Print every NBA team's name and abbreviation

```
from sportsipy.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 sportsipy.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 sportsipy.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 sportsipy.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 sportsipy.mlb.boxscore import Boxscore

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

1.7 Find the number of goals a football team has scored

```
from sportsipy.fb.team import Team

tottenham = Team('Tottenham Hotspur')
print(tottenham.goals_scored)
```

1.7.1 Examples

Thanks to the broad range of metrics that are pulled from sports-reference.com, there are multiple ways you can use the *sportsipy* 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 *sportsipy*.

1.7.1.1 Finding Tallest Players

For each team, find the tallest player on the roster and print out their name and height in inches.

```
from sportsipy.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():
    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)
```

1.7.1.2 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 sportsipy.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())
```

1.7.1.3 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 sportsipy.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)
```

1.7.2 Installation

The easiest way to install *sportsipy* is by downloading the latest released binary from PyPI using PIP. For instructions on installing PIP, visit PyPA.io for detailed steps on installing the package manager for your local environment.

Next, run:

```
pip install sportsipy
```

to download and install the latest official release of *sportsipy* on your machine. You now have the latest stable version of *sportsipy* installed and can begin using it following the examples!

If the bleeding-edge version of *sportsipy* is desired, clone this repository using git and install all of the package requirements with PIP:

```
git clone https://github.com/roclark/sportsipy
cd sportsipy
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
```

1.7.3 Testing

Sportsipy 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=sportsipy --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