Your Tests Are Lying To You

Kris Neuharth

kris.neuharth@gmail.com

@krisneuharth

github/krisneuharth

Hello.

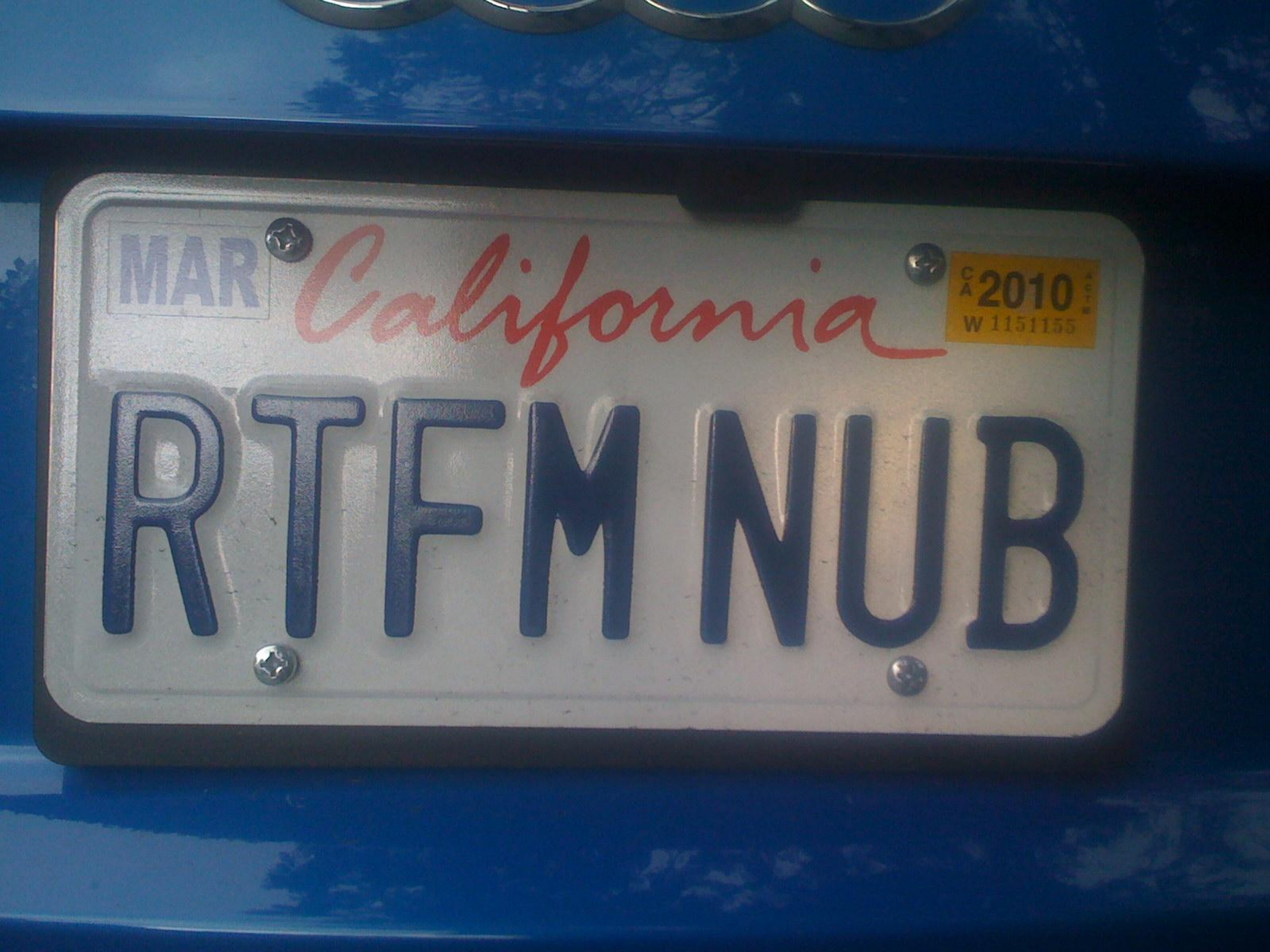
Credit.

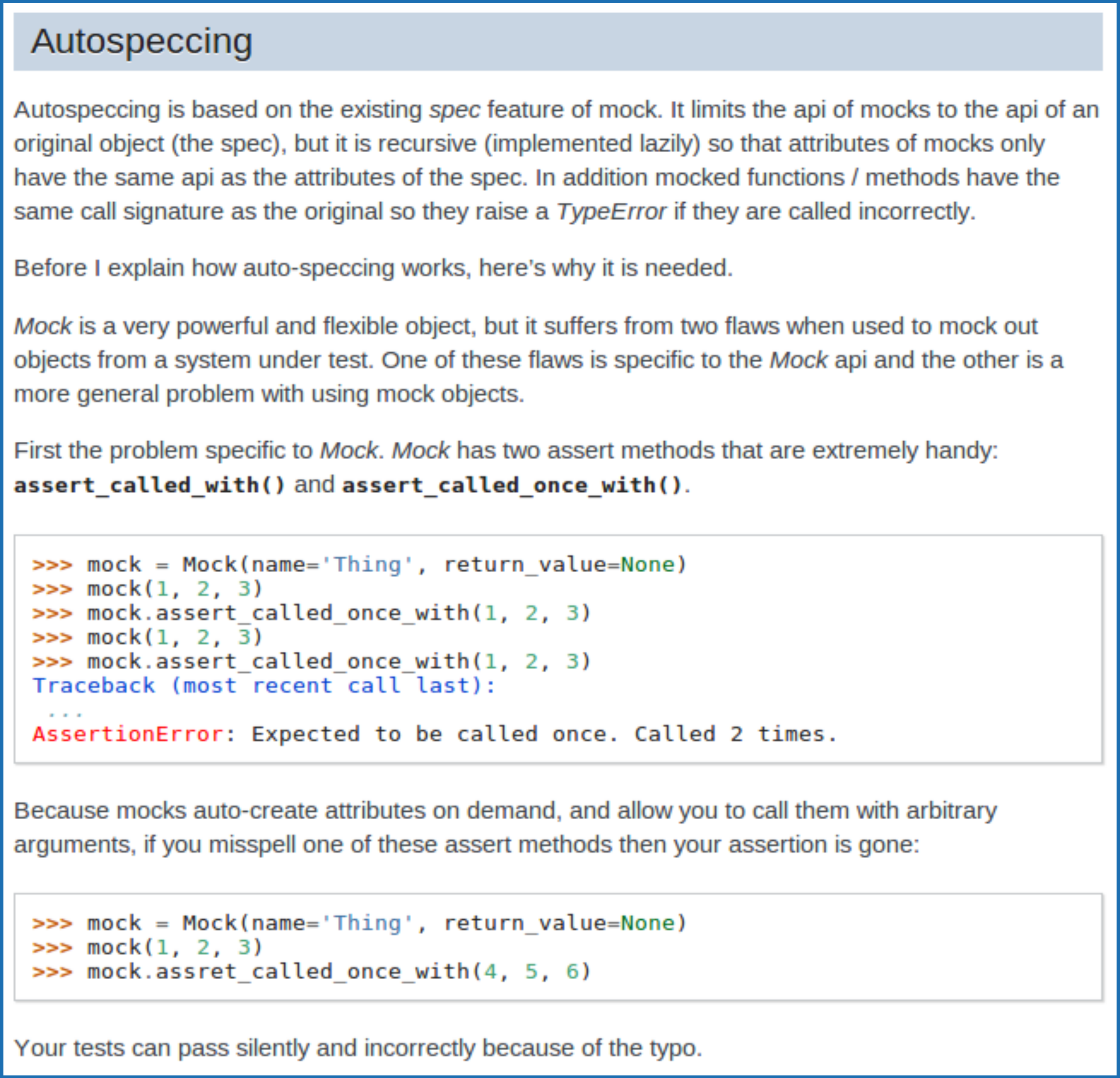
Drink.

500+ Tests
76% Coverage

assert_called_once()

NOPE!


MAR
California
CA 2010
W 1151155
RTFM NUB



Autospeccing

Autospeccing is based on the existing *spec* feature of mock. It limits the api of mocks to the api of an original object (the spec), but it is recursive (implemented lazily) so that attributes of mocks only have the same api as the attributes of the spec. In addition mocked functions / methods have the same call signature as the original so they raise a *TypeError* if they are called incorrectly.

Before I explain how auto-speccing works, here's why it is needed.

Mock is a very powerful and flexible object, but it suffers from two flaws when used to mock out objects from a system under test. One of these flaws is specific to the *Mock* api and the other is a more general problem with using mock objects.

First the problem specific to *Mock*. *Mock* has two assert methods that are extremely handy: **`assert_called_with()`** and **`assert_called_once_with()`**.

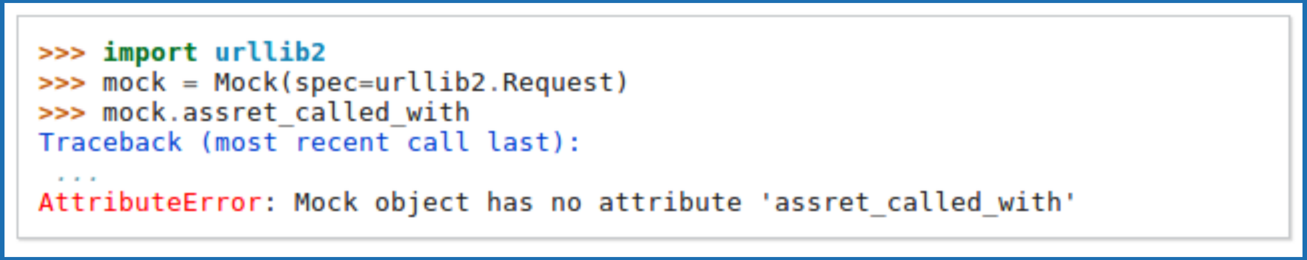
```
>>> mock = Mock(name='Thing', return_value=None)
>>> mock(1, 2, 3)
>>> mock.assert_called_once_with(1, 2, 3)
>>> mock(1, 2, 3)
>>> mock.assert_called_once_with(1, 2, 3)
Traceback (most recent call last):
 ...
AssertionError: Expected to be called once. Called 2 times.
```

Because mocks auto-create attributes on demand, and allow you to call them with arbitrary arguments, if you misspell one of these assert methods then your assertion is gone:

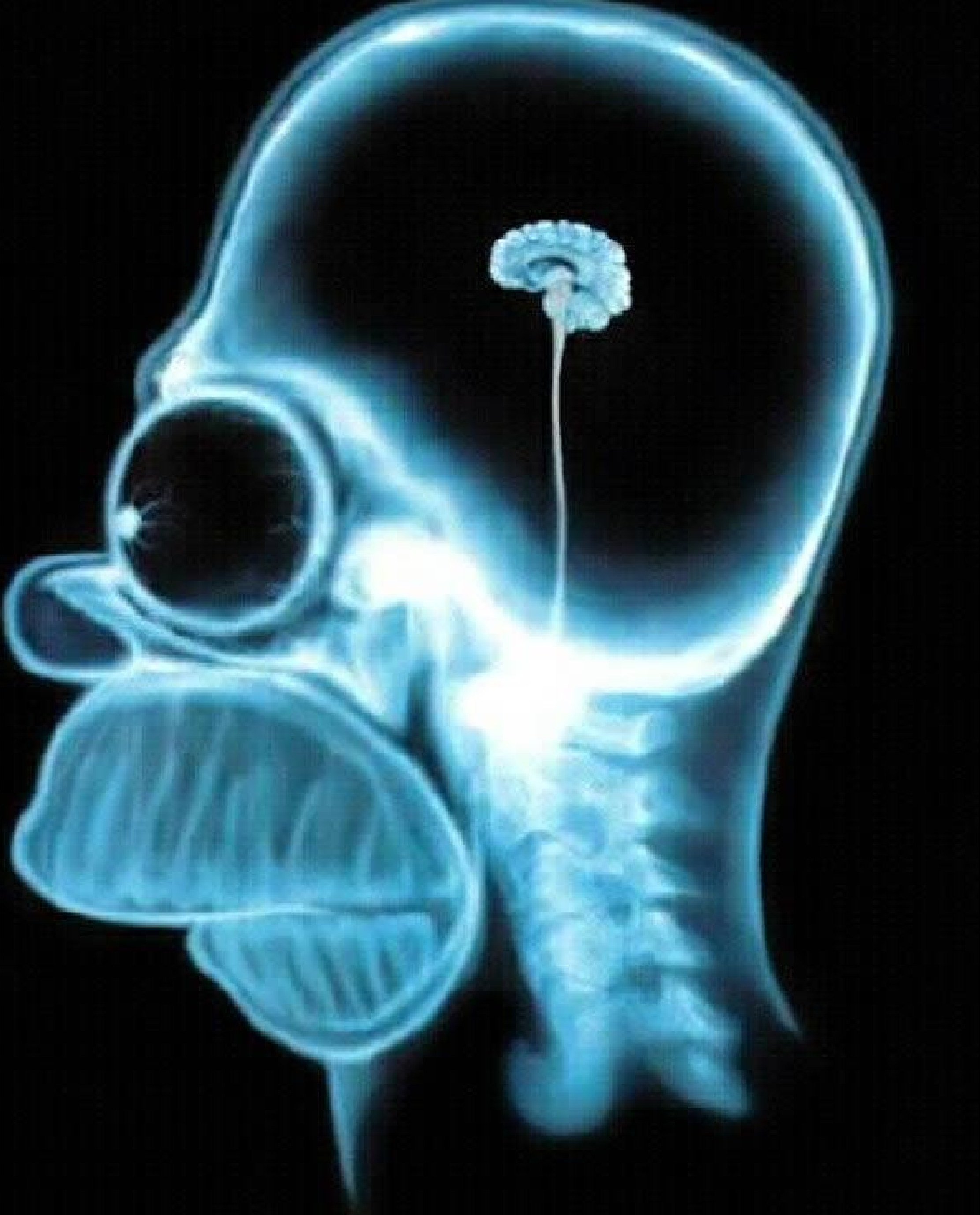
```
>>> mock = Mock(name='Thing', return_value=None)
>>> mock(1, 2, 3)
>>> mock.assret_called_once_with(4, 5, 6)
```

Your tests can pass silently and incorrectly because of the typo.

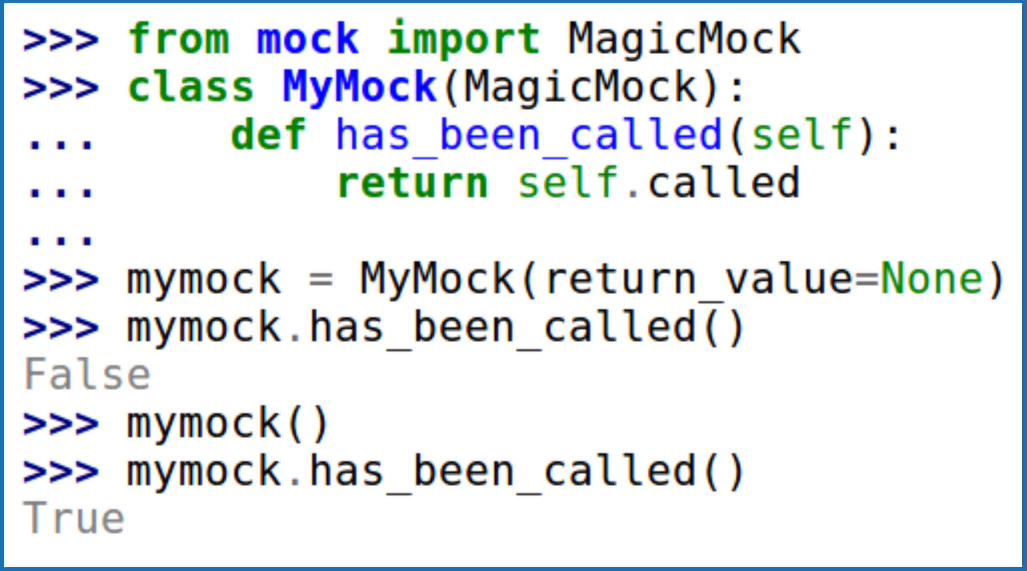
http://www.voidspace.org.uk/python/mock/helpers.html#autospeccing



```
>>> import urllib2
>>> mock = Mock(spec=urllib2.Request)
>>> mock.assret_called_with
Traceback (most recent call last):
  ...
AttributeError: Mock object has no attribute 'assret_called_with'
```



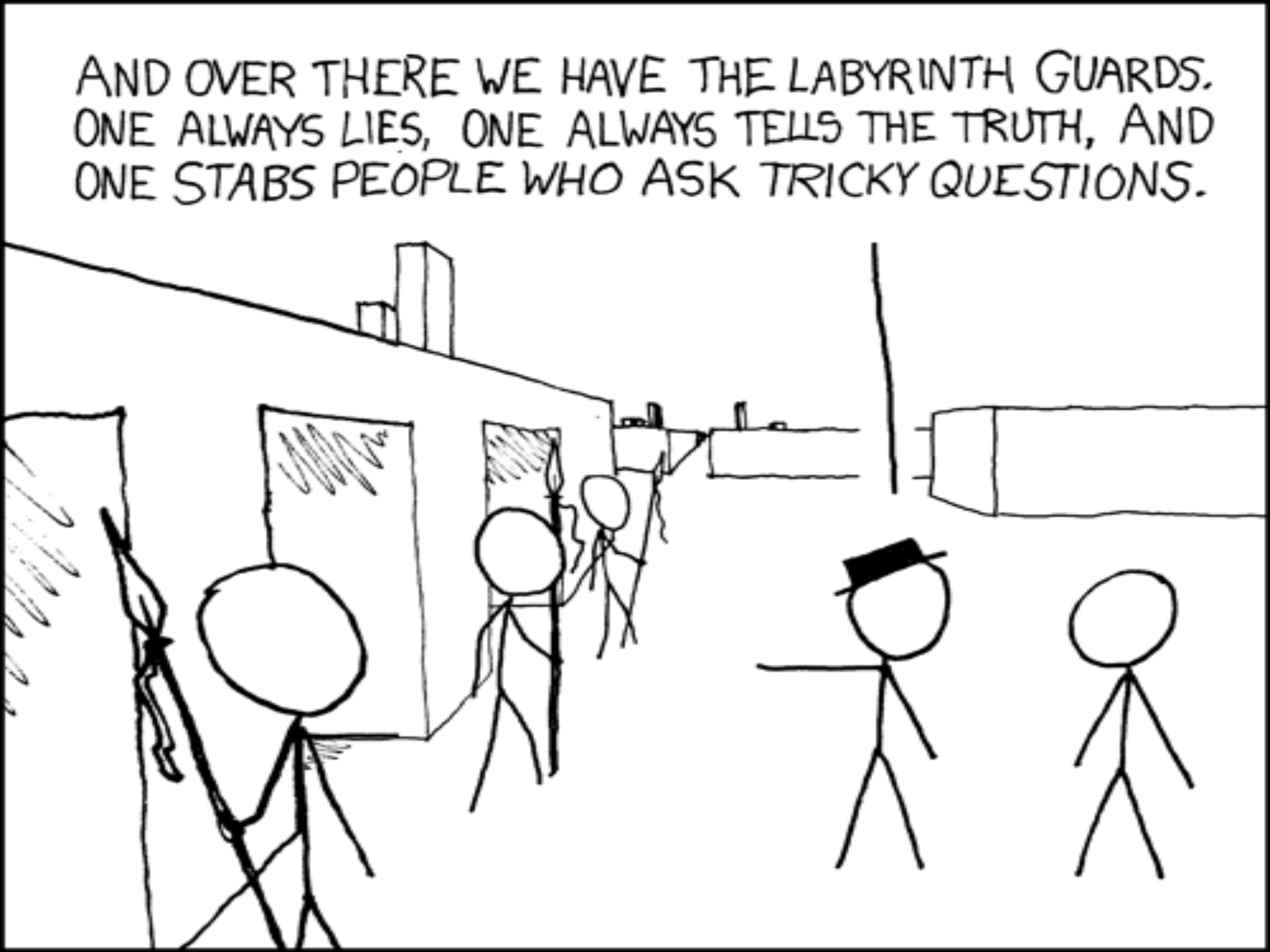
Extend



```
>>> from mock import MagicMock
>>> class MyMock(MagicMock):
...     def has_been_called(self):
...         return self.called
...
>>> mymock = MyMock(return_value=None)
>>> mymock.has_been_called()
False
>>> mymock()
>>> mymock.has_been_called()
True
```

http://www.voidspace.org.uk/python/weblog/arch_d7_2011_07_16.shtml#e1221

mock


AND OVER THERE WE HAVE THE LABYRINTH GUARDS. ONE ALWAYS LIES, ONE ALWAYS TELLS THE TRUTH, AND ONE STABS PEOPLE WHO ASK TRICKY QUESTIONS.

Excellent.

tl;dr