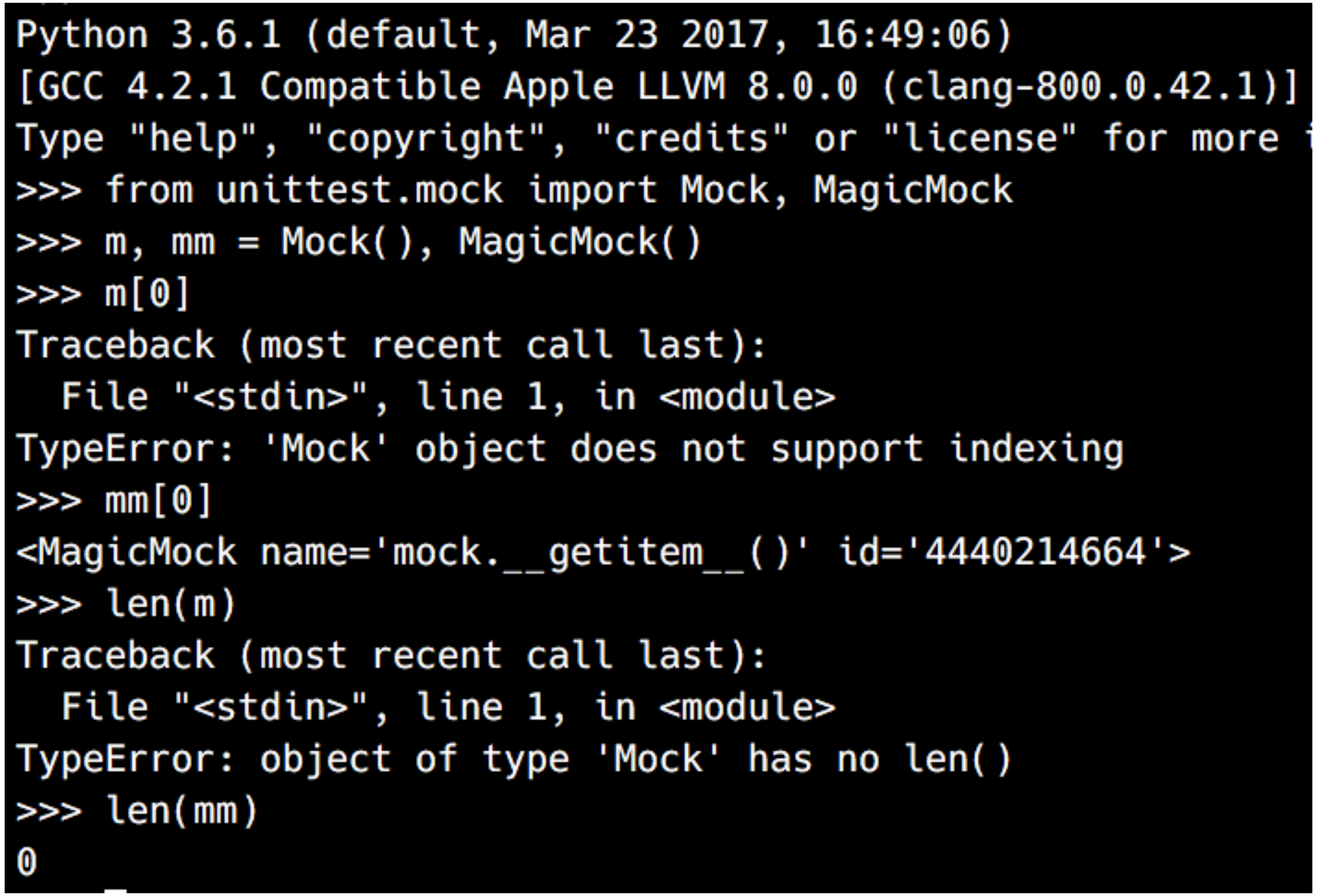
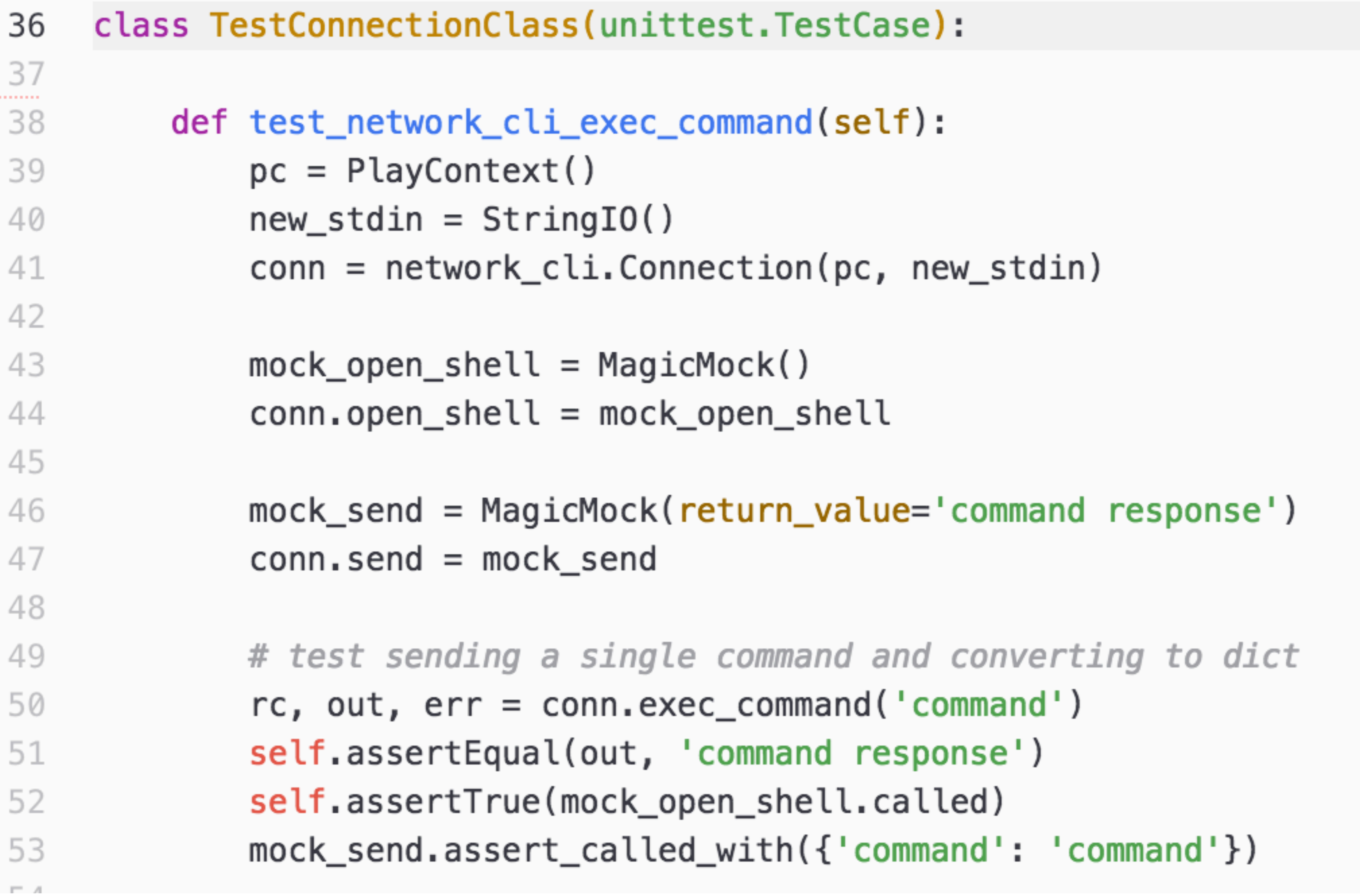
Example 2 - ✨MagicMock✨



```
Python 3.6.1 (default, Mar 23 2017, 16:49:06)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)]
Type "help", "copyright", "credits" or "license" for more i
>>> from unittest.mock import Mock, MagicMock
>>> m, mm = Mock(), MagicMock()
>>> m[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'Mock' object does not support indexing
>>> mm[0]
<MagicMock name='mock.__getitem__()' id='4440214664'>
>>> len(m)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'Mock' has no len()
>>> len(mm)
0
```

- Like Mock, but **acts more like an instance of an Object out of the box**
- For example, implements methods to act like it supports indexing and length calculations
- Helpful, but **easily abused by the lazy… (we'll see)**

Example 2 - ✨MagicMock✨



```python
class TestConnectionClass(unittest.TestCase):

    def test_network_cli_exec_command(self):
        pc = PlayContext()
        new_stdin = StringIO()
        conn = network_cli.Connection(pc, new_stdin)

        mock_open_shell = MagicMock()
        conn.open_shell = mock_open_shell

        mock_send = MagicMock(return_value='command response')
        conn.send = mock_send

        # test sending a single command and converting to dict
        rc, out, err = conn.exec_command('command')
        self.assertEqual(out, 'command response')
        self.assertTrue(mock_open_shell.called)
        mock_send.assert_called_with({'command': 'command'})
```

ANSIBLE

- Honestly, not sure if MagicMock() is that helpful here…
- But the key is you can use it just like Mock()