

---

# **pyanimelist Documentation**

***Release 1.2.2***

**Byron Vanstien**

February 13, 2017



<b>1</b>	<b>About</b>	<b>1</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
<b>3</b>	<b>Contents:</b>	<b>5</b>
3.1	Examples . . . . .	5
3.2	What's new! . . . . .	8
3.3	v1.2.3 . . . . .	8
3.4	API reference . . . . .	8
3.5	Client . . . . .	8
3.6	Exceptions . . . . .	8
3.7	Dataclasses . . . . .	8
3.8	Enumerations . . . . .	8
3.9	Abstractions . . . . .	9
3.10	FAQ . . . . .	9



---

### About

---

*Pyanimelist* is an asynchronous wrapper for the MyAnimeList API, using `lxml` and `aiohttp`.



---

## Installation

---

Pyanimelist will only work on Python 3.5+ as it heavily uses the [Typing](#) library, which is only available from 3.5 onwards. It also uses the 3.5 syntax *async def* rather than the *@asyncio.coroutine*.

Pyanimelist is installed through pip for the latest released version.

```
$ pip install pyanimelist
```

If you want the latest development version you can install from the git repo on [GitHub](#)

```
$ pip install git+https://github.com/GetRektByMe/Pyanimelist.git
```

**Note that the Git version isn't guaranteed to be work properly and lots of things may be broken.**





---

**Contents:**

---

## 3.1 Examples

### 3.1.1 Search anime

```
import asyncio

import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())
results = loop.run_until_complete(instance.search_all_anime("Mahouka koukou no rettousei"))

print(results)
```

### 3.1.2 Search manga

```
import asyncio

import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())
results = loop.run_until_complete(instance.search_all_manga("Mahouka koukou no rettousei"))

print(results)
```

### 3.1.3 Add anime

```
import asyncio

import pyanimelist
from pyanimelist.enumerations import AnimeStatus
```

```
instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

# The first argument is the anime id, the second argument is the status
loop.run_until_complete(instance.add_anime(1, AnimeStatus.WATCHING.value))
```

### 3.1.4 Add manga

```
import asyncio

import pyanimelist
from pyanimelist.enumerations import MangaStatus

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

# The first argument is the manga id, the second is the status
loop.run_until_complete(instance.add_manga(1, MangaStatus.READING.value))
```

### 3.1.5 Update anime

```
import asyncio
import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

# This can take anything specified in the docstring but nothing is required besides the anime's id
loop.run_until_complete(instance.update_anime(1))
```

### 3.1.6 Update manga

```
import asyncio
import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

# This can take anything specified in the docstring but nothing is required besides the anime's id
loop.run_until_complete(instance.update_manga(1))
```

### 3.1.7 Delete anime

```
import asyncio
import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

loop.run_until_complete(instance.delete_anime(1))
```

### 3.1.8 Delete manga

```
import asyncio
import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

loop.run_until_complete(instance.delete_manga(1))
```

### 3.1.9 Get user series

```
import asyncio
import pyanimelist

# This can be either anime or manga
series_type = "anime"

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

loop.run_until_complete(instance.get_user_series(username, series_type))
```

### 3.1.10 Get user data

```
import asyncio
import pyanimelist

instance = pyanimelist.PyAnimeList(username, password)
loop = asyncio.get_event_loop()

loop.run_until_complete(instance.verify_credentials())

loop.run_until_complete(instance.get_user_data(username))
```

## 3.2 What's new!

This page keeps a human readable list of what has changed throughout the versions, starting from v1.2.3

### 3.3 v1.2.3

- `pyanimelist.errors.InvalidCredentials` now inherits from `ResponseError`
- Switch from strings to datetime objects on `pyanimelist.abstractions.Dates`

## 3.4 API reference

---

**Note:** There is currently nothing logged on the backend of Pyanimelist; This is planned to be added in future commits.

---

---

**Note:** The use of BeautifulSoup will soon be depreciated and will switch to an lxml backend for the one function that uses BeautifulSoup.

---

### 3.4.1 Find out the version of Pyanimelist you have

`pyanimelist.__version__`

A string that represents the version of Pyanimelist you're using, for example "1.0.0"

## 3.5 Client

## 3.6 Exceptions

## 3.7 Dataclasses

---

**Note:** These are not for creating yourself, these are handled and given to you from the libraries backend.

---

## 3.8 Enumerations

The API returns integers which we plug into our [enumerations](#) for easy filtering

**class** `pyanimelist.enumerations.AnimeStatus`

Specifies the status of the Anime.

**WATCHING**

The anime is currently being watched

**COMPLETED**

The anime has been completely watched

**ON\_HOLD**

The anime is currently on hold

**DROPPED**

The anime was dropped mid way through watching

**PLAN\_TO\_WATCH**

The anime has been planned to be watched

**class** `pyanimelist.enumerations.MangaStatus`  
Specifies the status of the Manga.

**READING**

The manga is currently being read

**COMPLETED**

The manga has been completely read

**ON\_HOLD**

The manga is currently on hold

**DROPPED**

The manga was dropped mid way through reading

**PLAN\_TO\_READ**

The manga has been planned to be read

## 3.9 Abstractions

---

**Note:** These are used in the backend of the library and are not for using outside it!

---

### 3.10 FAQ



## Symbols

`__version__` (in module `pyanimelist`), 8

## C

`COMPLETED` (`pyanimelist.pyanimelist.enumerations.AnimeStatus` attribute), 8

`COMPLETED` (`pyanimelist.pyanimelist.enumerations.MangaStatus` attribute), 9

## D

`DROPPED` (`pyanimelist.pyanimelist.enumerations.AnimeStatus` attribute), 9

`DROPPED` (`pyanimelist.pyanimelist.enumerations.MangaStatus` attribute), 9

## O

`ON_HOLD` (`pyanimelist.pyanimelist.enumerations.AnimeStatus` attribute), 9

`ON_HOLD` (`pyanimelist.pyanimelist.enumerations.MangaStatus` attribute), 9

## P

`PLAN_TO_READ` (`pyanimelist.pyanimelist.enumerations.MangaStatus` attribute), 9

`PLAN_TO_WATCH` (`pyanimelist.pyanimelist.enumerations.AnimeStatus` attribute), 9

`pyanimelist.enumerations.AnimeStatus` (class in `pyanimelist`), 8

`pyanimelist.enumerations.MangaStatus` (class in `pyanimelist`), 9

## R

`READING` (`pyanimelist.pyanimelist.enumerations.MangaStatus` attribute), 9

## W

`WATCHING` (`pyanimelist.pyanimelist.enumerations.AnimeStatus` attribute), 8