Simple Examples

The Python mock class can mimic and other Python class. This allows you to examine what methods were called on your mocked class and even what parameters were passed to them. Let's start by looking at a couple of simple examples that demonstrate how to use the mock module:

```python
from unittest.mock import Mock
my_mock = Mock()
my_mock.__str__ = Mock(return_value='Mocking')
print(str(my_mock))
#'Mocking'
```

In this example, we import **Mock** class from the **unittest.mock** module. Then we create an instance of the Mock class. Finally we set our mock object's **__str__** method, which is the magic method that controls what happens if you call Python's **str** function on an object. In this case, we just return the string "Mocking", which is what you see when we actually execute the str() function at the end.

The mock module also supports five asserts. Let's take a look at how at a couple of those in action:

```python
from unittest.mock import Mock
class TestClass():
    pass

cls = TestClass()
cls.method = Mock(return_value='mocking is fun')
print (cls.method(1, 2, 3))
#'mocking is fun'

cls.method.assert_called_once_with(1, 2, 3)
print (cls.method(1, 2, 3))
#'mocking is fun'
```

```
from unittest.mock import Mock
class TestClass():
    pass

cls = TestClass()
cls.method = Mock(return_value='mocking is fun')
cls.method(1, 2, 3)
#'mocking is fun'

cls.method.assert_called_once_with(1, 2, 3)
cls.method(1, 2, 3)
#'mocking is fun'

cls.method.assert_called_once_with(1, 2, 3)
#Traceback (most recent call last):
# File "/usercode/__ed_file.py", line 14, in <module>
# cls.method.assert_called_once_with(1, 2, 3)
# File "/usr/lib/python3.5/unittest/mock.py", line 804, in assert_called_once_with
# raise AssertionError(msg)
# AssertionError: Expected 'mock' to be called once. Called 2 times.

cls.other_method = Mock(return_value='Something else')
cls.other_method.assert_not_called()
```

First off, we do our import and create an empty class. Then we create an instance of the class and add a method that returns a string using the Mock class. Then we call the method with three integers are arguments. As you will note, this returned the string that we set earlier as the return value. Now we can test out an assert! So we call the **assert_called_once_with** assert which will assert if we call our method two or more times with the same arguments. The first time we call the assert, it passes. So then we call the method again with the same methods and run the assert a second time to see what happens.

As you can see, we got an **AssertionError**. To round out the example, we go ahead and create a second method that we don't call at all and then assert that it wasn't called via the **assert_not_called** assert.