

---

# **apexpy Documentation**

***Release 0.1***

**surister**

**Mar 13, 2019**



---

## Contents

---

<b>1</b>	<b>Documentation Contents</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.1.1	Quick example . . . . .	3
1.1.2	Output . . . . .	4
1.2	Installation and Set up . . . . .	4
1.2.1	Downloading the library: . . . . .	4
1.2.2	Get an Api key . . . . .	4
1.2.3	Providing your Api Key . . . . .	4
1.3	Using the library . . . . .	5
1.4	Api Reference . . . . .	6
1.4.1	Apex Api . . . . .	6
1.4.2	Apex Character . . . . .	7
1.4.3	Apex request . . . . .	7
1.4.4	Exceptions . . . . .	8







### 1.1 Introduction

Apexpy it's an asynchronous api wrapper for Apex-legend written in [python 3.6+](#)

---

**Note:** This project or its documentation is in no way related to [EA](#) or the api it wraps [Apex Legends - API tracker](#)

---

Apex py is written using aiohttp and wrapps all the data given by the api in to a convenient scheme for your use.

#### 1.1.1 Quick example

```
import asyncio
from apexpy import ApexApi

async def main():

    player = ApexApi(key='SecretApiKey')
    await player.search('DiegosaurusTTV', 'pc')

    for legend in player.legends:
        print(f'{legend.name} -> {legend.stats}')

    print(player.stats)

asyncio.run(main())
```

## 1.1.2 Output

```
Wraith -> [{'Kills': 4329.0, 'rank': 658}, {'KillsPerMatch': 8.78, 'rank': 117}, {
↳ 'DamagePerMatch': 1696.19, 'rank': 128}, {'Damage': 836222.0, 'rank': 926}, {
↳ 'MatchesPlayed': 493.0, 'rank': 866}]
Mirage -> [{'Kills': 10324.0, 'rank': 1}, {'Damage': 2062356.0, 'rank': 1}, {
↳ 'Headshots': 6869.0, 'rank': 4}, {'CarePackageKills': 89.0, 'rank': 2}]
Lifeline -> [{'Kills': 14.0, 'rank': None}]
Bloodhound -> [{'Kills': 139.0, 'rank': 32243}]
Caustic -> [{'Kills': 41.0, 'rank': 7054}]
Gibraltar -> [{'Kills': 102.0, 'rank': 1915}]
Bangalore -> [{'Kills': 384.0, 'rank': 27236}, {'Damage': 78092.0, 'rank': 23243}]
Pathfinder -> [{'Kills': 37.0, 'rank': 27638}, {'Specific2': 17665.0, 'rank': 12108}]
[{'Level': 392.0, 'rank': None}, {'Kills': 15370.0, 'rank': 1}, {'KillsPerMatch': 31.
↳ 18, 'rank': None}, {'DamagePerMatch': 6037.87, 'rank': None}, {'Damage': 2976670.0,
↳ 'rank': 1}, {'MatchesPlayed': 493.0, 'rank': None}, {'CarePackageKills': 89.0, 'rank
↳ ': None}]
```

## 1.2 Installation and Set up

### 1.2.1 Downloading the library:

Using bare git:

```
$ git clone https://github.com/surister/apexpy.git
```

Using pip:

```
$ pip install apexlegendspy
```

Using pipenv:

```
$ pipenv install apexlegendspy
```

### 1.2.2 Get an Api key

To get your api key go [here](#) and follow the steps.

### 1.2.3 Providing your Api Key

Any api key should remain secret for your only use, to avoid leaking your key, you have it as an environment variable **APEX\_LEY=actual\_key** apexpy will look for it automatically.

---

**Note:** Quick way to have this done:

1. Install dotenv:

```
$ pip install python-dotenv
or
$ pipenv install python-dotenv
```

2. Create a .env file and put there your api key as:



```
APEX_KEY=KEY
```

*Apexpy automatically loads dotenv files if python-dotenv is installed*

## 1.3 Using the library

Because both are asynchronous, discord bots and Apexpy go really well together.

This snippet is written in python 3.7 using discord.py (rewrite branch) and shows a way of using Apexpy in Cog.

```
from discord.ext.commands import command
from apexpy import ApexApi

class ApexLegend:
    def __init__(self, bot):
        self.bot = bot

    @command()
    async def apexsearch(self, ctx, name: str, platform: str) -> None:
        player = ApexApi('secret_key')
        await player.search(name, platform)
        for legend in player.legends:
            await ctx.send(legend.stats)

def setup(bot):
    bot.add_cog(ApexLegend(bot))
```

Output:



The screenshot shows a Discord chat interface with a dark theme. At the top, a user named 'surister' has sent a command: `%apexsearch DiegosaurusTTV pc`. Below this, a series of bot responses are shown, each starting with 'BOT sur' followed by a JSON array of player statistics. The statistics include 'Kills', 'rank', 'KillsPerMatch', 'DamagePerMatch', 'Damage', 'MatchesPlayed', 'Headshots', 'CarePackageKills', and 'Specific2'. The data is as follows:

- 3:16 PM BOT sur `[{'Kills': 4408.0, 'rank': 658}, {'KillsPerMatch': 8.8, 'rank': 117}, {'DamagePerMatch': 1702.97, 'rank': 128}, {'Damage': 853186.0, 'rank': 926}, {'MatchesPlayed': 501.0, 'rank': 866}]`
- BOT sur `[{'Kills': 10326.0, 'rank': 1}, {'Damage': 2063040.0, 'rank': 1}, {'Headshots': 6869.0, 'rank': 4}, {'CarePackageKills': 89.0, 'rank': 2}]`
- BOT sur `[{'Kills': 14.0, 'rank': None}]`
- BOT sur `[{'Kills': 139.0, 'rank': 32243}]`
- BOT sur `[{'Kills': 41.0, 'rank': 7054}]`
- BOT sur `[{'Kills': 102.0, 'rank': 1915}]`
- BOT sur `[{'Kills': 384.0, 'rank': 27236}, {'Damage': 78092.0, 'rank': 23243}]`
- BOT sur `[{'Kills': 37.0, 'rank': 27638}, {'Specific2': 17665.0, 'rank': 12108}]`

You make sure you prettify everything, as most of the data is pure object oriented, its manipulation is very easy.

Apexpy can of course, be run as a standalone using asyncio.

```
import asyncio
from apexpy import ApexApi

async def main():

    player = ApexApi()

    await player.search('DiegosaurusTTV', 'pc')

    for legend in player.legends:
        print(f'{legend.name} -> {legend.stats}')

    print(player.stats)

asyncio.run(main())
```

For more detailed information of the library, check the api reference.

## 1.4 Api Reference

### 1.4.1 Apex Api

**class** apexpy.api.**ApexApi** (*key: str = None*)

Represents the Apex api containing all the data requested by [ApexRequest](#).

**Parameters** **key** (*str*) – The apex legends api key.

**name**

*str* Player's name

**key**

*str* api key

**legends**

List[[ApexCharacter](#)] Legends' data

**stats**

List[dict] Player's general stats

**id**

*str* Player's id

**\_populate** (*data*) → None

Populates object with data

**Parameters** **data** –

**Returns** None

**search** (*name: str, platform: str*) → None

Creates the [ApexRequest](#) object that takes care of the api's communication. Automatically sets variables and call the population method.

**Parameters**

- **name** – *str*

- **platform** – `str` Three platforms supported are pc, xbox and psn, they are transformed into their numeric code, 5, 2, 1 respectively.

**Returns** `None`

## 1.4.2 Apex Character

**class** `apexpy.characters.ApexCharacter` (*data*)

Represents an Apex legend character

**Parameters** `data` (`dict`) – Every characters data.

**raw\_data**

`dict` Raw data

**name**

`str` Character's name

**id**

`str` Every character id.

Wraith -> legend\_1, Bangalore -> legend\_2, Caustic -> legend\_3, Mirage -> legend\_4, Bloodhound -> legend\_5, Gibraltar -> legend\_6, Lifeline -> legend\_7, Pathfinder -> legend\_8

**icon**

`str` Character's icon image

**bgimage**

`str` Character's background image

**stats**

List[`dict`] Character's stats data

## 1.4.3 Apex request

**class** `apexpy.httprequest.ApexRequest` (*name: str, platform: int, api\_key: str = None*)

Represents the asynchronous http call made to the apex legends api

**Parameters**

- **name** (`str`) – Player's name
- **platform** (`str`) – Platform numeric code
- **api\_key** (`str`) – Apex key, if not provided directly it'll be looked in os.environ

**platform**

**name**

**api\_key**

**headers**

`dict` Headers sent with the call {'TRN-API-KEY': key}

**self.req\_url**

`str` End point

**session** () → <sphinx.ext.autodoc.importer.\_MockObject object at 0x7f4a6b884cc0>

Makes the api call. :return: `aiohttp.client.ClientResponse`

### 1.4.4 Exceptions

**class** apexpy.exceptions.**CustomBaseException** (*message=""*)

Base exception, exceptions are raised depending on the received code from the api.

```
API_ERROR_MAP = { 401: UnauthorizedError, 400: NotFoundError, 403:
↳UnauthorizedError, 404: PlayerNotFoundError, 500: ServerError}
```

**class** apexpy.exceptions.**ApiKeyNotProvidedError** (*message=""*)

Raised when the api key is not directly provided and doesn't exist in os.environ

**class** apexpy.exceptions.**NotFoundError** (*message=""*)

**class** apexpy.exceptions.**ServerError** (*message=""*)

**class** apexpy.exceptions.**UnauthorizedError** (*message=""*)

## Symbols

`_populate()` (apexpy.api.ApexApi method), 6

## A

ApexApi (class in apexpy.api), 6

ApexCharacter (class in apexpy.characters), 7

ApexRequest (class in apexpy.httprequest), 7

api\_key (apexpy.httprequest.ApexRequest attribute), 7

ApiKeyNotProvidedError (class in apexpy.exceptions), 8

## B

bgimage (apexpy.characters.ApexCharacter attribute), 7

## C

CustomBaseException (class in apexpy.exceptions), 8

## H

headers (apexpy.httprequest.ApexRequest attribute), 7

## I

icon (apexpy.characters.ApexCharacter attribute), 7

id (apexpy.api.ApexApi attribute), 6

id (apexpy.characters.ApexCharacter attribute), 7

## K

key (apexpy.api.ApexApi attribute), 6

## L

legends (apexpy.api.ApexApi attribute), 6

## N

name (apexpy.api.ApexApi attribute), 6

name (apexpy.characters.ApexCharacter attribute), 7

name (apexpy.httprequest.ApexRequest attribute), 7

NotFoundError (class in apexpy.exceptions), 8

## P

platform (apexpy.httprequest.ApexRequest attribute), 7

## R

raw\_data (apexpy.characters.ApexCharacter attribute), 7

req\_url (apexpy.httprequest.ApexRequest.self attribute),  
7

## S

search() (apexpy.api.ApexApi method), 6

ServerError (class in apexpy.exceptions), 8

session() (apexpy.httprequest.ApexRequest method), 7

stats (apexpy.api.ApexApi attribute), 6

stats (apexpy.characters.ApexCharacter attribute), 7

## U

UnauthorizedError (class in apexpy.exceptions), 8