fpl Documentation

Release 0.6.0

Amos Bastian

Jun 28, 2020

Contents

Note: The latest version of **fpl** is asynchronous, and requires Python 3.6+!

If you're interested in helping out the development of **fpl**, or have suggestions and ideas then please don't hesitate to create an issue on GitHub, join our Discord server or send an email to amosbastian@gmail.com!

A simple example:

```
>>> import aiohttp
>>> import asyncio
>>> from fpl import FPL
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         player = await fpl.get_player(302)
...     print(player)
...
>>> asyncio.run(main())
Pogba - Midfielder - Man Utd
```

With **fpl** you can easily use the Fantasy Premier League API in all your Python scripts, exactly how you expect it to work.

CHAPTER 1

The User Guide

This part of the documentation is mostly an introduction on how to use **fpl** and install it - including information for people newer to *asyncio*.

1.1 Installing fpl

The recommended way to install `fpl` is via `pip`.

```
pip install fpl
```

Note: Depending on your system, you may need to use `pip3` to install packages for Python 3.

1.1.1 Updating fpl with pip

To update **fpl** you can run:

```
pip install --upgrade fpl
```

Example output:

```
Installing collected packages: fpl
  Found existing installation: fpl 0.1.0
    Uninstalling fpl-0.1.0:
      Successfully uninstalled fpl-0.1.0
Successfully installed fpl-0.2.0
```

1.1.2 Installing older versions

Older versions of **fpl** can be installed by specifying the version number as part of the installation command:

```
pip install fpl==0.2.0
```

1.1.3 Installing from GitHub

The source code for **fpl** is available on GitHub repository https://github.com/amosbastian/fpl. To install the most recent version of **fpl** from here you can use the following command:

```
$ git clone git://github.com/amosbastian/fpl.git
```

You can also install a .tar file or .zip file

> $ curl -OL https://github.com/amosbastian/fpl/tarball/master $ curl -OL https://github.com/amosbastian/fpl/zipball/master # Windows

Once it has been downloaded you can easily install it using *pip*:

```
$ cd fpl
$ pip install .
```

1.2 Quickstart

This part of the user guide will try to make it a bit more easy for users to get started with using **fpl**! Before starting, make sure that **fpl** is *installed and up to date*.

1.2.1 Using the FPL class

The *FPL* class is the main way you will be accessing information from the Fantasy Premier League's API.

Begin by importing the *FPL* class from **fpl**:

```
>>> from fpl import FPL
```

Because **fpl** uses aiohttp, we must also import this and pass a Client Session as an argument to the *FPL* class. You can either create a session and pass it like this:

```
>>> import aiohttp
>>>
>>> async def main():
...     session = aiohttp.ClientSession()
...     fpl = FPL(session)
        # ...
...     await session.close()
```

or use a session context manager:

```
>>> async def main():
...     async with aiohttp.ClientSession as session:
...         fpl = FPL(session)
...         # ...
```

Now, let's try to get a player. For this example, let's get Manchester United's star midfielder Paul Pogba (replace # . . . with this code):

```
>>> player = await fpl.get_player(302)
>>> print(player)
Pogba - Midfielder - Man Utd
```

Now, we have a *Player* object called `player`. We can get all the information we need from this object. For example, if we want his points per game, or his total points, then we can simply do this:

```
>>> print(player.points_per_game)
5.7
>>> print(player.total_points)
113
```

Nearly all of *FPL*'s functions include the argument `return_json` - if you want to get a `dict` instead of e.g. a *Player* object, then you can simply do the following:

```
>>> player = await fpl.get_player(302, return_json=True)
>>> print(player["total_points"])
113
```

Nice, right? However, one important thing was left out. Because **fpl** is asynchronous, you must use `asyncio` to run the function:

```
>>> import asyncio
>>> asyncio.run(main())
```

1.2.2 Authentication

Some of the Fantasy Premier League's API endpoints require the user to be logged in. For example, the endpoint for my team) will return:

```
{"detail":"Authentication credentials were not provided."}
```

since you aren't logged in to my account. To still allow **fpl** users to access this, the `login` function was added to *FPL*. It must be called before using other functions where login authentication is required. Let's use my team as an example:

```
>>> import asyncio
>>> import aiohttp
>>> from fpl import FPL
>>>
>>> async def my_team(user_id):
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         await fpl.login()
...         user = await fpl.get_user(user_id)
...         team = await user.get_team()
...     print(team)
...
>>> asyncio.run(my_team(3808385))
[{'can_sub': True, 'has_played': False, 'is_sub': False, 'can_captain': True,
↪'selling_price': 46, 'multiplier': 1, 'is_captain': False, 'is_vice_captain': False,
↪ 'position': 1, 'element': 400}, ..., {'can_sub': True, 'has_played': False, 'is_sub
↪': True, 'can_captain': True, 'selling_price': 44, 'multiplier': 1, 'is_captain':␣
↪False, 'is_vice_captain': False, 'position': 15, 'element': 201}]
```

1.3 Examples

It always helps to have examples of how to implement certain things. Hopefully this page will help you if you are stuck on something, and need some inspiration. If there are certain examples you think would be helpful to add to this page, then don't hesitate to create an issue detailing it.

Note: There is no doubt that the below examples could be implemented in a much better way - if you want to improve them, then don't hestitate to!

1.3.1 The league's best . . .

One of things that is most interesting to see is which players are performing the best in certain metrics. For example, which players score the most points per game, or which players have the most goals + assists. This is really easy to implement using **fpl**!

```
import asyncio

import aiohttp
from prettytable import PrettyTable

from fpl import FPL


async def main():
    async with aiohttp.ClientSession() as session:
        fpl = FPL(session)
        players = await fpl.get_players()

    top_performers = sorted(
        players, key=lambda x: x.goals_scored + x.assists, reverse=True)

    player_table = PrettyTable()
    player_table.field_names = ["Player", "£", "G", "A", "G + A"]
    player_table.align["Player"] = "l"

    for player in top_performers[:10]:
        goals = player.goals_scored
        assists = player.assists
        player_table.add_row([player.web_name, f"£{player.now_cost / 10}",
                              goals, assists, goals + assists])

    print(player_table)

if __name__ == "__main__":
    asyncio.run(main())
```

which outputs the following table:

```
+------------+-------+----+----+-------+
| Player     |   £   | G  | A  | G + A |
+------------+-------+----+----+-------+
| Salah      | £13.6 | 16 | 8  |   24  |
| Hazard     | £11.0 | 10 | 10 |   20  |
| Sterling   | £11.3 | 10 | 10 |   20  |
```

(continues on next page)

(continued from previous page)

```
| Kane        | £12.4 | 14 | 6  |   20  |
| Aubameyang | £11.3 | 14 | 5  |   19  |
| Sané        |  £9.7 | 8  | 11 |   19  |
| Wilson      |  £6.5 | 10 | 8  |   18  |
| Agüero      | £11.3 | 10 | 8  |   18  |
| Lacazette   |  £9.3 | 8  | 8  |   16  |
| Pogba       |  £8.7 | 8  | 8  |   16  |
+-------------+-------+----+----+-------+
```

Of course this can be done with any of a *Player*'s attributes, so just experiment!

1.3.2 Alternative FDR

As we all know, the official FDR used by the Fantasy Premier League is not the best. With this in mind the function *FDR()* was created, which returns a dictionary containing an alternative FDR based on points scored for / against teams! Using this dictionary we can create a table containing each team's new FDR, which can then be used to decide which players you should play the next gameweek. Below an example of this table with colour highlighting is shown:

```
import asyncio

import aiohttp
from colorama import Fore, init
from prettytable import PrettyTable

from fpl import FPL


async def main():
    async with aiohttp.ClientSession() as session:
        fpl = FPL(session)
        fdr = await fpl.FDR()

    fdr_table = PrettyTable()
    fdr_table.field_names = [
        "Team", "All (H)", "All (A)", "GK (H)", "GK (A)", "DEF (H)", "DEF (A)",
        "MID (H)", "MID (A)", "FWD (H)", "FWD (A)"]

    for team, positions in fdr.items():
        row = [team]
        for difficulties in positions.values():
            for location in ["H", "A"]:
                if difficulties[location] == 5.0:
                    row.append(Fore.RED + "5.0" + Fore.RESET)
                elif difficulties[location] == 1.0:
                    row.append(Fore.GREEN + "1.0" + Fore.RESET)
                else:
                    row.append(f"{difficulties[location]:.2f}")

        fdr_table.add_row(row)

    fdr_table.align["Team"] = "l"
    print(fdr_table)

if __name__ == '__main__':
    asyncio.run(main())
```

which outputs the following table:

Team	All (H)	All (A)	GK (H)	GK (A)	DEF (H)	DEF (A)	MID (H)	MID (A)	FWD (H)	FWD (A)
Man City	4.45	5.0	3.62	5.0	3.75	5.0	4.61	5.0	5.0	3.94
Chelsea	3.99	3.47	3.72	3.01	3.62	3.35	4.01	3.93	4.09	4.42
West Ham	2.87	1.83	2.45	2.70	2.89	2.34	3.08	1.19	2.95	4.17
Cardiff	1.0	1.62	1.09	3.24	1.0	2.53	1.0	1.0	3.37	2.57
Newcastle	2.54	1.66	1.56	2.62	2.05	1.54	2.80	2.66	4.43	3.22
Everton	2.85	3.41	1.96	4.16	2.92	3.88	3.49	3.20	2.26	2.93
Watford	3.59	2.52	4.09	3.70	3.34	3.14	3.79	2.85	3.09	1.0
Fulham	1.09	1.48	1.81	2.17	1.16	2.32	1.77	1.26	1.26	2.78
Leicester	3.57	2.74	3.87	2.34	3.68	3.07	3.38	2.95	3.09	3.81
Crystal Palace	3.08	1.41	3.16	1.0	3.22	1.0	3.18	2.87	2.55	4.37
Liverpool	4.91	4.66	4.32	4.76	5.0	4.63	4.10	4.08	4.53	5.0
Wolves	3.20	2.34	2.15	3.62	3.06	2.82	3.74	1.42	2.82	4.14
Bournemouth	1.75	3.30	1.86	3.93	2.29	3.40	2.00	3.69	1.34	3.23
Spurs	5.0	3.17	5.0	3.10	4.85	3.21	5.0	3.85	3.09	3.40
Man Utd	3.94	3.21	3.78	2.84	4.49	3.63	3.25	3.06	3.44	3.79
Huddersfield	2.19	1.0	1.37	2.16	2.60	1.34	3.05	2.04	1.0	2.08
Southampton	2.11	2.03	1.0	3.01	2.30	2.37	2.56	1.80	2.42	3.70
Burnley	1.57	2.41	1.63	4.18	1.86	2.61	2.04	2.02	1.65	3.71
Brighton	2.24	3.39	2.53	4.18	1.97	3.61	2.34	3.61	3.53	2.96
Arsenal	3.44	4.29	4.11	4.39	3.67	4.34	3.35	4.07	2.51	4.21

1.3.3 Optimal captain choice?!

One of the most important aspects of the Fantasy Premier League is your captain choice each week. Of course, it's very difficult to get this correct each week! Because of this, it's quite interesting (or frustrating) to see what could've been. The code snippet below shows how you can create a table showing your captain and top scorer of each gameweek, and their respective difference in points scored:

```
import asyncio
from operator import attrgetter

import aiohttp
from prettytable import PrettyTable

from fpl import FPL
from fpl.utils import team_converter


def get_gameweek_score(player, gameweek):
    gameweek_history = next(history for history in player.history
                            if history["round"] == gameweek)
    return gameweek_history["total_points"]


def get_gameweek_opponent(player, gameweek):
    gameweek_history = next(history for history in player.history
                            if history["round"] == gameweek)
    return (f"{team_converter(gameweek_history['opponent_team'])} ("
            f"{'H' if gameweek_history['was_home'] else 'A'})")


def get_point_difference(player_a, player_b, gameweek):
    if player_a == player_b:
        return 0

    history_a = next(history for history in player_a.history
                     if history["round"] == gameweek)
    history_b = next(history for history in player_b.history
                     if history["round"] == gameweek)

    return history_a["total_points"] - history_b["total_points"]

async def main(user_id):
    player_table = PrettyTable()
    player_table.field_names = ["Gameweek", "Captain", "Top scorer", "Δ"]
    player_table.align = "r"
    total_difference = 0

    async with aiohttp.ClientSession() as session:
        fpl = FPL(session)
        user = await fpl.get_user(user_id)
        picks = await user.get_picks()

        for i, elements in enumerate(picks):
            gameweek = i + 1
            captain_id = next(player for player in elements
                              if player["is_captain"])["element"]
            players = await fpl.get_players(
                [player["element"] for player in elements],
                include_summary=True)

            captain = next(player for player in players
                           if player.id == captain_id)

            top_scorer = max(
```

(continues on next page)

(continued from previous page)

```
                players, key=lambda x: get_gameweek_score(x, gameweek))

            point_difference = get_point_difference(
                captain, top_scorer, gameweek)

            player_table.add_row([
                gameweek,
                (f"{captain.web_name} - "
                f"{get_gameweek_score(captain, gameweek)} points vs. "
                f"{get_gameweek_opponent(captain, gameweek)}"),
                (f"{top_scorer.web_name} - "
                f"{get_gameweek_score(top_scorer, gameweek)} points vs. "
                f"{get_gameweek_opponent(top_scorer, gameweek)}"),
                point_difference
            ])

            total_difference += point_difference

    print(player_table)
    print(f"Total point difference is {abs(total_difference)} points!")

if __name__ == '__main__':
    asyncio.run(main(3808385))
```

which outputs the following table:

```
+----------+-------------------------------------------+-------------------------------
↪------------+-----+
| Gameweek |                                   Captain |                               ␣
↪ Top scorer |   Δ |
+----------+-------------------------------------------+-------------------------------
↪------------+-----+
|        1 |     Sánchez - 5 points vs. Leicester (H) |         Mané - 16 points vs.␣
↪West Ham (H) | -11 |
|        2 |  Agüero - 20 points vs. Huddersfield (H) |   Agüero - 20 points vs.␣
↪Huddersfield (H) |   0 |
|        3 |         Agüero - 2 points vs. Wolves (A) |     Robertson - 9 points vs.␣
↪Brighton (H) |  -7 |
|        4 |      Agüero - 6 points vs. Newcastle (H) |    Hazard - 11 points vs.␣
↪Bournemouth (H) |  -5 |
|        5 |         Agüero - 7 points vs. Fulham (H) |        Hazard - 20 points vs.␣
↪Cardiff (H) | -13 |
|        6 |        Agüero - 6 points vs. Cardiff (A) |  Wan-Bissaka - 9 points vs.␣
↪Newcastle (H) |  -3 |
|        7 |       Agüero - 8 points vs. Brighton (H) |      Hazard - 10 points vs.␣
↪Liverpool (H) |  -2 |
|        8 |          Kane - 1 points vs. Cardiff (H) |    Hazard - 14 points vs.␣
↪Southampton (A) | -13 |
|        9 |      Sterling - 0 points vs. Burnley (H) |         Mendy - 10 points vs.␣
↪Burnley (H) | -10 |
|       10 |     Robertson - 0 points vs. Cardiff (H) |          Mané - 15 points vs.␣
↪Cardiff (H) | -15 |
|       11 | Sterling - 21 points vs. Southampton (H) |  Sterling - 21 points vs.␣
↪Southampton (H) |   0 |
|       12 |           Mané - 3 points vs. Fulham (H) |      Robertson - 12 points vs.
↪ Fulham (H) |  -9 |
|       13 |    Sterling - 16 points vs. West Ham (A) |     Sterling - 16 points vs.␣
↪West Ham (A) |   0 |
```

(continues on next page)

(continued from previous page)

```
|       14 |  Sterling - 9 points vs. Bournemouth (H) |   Sterling - 9 points vs.␣
↪Bournemouth (H) |   0 |
|       15 |          Sané - 7 points vs. Watford (A) |   Fraser - 12 points vs.␣
↪Huddersfield (H) |  -5 |
|       16 |        Kane - 1 points vs. Leicester (A) | Robertson - 11 points vs.␣
↪Bournemouth (A) | -10 |
|       17 |          Kane - 5 points vs. Burnley (H) |       Hazard - 13 points vs.␣
↪Brighton (A) |  -8 |
|       18 |   Sané - 2 points vs. Crystal Palace (H) |          Kane - 15 points vs.␣
↪Everton (A) | -13 |
|       19 |      Kane - 6 points vs. Bournemouth (H) |        Hazard - 15 points vs.␣
↪Watford (A) |  -9 |
|       20 |           Kane - 6 points vs. Wolves (H) |     Pogba - 18 points vs.␣
↪Bournemouth (H) | -12 |
|       21 |    Hazard - 3 points vs. Southampton (H) |        Fraser - 12 points vs.␣
↪Watford (H) |  -9 |
|       22 |       Salah - 11 points vs. Brighton (A) |     Digne - 12 points vs.␣
↪Bournemouth (H) |  -1 |
|       23 | Salah - 15 points vs. Crystal Palace (H) |  Salah - 15 points vs.␣
↪Crystal Palace (H) |   0 |
+----------+------------------------------------------+------------------------------
↪------------+-----+
Total point difference is 155 points!
```

CHAPTER 2

The Class Documentation / Guide

This part of the documentation is for people who want or need more information bout specific functions and classes found in **fpl**.

2.1 ClassicLeague

Information for the *ClassicLeague* is taken from e.g. the following endpoint:

> https://fantasy.premierleague.com/drf/leagues-classic-standings/1137

An example of what information a *ClassicLeague* contains is shown below:


```
{
  "new_entries": {
    "has_next": false,
    "number": 1,
    "results": [
      {
        "id": 42289277,
        "entry_name": "Atl\u00e9tico Alitera\u00e7\u00e3o",
        "player_first_name": "Liam",
        "player_last_name": "O`Brien",
        "joined_time": "2019-01-21T13:41:56Z",
        "entry": 2513270,
        "league": 1137
      },
      ...,
      {
        "id": 42313251,
        "entry_name": "restnowmywarrior",
        "player_first_name": "Daniel",
        "player_last_name": "Trudgill",
        "joined_time": "2019-01-23T11:44:00Z",
        "entry": 952466,
```


(continues on next page)

(continued from previous page)


```
        "league": 1137
      }
    ]
  },
  "league": {
    "id": 1137,
    "leagueban_set": [

    ],
    "name": "Official \/r\/FantasyPL Classic League",
    "short_name": null,
    "created": "2018-07-05T15:01:19Z",
    "closed": false,
    "forum_disabled": false,
    "make_code_public": false,
    "rank": null,
    "size": null,
    "league_type": "x",
    "_scoring": "c",
    "reprocess_standings": false,
    "admin_entry": 3027,
    "start_event": 1
  },
  "standings": {
    "has_next": true,
    "number": 1,
    "results": [
      {
        "id": 34680858,
        "entry_name": "Vaulen Tigers",
        "event_total": 72,
        "player_name": "Tore Bj\u00f8rheim",
        "movement": "same",
        "own_entry": false,
        "rank": 1,
        "last_rank": 1,
        "rank_sort": 1,
        "total": 1580,
        "entry": 226251,
        "league": 1137,
        "start_event": 1,
        "stop_event": 38
      },
      ...,
      {
        "id": 22006870,
        "entry_name": "( \u0361\u00b0 \u035c\u0296 \u0361\u00b0)",
        "event_total": 65,
        "player_name": "Amos Bastian",
        "movement": "down",
        "own_entry": true,
        "rank": 2185,
        "last_rank": 1943,
        "rank_sort": 2192,
        "total": 1404,
        "entry": 3808385,
        "league": 1137,
```


(continues on next page)

(continued from previous page)

```
        "start_event": 1,
        "stop_event": 38
      }
    ]
  },
  "update_status": 0
}
```

class fpl.models.classic_league.**ClassicLeague**(*league_information*, *session*)

A class representing a classic league in the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         await fpl.login()
...         classic_league = await fpl.get_classic_league(1137)
...     print(classic_league)
...
>>> asyncio.run(main())
Official /r/FantasyPL Classic League - 1137
```

get_standings(*page=1*, *page_new_entries=1*, *phase=1*)

Returns the league's standings of the given page.

Information is taken from e.g.: https://fantasy.premierleague.com/api/leagues-classic/967/standings/?page_new_entries=1&page_standings=1&phase=1

Parameters **page** (*string or int*) – A page of the league's standings (default is 50 managers per page).

Return type dict

2.2 Fixture

Information for the *Fixture* is taken from e.g. the following endpoints:

https://fantasy.premierleague.com/drf/fixtures/ https://fantasy.premierleague.com/drf/fixtures/?event=1

An example of what information a *Fixture* contains is shown below:

```
{
  "id": 6,
  "kickoff_time_formatted": "10 Aug 20:00",
  "started": true,
  "event_day": 1,
  "deadline_time": "2018-08-10T18:00:00Z",
  "deadline_time_formatted": "10 Aug 19:00",
  "stats": [...],
  "team_h_difficulty": 3,
  "team_a_difficulty": 4,
```

(continues on next page)

(continued from previous page)


```
  "code": 987597,
  "kickoff_time": "2018-08-10T19:00:00Z",
  "team_h_score": 2,
  "team_a_score": 1,
  "finished": true,
  "minutes": 90,
  "provisional_start_time": false,
  "finished_provisional": true,
  "event": 1,
  "team_a": 11,
  "team_h": 14
}
```


class fpl.models.fixture.**Fixture**(*fixture_information*)

A class representing fixtures in the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         fixture = await fpl.get_fixture(1)
...     print(fixture)
...
>>> asyncio.run(main())
Arsenal vs. Man City - 10 Aug 19:00
```

get_assisters()

Returns all players who made an assist in the fixture.

Return type dict

get_bonus(*provisional=False*)

Returns all players who received bonus points in the fixture.

Return type dict

get_bps()

Returns the bonus points of each player.

Return type dict

get_goalscorers()

Returns all players who scored in the fixture.

Return type dict

get_own_goalscorers()

Returns all players who scored an own goal in the fixture.

Return type dict

get_penalty_misses()

Returns all players who missed a penalty in the fixture.

Return type dict

get_penalty_saves()
Returns all players who saved a penalty in the fixture.

Return type dict

get_red_cards()
Returns all players who received a red card in the fixture.

Return type dict

get_saves()
Returns all players who made a save in the fixture.

Return type dict

get_yellow_cards()
Returns all players who received a yellow card in the fixture.

Return type dict

2.3 FPL

The *FPL* class is the main class used for interacting with Fantasy Premier League's API. It requires a `aiohttp.ClientSession` for sending requests, so typical usage of the *FPL* class can look something like this:

```
import asyncio
import aiohttp
from fpl import FPL


async def main():
    async with aiohttp.ClientSession() as session:
        fpl = FPL(session)
        await fpl.login()
        user = await fpl.get_user(3808385)
        my_team = await user.get_team()

    print(my_team)

asyncio.run(main())
```

class fpl.fpl.**FPL**(*session*)
The FPL class.

FDR()
Creates a new Fixture Difficulty Ranking (FDR) based on the number of points each team gives up to players in the Fantasy Premier League. These numbers are also between 1.0 and 5.0 to give a similar ranking system to the official FDR.

An example:

```
{
  "Man City": {
      "all": {
      "H": 4.4524439427082,
      "A": 5
      },
      "goalkeeper": {
```

(continues on next page)

(continued from previous page)


```
        "H": 3.6208195949129,
        "A": 5
        },
        "defender": {
        "H": 3.747999604078,
        "A": 5
        },
        "midfielder": {
        "H": 4.6103045986504,
        "A": 5
        },
        "forward": {
        "H": 5,
        "A": 3.9363219561895
        }
    },
    ...,
    "Arsenal": {
        "all": {
        "H": 3.4414041151234,
        "A": 4.2904529162594
        },
        "goalkeeper": {
        "H": 4.1106924163919,
        "A": 4.3867595818815
        },
        "defender": {
        "H": 3.6720291204673,
        "A": 4.3380917450181
        },
        "midfielder": {
        "H": 3.3537357534825,
        "A": 4.0706443384718
        },
        "forward": {
        "H": 2.5143403441683,
        "A": 4.205298013245
        }
    }
}
```


Return type dict

get_classic_league(*league_id*, *return_json=False*)

Returns the classic league with the given `league_id`. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/leagues-classic/967/standings/

Parameters

- **league_id** (*string or int*) – A classic league's ID.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `ClassicLeague` object. Defaults to `False`.

Return type `ClassicLeague` or `dict`

get_fixture(*fixture_id*, *return_json=False*)
Returns the fixture with the given `fixture_id`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/fixtures/ https://fantasy.premierleague.com/api/fixtures/?event=1

Parameters

- **fixture_id** (*int*) – The fixture's ID.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `Fixture` object. Defaults to `False`.

Return type `Fixture` or `dict`

Raises **ValueError** – if fixture with `fixture_id` not found

get_fixtures(*return_json=False*)
Returns a list of *all* fixtures.

Information is taken from e.g.: https://fantasy.premierleague.com/api/fixtures/ https://fantasy.premierleague.com/api/fixtures/?event=1

Parameters **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of `dict``s, if ``False` returns a list of `Fixture` objects. Defaults to `False`.

Return type list

get_fixtures_by_gameweek(*gameweek*, *return_json=False*)
Returns a list of all fixtures of the given `gameweek`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/fixtures/ https://fantasy.premierleague.com/api/fixtures/?event=1

Parameters

- **gameweek** (*string or int*) – A gameweek.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of `dict``s, if ``False` returns a list of `Player` objects. Defaults to `False`.

Return type list

get_fixtures_by_id(*fixture_ids*, *return_json=False*)
Returns a list of all fixtures with IDs included in the *fixture_ids* list.

Information is taken from e.g.: https://fantasy.premierleague.com/api/fixtures/ https://fantasy.premierleague.com/api/fixtures/?event=1

Parameters

- **fixture_ids** (*list*) – A list of fixture IDs.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of `dict``s, if ``False` returns a list of `Fixture` objects. Defaults to `False`.

Return type list

get_gameweek(*gameweek_id*, *include_live=False*, *return_json=False*)
Returns the gameweek with the ID `gameweek_id`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/bootstrap-static/ https://fantasy.premierleague.com/api/event/1/live/

Parameters

- **gameweek_id** (*int*) – A gameweek's ID.
- **include_summary** (*bool*) – (optional) Includes a gameweek's live data if `True`.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `Gameweek` object. Defaults to `False`.

Return type `Gameweek` or `dict`

get_gameweeks (*gameweek_ids=None*, *include_live=False*, *return_json=False*)

Returns either a list of *all* gamweeks, or a list of gameweeks whose IDs are in the `gameweek_ids` list.

Information is taken from e.g.: https://fantasy.premierleague.com/api/bootstrap-static/ https://fantasy.premierleague.com/api/event/1/live/

Parameters

- **gameweek_ids** (*list*) – (optional) A list of gameweek IDs.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of `dict``s, if ``False` returns a list of `Gameweek` objects. Defaults to `False`.

Return type list

get_h2h_league (*league_id*, *return_json=False*)

Returns a *H2HLeague* object with the given *league_id*. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/leagues-h2h-matches/league/946125/

Parameters

- **league_id** (*string or int*) – A H2H league's ID.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `H2HLeague` object. Defaults to `False`.

Return type `H2HLeague` or `dict`

get_player (*player_id*, *players=None*, *include_summary=False*, *return_json=False*)

Returns the player with the given `player_id`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/bootstrap-static/ https://fantasy.premierleague.com/api/element-summary/1/ (optional)

Parameters

- **player_id** (*string or int*) – A player's ID.
- **players** (*list*) – (optional) A list of players.
- **include_summary** (*bool*) – (optional) Includes a player's summary if `True`.
- **return_json** – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `Player` object. Defaults to `False`.

Return type `Player` or `dict`

Raises **ValueError** – Player with player_id not found

get_player_summaries(*player_ids*, *return_json=False*)
Returns a list of summaries of players whose ID are in the player_ids list.

Information is taken from e.g.: https://fantasy.premierleague.com/api/element-summary/1/

Parameters

- **player_ids** (*list*) – A list of player IDs.
- **return_json** (*bool*) – (optional) Boolean. If True returns a list of dict``s, if ``False returns a list of PlayerSummary objects. Defaults to False.

Return type list

get_player_summary(*player_id*, *return_json=False*)
Returns a summary of the player with the given player_id.

Information is taken from e.g.: https://fantasy.premierleague.com/api/element-summary/1/

Parameters

- **player_id** (*int*) – A player's ID.
- **return_json** (*bool*) – (optional) Boolean. If True returns a dict, if False returns a PlayerSummary object. Defaults to False.

Return type PlayerSummary or dict

get_players(*player_ids=None*, *include_summary=False*, *return_json=False*)
Returns either a list of *all* players, or a list of players whose IDs are in the given player_ids list.

Information is taken from e.g.: https://fantasy.premierleague.com/api/bootstrap-static/ https://fantasy.premierleague.com/api/element-summary/1/ (optional)

Parameters

- **player_ids** (*list*) – (optional) A list of player IDs
- **include_summary** (*boolean*) – (optional) Includes a player's summary if True.
- **return_json** (*bool*) – (optional) Boolean. If True returns a list of dict``s, if ``False returns a list of Player objects. Defaults to False.

Return type list

get_points_against()
Returns a dictionary containing the points scored against all teams in the Premier League, split by position and location.

An example:

```
{
  "Man City": {
      "all": {
      "H": [3, ..., 1],
      "A": [2, ..., 2]
      },
      "goalkeeper": {
      "H": [3, ..., 3],
```

(continues on next page)

(continued from previous page)

```
      "A": [2, ..., 3]
      },
      "defender": {
      "H": [1, ..., 2],
      "A": [4, ..., 1]
      },
      "midfielder": {
      "H": [2, ..., 1],
      "A": [2, ..., 2]
      },
      "forward": {
      "H": [1, ..., 2],
      "A": [6, ..., 1]
      }
  },
  ...
}
```

Return type dict

get_team(*team_id*, *return_json=False*)

Returns the team with the given `team_id`.

Information is taken from: https://fantasy.premierleague.com/api/bootstrap-static/

Parameters

- **team_id** (*string or int*) – A team's ID.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `Team` object. Defaults to `False`.

Return type `Team` or `dict`

For reference here is the mapping from team ID to team name:

```
 1 - Arsenal
 2 - Bournemouth
 3 - Brighton
 4 - Burnley
 5 - Cardiff
 6 - Chelsea
 7 - Crystal Palace
 8 - Everton
 9 - Fulham
10 - Huddersfield
11 - Leicester
12 - Liverpool
13 - Man City
14 - Man Utd
15 - Newcastle
16 - Southampton
17 - Spurs
18 - Watford
19 - West Ham
20 - Wolves
```

get_teams (*team_ids=None*, *return_json=False*)
Returns either a list of *all* teams, or a list of teams with IDs in the optional `team_ids` list.

Information is taken from: https://fantasy.premierleague.com/api/bootstrap-static/

Parameters

- **team_ids** (*list*) – (optional) List containing the IDs of teams. If not set a list of *all* teams will be returned.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of `dict``s, if ``False` returns a list of `Team` objects. Defaults to `False`.

Return type list

get_user (*user_id=None*, *return_json=False*)
Returns the user with the given `user_id`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/

Parameters

- **user_id** (*string or int*) – A user's ID.
- **return_json** (*bool*) – (optional) Boolean. If `True` returns a `dict`, if `False` returns a `User` object. Defaults to `False`.

Return type `User` or *dict*

login (*email=None*, *password=None*)
Returns a requests session with FPL login authentication.

Parameters

- **email** (*string*) – Email address for the user's Fantasy Premier League account.
- **password** (*string*) – Password for the user's Fantasy Premier League account.

2.4 Gameweek

Information for the *Gameweek* is taken from e.g. the following endpoints:

https://fantasy.premierleague.com/drf/events/ https://fantasy.premierleague.com/drf/event/1/live

An example of what information a *Gameweek* is too large to show, so it is recommended you check about the above two endpoints yourself to get an idea.

class fpl.models.gameweek.**Gameweek** (*gameweek_information*)
A class representing a gameweek of the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         gameweek = await fpl.get_gameweek(1)
```

(continues on next page)

(continued from previous page)

```
...     print(gameweek)
...
>>> asyncio.run(main())
Gameweek 1 - 10 Aug 19:00
```

2.5 H2HLeague

Information for the *H2HLeague* is taken from the following endpoints:

> https://fantasy.premierleague.com/drf/leagues-h2h-standings/829116 https://fantasy.premierleague.com/drf/leagues-entries-and-h2h-matches/829116/?page=1

An example of what information a *H2HLeague* contains is shown below:


```
{
  "new_entries": {
    "has_next": false,
    "number": 1,
    "results": [

    ]
  },
  "league": {
    "id": 829116,
    "leagueban_set": [

    ],
    "name": "League 829116",
    "has_started": true,
    "can_delete": false,
    "short_name": null,
    "created": "2018-08-09T18:10:37Z",
    "closed": true,
    "forum_disabled": false,
    "make_code_public": false,
    "rank": null,
    "size": null,
    "league_type": "c",
    "_scoring": "h",
    "ko_rounds": 2,
    "admin_entry": null,
    "start_event": 1
  },
  "standings": {
    "has_next": false,
    "number": 1,
    "results": [
      {
        "id": 1230859,
        "entry_name": "fcjeff",
        "player_name": "Khalid Jeffal",
        "movement": "same",
        "own_entry": false,
        "rank": 1,
        "last_rank": 1,
```


(continues on next page)

(continued from previous page)


```
        "rank_sort": 1,
        "total": 0,
        "matches_played": 23,
        "matches_won": 16,
        "matches_drawn": 1,
        "matches_lost": 6,
        "points_for": 1330,
        "points_against": 0,
        "points_total": 49,
        "division": 141015,
        "entry": 21127
      },
      ...,
      {
        "id": 1230854,
        "entry_name": "Wilson-fc",
        "player_name": "Liam Wilson",
        "movement": "same",
        "own_entry": false,
        "rank": 20,
        "last_rank": 20,
        "rank_sort": 20,
        "total": 0,
        "matches_played": 23,
        "matches_won": 6,
        "matches_drawn": 1,
        "matches_lost": 16,
        "points_for": 1115,
        "points_against": 0,
        "points_total": 19,
        "division": 141015,
        "entry": 3649536
      }
    ]
  },
  "matches_next": {
    "has_next": false,
    "number": 1,
    "results": [
      {
        "id": 33164099,
        "entry_1_entry": 3651588,
        "entry_1_name": "subi",
        "entry_1_player_name": "subi ebrahim",
        "entry_2_entry": 3648783,
        "entry_2_name": "Baugveien FC",
        "entry_2_player_name": "Nina Simonsen",
        "is_knockout": false,
        "winner": null,
        "tiebreak": null,
        "own_entry": false,
        "entry_1_points": 0,
        "entry_1_win": 0,
        "entry_1_draw": 0,
        "entry_1_loss": 0,
        "entry_2_points": 0,
        "entry_2_win": 0,
```


(continues on next page)

(continued from previous page)


```
        "entry_2_draw": 0,
        "entry_2_loss": 0,
        "entry_1_total": 0,
        "entry_2_total": 0,
        "seed_value": null,
        "event": 24
      },
      ...,
      {
        "id": 33164094,
        "entry_1_entry": 367548,
        "entry_1_name": "Spartans fc",
        "entry_1_player_name": "Shehryar Gaba",
        "entry_2_entry": 303318,
        "entry_2_name": "Red Devils",
        "entry_2_player_name": "Ajay Bhullar",
        "is_knockout": false,
        "winner": null,
        "tiebreak": null,
        "own_entry": false,
        "entry_1_points": 51,
        "entry_1_win": 1,
        "entry_1_draw": 0,
        "entry_1_loss": 0,
        "entry_2_points": 39,
        "entry_2_win": 0,
        "entry_2_draw": 0,
        "entry_2_loss": 1,
        "entry_1_total": 3,
        "entry_2_total": 0,
        "seed_value": null,
        "event": 23
      }
    ]
  }
}
```


class fpl.models.h2h_league.**H2HLeague**(*league_information*, *session*)

A class representing a H2H league in the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         await fpl.login()
...         h2h_league = await fpl.get_h2h_league(760869)
...     print(h2h_league)
...
>>> asyncio.run(main())
League 760869 - 760869
```

get_fixtures(*gameweek=None*, *page=1*)

Returns a list of fixtures / results of the H2H league.

Information is taken from e.g.: https://fantasy.premierleague.com/api/leagues-h2h-matches/league/946125/?page=1

Parameters

- **gameweek** (*string or int*) – (optional) The gameweek of the fixtures / results.
- **page** (*string or int*) – (optional) The fixtures / results page.

Return type list

2.6 Player

Information for the *Player* is taken from e.g. the following endpoints:

https://fantasy.premierleague.com/drf/elements https://fantasy.premierleague.com/drf/element-summary/1 (optional)

The information from the latter endpoint is only included when `include_summary` is `True`.

An example of what information a *Player* (Petr Cech) contains is shown below (without summary included):


```
{
  "id": 1,
  "photo": "11334.jpg",
  "web_name": "Cech",
  "team_code": 3,
  "status": "a",
  "code": 11334,
  "first_name": "Petr",
  "second_name": "Cech",
  "squad_number": 1,
  "news": "",
  "now_cost": 48,
  "news_added": "2018-09-29T17:31:14Z",
  "chance_of_playing_this_round": 100,
  "chance_of_playing_next_round": 100,
  "value_form": "0.0",
  "value_season": "5.0",
  "cost_change_start": -2,
  "cost_change_event": 0,
  "cost_change_start_fall": 2,
  "cost_change_event_fall": 0,
  "in_dreamteam": false,
  "dreamteam_count": 0,
  "selected_by_percent": "1.2",
  "form": "0.0",
  "transfers_out": 130119,
  "transfers_in": 81105,
  "transfers_out_event": 644,
  "transfers_in_event": 122,
  "loans_in": 0,
  "loans_out": 0,
  "loaned_in": 0,
  "loaned_out": 0,
  "total_points": 24,
```


(continues on next page)

(continued from previous page)


```
  "event_points": 0,
  "points_per_game": "3.4",
  "ep_this": "0.0",
  "ep_next": "1.0",
  "special": false,
  "minutes": 585,
  "goals_scored": 0,
  "assists": 0,
  "clean_sheets": 1,
  "goals_conceded": 9,
  "own_goals": 0,
  "penalties_saved": 0,
  "penalties_missed": 0,
  "yellow_cards": 0,
  "red_cards": 0,
  "saves": 27,
  "bonus": 3,
  "bps": 130,
  "influence": "205.0",
  "creativity": "0.0",
  "threat": "0.0",
  "ict_index": "20.4",
  "ea_index": 0,
  "element_type": 1,
  "team": 1
}
```


class fpl.models.player.**Player**(*player_information*, *session*)

A class representing a player in the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         player = await fpl.get_player(302)
...     print(player)
...
>>> asyncio.run(main())
Pogba - Midfielder - Man Utd
```

games_played

The number of games where the player has played at least 1 minute.

Return type int

pp90

Points per 90 minutes.

Return type float

vapm

Value added per million An explanation of VAPM can be found here:

https://www.reddit.com/r/FantasyPL/comments/6r60fu/exploring_a_key_metric_value_added_

per_1m/

Return type float

2.7 Team

Information for the *Team* is taken from the following endpoint:

https://fantasy.premierleague.com/drf/teams

An example of what information a *Team* (Manchester United) contains is shown below:


```
{
  "id": 14,
  "current_event_fixture": [
    {
      "is_home": true,
      "month": 1,
      "event_day": 1,
      "id": 226,
      "day": 19,
      "opponent": 3
    }
  ],
  "next_event_fixture": [
    {
      "is_home": true,
      "month": 1,
      "event_day": 1,
      "id": 235,
      "day": 29,
      "opponent": 4
    }
  ],
  "name": "Man Utd",
  "code": 1,
  "short_name": "MUN",
  "unavailable": false,
  "strength": 4,
  "position": 0,
  "played": 0,
  "win": 0,
  "loss": 0,
  "draw": 0,
  "points": 0,
  "form": null,
  "link_url": "",
  "strength_overall_home": 1280,
  "strength_overall_away": 1290,
  "strength_attack_home": 1250,
  "strength_attack_away": 1260,
  "strength_defence_home": 1310,
  "strength_defence_away": 1340,
  "team_division": 1
}
```

class fpl.models.team.**Team**(*team_information*, *session*)

A class representing a real team in the Fantasy Premier League.

Basic usage:

```
>>> from fpl import FPL
>>> import aiohttp
>>> import asyncio
>>>
>>> async def main():
...     async with aiohttp.ClientSession() as session:
...         fpl = FPL(session)
...         team = await fpl.get_team(14)
...     print(team)
...
>>> asyncio.run(main())
Man Utd
```

get_fixtures(*return_json=False*)

Returns a list containing the team's fixtures.

Parameters **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of dicts, if `False` returns a list of TeamFixture objects. Defaults to `False`.

Return type list

get_players(*return_json=False*)

Returns a list containing the players who play for the team. Does not include the player's summary.

Parameters **return_json** (*bool*) – (optional) Boolean. If `True` returns a list of dicts, if `False` returns a list of Player objects. Defaults to `False`.

Return type list

2.8 User

Information for the *User* is taken from the following endpoint:

https://fantasy.premierleague.com/drf/entry/3808385

An example of what information a *User* contains is shown below:


```
{
  "entry": {
    "id": 3808385,
    "player_first_name": "Amos",
    "player_last_name": "Bastian",
    "player_region_id": 152,
    "player_region_name": "Netherlands",
    "player_region_short_iso": "NL",
    "summary_overall_points": 1404,
    "summary_overall_rank": 49225,
    "summary_event_points": 65,
    "summary_event_rank": 1480275,
    "joined_seconds": 18267,
    "current_event": 23,
    "total_transfers": 21,
    "total_loans": 0,
```


(continues on next page)

(continued from previous page)


```
    "total_loans_active": 0,
    "transfers_or_loans": "transfers",
    "deleted": false,
    "email": false,
    "joined_time": "2018-08-09T22:44:21Z",
    "name": "( \u0361\u00b0 \u035c\u0296 \u0361\u00b0)",
    "bank": 22,
    "value": 1037,
    "kit": "{\"kit_shirt_type\":\"plain\",\"kit_shirt_base\":\"#ff0000\",\"kit_shirt_
↪sleeves\":\"#ff0000\",\"kit_shirt_secondary\":\"#e1e1e1\",\"kit_shirt_logo\":\"none\
↪",\"kit_shorts\":\"#000000\",\"kit_socks_type\":\"plain\",\"kit_socks_base\":\"
↪#ffffff\",\"kit_socks_secondary\":\"#e1e1e1\"}",
    "event_transfers": 0,
    "event_transfers_cost": 0,
    "extra_free_transfers": 0,
    "strategy": null,
    "favourite_team": 14,
    "started_event": 1,
    "player": 7425806
  },
  "leagues": {
    "cup": [

    ],
    "h2h": [

    ],
    "classic": [
      {
        "id": 14,
        "entry_rank": 8454,
        "entry_last_rank": 7255,
        "entry_movement": "down",
        "entry_change": 1199,
        "entry_can_leave": false,
        "entry_can_admin": false,
        "entry_can_invite": false,
        "entry_can_forum": false,
        "entry_code": null,
        "name": "Man Utd",
        "short_name": "team-14",
        "created": "2018-07-05T12:12:23Z",
        "closed": false,
        "forum_disabled": false,
        "make_code_public": false,
        "rank": null,
        "size": null,
        "league_type": "s",
        "_scoring": "c",
        "reprocess_standings": false,
        "admin_entry": null,
        "start_event": 1
      },
      ...,
      {
        "id": 890172,
        "entry_rank": 2,
```


(continues on next page)

(continued from previous page)


```
        "entry_last_rank": 2,
        "entry_movement": "same",
        "entry_change": null,
        "entry_can_leave": true,
        "entry_can_admin": false,
        "entry_can_invite": false,
        "entry_can_forum": true,
        "entry_code": null,
        "name": "AJ's Angels",
        "short_name": null,
        "created": "2018-08-10T08:15:37Z",
        "closed": false,
        "forum_disabled": false,
        "make_code_public": false,
        "rank": null,
        "size": null,
        "league_type": "x",
        "_scoring": "c",
        "reprocess_standings": false,
        "admin_entry": 9346,
        "start_event": 1
      }
    ]
  }
}
```


class fpl.models.user.**User**(*user_information*, *session*)

A class representing a user of the Fantasy Premier League.

```
>>> from fpl import FPL
  >>> import aiohttp
  >>> import asyncio
  >>>
  >>> async def main():
  ...     async with aiohttp.ClientSession() as session:
  ...         fpl = FPL(session)
  ...         user = await fpl.get_user(3808385)
  ...     print(user)
  ...
  >>> asyncio.run(main())
  Amos Bastian - Netherlands
```

captain(*captain*)

Set the captain of the user's team.

Parameters **captain** (*int*) – ID of the captain.

get_active_chips(*gameweek=None*)

Returns a list containing the user's active chip for each gameweek, or the active chip of the given gameweek.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/event/1/picks/

Parameters **gameweek** – (optional): The gameweek. Defaults to None.

Return type list

get_automatic_substitutions(*gameweek=None*)
Returns a list containing the user's automatic substitutions each gameweek.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/event/1/picks/

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type list

get_chips()
Returns a logged in user's list of chips. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/my-team/91928/

Return type list

get_chips_history(*gameweek=None*)
Returns a list containing the chip history of the user.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/history

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type list

get_cup_matches(*gameweek=None*)
Returns either a list of all the user's cup matches, dictionary of the cup match in the given gameweek (gameweek 17 and onwards).

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/cup/

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type list or dict

get_cup_status()
Returns the user's cup status.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/cup/

Return type dict

get_gameweek_history(*gameweek=None*)
Returns a list containing the gameweek history of the user.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/history

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type list if gameweek is `None`, otherwise dict.

get_latest_transfers()
Returns a list of transfers made by the user in the current gameweek. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/transfers-latest/

Return type list

get_picks(*gameweek=None*)
Returns a dict containing the user's picks each gameweek.

Key is the gameweek number, value contains picks of the gameweek.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/event/1/picks/

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type dict

get_season_history()
Returns a list containing the seasonal history of the user.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/history

Return type list

get_team()
Returns a logged in user's current team. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/my-team/91928/

Return type list

get_transfers(*gameweek=None*)
Returns either a list of all the user's transfers, or a list of transfers made in the given gameweek.

Information is taken from e.g.: https://fantasy.premierleague.com/api/entry/91928/transfers/

Parameters **gameweek** – (optional): The gameweek. Defaults to `None`.

Return type list

get_transfers_status()
Returns a logged in user's transfer status, which is a dictionary containing their bank value, how many free transfers they have left and so on. Requires the user to have logged in using `fpl.login()`.

Information is taken from e.g.: https://fantasy.premierleague.com/api/my-team/91928/

Return type dict

get_user_history(*gameweek=None*)
Returns a list containing the user's history for each gameweek, or a dictionary of the user's history for the given gameweek.

Return type list or dict

get_watchlist()
Returns the user's watchlist. Requires the user to have logged in using `fpl.login()`.

Information is taken from here: https://fantasy.premierleague.com/api/me/

Return type list

substitute(*players_in*, *players_out*, *captain=None*, *vice_captain=None*)
Substitute players on the bench for players in the starting eleven. Also allows the user to simultaneously set the new (vice) captain(s). A maximum of 4 substitutes is set to force proper usage.

Parameters

- **players_in** (*list*) – List of IDs of players who will be substituted in.
- **players_out** (*list*) – List of IDS of players who will be substituted out.
- **captain** – ID of the captain, defaults to None.
- **captain** – int, optional
- **vice_captain** – ID of the vice captain, defaults to None.
- **vice_captain** – int, optional

transfer(*players_out*, *players_in*, *max_hit=60*, *wildcard=False*, *free_hit=False*)
Transfers given players out and transfers given players in.

Parameters

- **players_out** (*list*) – List of IDs of players who will be transferred out.
- **players_in** (*list*) – List of IDs of players who will be transferred in.
- **max_hit** – Maximum hit that should be taken by making the transfer(s), defaults to 60
- **max_hit** – int, optional
- **wildcard** – Boolean for playing wildcard, defaults to False
- **wildcard** – bool, optional
- **free_hit** – Boolean for playing free hit, defaults to False
- **free_hit** – bool, optional

Returns Returns the response given by a succesful transfer.

Return type dict

vice_captain(*vice_captain*)
Set the vice captain of the user's team.

Parameters **vice_captain** (*int*) – ID of the vice captain.

CHAPTER 3

The Contributor Guide

If you want to help **fpl** out and contribute to the project, be it via development, suggestions, hunting bugs etc. then this part of the documentation is for you!

3.1 Contributing

If you're reading this, then you're probably interested in helping out with the development of *fpl*! On this page you will be able to find information that *should* make it easier for you to start contributing. Since contributions can be in all kinds of different forms, the contributing guide has been split up into sections.

To contact me directly you can send an email to amosbastian@gmail.com. If you are looking for other people interested in FPL related programming, then you can also join our Discord server.

3.1.1 Code contributions

Submitting code

When contributing code, you'll want to follow this checklist:

1. Fork the repository on GitHub.
2. Run the tests with *pytest tests/* to confirm they all pass on your system. If the tests fail, then try and find out why this is happening. If you aren't able to do this yourself, then don't hesitate to either create an issue on GitHub (see *Reporting bugs*), contact me on Discord or send an email to amosbastian@gmail.com.
3. Either create your feature and then write tests for it, or do this the other way around.
4. Run all tests again with with *pytest tests/* to confirm that everything still passes, including your newly added test(s).
5. Create a pull request for the main repository's `master` branch.

If you want, you can also add your name AUTHORS.

Code review

Currently I am the only maintainer of this project. Because of this I will review each pull request myself and provide feedback if necessary. I would like this to happen in a clear and calm manner (from both sides)!

New contributors

If you are new or relatively new to contributing to open source projects, then please don't hesitate to contact me directly! I am more than willing to help out, and will try and assign issues to you if possible.

Code style

The *fpl* package follows PEP 8 code style. Currently there is only one specific additions to this, but if you think more should be added, then this can always be discussed.

- Always use double-quoted strings, unless it is not possible.

3.1.2 Documentation contributions

Documentation improvements and suggestions are always welcome! The documentation files live in the `docs/` directory. They're written in reStructuredText, and use Sphinx to generate the full suite of documentation.

Of course the documentation doesn't have to be too serious, but try and keep it semi-formal.

3.1.3 Reporting bugs

If you encounter any bugs while using **fpl** then please don't hesitate to open an issue. However, before you do, please check the GitHub issues (make sure to also check closed ones) to see if the bug has already been reported.

A template is provided below to make it easier to understand the issue:

```
#### Expected behaviour
What did you expect to happen?

#### Actual behaviour
What actually happened?

#### How to reproduce
When did it happen? Include a code snippet if possible!
```

3.1.4 Feature requests

Currently **fpl** is in active development, so feature requests are more than welcome. If you have any ideas for features you'd like to see added, then simply create an issue with an **enhancement** label.

3.2 Authors

3.2.1 Maintainer

- Amos Bastian <amosbastian@gmail.com> @amosbastian,

3.2.2 Contributors

- David MacLeod

Python Module Index

f

Index

H

L

P

S

T

U

V