pyfootball Documentation

Release 1.0.1

Timothy "xozzo" Ng

Jun 03, 2017

User Documentation

1 Requirements **3**

2 Installation **5**

3 Example Usage **7**
- 3.1 Getting Started . . . 7
- 3.2 Data Model . . . 8
- 3.3 API . . . 11
- 3.4 Frequently Asked Questions . . . 14
- 3.5 Support . . . 14
- 3.6 Change Log . . . 14

4 License **15**

Python Module Index **17**

pyfootball is a client library for football-data.org written in Python.

This library was written to allow for easier access to football-data's resources by abstracting HTTP requests and representing the JSON responses as Python classes.

Warning: pyfootball **does not** rate limit methods that send HTTP requests to football-data's endpoints. You are responsible for adhering to the 50-requests-per-minute rule — you risk having your API key revoked and/or your IP banned if you don't!

CHAPTER 1

Requirements

- A valid API key for football-data. You can request for one here.
- Python 3.5+
- The `requests` library. pip should handle this for you when installing pyfootball.

CHAPTER 2

Installation

Installation is easy using pip:

```
$ pip install pyfootball
```

CHAPTER 3

Example Usage

```
>>> import pyfootball
>>> f = pyfootball.Football(api_key='your_api_key')
>>> bayern = f.get_team(5)
>>> bayern.market_value
582,225,000 €
```

Getting Started

In this tutorial, you'll be introduced to pyfootball's API as well as its data mapping.

If you're not familiar with football-data.org, it'd be better for you to get acquainted with it by reading the football-data.org documentation before proceeding with pyfootball.

If you don't have pyfootball set up, see *the home page*. Otherwise, let's get started!

First, you're going to want to create a `Football` instance:

```
>>> import pyfootball
>>> f = pyfootball.Football(api_key='your_api_key')
```

You can also choose to instantiate `Football` without any arguments and make it use an API key obtained from an environmental variable named `PYFOOTBALL_API_KEY`. Here is an example in *nix:

```
$ export PYFOOTBALL_API_KEY='your_api_key'
```

and then in your program:

```
>>> import pyfootball
>>> f = pyfootball.Football()
```

If you provide an invalid API key, an HTTPError exception will be raised.

Note: Instantiating a `Football` object will use one request out of the 50 allowed per minute by football-data.org's API. You can see the full list of which functions send requests and which ones don't at *API*.

The `Football` class serves as an entry point for the library. Now, we want to get the data of a team — for example, Manchester United — but since we don't know its ID in football-data.org's database, we're going to have to look it up:

```
>>> matches = f.search_teams("manchester")
>>> matches
{65: 'Manchester City FC', 66: 'Manchester United FC'}
```

`Football.search_teams(name)` queries the database for matches to `name` and returns key-value pairs of team IDs and team names respectively.

Now that we have Manchester United's ID, we can get more information about it:

```
>>> man_utd = f.get_team(66)
```

`Football.get_team(id)` returns a `Team` object. It contains all the information you'd get in a JSON response from football-data.org, along with some cool functions. We can call `Team.get_fixtures()` to get its fixtures or `Team.get_players()` to get its players.

Hint: The `Football` class provides a useful method `Football.get_prev_response()` to give you information about the most recently-used response. Any time you use a method in the library that sends a HTTP request, this value is updated. You can use it to keep track of useful stuff like response status code or how many requests you have left.

```
>>> players = man_utd.get_players()
```

`Team.get_players()` returns a list of `Player` objects. Like `Team` objects, `Player` objects are objects from JSON responses mapped to Python classes:

```
>>> players[0].name
Paul Pogba
>>> players[0].market_value
70,000,000 €
```

A comprehensive list of object models and their attributes are available at *Data Model*. A full list of functions available are available at *API*.

Data Model

The data model was designed to keep to the original data's structure as closely as possible. There were mostly minor changes as a result of following the PEP8 guidelines such as turning variable names from using `camelCase` to `under_scores`.

Each football-data.org resource is mapped into an object. Each value in a JSON resource is mapped to an attribute of the object. You can access these values using the syntax `Object.attribute`. For example:

```
>>> import pyfootball
>>> f = pyfootball.Football(api_key='your_api_key')
>>> my_team = f.get_team(5)
```

```
>>> my_team.name
FC Bayern München
```

Competition

Attribute	Type	Description
id	integer	The ID of the competition.
name	string	The name of the competition.
code	string	The League Code of the competition.
year	integer	The year in which the competition started. For example, the `year` for a 16/17 competition would be 2016.
current_matchday	integer	The competition's current matchday.
number_of_matchdays	integer	The number of matchdays in this competition.
number_of_teams	integer	The number of teams competing in this competition.
number_of_games	integer	The number of games in this competition.
last_updated	datetime	The date and time at which this resource was last updated.

LeagueTable

Attribute	Type	Description
competition_id	integer	The competition ID for this league table.
competition_name	string	The competition name for this league table.
current_matchday	id	The current matchday.
standings	list	A list of `Standing` objects. The list is one-indexed so as to correspond with the position in the table (i.e. standings[1] is the top of the table)

Standing

Each `Standing` object represents a "row" in the league table.

Attribute	Type	Description
team_id	integer	The team ID.
team_name	string	The team name.
crest_url	string	A link to an image of the team's crest.
position	integer	The current team's position.
games_played	integer	The number of games played by this team.
points	integer	The number of points that this team has.
goals	integer	The number of goals scored by this team.
goals_against	integer	The number of goals conceded by this team.
goal_difference	integer	`(goals - goals_against)`
wins	integer	The number of wins this team has.
draws	integer	The number of draws this team has.
losses	integer	The number of losses this team has.
home	dict	Contains `goals`, `goals_against`, `wins`, `draws`, and `losses` keys with integer values that represent home stats.
away	dict	Contains `goals`, `goals_against`, `wins`, `draws`, and `losses` keys with integer values that represent away stats.

Fixture

Attribute	Type	Description
date	datetime	The fixture date and time.
status	string	The status of this fixture.
matchday	integer	The matchday on which this fixture is set.
home_team	string	The name of the home team.
home_team_id	integer	The ID of the home team.
away_team	string	The name of the away team.
away_team_id	integer	The ID of the away team.
competition_id	integer	The ID of the competition for this fixture.
result	dict	The result for this fixture. `None` if the match is not complete. Otherwise, contains `home_team_goals` and `away_team_goals` keys with integer values. Some `Fixtures` have a `half_time` key set for the score at half time.
odds	dict	The betting odds for this fixture. `None` if not available. Otherwise, contains `home_win`, `draw` and `away_win` keys with float values.

Team

Attribute	Type	Description
id	integer	The team ID.
name	string	The team name.
code	string	The team code (e.g. Borussia Dortmund's code is BVB).
short_name	string	The team's short name.
market_value	string	The collective market value of the team's squad.
crest_url	string	A link to an image of the team's crest.

Player

Attribute	Type	Description
name	string	The player's name.
position	string	The player's position on the field.
jersey_number	integer	The player's kit number.
date_of_birth	date	The player's date of birth.
nationality	string	The player's nationality.
contract_until	date	The date of the player's contract expiry with their team.
market_value	string	The player's market value.

API

For every function that sends a HTTP request, an `HTTPError` is raised whenever the response status code is `4XX` or `5XX` which signifies that something went wrong between pyfootball sending the API a request and the API giving a response. If you believe this to be an issue with pyfootball itself, please see *Support* for more information.

Football

This class serves as the driver/entry point for this library.

class `pyfootball.football.`**Football**(*api_key=None*)

__init__(*api_key=None*)

Takes either an api_key as a keyword argument or tries to access an environmental variable `PYFOOTBALL_API_KEY`, then uses the key to send a test request to make sure that it's valid. The api_key kwarg takes precedence over the envvar.

Sends one request to api.football-data.org.

Parameters **api_key** (*string*) – The user's football-data.org API key.

get_all_competitions()

Returns a list of Competition objects representing the current season's competitions.

Sends one request to api.football-data.org.

Returns comp_list: List of Competition objects.

get_all_fixtures()

Returns a list of all Fixture objects in the specified time frame. Defaults to the next 7 days or "n7". TODO: Include timeFrameStart and timeFrameEnd, and filter for specifying time frame.

Sends one request to api.football-data.org.

Returns fixture_list: A list of Fixture objects.

get_comp_fixtures(*comp_id*)
Given an ID, returns a list of Fixture objects associated with the given competition.

Sends one request to api.football-data.org.

Parameters **comp_id** (*integer*) – The competition ID.

Returns fixture_list: A list of Fixture objects.

get_competition(*comp_id*)
Returns a Competition object associated with the competition ID.

Sends one request to api.football-data.org.

Parameters **comp_id** (*integer*) – The competition ID.

Returns Competition: The Competition object.

get_competition_teams(*comp_id*)
Given an ID, returns a list of Team objects associated with the given competition.

Sends one request to api.football-data.org.

Parameters **comp_id** (*integer*) – The competition ID.

Returns team_list: A list of Team objects.

get_fixture(*fixture_id*)
Returns a Fixture object associated with the given ID. The response includes a head-to-head between teams; this will be implemented in the near future.

Sends one request to api.football-data.org.

Parameters **fixture_id** (*integer*) – The fixture ID.

Returns Fixture: A Fixture object.

get_league_table(*comp_id*)
Given a competition ID, returns a LeagueTable object for the league table associated with the competition.

Sends one request to api.football-data.org.

Parameters **comp_id** (*integer*) – The competition ID.

Returns LeagueTable: A LeagueTable object.

get_prev_response()
Returns information about the most recent response.

Returns prev_response: Information about the most recent response.

get_team(*team_id*)
Given an ID, returns a Team object for the team associated with the ID.

Sends one request to api.football-data.org.

Parameters **team_id** (*integer*) – The team ID.

Returns Team: A Team object.

get_team_fixtures(*team_id*)
Given a team ID, returns a list of Fixture objects associated with the team.

Sends one request to api.football-data.org.

Parameters **team_id** (*integer*) – The team ID.

Returns fixture_list: A list of Fixture objects for the team.

get_team_players(*team_id*)
Given a team ID, returns a list of Player objects associated with the team.

Sends one request to api.football-data.org.

Parameters **team_id** (*integer*) – The team ID.

Returns player_list: A list of Player objects in the specified team.

search_teams(*team_name*)
Given a team name, queries the database for matches and returns a dictionary containing key-value pairs of their team IDs and team names.

Sends one request to api.football-data.org.

Parameters **team_name** (*string*) – The partial or full team name.

Returns matches: A dict with team ID as keys and team name as values.

Returns `None`: If no matches are found for the given team_name.

Competition

class pyfootball.models.competition.**Competition**(*data*)

get_fixtures()
Return a list of Fixture objects representing the fixtures in this competition for the current season.

Sends one request to api.football-data.org.

Returns fixture_list: A list of Fixture objects.

get_league_table()
Return the league table for this competition.

Sends one request to api.football-data.org.

Returns LeagueTable: A LeagueTable object.

get_teams()
Return a list of Team objects representing the teams in this competition for the current season.

Sends one request to api.football-data.org.

Returns team_list: A list of Team objects.

Team

class pyfootball.models.team.**Team**(*data*)

get_fixtures()
Return a list of Fixture objects representing this season's fixtures for the current team.

Sends one request to api.football-data.org.

Returns fixture_list: A list of Fixture objects.

get_players()
Return a list of Player objects representing players on the current team.

Sends one request to api.football-data.org.

Returns player_list: A list of Player objects.

Frequently Asked Questions

Intentionally left empty for now.

Support

Bugs

If you believe you've found a bug with the library, feel free to create an issue on our issue tracker **with information on how to reproduce the problem**.

The pyfootball issue tracker is located at https://github.com/xozzo/pyfootball/issues.

Other

For anything else, like questions on how to use the library or why something is behaving the way it is, you can tweet me @timorthi.

Change Log

1.0.1 (2016.11.15)

- **[FEATURE]** The `Football` object now uses either a kwarg or an envvar `PYFOOTBALL_API_KEY` to obtain an API key.
- **[FIX]** Fixed models not returning expected data types. Namely, numerical types were being returned as strings.
- **[DEV]** Wrote tests that cover most of the library.
- **[DEV]** Added Travis CI integration.
- **[OTHER]** Removed To-Do List from README file.
- **[OTHER]** Added a CONTRIBUTING file including contributing guidelines.

1.0.0 (2016.10.17)

- Initial release! :)

CHAPTER 4

License

The project is licensed under the MIT license.

Python Module Index

p

pyfootball.football, 11
pyfootball.models.competition, 13
pyfootball.models.team, 13

Index

Symbols

__init__() (pyfootball.football.Football method), 11

C

Competition (class in pyfootball.models.competition), 13

F

Football (class in pyfootball.football), 11

G

get_all_competitions() (pyfootball.football.Football method), 11
get_all_fixtures() (pyfootball.football.Football method), 11
get_comp_fixtures() (pyfootball.football.Football method), 12
get_competition() (pyfootball.football.Football method), 12
get_competition_teams() (pyfootball.football.Football method), 12
get_fixture() (pyfootball.football.Football method), 12
get_fixtures() (pyfootball.models.competition.Competition method), 13
get_fixtures() (pyfootball.models.team.Team method), 13
get_league_table() (pyfootball.football.Football method), 12
get_league_table() (pyfootball.models.competition.Competition method), 13
get_players() (pyfootball.models.team.Team method), 13
get_prev_response() (pyfootball.football.Football method), 12
get_team() (pyfootball.football.Football method), 12
get_team_fixtures() (pyfootball.football.Football method), 12
get_team_players() (pyfootball.football.Football method), 13
get_teams() (pyfootball.models.competition.Competition method), 13

P

pyfootball.football (module), 11
pyfootball.models.competition (module), 13
pyfootball.models.team (module), 13

S

search_teams() (pyfootball.football.Football method), 13

T

Team (class in pyfootball.models.team), 13