
Understat Documentation

Release 0.1.1

Amos Bastian

Apr 13, 2023

Contents

1	The User Guide	3
1.1	Installing understat	3
2	The Class Documentation / Guide	5
2.1	Understat	5
3	The Contributor Guide	47
3.1	Contributing	47
3.2	Authors	48
	Python Module Index	51
	Index	53

Note: I have nothing to do with Understat, and have simply created this package for fun!

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

If you're interested in helping out the development of **understat**, 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 asyncio
import json

import aiohttp

from understat import Understat

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        player = await understat.get_players(
            "epl", 2018,
            player_name="Paul Pogba",
            team_title="Manchester United"
        )
        print(json.dumps(player))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

>>>[{"id": "1740", "player_name": "Paul Pogba", "games": "27", "time": "2293", "goals": "11", "xG": "13.361832823604345", "assists": "9", "xA": "4.063152700662613", "shots": "87", "key_passes": "40", "yellow_cards": "5", "red_cards": "0", "position": "M S", "team_title": "Manchester United", "npg": "6", "npxG": "7.272482139989734", "xGChain": "17.388037759810686", "xGBuildup": "8.965998269617558"}]
```

With **understat** you can easily get all the data available on [Understat](#)!

This part of the documentation simply shows you have to install **understat**.

1.1 Installing understat

The recommended way to install **understat** is via `pip`.

```
pip install understat
```

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

1.1.1 Updating understat with pip

To update **understat** you can run:

```
pip install --upgrade understat
```

Example output:

```
Installing collected packages: understat
  Found existing installation: understat 0.1.0
    Uninstalling understat-0.1.0:
      Successfully uninstalled understat-0.1.0
Successfully installed understat-0.1.1
```

1.1.2 Installing older versions

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

```
pip install understat==0.1.1
```

1.1.3 Installing from GitHub

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

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

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

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

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

```
$ cd understat
$ pip install .
```

The Class Documentation / Guide

This part of the documentation is for people who want or need more information about specific functions and classes found in **understat**. It includes example output for each of the functions, and also screenshots showing where you would find the equivalent data on [Understat](#).

2.1 Understat

The Understat class is the main, and only class used for interacting with Understat's data. It requires a `aiohttp.ClientSession` for sending requests, so typical usage of the Understat class can look something like this:

```
import asyncio
import json

import aiohttp

from understat import Understat

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        player = await understat.get_league_player(
            "epl", 2018,
            player_name="Paul Pogba",
            team_title="Manchester United"
        )
        print(json.dumps(player))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

>>>[{"id": "1740", "player_name": "Paul Pogba", "games": "27", "time": "2293", "goals": "11", "xG": "13.361832823604345", "assists": "9", "xA": "4.063152700662613", "shots": "87", "key_passes": "40", "yellow_cards": "5", "red_cards": "0", "position": "M S", "team_title": "Manchester United", "npg": "6", "npG": "7.272482139989734", "xGChain": "17.388037759810686", "xGBuildup": "8.965998269617558"}]
```

(continues on next page)

2.1.1 The functions

Below each function of the `Understat` class will be documented separately. It will also show a screenshot of the equivalent data on understat.com, and an example of how the function itself could be used.

Most of the functions come with the `options` keyword argument, and the `**kwargs` magic variable, which means that their output can be filtered (the ways this can be done depends entirely on the output). It was the easiest way to implement something like this, but may not always be optimal (e.g. filtering by home team may require an object for example), and so this could be changed in the future.

If you have any suggestions on what kind of filtering options you'd like to see for certain functions, then you can create an [issue](#) for this. Also, any help with adding better filtering, if necessary, is also very much appreciated!

—
`Understat.get_league_fixtures` (*league_name*, *season*, *options=None*, ***kwargs*)

Returns a list containing information about all the upcoming fixtures of the given league in the given season.

Parameters

- **league_name** (*str*) – The league's name.
- **season** (*str or int*) – The season.
- **options** – Options to filter the data by, defaults to `None`.
- **options** – dict, optional

Returns A list of the fixtures as seen on Understat's league overview.

Return type list

It returns the fixtures (not results) of the given league, in the given season. So for example, the fixtures as seen in the screenshot below

Saturday, March 30, 2019		
Fulham	13:30	Manchester City
Manchester United	16:00	Watford
Leicester	16:00	Bournemouth
Crystal Palace	16:00	Huddersfield
Burnley	16:00	Wolverhampton Wanderers
Brighton	16:00	Southampton
West Ham	18:30	Everton
Arsenal	19:00	Newcastle United
Sunday, March 31, 2019		
Cardiff	15:05	Chelsea
Liverpool	17:30	Tottenham

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get all Manchester United's upcoming fixtures at **home**, then you could do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        fixtures = await understat.get_league_fixtures(
            "epl",
            2018,
            {
                "h": {"id": "89",
                      "title": "Manchester United",
                      "short_title": "MUN"}
            }
        )
        print(json.dumps(fixtures))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  {
    "id": "9501",
    "isResult": false,
    "h": {
      "id": "89",
      "title": "Manchester United",
      "short_title": "MUN"
    },
  },

```

(continues on next page)

(continued from previous page)

```
    "a": {
        "id": "88",
        "title": "Manchester City",
        "short_title": "MCI"
    },
    "goals": {
        "h": null,
        "a": null
    },
    "xG": {
        "h": null,
        "a": null
    },
    "datetime": "2019-03-16 18:00:00"
},
...
{
    "id": "9570",
    "isResult": false,
    "h": {
        "id": "89",
        "title": "Manchester United",
        "short_title": "MUN"
    },
    "a": {
        "id": "227",
        "title": "Cardiff",
        "short_title": "CAR"
    },
    "goals": {
        "h": null,
        "a": null
    },
    "xG": {
        "h": null,
        "a": null
    },
    "datetime": "2019-05-12 17:00:00"
}
]
```

`Understat.get_league_table(league_name, season, with_headers=True, h_a='overall', start_date=None, end_date=None)`

Returns the latest league table of a specified league in a specified year.

Parameters

- **league_name** (*str*) – The league’s name.
- **season** (*str* or *int*) – The season.
- **with_headers** (*bool*) – whether or not to include headers in the returned table.
- **h_a** (*str*) – whether to return the overall table (“overall”), home table (“home”), or away table (“away”).
- **start_date** (*str*) – start date to filter the table by (format: YYYY-MM-DD).

- **end_date** (*str*) – end date of the table to filter the table by (format: YYYY-MM-DD).

Returns List of lists.

Return type list

It returns the standings of the given league in the given year, as seen in the screenshot below

Nº	Team	M	W	D	L	G	GA	PTS	xG	NPxG	xGA	NPxGA	NPxGD	PPDA	OPPPDA	DC	ODC	xPTS
1	Liverpool	38	32	3	3	85	33	99	75.19 ^{-8.61}	71.39	39.57 ^{+8.57}	38.81	+32.58	8.01	21.33	429	145	74.28 ^{-24.72}
2	Manchester City	38	26	3	9	102	35	81	102.21 ^{+0.21}	93.53	37.00 ^{+2.00}	34.71	+58.82	8.49	23.77	547	135	86.76 ^{+5.76}
3	Manchester United	38	18	12	8	66	36	66	66.19 ^{+0.19}	55.53	38.06 ^{+2.08}	35.78	+19.75	9.64	11.10	290	178	70.99 ^{+4.99}
4	Chelsea	38	20	6	12	69	54	66	76.23 ^{+7.23}	70.90	41.09 ^{-12.91}	39.57	+31.33	9.01	14.66	326	169	73.49 ^{+7.49}
5	Leicester	38	18	8	12	67	41	62	61.02 ^{-5.98}	55.67	47.89 ^{+8.89}	40.28	+15.39	7.95	12.75	293	203	61.16 ^{-0.84}
6	Tottenham	38	16	11	11	61	47	59	49.02 ^{-11.98}	45.83	54.13 ^{+7.13}	48.80	-2.97	10.94	10.11	224	264	49.26 ^{-9.74}
7	Wolverhampton Wanderers	38	15	14	9	51	40	59	54.22 ^{+3.22}	51.18	37.39 ^{-2.81}	34.96	+16.22	13.39	10.29	172	193	63.82 ^{+4.82}
8	Arsenal	38	14	14	10	56	48	56	50.82 ^{-5.18}	48.54	57.25 ^{+9.25}	51.01	-2.47	11.02	11.45	265	253	50.15 ^{-5.85}
9	Sheffield United	38	14	12	12	39	39	54	45.81 ^{+8.81}	45.05	52.04 ^{+13.04}	49.76	-4.71	13.71	11.35	198	231	49.34 ^{-4.66}
10	Burnley	38	15	9	14	43	50	54	49.35 ^{+8.35}	47.06	53.84 ^{+3.84}	50.02	-2.95	11.89	9.35	145	263	49.54 ^{-4.46}
11	Southampton	38	15	7	16	51	60	52	56.50 ^{+5.50}	52.54	56.59 ^{-3.41}	53.55	-1.01	8.26	9.97	230	249	56.87 ^{+4.87}
12	Everton	38	13	10	15	44	56	49	53.71 ^{+9.71}	52.95	49.21 ^{-6.79}	46.17	+6.78	10.17	10.38	203	215	55.78 ^{+8.78}
13	Newcastle United	38	11	11	16	38	58	44	36.49 ^{-1.51}	35.73	67.03 ^{+9.03}	65.51	-29.78	19.00	7.55	132	344	31.92 ^{-12.08}
14	Crystal Palace	38	11	10	17	31	50	43	34.45 ^{+3.45}	32.16	57.39 ^{+7.39}	56.63	-24.47	13.00	7.99	225	281	38.26 ^{-4.74}
15	Brighton	38	9	14	15	39	54	41	47.42 ^{+8.42}	45.90	60.42 ^{+8.42}	58.89	-12.99	10.36	12.78	242	264	48.02 ^{+7.02}
16	West Ham	38	10	9	19	49	62	39	49.07 ^{+0.07}	46.02	68.32 ^{+8.32}	63.76	-17.73	13.22	9.50	217	308	38.75 ^{-0.25}
17	Aston Villa	38	9	8	21	41	67	35	45.09 ^{+4.09}	42.65	71.60 ^{+4.60}	66.88	-24.23	12.34	7.89	186	343	37.23 ^{+2.23}
18	Bournemouth	38	9	7	22	40	65	34	44.67 ^{+4.67}	41.63	63.29 ^{-1.71}	58.73	-17.10	13.38	9.15	210	326	39.20 ^{+5.20}
19	Watford	38	8	10	20	36	64	34	48.56 ^{+12.56}	42.47	59.53 ^{-4.47}	52.52	-10.05	12.20	9.64	227	259	47.87 ^{+13.87}
20	Norwich	38	5	6	27	26	75	21	37.23 ^{+11.23}	35.71	71.61 ^{-3.39}	66.13	-30.41	12.59	9.65	207	345	33.12 ^{+12.12}

There are also optional “start_date” and “end_date” arguments, which can be used to get the table from a specific date range from given season, like on screenshot below

Table	Charts	overall	home	away	Aug 11, 2018	Sep 1, 2018					
Nº	Team	M	W	D	L	G	GA	PTS	xG	xGA	xPTS
1	Liverpool	4	4	0	0	9	1	12	10.21 ^{+1.21}	2.29 ^{+1.29}	9.74 ^{-2.26}
2	Chelsea	4	4	0	0	10	3	12	6.69 ^{-3.31}	4.01 ^{+1.01}	7.67 ^{-4.33}
3	Manchester City	4	3	1	0	11	3	10	10.93 ^{-0.07}	2.93 ^{-0.07}	9.98 ^{-0.02}
4	Tottenham	3	3	0	0	8	2	9	7.66 ^{-0.34}	3.82 ^{+1.82}	6.65 ^{-2.35}
5	Watford	3	3	0	0	7	2	9	4.49 ^{-2.51}	2.07 ^{+0.07}	6.13 ^{-2.87}
6	Bournemouth	4	2	1	1	6	5	7	7.83 ^{+1.83}	4.32 ^{-0.68}	8.00 ^{+1.00}
7	Everton	4	1	3	0	7	6	6	3.90 ^{-3.10}	5.91 ^{-0.09}	4.14 ^{-1.86}
8	Leicester	4	2	0	2	6	5	6	3.46 ^{-2.54}	4.38 ^{-0.62}	4.35 ^{-1.65}
9	Wolverhampton Wanderers	4	1	2	1	4	5	5	4.47 ^{+0.47}	3.67 ^{-1.33}	6.29 ^{+1.29}
10	Southampton	4	1	1	2	4	4	4	6.47 ^{+2.47}	6.30 ^{-2.30}	5.39 ^{+1.39}
11	Fulham	4	1	1	2	7	9	4	5.52 ^{-1.48}	8.60 ^{-0.40}	3.50 ^{-0.50}
12	Brighton	4	1	1	2	5	7	4	5.31 ^{+0.31}	6.32 ^{-0.68}	4.63 ^{+0.63}
13	Arsenal	3	1	0	2	5	6	3	4.69 ^{-0.31}	4.92 ^{-1.08}	4.00 ^{+1.00}
14	Manchester United	3	1	0	2	4	7	3	4.48 ^{+0.48}	5.18 ^{-1.82}	3.47 ^{+0.47}
15	Crystal Palace	4	1	0	3	3	6	3	5.02 ^{+2.02}	7.74 ^{-1.74}	4.14 ^{+1.14}
16	Cardiff	3	0	2	1	0	2	2	3.73 ^{+3.73}	4.36 ^{+2.36}	3.73 ^{+1.73}
17	Huddersfield	4	0	2	2	2	10	2	1.92 ^{-0.08}	8.60 ^{-1.40}	2.06 ^{+0.06}
18	Newcastle United	4	0	1	3	3	6	1	3.29 ^{+0.29}	7.70 ^{-1.70}	2.59 ^{+1.59}
19	Burnley	3	0	1	2	3	7	1	3.37 ^{+0.37}	4.86 ^{-2.14}	3.02 ^{+2.02}
20	West Ham	4	0	0	4	2	10	0	3.58 ^{+1.58}	9.04 ^{-0.96}	2.85 ^{+2.85}

An example of getting the standings from the EPL in 2019 can be found below

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        table = await understat.get_league_table("EPL", "2019")
        print(table)

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  ['Team', 'M', 'W', 'D', 'L', 'G', 'GA', 'PTS', 'xG', 'NPxG', 'xGA', 'NPxGA', 'NPxGD',
   ↪ 'PPDA', 'OPPDA', 'DC', 'ODC', 'xPTS'],
  ['Liverpool', 38, 32, 3, 3, 85, 33, 99, 75.19, 71.39, 39.57, 38.81, 32.58, 8.01, 21.
   ↪ 33, 429, 145, 74.28],
  ['Manchester City', 38, 26, 3, 9, 102, 35, 81, 102.21, 93.53, 37.0, 34.71, 58.82, 8.
   ↪ 49, 23.77, 547, 135, 86.76],
  ['Manchester United', 38, 18, 12, 8, 66, 36, 66, 66.19, 55.53, 38.06, 35.78, 19.75,
   ↪ 9.64, 11.1, 290, 178, 70.99],
  ...,
  ['Aston Villa', 38, 9, 8, 21, 41, 67, 35, 45.09, 42.65, 71.6, 66.88, -24.23, 12.34,
   ↪ 7.89, 186, 343, 37.23],
  ['Bournemouth', 38, 9, 7, 22, 40, 65, 34, 44.67, 41.63, 63.29, 58.73, -17.1, 13.38,
   ↪ 9.15, 210, 326, 39.2],
  ['Watford', 38, 8, 10, 20, 36, 64, 34, 48.56, 42.47, 59.53, 52.52, -10.05, 12.2, 9.
   ↪ 64, 227, 259, 47.87],

```

(continues on next page)

(continued from previous page)

```
[ 'Norwich', 38, 5, 6, 27, 26, 75, 21, 37.23, 35.71, 71.61, 66.13, -30.41, 12.59, 9.
↪ 65, 207, 345, 33.12 ]
```

Understat.**get_player_grouped_stats**(*player_id*)

Returns the player with the given ID's grouped stats (as seen at the top of a player's page).

Parameters *player_id* (*int* or *str*) – The player's Understat ID.

Returns Dictionary of the player's grouped stats.

Return type dict

It returns all the statistics of a given player, which includes stuff like their performance per season, position and more. Basically, it's everything that can be found in the table shown in the screenshot below

Season	Position	Situation	Shot zones	Shot types										
№	Season	Team	Apps	Min	G	A	Sh90	KP90	xG	xA	xG90	xA90		
1	2018/2019	Manchester City	26	1960	18	6	4.36	1.15	17.52 -0.48	3.78 -2.22	0.80	0.17		
2	2017/2018	Manchester City	25	1985	21	6	4.31	1.77	18.57 -2.43	6.53 +0.53	0.84	0.30		
3	2016/2017	Manchester City	31	2408	20	3	5.20	1.16	22.67 +2.67	4.67 +1.67	0.85	0.17		
4	2015/2016	Manchester City	30	2399	24	2	4.46	1.01	20.08 -3.92	2.03 +0.03	0.75	0.08		
5	2014/2015	Manchester City	33	2551	26	8	5.22	1.16	25.27 -0.73	5.57 -2.43	0.89	0.20		
			145	11303	109	25	4.75	1.23	104.10 -4.90	22.57 -2.43	0.83	0.18		

An example of getting Sergio Agüero's grouped data can be found below

```
async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        grouped_stats = await understat.get_player_grouped_stats(619)
        print(json.dumps(grouped_stats))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

which outputs (with parts omitted)

```
{
  "season": [
    {
      "position": "FW",
      "games": "26",
      "goals": "18",
      "shots": "95",
      "time": "1960",
      "xG": "17.515484783798456",
      "assists": "6",
      "xA": "3.776376834139228",
      "key_passes": "25",
      "season": "2018",
      "team": "Manchester City",
      "yellow": "3",
```

(continues on next page)

(continued from previous page)

```

    "red": "0",
    "npg": "16",
    "npG": "15.9931472055614",
    "xGChain": "23.326821692287922",
    "xGBuildup": "6.351545065641403"
  },
  ...,
  {
    "position": "Sub",
    "games": "33",
    "goals": "26",
    "shots": "148",
    "time": "2551",
    "xG": "25.270159743726254",
    "assists": "8",
    "xA": "5.568922242149711",
    "key_passes": "33",
    "season": "2014",
    "team": "Manchester City",
    "yellow": "4",
    "red": "0",
    "npg": "21",
    "npG": "20.70318364351988",
    "xGChain": "27.805154908448458",
    "xGBuildup": "6.878173082135618"
  }
],
"position": {
  "2018": {
    "FW": {
      "position": "FW",
      "games": "24",
      "goals": "18",
      "shots": "94",
      "time": "1911",
      "xG": "17.464063242077827",
      "assists": "6",
      "xA": "3.776376834139228",
      "key_passes": "25",
      "season": "2018",
      "yellow": "3",
      "red": "0",
      "npg": "16",
      "npG": "15.94172566384077",
      "xGChain": "23.258203461766243",
      "xGBuildup": "6.334348376840353"
    },
    "Sub": {
      "position": "Sub",
      "games": "2",
      "goals": "0",
      "shots": "1",
      "time": "49",
      "xG": "0.05142154172062874",
      "assists": "0",
      "xA": "0",
      "key_passes": "0",

```

(continues on next page)

(continued from previous page)

```

        "season": "2018",
        "yellow": "0",
        "red": "0",
        "npg": "0",
        "npG": "0.05142154172062874",
        "xGChain": "0.06861823052167892",
        "xGBuildup": "0.017196688801050186"
    }
},
...,
},
"2014": {
    "FW": {
        "position": "FW",
        "games": "30",
        "goals": "24",
        "shots": "142",
        "time": "2504",
        "xG": "24.362012460827827",
        "assists": "8",
        "xA": "5.568922242149711",
        "key_passes": "33",
        "season": "2014",
        "yellow": "4",
        "red": "0",
        "npg": "19",
        "npG": "19.795036360621452",
        "xGChain": "26.94415594637394",
        "xGBuildup": "6.878173082135618"
    },
    "Sub": {
        "position": "Sub",
        "games": "3",
        "goals": "2",
        "shots": "6",
        "time": "47",
        "xG": "0.9081472828984261",
        "assists": "0",
        "xA": "0",
        "key_passes": "0",
        "season": "2014",
        "yellow": "0",
        "red": "0",
        "npg": "2",
        "npG": "0.9081472828984261",
        "xGChain": "0.8609989620745182",
        "xGBuildup": "0"
    }
},
},
"situation": {
    "2015": {
        "OpenPlay": {
            "situation": "OpenPlay",
            "season": "2015",
            "goals": "17",
            "shots": "97",

```

(continues on next page)

(continued from previous page)

```

        "xG": "13.971116883680224",
        "assists": "2",
        "key_passes": "26",
        "xA": "2.0287596937268972",
        "npg": "17",
        "npG": "13.971116883680224",
        "time": 2399
    },
    "FromCorner": {
        "situation": "FromCorner",
        "season": "2015",
        "goals": "2",
        "shots": "11",
        "xG": "1.8276203628629446",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "2",
        "npG": "1.8276203628629446",
        "time": 2399
    },
    "Penalty": {
        "situation": "Penalty",
        "season": "2015",
        "goals": "4",
        "shots": "5",
        "xG": "3.8058441877365112",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "0",
        "npG": "0",
        "time": 2399
    },
    ...,
    "2014": {
        "OpenPlay": {
            "situation": "OpenPlay",
            "season": "2014",
            "goals": "19",
            "shots": "128",
            "xG": "18.23446972388774",
            "assists": "7",
            "key_passes": "32",
            "xA": "4.622839629650116",
            "npg": "19",
            "npG": "18.23446972388774",
            "time": 2551
        },
        "FromCorner": {
            "situation": "FromCorner",
            "season": "2014",
            "goals": "1",
            "shots": "12",
            "xG": "1.8788630235940218",
            "assists": "1",
            "key_passes": "1",

```

(continues on next page)

(continued from previous page)

```

        "xA": "0.9460826516151428",
        "npg": "1",
        "npG": "1.8788630235940218",
        "time": 2551
    },
    "Penalty": {
        "situation": "Penalty",
        "season": "2014",
        "goals": "5",
        "shots": "6",
        "xG": "4.566976249217987",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "0",
        "npG": "0",
        "time": 2551
    },
    "SetPiece": {
        "situation": "SetPiece",
        "season": "2014",
        "goals": "1",
        "shots": "2",
        "xG": "0.5898510366678238",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "1",
        "npG": "0.5898510366678238",
        "time": 2551
    }
}
},
"shotZones": {
    "2014": {
        "shotOboxTotal": {
            "shotZones": "shotOboxTotal",
            "season": "2014",
            "goals": "2",
            "shots": "33",
            "xG": "1.5900825830176473",
            "assists": "2",
            "key_passes": "9",
            "xA": "0.3100438117980957",
            "npg": "2",
            "npG": "1.5900825830176473"
        },
        "shotPenaltyArea": {
            "shotZones": "shotPenaltyArea",
            "season": "2014",
            "goals": "22",
            "shots": "108",
            "xG": "19.79369100742042",
            "assists": "5",
            "key_passes": "22",
            "xA": "3.9576267898082733",
            "npg": "17",

```

(continues on next page)

(continued from previous page)

```

        "npG": "15.226714758202434"
    },
    "shotSixYardBox": {
        "shotZones": "shotSixYardBox",
        "season": "2014",
        "goals": "2",
        "shots": "7",
        "xG": "3.8863864429295063",
        "assists": "1",
        "key_passes": "2",
        "xA": "1.3012516796588898",
        "npg": "2",
        "npG": "3.8863864429295063"
    }
},
...,
"2018": {
    "shotOboxTotal": {
        "shotZones": "shotOboxTotal",
        "season": "2018",
        "goals": "2",
        "shots": "21",
        "xG": "0.8707829182967544",
        "assists": "1",
        "key_passes": "9",
        "xA": "0.31408058758825064",
        "npg": "2",
        "npG": "0.8707829182967544"
    },
    "shotPenaltyArea": {
        "shotZones": "shotPenaltyArea",
        "season": "2018",
        "goals": "12",
        "shots": "65",
        "xG": "11.844964944757521",
        "assists": "4",
        "key_passes": "14",
        "xA": "2.1070052348077297",
        "npg": "10",
        "npG": "10.322627269662917"
    },
    "shotSixYardBox": {
        "shotZones": "shotSixYardBox",
        "season": "2018",
        "goals": "4",
        "shots": "9",
        "xG": "4.799736991524696",
        "assists": "1",
        "key_passes": "2",
        "xA": "1.3552910089492798",
        "npg": "4",
        "npG": "4.799736991524696"
    }
}
},
"shotTypes": {
    "2014": {

```

(continues on next page)

(continued from previous page)

```

    "RightFoot": {
        "shotTypes": "RightFoot",
        "season": "2014",
        "goals": "18",
        "shots": "96",
        "xG": "17.13349057827145",
        "assists": "5",
        "key_passes": "19",
        "xA": "3.883937703445554",
        "npg": "13",
        "npG": "12.566514329053462"
    },
    "LeftFoot": {
        "shotTypes": "LeftFoot",
        "season": "2014",
        "goals": "7",
        "shots": "40",
        "xG": "6.236775731667876",
        "assists": "3",
        "key_passes": "13",
        "xA": "1.6454832945019007",
        "npg": "7",
        "npG": "6.236775731667876"
    },
    "Head": {
        "shotTypes": "Head",
        "season": "2014",
        "goals": "1",
        "shots": "12",
        "xG": "1.8998937234282494",
        "assists": "0",
        "key_passes": "1",
        "xA": "0.03950128331780434",
        "npg": "1",
        "npG": "1.8998937234282494"
    }
},
...,
},
"2018": {
    "RightFoot": {
        "shotTypes": "RightFoot",
        "season": "2018",
        "goals": "9",
        "shots": "58",
        "xG": "9.876922971569002",
        "assists": "3",
        "key_passes": "9",
        "xA": "1.6752301333472133",
        "npg": "7",
        "npG": "8.354585296474397"
    },
    "LeftFoot": {
        "shotTypes": "LeftFoot",
        "season": "2018",
        "goals": "6",
        "shots": "26",

```

(continues on next page)

(continued from previous page)

```
        "xG": "4.921279687434435",
        "assists": "3",
        "key_passes": "16",
        "xA": "2.101146697998047",
        "npg": "6",
        "npG": "4.921279687434435"
    },
    "Head": {
        "shotTypes": "Head",
        "season": "2018",
        "goals": "2",
        "shots": "10",
        "xG": "1.8183354930952191",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "2",
        "npG": "1.8183354930952191"
    },
    "OtherBodyPart": {
        "shotTypes": "OtherBodyPart",
        "season": "2018",
        "goals": "1",
        "shots": "1",
        "xG": "0.8989467024803162",
        "assists": "0",
        "key_passes": "0",
        "xA": "0",
        "npg": "1",
        "npG": "0.8989467024803162"
    }
}
}
```

`Understat.get_player_matches(player_id, options=None, **kwargs)`
Returns the player with the given ID's matches data.

Parameters

- **player_id** (*int* or *str*) – The player's Understat ID.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of the player's matches data.

Return type list

It returns the information about the matches played by the given player. So for example, the matches Sergio Agüero has played, as seen in the screenshot

Nº	Date	Home	Score	Away	Pos	Min	Sh	G	KP	A	xG	xA
1	2021-03-13	Fulham	0-3	Manchester City	FW	90	2	1	0	0	0.80 ^{-0.20}	0.00
2	2021-03-10	Manchester City	5-2	Southampton	Sub	15	1	0	0	0	0.05 ^{+0.05}	0.00
3	2021-02-27	Manchester City	2-1	West Ham	FW	62	0	0	1	0	0.00	0.02 ^{+0.02}
4	2021-01-03	Chelsea	1-3	Manchester City	Sub	2	0	0	0	0	0.00	0.00
5	2020-12-26	Manchester City	2-0	Newcastle United	Sub	11	1	0	0	0	0.70 ^{+0.70}	0.00
6	2020-12-15	Manchester City	1-1	West Bromwich Albion	Sub	13	2	0	0	0	0.12 ^{+0.12}	0.00
7	2020-10-24	West Ham	1-1	Manchester City	FW	47	0	0	0	0	0.00	0.00
8	2020-10-17	Manchester City	1-0	Arsenal	FW	66	1	0	3	0	0.05 ^{+0.05}	0.71 ^{+0.71}
9	2020-06-22	Manchester City	5-0	Burnley	FW	48	1	0	0	0	0.08 ^{+0.08}	0.00
10	2020-06-17	Manchester City	3-0	Arsenal	Sub	7	3	0	0	0	0.53 ^{+0.53}	0.00

This function also comes with the *options* keyword argument, and also the ***kwargs* magic variable. An example of how you could use either of these to filter Sergio Agüero's matches to only include matches where Manchester United were the home team is shown below

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        # Using **kwargs
        player_matches = await understat.get_player_matches(
            619, h_team="Manchester United")
        # Or using options keyword argument
        player_matches = await understat.get_player_matches(
            619, {"h_team": "Manchester United"})
        print(json.dumps(player_matches))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs

```

[
  {
    "goals": "2",
    "shots": "5",
    "xG": "1.4754852056503296",
    "time": "90",
    "position": "FW",
    "h_team": "Manchester United",
    "a_team": "Manchester City",
    "h_goals": "4",
    "a_goals": "2",
    "date": "2015-04-12",
    "id": "4459",
    "season": "2014",
    "roster_id": "23306",
    "xA": "0",
    "assists": "0",
    "key_passes": "0",
    "npg": "2",
    "npG": "1.4754852056503296",
    "xGChain": "1.4855852127075195",
    "xGBuildup": "0.04120262712240219"
  }
]

```

(continues on next page)

(continued from previous page)

```
}
]
```

Since the usage of both the *options* keyword argument and the ***kwargs* magic variable have been shown, the examples following this will only show *one* of the two.

Understat.**get_player_shots**(*player_id*, *options=None*, ***kwargs*)

Returns the player with the given ID's shot data.

Parameters

- **player_id** (*int or str*) – The player's Understat ID.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of the player's shot data.

Return type list

It returns the given player's shot data, which includes information about the situation (open play, freekick etc.), if it hit the post or was a goal, and more. Basically, all the information that you can get from a player's page in the section shown below



The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get all Sergio Agüero's shots (not necessarily goals) that were assisted by Fernandinho, then you could do the following


```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        player_shots = await understat.get_player_shots(
            619, {"player_assisted": "Fernandinho"})
        print(json.dumps(player_shots))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  {
    "id": "14552",
    "minute": "91",
    "result": "SavedShot",
    "X": "0.9259999847412109",
    "Y": "0.6809999847412109",
    "xG": "0.0791548416018486",
    "player": "Sergio Ag\u00f0cero",
    "h_a": "a",
    "player_id": "619",
    "situation": "OpenPlay",
    "season": "2014",
    "shotType": "LeftFoot",
    "match_id": "4757",
    "h_team": "Newcastle United",
    "a_team": "Manchester City",
    "h_goals": "0",
    "a_goals": "2",
    "date": "2014-08-17 16:00:00",
    "player_assisted": "Fernandinho",
    "lastAction": "Pass"
  },
  ...,
  {
    "id": "233670",
    "minute": "15",
    "result": "MissedShots",
    "X": "0.7419999694824219",
    "Y": "0.5359999847412109",
    "xG": "0.029104366898536682",
    "player": "Sergio Ag\u00f0cero",
    "h_a": "h",
    "player_id": "619",
    "situation": "OpenPlay",
    "season": "2018",
    "shotType": "RightFoot",
    "match_id": "9234",
    "h_team": "Manchester City",
    "a_team": "Newcastle United",
    "h_goals": "2",
    "a_goals": "1",
    "date": "2018-09-01 16:30:00",
    "player_assisted": "Fernandinho",
    "lastAction": "Pass"
  }
]

```

(continues on next page)

(continued from previous page)

```
}
]
```

Understat.**get_player_stats** (*player_id*, *positions=None*)

Returns the player with the given ID's min / max stats, per position(s).

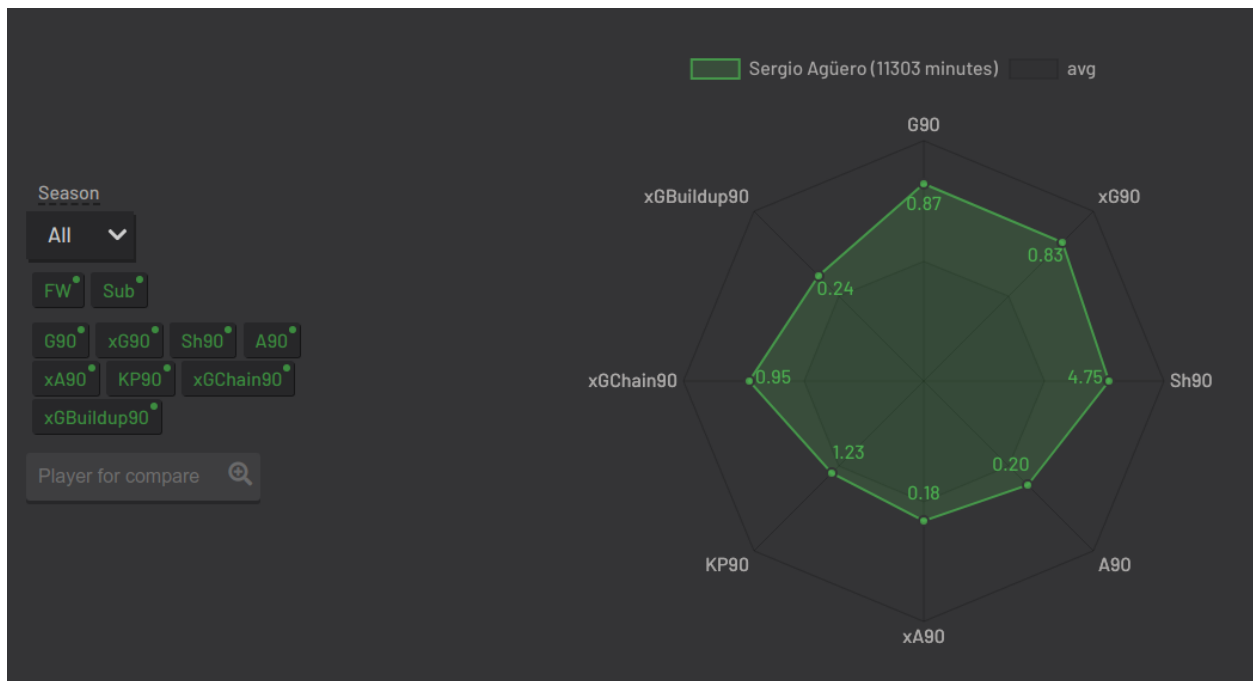
Parameters

- **player_id** (*int* or *str*) – The player's Understat ID.
- **positions** – Positions to filter the data by, defaults to None.
- **positions** – list, optional

Returns List of the player's stats per position.

Return type list

It returns the player's average stats overall, which includes stuff like their average goals per 90 minutes, average expected assists per 90 minutes and more. Basically everything you can see on a player's page in the section shown below



The function comes with the *positions* argument, which can be used to filter the stats by position(s). So for example, if you wanted to get Sergio Agüero's performance as a forward, then you could do the following

```
async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        player_stats = await understat.get_player_stats(619, ["FW"])
        print(json.dumps(player_stats))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

which outputs

```
[
  {
    "goals": {
      "min": 0.0011,
      "max": 0.0126,
      "avg": 0.0042
    },
    "xG": {
      "min": 0.00172821,
      "max": 0.0120816,
      "avg": 0.00415549
    },
    "shots": {
      "min": 0.015,
      "max": 0.0737,
      "avg": 0.028
    },
    "assists": {
      "min": 0,
      "max": 0.0048,
      "avg": 0.0014
    },
    "xA": {
      "min": 0.000264191,
      "max": 0.00538174,
      "avg": 0.00131568
    },
    "key_passes": {
      "min": 0.0036,
      "max": 0.0309,
      "avg": 0.012
    },
    "xGChain": {
      "min": 0.00272705,
      "max": 0.0169137,
      "avg": 0.00533791
    },
    "xGBuildup": {
      "min": 0.000243189,
      "max": 0.00671256,
      "avg": 0.00131848
    },
    "position": "FW"
  }
]
```

Understat.**get_league_players**(*league_name*, *season*, *options=None*, ***kwargs*)

Returns a list containing information about all the players in the given league in the given season.

Parameters

- **league_name** (*str*) – The league’s name.
- **season** (*str* or *int*) – The season.
- **options** – Options to filter the data by, defaults to None.

- **options** – dict, optional

Returns A list of the players as seen on Understat’s league overview.

Return type list

It returns all the information about the players in a given league in the given season. This includes stuff like their number of goals scored, their total expected assists and more. Basically, it’s all the information you can find in the player table shown on all league overview pages on understat.com.

Positions	Last												
All	All games	Start date		End date			1 2 3 4 5 ... 50						
No	Player	Team	Apps	Min	G	A	xG	xA	xG90	xA90			
1	Sergio Agüero	Manchester City		26	1960	18	6	17.52 ^{-0.48}	3.78 ^{-2.22}	0.80	0.17		
2	Pierre-Emerick Aubameyang	Arsenal		29	2220	17	4	18.68 ^{+1.68}	4.13 ^{+0.13}	0.76	0.17		
3	Harry Kane	Tottenham		26	2257	17	4	15.20 ^{-1.80}	4.54 ^{+0.54}	0.61	0.18		
4	Sadio Mané	Liverpool		29	2470	17	1	12.72 ^{-4.28}	3.80 ^{+2.80}	0.46	0.14		
5	Mohamed Salah	Liverpool		31	2659	17	7	18.36 ^{+1.36}	8.69 ^{+1.69}	0.62	0.29		
6	Raheem Sterling	Manchester City		27	2159	15	9	12.54 ^{-2.46}	9.91 ^{+0.91}	0.52	0.41		
7	Eden Hazard	Chelsea		29	2343	13	11	10.45 ^{-2.55}	8.02 ^{-2.98}	0.40	0.31		
8	Romelu Lukaku	Manchester United		27	1768	12	0	12.05 ^{+0.05}	1.68 ^{+1.68}	0.61	0.09		
9	Gylfi Sigurdsson	Everton		31	2541	12	3	10.23 ^{-1.77}	4.80 ^{+1.80}	0.36	0.17		
10	Jamie Vardy	Leicester		27	2110	12	4	13.83 ^{+1.83}	3.61 ^{-0.39}	0.59	0.15		
					841	599	879.71 ^{+38.71}	617.30 ^{+18.30}					

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get all the players who play for Manchester United, then you could do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        players = await understat.get_league_players(
            "epl",
            2018,
            team_title="Manchester United"
        )
        print(json.dumps(players))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  {
    "id": "594",
    "player_name": "Romelu Lukaku",
    "games": "27",
    "time": "1768",
    "goals": "12",
    "xG": "12.054240763187408",
    "assists": "0",
    "xA": "1.6836179178208113",

```

(continues on next page)

(continued from previous page)

```

        "shots": "50",
        "key_passes": "17",
        "yellow_cards": "4",
        "red_cards": "0",
        "position": "F S",
        "team_title": "Manchester United",
        "npg": "12",
        "npG": "12.054240763187408",
        "xGChain": "12.832402393221855",
        "xGBuildup": "3.366600174456835"
    },
    ...,
    {
        "id": "1740",
        "player_name": "Paul Pogba",
        "games": "27",
        "time": "2293",
        "goals": "11",
        "xG": "13.361832823604345",
        "assists": "9",
        "xA": "4.063152700662613",
        "shots": "87",
        "key_passes": "40",
        "yellow_cards": "5",
        "red_cards": "0",
        "position": "M S",
        "team_title": "Manchester United",
        "npg": "6",
        "npG": "7.272482139989734",
        "xGChain": "17.388037759810686",
        "xGBuildup": "8.965998269617558"
    }
]

```

Understat.**get_league_results** (*league_name*, *season*, *options=None*, ***kwargs*)

Returns a list containing information about all the results (matches) played by the teams in the given league in the given season.

Parameters

- **league_name** (*str*) – The league’s name.
- **season** (*str or int*) – The season.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns A list of the results as seen on Understat’s league overview.

Return type list

It returns the results (not fixtures) of the given league, in the given season. So for example, the results as seen in the screenshot below

Saturday, March 16, 2019		
Wolverhampton Wanderers	16:00	Arsenal
West Ham	<div>4 3</div> <div>1.88 1.67</div>	Huddersfield
Watford	16:00	Southampton
Burnley	<div>1 2</div> <div>1.17 0.61</div>	Leicester
Brighton	16:00	Cardiff
Bournemouth	<div>2 2</div> <div>1.62 1.03</div>	Newcastle United
Manchester United	19:00	Manchester City
Sunday, March 17, 2019		
Tottenham	13:00	Crystal Palace
Fulham	<div>1 2</div> <div>0.92 2.73</div>	Liverpool
Everton	<div>2 0</div> <div>2.24 1.24</div>	Chelsea

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get all Manchester United's results away from home, then you could do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        fixtures = await understat.get_league_results(
            "epl",
            2018,
            {
                "a": {"id": "89",
                      "title": "Manchester United",
                      "short_title": "MUN"}
            }
        )
        print(json.dumps(fixtures))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  {
    "id": "9215",
    "isResult": true,
    "h": {
      "id": "220",
      "title": "Brighton",
      "short_title": "BRI"
    }
  }
]

```

(continues on next page)

(continued from previous page)

```

    },
    "a": {
        "id": "89",
        "title": "Manchester United",
        "short_title": "MUN"
    },
    "goals": {
        "h": "3",
        "a": "2"
    },
    "xG": {
        "h": "1.63672",
        "a": "1.56579"
    },
    "datetime": "2018-08-19 18:00:00",
    "forecast": {
        "w": "0.3538",
        "d": "0.3473",
        "l": "0.2989"
    }
},
...,
{
    "id": "9496",
    "isResult": true,
    "h": {
        "id": "83",
        "title": "Arsenal",
        "short_title": "ARS"
    },
    "a": {
        "id": "89",
        "title": "Manchester United",
        "short_title": "MUN"
    },
    "goals": {
        "h": "2",
        "a": "0"
    },
    "xG": {
        "h": "1.52723",
        "a": "2.3703"
    },
    "datetime": "2019-03-10 16:30:00",
    "forecast": {
        "w": "0.1667",
        "d": "0.227",
        "l": "0.6063"
    }
}
]

```

Understat.**get_match_players** (*match_id*, *options=None*, ***kwargs*)

Returns a dictionary containing information about the players who played in the given match.


Parameters

- **fixture_id**(*int*) – A match’s ID.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns Dictionary containing information about the players who played in the match.

Return type dict

It returns information about the players who played in the given match. So for example, the players seen in the screenshot below

Manchester United										Chelsea			
No	Player	Pos	Min	Sh	G	KP	A	xG	xA				
1	David de Gea	GK	90	0	0	0	0	0.00	0.00				
2	Aaron Wan-Bissaka	DR	90	0	0	1	0	0.00	0.12	+0.12			
3	Victor Lindelöf	DC	90	0	0	0	0	0.00	0.00				
4	Harry Maguire	DC	90	0	0	1	0	0.00	0.06	+0.06			
5	Luke Shaw	DL	90	0	0	0	0	0.00	0.00				
6	Paul Pogba	DMC	90	1	0	4	2	0.06	+0.06	0.70	-1.30		
7	Scott McTominay	DMC	90	0	0	0	0	0.00	0.00				
8	Jesse Lingard	AMR	88	0	0	1	0	0.00	0.03	+0.03			
9	Andreas Perelra	AMC	76	0	0	1	1	0.00	0.61	-0.39			
10	Marcus Rashford	AML	88	4	2	0	0	1.30	-0.70	0.00			
11	Anthony Martial	FW	90	4	1	1	0	0.77	-0.23	0.06	+0.06		
12	Juan Mata	Sub	2	0	0	0	0	0.00	0.00				
13	Mason Greenwood	Sub	2	1	0	0	0	0.12	+0.12	0.00			
14	Daniel James	Sub	14	1	1	0	0	0.13	-0.87	0.00			
				11	4	9	3	2.37	-1.63	1.57	-1.43		

An example of getting the players who played in the match between Manchester United and Chelsea on 11 August, 2019 which ended 4-0 can be seen below:

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        players = await understat.get_match_players(11652)
        print(json.dumps(players))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

{
  "h": {
    "341628": {
      "id": "341628",
      "goals": "2",
      "own_goals": "0",
      "shots": "4",
      "xG": "1.3030972480773926",
      "time": "88",

```

(continues on next page)

(continued from previous page)

```

    "player_id": "556",
    "team_id": "89",
    "position": "AML",
    "player": "Marcus Rashford",
    "h_a": "h",
    "yellow_card": "0",
    "red_card": "0",
    "roster_in": "341631",
    "roster_out": "0",
    "key_passes": "0",
    "assists": "0",
    "xA": "0",
    "xGChain": "1.1517746448516846",
    "xGBuildup": "0.6098462343215942",
    "positionOrder": "13"
  },
  ...,
  "341629": {
    "id": "341629",
    "goals": "1",
    "own_goals": "0",
    "shots": "4",
    "xG": "0.7688590884208679",
    "time": "90",
    "player_id": "553",
    "team_id": "89",
    "position": "FW",
    "player": "Anthony Martial",
    "h_a": "h",
    "yellow_card": "1",
    "red_card": "0",
    "roster_in": "0",
    "roster_out": "0",
    "key_passes": "1",
    "assists": "0",
    "xA": "0.05561231076717377",
    "xGChain": "0.9395027160644531",
    "xGBuildup": "0.11503136157989502",
    "positionOrder": "15"
  }
},
"a": {
  "341633": {
    "id": "341633",
    "goals": "0",
    "own_goals": "0",
    "shots": "0",
    "xG": "0",
    "time": "90",
    "player_id": "5061",
    "team_id": "80",
    "position": "GK",
    "player": "Kepa",
    "h_a": "a",
    "yellow_card": "0",
    "red_card": "0",
    "roster_in": "0",

```

(continues on next page)

(continued from previous page)

```
        "roster_out": "0",
        "key_passes": "0",
        "assists": "0",
        "xA": "0",
        "xGChain": "0.04707280918955803",
        "xGBuildup": "0.04707280918955803",
        "positionOrder": "1"
    },
    ...,
    "341642": {
        "id": "341642",
        "goals": "0",
        "own_goals": "0",
        "shots": "2",
        "xG": "0.08609434962272644",
        "time": "60",
        "player_id": "592",
        "team_id": "80",
        "position": "AML",
        "player": "Ross Barkley",
        "h_a": "a",
        "yellow_card": "0",
        "red_card": "0",
        "roster_in": "341646",
        "roster_out": "0",
        "key_passes": "1",
        "assists": "0",
        "xA": "0.024473881348967552",
        "xGChain": "0.11056823283433914",
        "xGBuildup": "0",
        "positionOrder": "13"
    }
}
```

Understat.**get_match_shots** (*match_id*, *options=None*, ***kwargs*)

Returns a dictionary containing information about shots taken by the players in the given match.

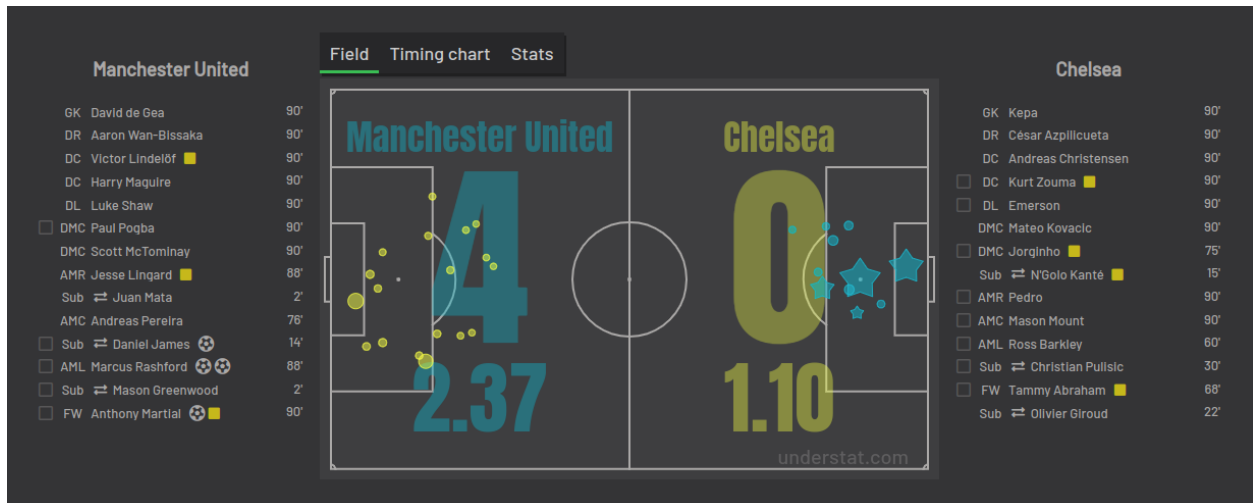
Parameters

- **fixture_id** (*int*) – A match’s ID.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns Dictionary containing information about the players who played in the match.

Return type dict

It returns information about the shots made by players who played in the given match. So for example, the shots seen in the screenshot below



An example of getting the shots made in the match between Manchester United and Chelsea on 11 August, 2019 which ended 4-0 can be seen below:

```
async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        players = await understat.get_match_shots(11652)
        print(json.dumps(players))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

which outputs (with parts omitted)

```
{
  "h": [
    {
      "id": "310295",
      "minute": "6",
      "result": "SavedShot",
      "X": "0.8280000305175781",
      "Y": "0.639000015258789",
      "xG": "0.04247729107737541",
      "player": "Anthony Martial",
      "h_a": "h",
      "player_id": "553",
      "situation": "OpenPlay",
      "season": "2019",
      "shotType": "RightFoot",
      "match_id": "11652",
      "h_team": "Manchester United",
      "a_team": "Chelsea",
      "h_goals": "4",
      "a_goals": "0",
      "date": "2019-08-11 16:30:00",
      "player_assisted": null,
      "lastAction": "None"
    },
    ...,
    {
```

(continues on next page)

(continued from previous page)

```

        "id": "310318",
        "minute": "86",
        "result": "BlockedShot",
        "X": "0.8669999694824219",
        "Y": "0.47299999237060547",
        "xG": "0.11503136157989502",
        "player": "Mason Greenwood",
        "h_a": "h",
        "player_id": "7490",
        "situation": "OpenPlay",
        "season": "2019",
        "shotType": "RightFoot",
        "match_id": "11652",
        "h_team": "Manchester United",
        "a_team": "Chelsea",
        "h_goals": "4",
        "a_goals": "0",
        "date": "2019-08-11 16:30:00",
        "player_assisted": "Aaron Wan-Bissaka",
        "lastAction": "Cross"
    }
],
"a": [
    {
        "id": "310293",
        "minute": "3",
        "result": "ShotOnPost",
        "X": "0.835999984741211",
        "Y": "0.38599998474121094",
        "xG": "0.03392893448472023",
        "player": "Tammy Abraham",
        "h_a": "a",
        "player_id": "702",
        "situation": "FromCorner",
        "season": "2019",
        "shotType": "RightFoot",
        "match_id": "11652",
        "h_team": "Manchester United",
        "a_team": "Chelsea",
        "h_goals": "4",
        "a_goals": "0",
        "date": "2019-08-11 16:30:00",
        "player_assisted": "Mateo Kovacic",
        "lastAction": "BallTouch"
    },
    ...,
    {
        "id": "310321",
        "minute": "93",
        "result": "SavedShot",
        "X": "0.850999984741211",
        "Y": "0.7",
        "xG": "0.043492574244737625",
        "player": "Emerson",
        "h_a": "a",
        "player_id": "1245",
        "situation": "OpenPlay",

```

(continues on next page)

(continued from previous page)

```

    "season": "2019",
    "shotType": "LeftFoot",
    "match_id": "11652",
    "h_team": "Manchester United",
    "a_team": "Chelsea",
    "h_goals": "4",
    "a_goals": "0",
    "date": "2019-08-11 16:30:00",
    "player_assisted": "Christian Pulisic",
    "lastAction": "Pass"
  }
]
}

```

Understat.**get_stats** (*options=None, **kwargs*)

Returns a list containing stats of every league, grouped by month.

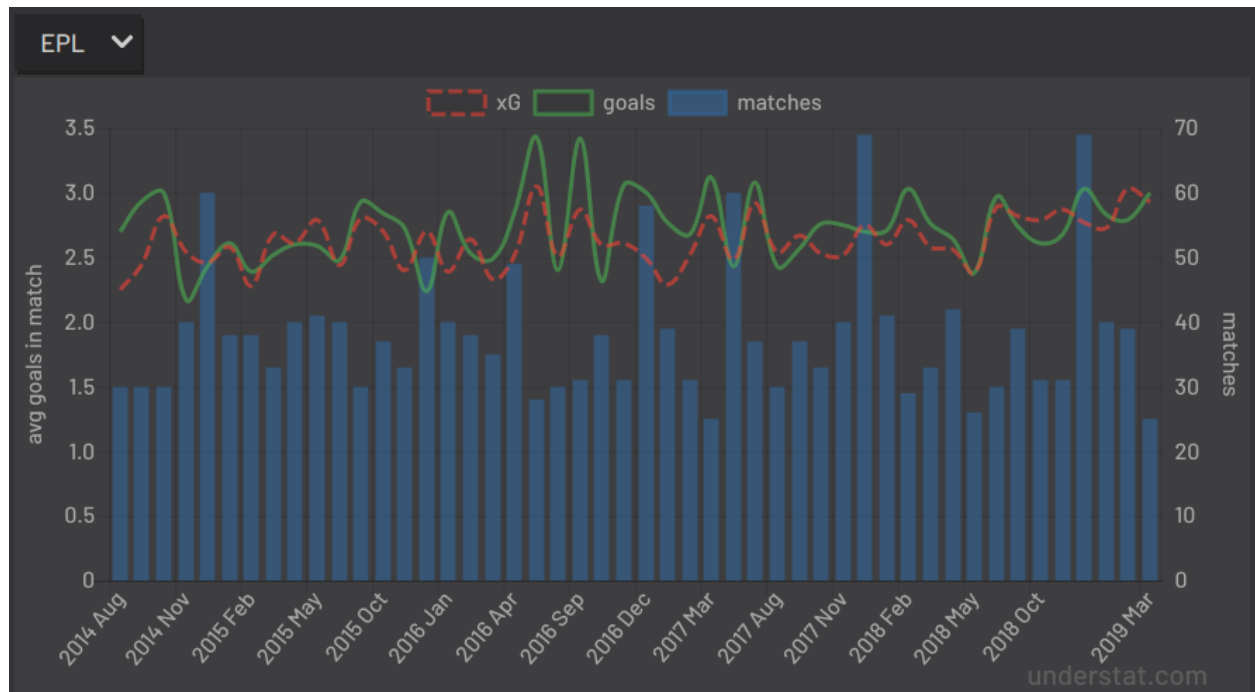
Parameters

- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of dictionaries.

Return type list

It returns the average stats of all the leagues tracked on understat.com, split by month. Basically, it is all the information you see on their homepage, as seen in the screenshot below



The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get the stats for the Premier League in the 8th month of each year they have been tracking the stats, then you could do the following

```
async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        stats = await understat.get_stats({"league": "EPL", "month": "8"})
        print(json.dumps(stats))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

which outputs

```
[
  {
    "league_id": "1",
    "league": "EPL",
    "h": "1.3000",
    "a": "1.4000",
    "hxd": "1.141921697060267",
    "axd": "1.110964298248291",
    "year": "2014",
    "month": "8",
    "matches": "30"
  },
  {
    "league_id": "1",
    "league": "EPL",
    "h": "1.1000",
    "a": "1.3750",
    "hxd": "1.2151590750552714",
    "axd": "1.221375621855259",
    "year": "2015",
    "month": "8",
    "matches": "40"
  },
  {
    "league_id": "1",
    "league": "EPL",
    "h": "1.2000",
    "a": "1.2000",
    "hxd": "1.3605596815546355",
    "axd": "1.145853524406751",
    "year": "2016",
    "month": "8",
    "matches": "30"
  },
  {
    "league_id": "1",
    "league": "EPL",
    "h": "1.3000",
    "a": "1.1333",
    "hxd": "1.4422248949607213",
    "axd": "1.096401752779881",
    "year": "2017",
    "month": "8",
    "matches": "30"
  },
]
```

(continues on next page)

(continued from previous page)

```

        "league_id": "1",
        "league": "EPL",
        "h": "1.6333",
        "a": "1.3333",
        "hxdg": "1.453833992779255",
        "axg": "1.4325587471326193",
        "year": "2018",
        "month": "8",
        "matches": "30"
    }
]

```

Understat.**get_team_fixtures**(*team_name*, *season*, *options=None*, ***kwargs*)

Returns a team's upcoming fixtures in the given season.

Parameters

- **team_name** (*str*) – A team's name.
- **season** (*int* or *str*) – The season.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of the team's upcoming fixtures in the given season.

Return type list

It returns the upcoming fixtures (not results) of the given team, in the given season. So for example, the fixtures as seen in the screenshot below

Mar 30, 2019	Apr 06, 2019	Apr 13, 2019	Apr 20, 2019	Apr 27, 2019	May 04, 2019	May 12, 2019
H	A	H	A	H	A	H
16:00	18:00	18:00	18:00	18:00	18:00	18:00
Watford	Wolverhampton Wanderers	West Ham	Everton	Chelsea	Huddersfield	Cardiff

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. This is similar to the *get_league_fixtures* function, but it makes certain options for filtering much easier. For example, if you, once again, wanted to get all Manchester United's upcoming fixtures at **home**, then instead of passing a dictionary as keyword argument, you could simply do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        results = await understat.get_team_fixtures(
            "Manchester United",
            2018,
            side="h"
        )
        print(json.dumps(results))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```
[
  {
    "id": "9501",
    "isResult": false,
    "side": "h",
    "h": {
      "id": "89",
      "title": "Manchester United",
      "short_title": "MUN"
    },
    "a": {
      "id": "88",
      "title": "Manchester City",
      "short_title": "MCI"
    },
    "goals": {
      "h": null,
      "a": null
    },
    "xG": {
      "h": null,
      "a": null
    },
    "datetime": "2019-03-16 18:00:00"
  },
  ...,
  {
    "id": "9570",
    "isResult": false,
    "side": "h",
    "h": {
      "id": "89",
      "title": "Manchester United",
      "short_title": "MUN"
    },
    "a": {
      "id": "227",
      "title": "Cardiff",
      "short_title": "CAR"
    },
    "goals": {
      "h": null,
      "a": null
    },
    "xG": {
      "h": null,
      "a": null
    },
    "datetime": "2019-05-12 17:00:00"
  }
]
```

Understat.**get_team_players** (*team_name*, *season*, *options=None*, ***kwargs*)

Returns a team's player statistics in the given season.

Parameters

- **team_name** (*str*) – A team's name.
- **season** (*int or str*) – The season.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of the team's players' statistics in the given season.

Return type list

It returns all the information about the players of a given team in the given season. This includes stuff like their number of goals scored, their total expected assists and more. Basically, it's all the information you can find in the player table shown on all team overview pages on understat.com.

Positions		Last											
All	▼	All games	▼	Start date	📅	End date	📅	🔍					
No	Player	Pos	Apps	Min	G	A	Sh90	KP90	xG	xA	xG90	xA90	
1	Romelu Lukaku	F	27	1768	12	0	2.55	0.87	12.05	+0.05	1.68	+1.68	0.09
2	Paul Pogba	M	27	2293	11	9	3.41	1.57	13.36	+2.36	4.06	+4.94	0.16
3	Anthony Martial	F M	21	1309	9	2	2.06	2.06	5.49	+3.51	3.00	+1.00	0.21
4	Marcus Rashford	F M	26	1820	9	6	3.21	1.34	9.60	+0.60	4.30	+1.70	0.21
5	Jesse Lingard	F M	22	1297	4	2	1.73	0.90	4.13	+0.13	2.27	+0.27	0.16
6	Ander Herrera	D M	19	1213	2	3	1.41	1.04	1.43	+0.57	2.04	+0.98	0.15
7	Juan Mata	F M	18	1010	2	2	1.43	1.78	0.98	+1.02	3.03	+1.03	0.27
8	Ashley Young	D	23	2016	2	2	0.45	1.43	0.45	+1.55	3.06	+1.08	0.14
9	Alexis Sánchez	F M	17	792	1	3	1.93	2.27	1.71	+0.71	3.39	+0.39	0.39
10	Chris Smalling	D	18	1590	1	0	0.51	0.06	1.04	+0.04	0.06	+0.06	0.00
11	Nemanja Matic	D M	24	2125	1	0	0.42	0.55	1.62	+0.62	0.56	+0.56	0.02
12	Andreas Pereira	M	9	390	1	1	0.92	0.69	0.11	+0.89	0.50	+0.50	0.12
13	Luke Shaw	D M	24	2144	1	2	0.71	1.13	1.26	+0.26	3.09	+1.09	0.13

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. This is similar to the *get_league_players* function, but is quicker and easier. For example, if you, once again, wanted to get all Manchester United's players who have only played games as a forward, then you could do the following

```
async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        results = await understat.get_team_players(
            "Manchester United",
            2018,
            position="F S"
        )
        print(json.dumps(results))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```

which outputs

```
[
  {
```

(continues on next page)

(continued from previous page)

```

        "id": "594",
        "player_name": "Romelu Lukaku",
        "games": "27",
        "time": "1768",
        "goals": "12",
        "xG": "12.054240763187408",
        "assists": "0",
        "xA": "1.6836179178208113",
        "shots": "50",
        "key_passes": "17",
        "yellow_cards": "4",
        "red_cards": "0",
        "position": "F S",
        "team_title": "Manchester United",
        "npg": "12",
        "npxG": "12.054240763187408",
        "xGChain": "12.832402393221855",
        "xGBuildup": "3.366600174456835"
    }
]

```

Understat.**get_team_results**(*team_name*, *season*, *options=None*, ***kwargs*)

Returns a team's results in the given season.

Parameters

- **team_name** (*str*) – A team's name.
- **season** (*int* or *str*) – The season.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns List of the team's results in the given season.

Return type list

It returns the results (not fixtures) of the given team, in the given season. So for example, the fixtures as seen in the screenshot below

Jan 19, 2019	Jan 29, 2019	Feb 03, 2019	Feb 09, 2019	Feb 24, 2019	Feb 27, 2019	Mar 02, 2019
H	H	A	A	H	A	H
2 1	2 2	0 1	0 3	0 0	1 3	3 2
2.64 0.64	3.23 0.93	1.40 1.50	2.19 2.13	0.47 0.38	1.89 1.54	2.54 0.58
Brighton	Burnley	Leicester	Fulham	Liverpool	Crystal Palace	Southampton

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. This is similar to the *get_league_results* function, but it makes certain options for filtering much easier. For example, if you, once again, wanted to get all Manchester United's results at **home**, then instead of passing a dictionary as keyword argument, you could simply do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        results = await understat.get_team_results(
            "Manchester United",

```

(continues on next page)

(continued from previous page)

```

        2018,
        side="h"
    )
    print(json.dumps(results))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

[
  {
    "id": "9197",
    "isResult": true,
    "side": "h",
    "h": {
      "id": "89",
      "title": "Manchester United",
      "short_title": "MUN"
    },
    "a": {
      "id": "75",
      "title": "Leicester",
      "short_title": "LEI"
    },
    "goals": {
      "h": "2",
      "a": "1"
    },
    "xG": {
      "h": "1.5137",
      "a": "1.73813"
    },
    "datetime": "2018-08-10 22:00:00",
    "forecast": {
      "w": 0.33715468577027,
      "d": 0.23067469101496,
      "l": 0.43217062251974
    },
    "result": "w"
  },
  ...,
  {
    "id": "9226",
    "isResult": true,
    "side": "h",
    "h": {
      "id": "89",
      "title": "Manchester United",
      "short_title": "MUN"
    },
    "a": {
      "id": "82",
      "title": "Tottenham",
      "short_title": "TOT"
    },
    "goals": {

```

(continues on next page)

(continued from previous page)

```

        "h": "0",
        "a": "3"
    },
    "xG": {
        "h": "1.40321",
        "a": "1.80811"
    },
    "datetime": "2018-08-27 22:00:00",
    "forecast": {
        "w": 0.29970781519619,
        "d": 0.22891929318443,
        "l": 0.47137289056693
    },
    "result": "1"
}
]

```

Understat.**get_team_stats**(*team_name*, *season*)

Returns a team's stats, as seen on their page on Understat, in the given season.

Parameters

- **team_name** (*str*) – A team's name, e.g. Manchester United.
- **season** (*int* or *str*) – A season / year, e.g. 2018.

Returns A dictionary containing a team's stats.

Return type dict

It returns all the statistics of a given team, which includes stuff like their performance per season, formation and more. Basically, it's everything that can be found in the table shown in the screenshot below

Situation	Formation	Game state	Timing	Shot zones	Attack speed	Result						
No	Situation	Sh	G	ShA	GA	xG	xGA	xGD	xG/Sh	xGA/Sh		
1	Open play	297	39	279	25	36.67 ^{+2.33}	28.87 ^{+3.87}	7.80	0.12	0.10		
2	From corner	62	7	63	7	7.36 ^{+0.36}	6.59 ^{+0.41}	0.77	0.12	0.10		
3	Set piece	21	3	28	1	4.63 ^{+1.63}	3.94 ^{+2.94}	0.69	0.22	0.14		
4	Direct Freekick	21	2	17	2	1.05 ^{+0.95}	1.09 ^{+0.91}	-0.04	0.05	0.06		
5	Penalty	10	7	5	5	7.61 ^{+0.61}	3.81 ^{+1.19}	3.81	0.76	0.76		

An example of getting Manchester United's data can be found below

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        team_stats = await understat.get_team_stats("Manchester United", 2018)
        print(json.dumps(team_stats))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```

{
  "situation": {
    "OpenPlay": {
      "shots": 297,
      "goals": 39,
      "xG": 36.671056651045,
      "against": {
        "shots": 279,
        "goals": 25,
        "xG": 28.870285989717
      }
    },
    ...,
    "Penalty": {
      "shots": 10,
      "goals": 7,
      "xG": 7.611688375473,
      "against": {
        "shots": 5,
        "goals": 5,
        "xG": 3.8058441877365
      }
    }
  },
  "formation": {
    "4-3-3": {
      "stat": "4-3-3",
      "time": 1295,
      "shots": 185,
      "goals": 30,
      "xG": 27.7899469533,
      "against": {
        "shots": 176,
        "goals": 18,
        "xG": 20.478145442903
      }
    },
    ...,
    "4-4-2": {
      "stat": "4-4-2",
      "time": 38,
      "shots": 8,
      "goals": 0,
      "xG": 0.87938431277871,
      "against": {
        "shots": 11,
        "goals": 1,
        "xG": 0.66449437476695
      }
    }
  },
  "gameState": {
    "Goal diff 0": {
      "stat": "Goal diff 0",
      "time": 1284,
      "shots": 154,
      "goals": 20,

```

(continues on next page)

(continued from previous page)

```

        "xG": 20.433959940448,
        "against": {
            "shots": 170,
            "goals": 15,
            "xG": 17.543024708517
        }
    },
    ...,
    "Goal diff < -1": {
        "stat": "Goal diff < -1",
        "time": 253,
        "shots": 43,
        "goals": 7,
        "xG": 6.4928285568021,
        "against": {
            "shots": 21,
            "goals": 1,
            "xG": 2.9283153852448
        }
    }
},
"timing": {
    "1-15": {
        "stat": "1-15",
        "shots": 51,
        "goals": 6,
        "xG": 7.2566251829267,
        "against": {
            "shots": 72,
            "goals": 7,
            "xG": 8.5656435946003
        }
    },
    ...,
    "76+": {
        "stat": "76+",
        "shots": 70,
        "goals": 12,
        "xG": 10.272770666517,
        "against": {
            "shots": 77,
            "goals": 8,
            "xG": 10.18940022774
        }
    }
},
"shotZone": {
    "ownGoals": {
        "stat": "ownGoals",
        "shots": 0,
        "goals": 0,
        "xG": 0,
        "against": {
            "shots": 2,
            "goals": 2,
            "xG": 2
        }
    }
}

```

(continues on next page)

(continued from previous page)

```

    },
    "shotOboxTotal": {
        "stat": "shotOboxTotal",
        "shots": 158,
        "goals": 8,
        "xG": 4.8084309450351,
        "against": {
            "shots": 170,
            "goals": 6,
            "xG": 5.4022304248065
        }
    },
    ...,
    "shotSixYardBox": {
        "stat": "shotSixYardBox",
        "shots": 36,
        "goals": 13,
        "xG": 13.912872407585,
        "against": {
            "shots": 32,
            "goals": 8,
            "xG": 11.533062046394
        }
    },
    },
    "attackSpeed": {
        "Normal": {
            "stat": "Normal",
            "shots": 258,
            "goals": 34,
            "xG": 30.690259062219,
            "against": {
                "shots": 230,
                "goals": 18,
                "xG": 23.094043077901
            }
        },
        ...,
        "Slow": {
            "stat": "Slow",
            "shots": 18,
            "goals": 2,
            "xG": 0.71848054975271,
            "against": {
                "shots": 26,
                "goals": 5,
                "xG": 2.9855494443327
            }
        }
    },
    },
    "result": {
        "MissedShots": {
            "shots": 122,
            "goals": 0,
            "xG": 12.353983599227,
            "against": {
                "shots": 155,

```

(continues on next page)

(continued from previous page)

```
        "goals": 0,
        "xG": 13.091518453322
    },
    ...,
    "ShotOnPost": {
        "shots": 4,
        "goals": 0,
        "xG": 0.81487018615007,
        "against": {
            "shots": 2,
            "goals": 0,
            "xG": 0.61989105120301
        }
    }
}
```

Understat.**get_teams**(*league_name*, *season*, *options=None*, ***kwargs*)

Returns a list containing information about all the teams in the given league in the given season.

Parameters

- **league_name** (*str*) – The league's name.
- **season** (*str or int*) – The season.
- **options** – Options to filter the data by, defaults to None.
- **options** – dict, optional

Returns A list of the league's table as seen on Understat's league overview.

Return type list

It returns all the information for the teams in a given league, in a given season. Basically it is all the information that is shown in the league's table, as shown in the screenshot below

Table	Charts	overall	home	away	Start date		End date								
No	Team	M	W	D	L	G	GA	PTS	xG	xGA	xPTS				
1	Liverpool	31	23	7	1	70	18	76	65.85 ^{+4.15}	23.04 ^{+5.04}	69.26 ^{-6.74}				
2	Manchester City	30	24	2	4	79	21	74	76.75 ^{+2.25}	21.68 ^{+0.68}	71.82 ^{-2.18}				
3	Tottenham	30	20	1	9	57	32	61	50.19 ^{+6.81}	39.01 ^{+7.01}	49.50 ^{+11.50}				
4	Arsenal	30	18	6	6	63	39	60	52.96 ^{+10.04}	42.57 ^{+3.57}	48.33 ^{+11.67}				
5	Manchester United	30	17	7	6	58	40	58	56.23 ^{+1.77}	41.74 ^{+1.74}	49.76 ^{-8.24}				
6	Chelsea	30	17	6	7	50	33	57	48.52 ^{+1.48}	31.91 ^{-1.09}	54.33 ^{-2.87}				
7	Wolverhampton Wanderers	30	12	8	10	38	36	44	41.36 ^{+3.36}	32.11 ^{-3.89}	47.68 ^{+3.68}				
8	Watford	30	12	7	11	42	44	43	41.07 ^{-0.93}	45.31 ^{+1.31}	40.04 ^{-2.96}				
9	West Ham	31	12	6	13	41	46	42	38.42 ^{-2.58}	53.44 ^{+7.44}	35.16 ^{-8.84}				
10	Leicester	31	12	5	14	40	43	41	40.36 ^{+0.36}	36.96 ^{-6.04}	45.10 ^{+4.10}				
11	Everton	31	11	7	13	43	42	40	40.72 ^{-2.28}	45.43 ^{+3.43}	39.33 ^{-0.67}				
12	Bournemouth	31	11	5	15	43	56	38	44.46 ^{+1.46}	50.01 ^{-5.99}	40.84 ^{+2.84}				
13	Newcastle United	31	9	8	14	31	40	35	33.05 ^{+2.05}	50.19 ^{+10.19}	30.55 ^{+4.45}				
14	Crystal Palace	30	9	6	15	36	41	33	38.51 ^{+2.51}	39.41 ^{-1.59}	41.34 ^{+8.34}				
15	Brighton	29	9	6	14	32	42	33	31.27 ^{-0.73}	46.26 ^{+4.26}	29.93 ^{-3.07}				
16	Southampton	30	7	9	14	34	50	30	36.78 ^{+2.78}	48.08 ^{-1.92}	33.62 ^{+3.62}				
17	Burnley	31	8	6	17	35	59	30	36.85 ^{+1.85}	55.37 ^{-3.63}	34.27 ^{+4.27}				
18	Cardiff	30	8	4	18	27	57	28	34.05 ^{+7.05}	50.81 ^{-6.19}	30.55 ^{+2.55}				
19	Fulham	31	4	5	22	29	70	17	34.17 ^{+5.17}	60.38 ^{-6.62}	27.35 ^{+10.35}				
20	Huddersfield	31	3	5	23	18	57	14	22.62 ^{+4.62}	50.45 ^{-6.55}	25.49 ^{+11.49}				

The function comes with the *options* keyword argument, and the ***kwargs* magic variable, and so that can be used to filter the output. So for example, if you wanted to get Manchester United's stats (as shown in the table), you could do the following

```

async def main():
    async with aiohttp.ClientSession() as session:
        understat = Understat(session)
        teams = await understat.get_teams(
            "epl",
            2018,
            title="Manchester United"
        )
        print(json.dumps(teams))

loop = asyncio.get_event_loop()
loop.run_until_complete(main())

```

which outputs (with parts omitted)

```
[
  {
    "id": "89",
    "title": "Manchester United",
    "history": [
      {
        "h_a": "h",
        "xG": 1.5137,
```

(continues on next page)

(continued from previous page)

```

        "xGA": 1.73813,
        "npvG": 0.75253,
        "npvGA": 1.73813,
        "ppda": {
            "att": 285,
            "def": 18
        },
        "ppda_allowed": {
            "att": 298,
            "def": 26
        },
        "deep": 3,
        "deep_allowed": 10,
        "scored": 2,
        "missed": 1,
        "xpts": 1.1711,
        "result": "w",
        "date": "2018-08-10 22:00:00",
        "wins": 1,
        "draws": 0,
        "loses": 0,
        "pts": 3,
        "npvGD": -0.9856
    },
    ...,
    {
        "h_a": "a",
        "xG": 2.3703,
        "xGA": 1.52723,
        "npvG": 2.3703,
        "npvGA": 0.766059,
        "ppda": {
            "att": 203,
            "def": 25
        },
        "ppda_allowed": {
            "att": 271,
            "def": 21
        },
        "deep": 7,
        "deep_allowed": 9,
        "scored": 0,
        "missed": 2,
        "xpts": 2.0459,
        "result": "l",
        "date": "2019-03-10 16:30:00",
        "wins": 0,
        "draws": 0,
        "loses": 1,
        "pts": 0,
        "npvGD": 1.604241
    }
]

```

The Contributor Guide

If you want to help **understat** 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 **understat**! 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 programming stuff related to football, 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 `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 *understat* 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 **understat** 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 **understat** 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

- Chris Musson <chris.musson@hotmail.com> @ChrisMusson
- Gracjan Strzelec <gracjanss98@gmail.com> @gracjans

u

`understat`, 5

G

`get_league_fixtures()` (*understat.Understat method*), [6](#)
`get_league_players()` (*understat.Understat method*), [23](#)
`get_league_results()` (*understat.Understat method*), [25](#)
`get_league_table()` (*understat.Understat method*), [8](#)
`get_match_players()` (*understat.Understat method*), [27](#)
`get_match_shots()` (*understat.Understat method*), [30](#)
`get_player_grouped_stats()` (*understat.Understat method*), [11](#)
`get_player_matches()` (*understat.Understat method*), [18](#)
`get_player_shots()` (*understat.Understat method*), [20](#)
`get_player_stats()` (*understat.Understat method*), [22](#)
`get_stats()` (*understat.Understat method*), [33](#)
`get_team_fixtures()` (*understat.Understat method*), [35](#)
`get_team_players()` (*understat.Understat method*), [36](#)
`get_team_results()` (*understat.Understat method*), [38](#)
`get_team_stats()` (*understat.Understat method*), [40](#)
`get_teams()` (*understat.Understat method*), [44](#)

U

`understat` (*module*), [5](#)