# younit Documentation

***Release 0.1.0***

**Jeremy Arr**

**Sep 14, 2017**

# Contents:

younit is a collection of helpers for the `unittest` module.

CHAPTER 1

# Helpers

| I want to | Helpers to Use |
|---|---|
| Test coroutines | *@asyncio_test* |
| Mock out coroutines | *AsyncMock()*, *@asyncio_test* |
| Print the name of a test before running it | *@test_name* |
| Fail a test if it hangs | *@set_test_hang_alarm*, *@clear_test_hang_alarm*, or *@test_hang_alarm* |
| Close all threads associated with a test | *@close_all_threads* |

CHAPTER 2

# Get It Now

```
$ pip install younit
```

CHAPTER 3

# Examples

Testing and mocking coroutines:

```
class MyTestCase(unittest.TestCase):
    async def async_setUp(self):
        pass

    async def async_tearDown(self):
        pass

    @asyncio_test
    async def test_this(self):
        x = AsyncMock()
        await x()
        x.mock.assert_called_once()
```

Setting up test hang alarms:

```
class MyTestCase(unittest.TestCase):
    @set_test_hang_alarm
    def setUp(self):
        pass

    @clear_test_hang_alarm
    def tearDown(self):
        pass
```

Check out the individual *helpers* for more examples.

CHAPTER 4

# Project Documentation

## API

younit is a collection of helpers for the `unittest` module.

`younit.`**`test_name`**(*func*)

A decorator that prints the test name before starting a test.

Convenient if you want to separate the output of different tests.

Usage:

```
class MyTestCase(unittest.TestCase):
    @test_name
    def test_this(self):
        print("im testing this")

    @test_name
    def test_that(self):
        print("im testing that")
```

`younit.`**`set_test_hang_alarm`**(*func*)

A decorator that sets an alarm of 1 second before starting any test.

If a test takes longer than 1 second, a *`TestHang`* exception is raised.

Should be used during set up and in conjunction with *`clear_test_hang_alarm()`* during tear down.

Usage:

```
class MyTestCase(unittest.TestCase):
    @set_test_hang_alarm
    def setUp(self):
        pass

    @clear_test_hang_alarm
```

```
    def tearDown(self):
        pass
```

younit.**clear_test_hang_alarm**(*func*)

A decorator that resets an alarm set by *set_test_hang_alarm()*

Should be used during tear down and in conjunction with *set_test_hang_alarm()* during set up.

Usage:

```
class MyTestCase(unittest.TestCase):
    @set_test_hang_alarm
    def setUp(self):
        pass

    @clear_test_hang_alarm
    def tearDown(self):
        pass
```

younit.**test_hang_alarm**(*func*)

A decorator that sets an alarm of 1 second before starting any test.

If a test takes longer than 1 second, a *TestHang* exception is raised.

If a test takes less than 1 second, the alarm is cancelled.

Usage:

```
class MyTestCase(unittest.TestCase):

    @test_hang_alarm
    def test_this(self):
        time.sleep(3)
```

younit.**close_all_threads**(*func*)

A decorator that closes any threads that are created as part of running a test.

To use, ensure your threads are able to be closed by invoking a `close()` method on an object related to the thread. Then add the object to the `self.threads_to_close` list.

Usage:

```
class MyTestCase(unittest.TestCase):
    def setUp(self):
        self.threads_to_close = []
        x = start_a_new_thread()
        #x is an object with a close() method
        #that closes the thread
        self.threads_to_close.append(x)

    @close_all_threads
    def test_this(self):
        y = start_a_new_thread()
        self.threads_to_close.append(y)
```

younit.**asyncio_test**(*func*)

A decorator that runs a test as a coroutine including any set up and tear down coroutines.

Usage:

```
class MyTestCase(unittest.TestCase):
    async def async_setUp(self):
        pass

    async def async_tearDown(self):
        pass

    @asyncio_test
    async def test_this(self):
        pass
```

younit.**AsyncMock**(**args*, ***kwargs*)

A function that can be used to mock a coroutine.

Returns a coroutine function with a mock attribute. The mock attribute is a unittest.mock.MagicMock object that records usage.

Usage:

```
class MyTestCase(unittest.TestCase):
    async def async_setUp(self):
        pass

    async def async_tearDown(self):
        pass

    @asyncio_test
    async def test_this(self):
        x = AsyncMock()
        await x()
        x.mock.assert_called_once()
```

**exception** younit.**TestHang**

# Changelog

## 0.1.0 (2017-07-21)

- Initial release

# License

```
MIT License

Copyright (c) 2017 Jeremy Arr

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

# Authors

## Leads

- Jeremy Arr @jeremyarr

## Contributors

# Kudos

- Blog posts here and here on some simple ways to unit test asyncio code

# Contributing

Source code for *younit* can be found here . New issues, feature requests and pull requests are all welcome.

CHAPTER 5

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## y

# Index