<OPENDOTA/> API Documentation

Release 0.2.9

Hrishikesh Terdalkar

Apr 30, 2023

CONTENTS:

A python interface for <OPENDOTA/> API

The `OpenDota` class provided with the package serves as a python interface for the original OpenDota API in the form of a thin wrapper. The class assumes some familiarity with the OpenDota API.

All method calls return serializable python objects, as return by the API, in most cases a dict or a list. Response data is stored as JSON in a local directory (Default: `~/dota2`), to prevent the load on OpenDota API.

- Free software: MIT license
- Documentation: https://pyopendota.readthedocs.io.

CHAPTER

ONE

FEATURES

- Transparent wrapper for majority of the API calls
- Ability to authenticate using API key
- In-built and cusomizable limit to protect against frequent API calls
- Local file-based storage for frequent requests
- Basic CLI using `fire`

CHAPTER

TWO

ABOUT OPENDOTA API

The OpenDota API provides Dota 2 related data including advanced match data extracted from match replays.

OpenDota API Documentation: https://docs.opendota.com/

2.1 <OPENDOTA/> API

A python interface for <OPENDOTA/> API

The `OpenDota` class provided with the package serves as a python interface for the original OpenDota API in the form of a thin wrapper. The class assumes some familiarity with the OpenDota API.

All method calls return serializable python objects, as return by the API, in most cases a dict or a list. Response data is stored as JSON in a local directory (Default: `~/dota2`), to prevent the load on OpenDota API.

- Free software: MIT license
- Documentation: https://pyopendota.readthedocs.io.

2.1.1 Features

- Transparent wrapper for majority of the API calls
- Ability to authenticate using API key
- In-built and cusomizable limit to protect against frequent API calls
- Local file-based storage for frequent requests
- Basic CLI using `fire`

2.1.2 Usage

Use <OPENDOTA/> API in a project

```
import opendota

# Initialize the API-connection object
client = opendota.OpenDota()
```

Get Common Entities

```
client.get_matches('match-id')
client.get_player('player-id')
client.get_team('team-id')
```

Search Functionality

```
players = client.search_player('Dendi')
teams = client.search_team('Alliance')
heroes = client.search_hero('Crystal')
leagues = client.search_league('International')
```

PostgreSQL Query

OpenDota API supports arbitrary PostgreSQL query.

Check Database Schema:

```
client.get_schema()           # Lists all tables
client.get_schema('matches')  # Lists schema for a specific table
```

Arbitrary PostgreSQL Query:

```
client.explorer("select * from matches where limit 1")
```

Use <OPENDOTA/> API Command Line Interface

Information about OpenDota class initialization:

```
opendota --help
```

Information about OpenDota methods:

```
opendota - --help
```

Run methods

```
opendota search_team Virtus
opendota get_match 4080778303
```

Powered by :code:`fire`

2.1.3 About OpenDota API

The OpenDota API provides Dota 2 related data including advanced match data extracted from match replays.

OpenDota API Documentation: https://docs.opendota.com/

2.1.4 Credits

- This package uses data provided by The OpenDota API.

2.2 Installation

2.2.1 Stable release

To install <OPENDOTA/> API, run this command in your terminal:

```
$ pip install pyopendota
```

This is the preferred method to install <OPENDOTA/> API, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2.2 From sources

The sources for <OPENDOTA/> API can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/hrishikeshrt/pyopendota
```

Or download the tarball:

```
$ curl -OJL https://github.com/hrishikeshrt/pyopendota/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

2.3 Usage

2.3.1 Use <OPENDOTA/> API in a project

```
import opendota

# Initialize the API-connection object
client = opendota.OpenDota()
```

Get Common Entities

```
client.get_matches('match-id')
client.get_player('player-id')
client.get_team('team-id')
```

Search Functionality

```
players = client.search_player('Dendi')
teams = client.search_team('Alliance')
heroes = client.search_hero('Crystal')
leagues = client.search_league('International')
```

PostgreSQL Query

OpenDota API supports arbitrary PostgreSQL query.

Check Database Schema:

```
client.get_schema()           # Lists all tables
client.get_schema('matches')  # Lists schema for a specific table
```

Arbitrary PostgreSQL Query:

```
client.explorer("select * from matches where limit 1")
```

2.3.2 Use <OPENDOTA/> API Command Line Interface

Information about OpenDota class initialization:

```
opendota --help
```

Information about OpenDota methods:

```
opendota - --help
```

Run methods

```
opendota search_team Virtus
opendota get_match 4080778303
```

Powered by :code:`fire`

2.4 opendota

2.4.1 opendota package

Submodules

opendota.cli module

Console Script for <OPENDOTA/> API

@author: Hrishikesh Terdalkar

opendota.cli.**main**()

Console Script for <OPENDOTA/> API Powered by *python-fire*

opendota.opendota module

Python Wrapper for <OPENDOTA/> API

The OpenDota API provides Dota 2 related data including advanced match data extracted from match replays.

OpenDota API Documentation: https://docs.opendota.com/

About

The OpenDota class serves as a python interface for the original OpenDota API in the form of a thin wrapper. The class assumes some familiarity with the OpenDota API.

All method calls return serializable python objects, as return by the API, in most cases a dict or a list. Response data is stored as JSON in a local directory (Default: ~/dota2), to prevent the load on OpenDota API.

Features

- Functions for the most frequently used API calls
- Ability to authenticate using API key
- In-built and cusomizable limit to protect against frequent API calls
- Local file-based storage for frequent requests (persistent cache)

class opendota.opendota.**OpenDota**(*data_dir: Optional[str] = None*, *api_key: Optional[str] = None*, *delay: int = 3*, *fantasy: Optional[dict] = None*, *api_url: str = 'https://api.opendota.com/api'*)

Bases: object

<OPENDOTA/> API Interface

Instance of a connection to OpenDota API. All methods return serializable python objects, which are also stored as JSON in the `data_dir` for future calls. All methods take a boolean argument `force` which, if True, will fetch the data again even if it is available in the data directory.

Parameters

- **data_dir** (*str, (optional)*) – Path to data directory for storing responses to API calls The default is `~/dota2`.
- **api_key** (*str, (optional)*) – If you have an OpenDota API key The default is None.
- **delay** (*int, (optional)*) – Delay in seconds between two consecutive API calls. It is recommended to keep this at least 3 seconds, to prevent hitting the daily API limit. If you have an API key, this value is ignored. The default is 3.
- **fantasy** (*dict, (optional)*) – Fantasy DotA2 Configuration Utility constant FANTASY holds the standard values and is used as default. Keys of the `fantasy` will override the default values. They must be a subset of the keys of `FANTASY`.

  Parameters ending with `_base` are used as base values, while others are used as multipliers. e.g. `deaths = -0.3` and `deaths_base = 3` results in the calculation as, `death_score = 3 + (number_of_deaths * -0.3)` If `_base` parameter is absent, it's assumed to be 0.
- **api_url** (*str, (optional)*) – URL to OpenDota API. It is recommended to not change this value.

data_dir: str = None

api_key: str = None

delay: int = 3

fantasy: dict = None

api_url: str = 'https://api.opendota.com/api'

request(*url: str*, ***, *post: bool = False*, *data: Optional[dict] = None*, *filename: Optional[str] = None*, *force: bool = False*) → Any

Make a GET or POST request to <OPENDOTA/> API

Parameters

- **url** (*str*) – API path to query
- **post** (*bool, (optional)*) – Make a POST request. The default is False.
- **data** (*dict, (optional)*) – Query Data. The default is None.
- **filename** (*str, (optional)*) – Save the data to this file. The default is None.
- **force** (*bool, (optional)*) – Force-fetch and overwrite data. The default is False.

Returns
Result of the API call deserialized as a python object

Return type
object

get(**args*, ***kwargs*)

Make a GET request to <OPENDOTA/> API.

Calls .request() with *post=False*

post(**args*, ***kwargs*)

Make a POST request to <OPENDOTA/> API

Calls .request() with *post=True*

request_parse(*match_id: int*)

Submit a new parse request

request_status(*job_id: int*)

Get parse request state

get_constant_names(*force: bool = False*)

Get an array of available resources

get_constants(*resource: Optional[str] = None*, *force: bool = False*)

Get static game data for specified resource(s) (mirrored from the dotaconstants repository)

Parameters

resource (*str or list,*) – Name or names of resources

get_heroes(*force: bool = False*)

Get hero data

get_hero_stats(*force: bool = False*)

Get stats about hero performance in recent matches

get_hero_benchmarks(*hero_id: int*, *force: bool = False*)

Get benchmarks for a hero

get_leagues(*force: bool = False*)

Get a list of leagues

get_league(*league_id: int*, *force: bool = False*)

Get data for a league

get_league_matches(*league_id: int*, *force: bool = False*)

Get matches from a league

get_league_teams(*league_id: int*, *force: bool = False*)

Get teams from a league

get_teams(*force: bool = False*)

Get team data

get_team(*team_id: int*, *force: bool = False*)

Get data for a team

get_team_matches(*team_id: int*, *force: bool = False*)

Get matches for a team

get_team_players(*team_id: int*, *current: bool = True*, *force: bool = False*)

Get players who have played for a team

get_team_heroes(*team_id: int*, *force: bool = False*)

Get heroes for a team

get_match(*match_id: int*, *force: bool = False*)

Get match data

get_pro_matches(*match_id: Optional[int] = None*, *force: bool = False*)

Get a list of pro matches

get_live()

Get top currently ongoing live games

get_player(*account_id: int*, *force: bool = False*)

Player data

get_pro_players(*force: bool = False*)

Get a list of pro players

get_player_heroes(*player_id: int*, *force: bool = False*)

Get heroes played by a player

get_player_matches(*player_id: int*, *request_parse: bool = False*, *days: int = 180*, *force: bool = False*)

Matches played by a player

get_player_ratings(*player_id: int*, *force: bool = False*)

Player rating history

get_player_rankings(*player_id: int*, *force: bool = False*)

Player hero rankings

search_hero(*search_key: Optional[str] = None*, *attack_type: Optional[str] = None*, *roles: Optional[List[str]] = None*)

Search for a hero by name, attack type or roles

search_league(*search_key: str*)

Search for a league

search_team(*search_key: str*)

Search for a team by name or tag

search_player(*search_key: str*)

Search for a player

get_match_fantasy(*match_id: int*, *force: bool = False*)

Get Fantasy Points of All Players from a Match

Parameters
match_id (*int or str*) – Match ID

Returns
Fantasy profiles of players from the specified match

Return type
Dict

get_schema(*table_name: Optional[str] = None*, *force: bool = False*)

Get database schema

Parameters
table_name (*str*) – Get schema for table_name If None, list the available table names

explorer(*sql: str*, *debug: bool = False*)

Submit arbitrary PostgreSQL queries to the database

query(**args*, ***kwargs*)

Submit arbitrary PostgreSQL queries to the database

update_data(*frequency: int = 30*)

Update core data

Parameters

frequency (*int*) – It is recommended to use utility constants, FREQ_LOW, FREQ_MEDIUM or FREQ_HIGH to specifcy frequency.

FREQ_HIGH: update data that changes frequently
(e.g. teams)

FREQ_MEDIUM: update data that changes with a moderate frequency
(e.g. hero benchmarks)

FREQ_LOW: update data that changes very infrequently
(e.g. heroes)

Module contents

Top-level package for <OPENDOTA/> API.

class opendota.**OpenDota**(*data_dir: Optional[str] = None*, *api_key: Optional[str] = None*, *delay: int = 3*, *fantasy: Optional[dict] = None*, *api_url: str = 'https://api.opendota.com/api'*)

Bases: `object`

<OPENDOTA/> API Interface

Instance of a connection to OpenDota API. All methods return serializable python objects, which are also stored as JSON in the `data_dir` for future calls. All methods take a boolean argument `force` which, if True, will fetch the data again even if it is available in the data directory.

Parameters

- **data_dir** (*str, (optional)*) – Path to data directory for storing responses to API calls The default is `~/dota2`.
- **api_key** (*str, (optional)*) – If you have an OpenDota API key The default is None.
- **delay** (*int, (optional)*) – Delay in seconds between two consecutive API calls. It is recommended to keep this at least 3 seconds, to prevent hitting the daily API limit. If you have an API key, this value is ignored. The default is 3.
- **fantasy**(*dict, (optional)*) – Fantasy DotA2 Configuration Utility constant FANTASY holds the standard values and is used as default. Keys of the `fantasy` will override the default values. They must be a subset of the keys of `FANTASY`.

  Parameters ending with `_base` are used as base values, while others are used as multipliers. e.g. `deaths = -0.3` and `deaths_base = 3` results in the calculation as, `death_score = 3 + (number_of_deaths * -0.3)` If `_base` parameter is absent, it's assumed to be 0.
- **api_url** (*str, (optional)*) – URL to OpenDota API. It is recommended to not change this value.

data_dir: **str = None**

api_key: **str = None**

delay: **int = 3**

`fantasy: dict = None`

`api_url: str = 'https://api.opendota.com/api'`

`request`(*url: str*, ***, *post: bool = False*, *data: Optional[dict] = None*, *filename: Optional[str] = None*, *force: bool = False*) → Any

Make a GET or POST request to <OPENDOTA/> API

Parameters

- **url** (*str*) – API path to query
- **post** (*bool, (optional)*) – Make a POST request. The default is False.
- **data** (*dict, (optional)*) – Query Data. The default is None.
- **filename** (*str, (optional)*) – Save the data to this file. The default is None.
- **force** (*bool, (optional)*) – Force-fetch and overwrite data. The default is False.

Returns
Result of the API call deserialized as a python object

Return type
object

`get`(**args*, ***kwargs*)

Make a GET request to <OPENDOTA/> API.

Calls .request() with *post=False*

`post`(**args*, ***kwargs*)

Make a POST request to <OPENDOTA/> API

Calls .request() with *post=True*

`request_parse`(*match_id: int*)

Submit a new parse request

`request_status`(*job_id: int*)

Get parse request state

`get_constant_names`(*force: bool = False*)

Get an array of available resources

`get_constants`(*resource: Optional[str] = None*, *force: bool = False*)

Get static game data for specified resource(s) (mirrored from the dotaconstants repository)

Parameters
resource (*str or list,*) – Name or names of resources

`get_heroes`(*force: bool = False*)

Get hero data

`get_hero_stats`(*force: bool = False*)

Get stats about hero performance in recent matches

`get_hero_benchmarks`(*hero_id: int*, *force: bool = False*)

Get benchmarks for a hero

`get_leagues`(*force: bool = False*)

Get a list of leagues

get_league(*league_id: int*, *force: bool = False*)

Get data for a league

get_league_matches(*league_id: int*, *force: bool = False*)

Get matches from a league

get_league_teams(*league_id: int*, *force: bool = False*)

Get teams from a league

get_teams(*force: bool = False*)

Get team data

get_team(*team_id: int*, *force: bool = False*)

Get data for a team

get_team_matches(*team_id: int*, *force: bool = False*)

Get matches for a team

get_team_players(*team_id: int*, *current: bool = True*, *force: bool = False*)

Get players who have played for a team

get_team_heroes(*team_id: int*, *force: bool = False*)

Get heroes for a team

get_match(*match_id: int*, *force: bool = False*)

Get match data

get_pro_matches(*match_id: Optional[int] = None*, *force: bool = False*)

Get a list of pro matches

get_live()

Get top currently ongoing live games

get_player(*account_id: int*, *force: bool = False*)

Player data

get_pro_players(*force: bool = False*)

Get a list of pro players

get_player_heroes(*player_id: int*, *force: bool = False*)

Get heroes played by a player

get_player_matches(*player_id: int*, *request_parse: bool = False*, *days: int = 180*, *force: bool = False*)

Matches played by a player

get_player_ratings(*player_id: int*, *force: bool = False*)

Player rating history

get_player_rankings(*player_id: int*, *force: bool = False*)

Player hero rankings

search_hero(*search_key: Optional[str] = None*, *attack_type: Optional[str] = None*, *roles: Optional[List[str]] = None*)

Search for a hero by name, attack type or roles

search_league(*search_key: str*)

Search for a league

search_team(*search_key: str*)

Search for a team by name or tag

search_player(*search_key: str*)

Search for a player

get_match_fantasy(*match_id: int*, *force: bool = False*)

Get Fantasy Points of All Players from a Match

Parameters
match_id (*int or str*) – Match ID

Returns
Fantasy profiles of players from the specified match

Return type
Dict

get_schema(*table_name: Optional[str] = None*, *force: bool = False*)

Get database schema

Parameters
table_name (*str*) – Get schema for table_name If None, list the available table names

explorer(*sql: str*, *debug: bool = False*)

Submit arbitrary PostgreSQL queries to the database

query(**args*, ***kwargs*)

Submit arbitrary PostgreSQL queries to the database

update_data(*frequency: int = 30*)

Update core data

Parameters
frequency (*int*) – It is recommended to use utility constants, FREQ_LOW, FREQ_MEDIUM or FREQ_HIGH to specifcy frequency.

FREQ_HIGH: update data that changes frequently
(e.g. teams)

FREQ_MEDIUM: update data that changes with a moderate frequency
(e.g. hero benchmarks)

FREQ_LOW: update data that changes very infrequently
(e.g. heroes)

2.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

2.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/hrishikeshrt/pyopendota/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

<OPENDOTA/> API could always use more documentation, whether as part of the official <OPENDOTA/> API docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/hrishikeshrt/pyopendota/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.5.2 Get Started!

Ready to contribute? Here's how to set up *pyopendota* for local development.

1. Fork the *pyopendota* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/pyopendota.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv pyopendota
$ cd pyopendota/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 opendota tests
$ python setup.py test or pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5, 3.6, 3.7 and 3.8, and for PyPy. Check https://travis-ci.com/hrishikeshrt/pyopendota/pull_requests and make sure that the tests pass for all supported Python versions.

2.5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_opendota
```

2.5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

2.6 Credits

2.6.1 Development Lead

- Hrishikesh Terdalkar <hrishikeshrt@linuxmail.org>

2.6.2 Contributors

None yet. Why not be the first?

2.7 History

2.7.1 0.2.0 (2021-10-04)

- Improved Documentation
- League specific functions
- Basic CLI using *python-fire*
- Fantasy score calculation
- Bugfixes

2.7.2 0.1.0 (2021-07-20)

- First release on PyPI.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

O

INDEX

M

O

P

Q

R

S

U