Object Oriented Programming with Python

Gramsci Hermozo

Session 1

Content

- Unit test

Unit test

unittest

`unittest` is python library that help us to create unit tests for our application. This package cames by default with python instalation.

unittest

How to name unit test in python?

We need to follow some conventions to name the tests.

```
*test.py
*_test.py
test*.py
test_*.py
*test*.py
```

unittest

How to write the unit test?

```
import unittest

class TestObject(unittest.TestCase)
  pass
```

unittest

How to name methods for unit test?

I use to split the function naming in two:

```
# basic tests
#test_<function_name>
def test_add(self):
  pass

# negative tests
# test_<function_name>_<expecter_results>
def test_divide_raise_value_error(self):
  pass
```

Asserts

`assert*` functions are used to validate that the value returned is something that we expect. Below some commong functions.

```
self.assertEqual(<result>, <expected_result>)
self.assertTrue(<result>)
self.assertFalse(<result>)
self.assertRaises(<exception>, <function>, <params>, )
self.assertIn(<value>, <list/dict>)
```

Setup and Teardown

Sometimes we want to prepare a context for each test to be run under. The `setUp` method is run prior to each test in the class and `tearDown` is run at the end of every test. This methods are optional.

```
import unittest

class SomeTest(unittest.TestCase):
  def serUp(self):
    self.mock_data = [1, 2, 3, 4, 5]
  def tearDown(self):
    self.mock_data = []
  def test(self):
    self.assertEqual(len(self.mock_data), 5)
```

mock -> patch

The `patch` decorator/context manager makes it easy to mock classes or objects in a module under test

```
from unittest.mock import patch

def test(mock_class1, mock_class2):
  pass
```

Lets practice

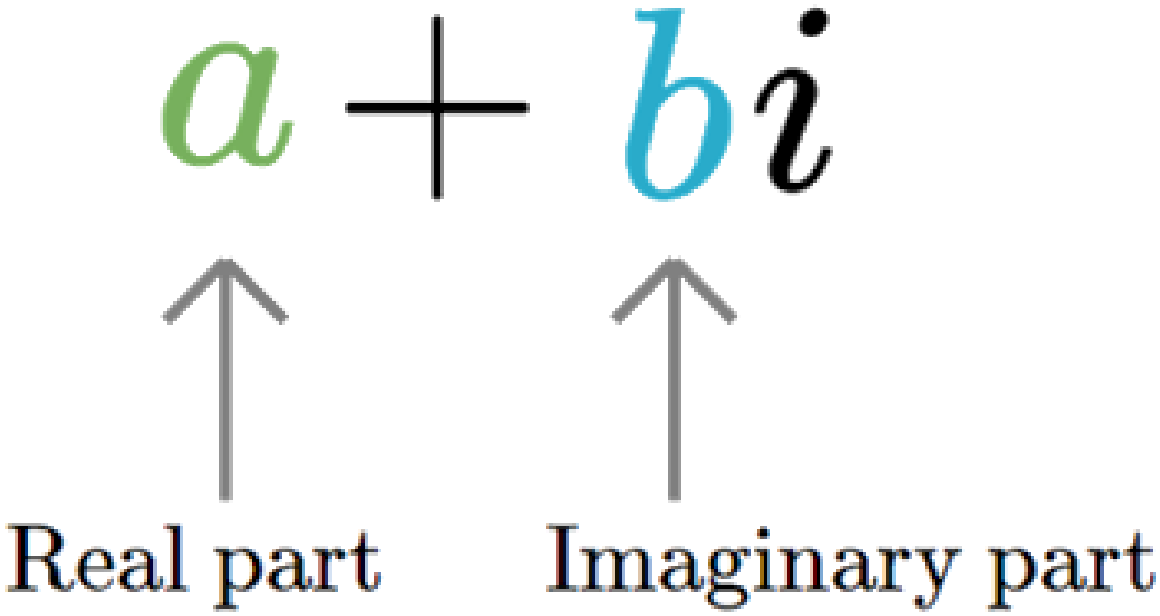
Complex numbers

Implement your object complex numbers and add the following function:

- Adding (numComplex1 + numComplex2)
- Multiplying (numComplex1 * numComplex2)
- Print

What is a complex number

Is a combination of a **Real Number** and an **Imaginary Number**



operation with complex number

Adding:

$$(a + b\mathbf{i}) + (c + d\mathbf{i}) = (a + c) + (b + d)\mathbf{i}$$

Multiplying:

$$(a + b\mathbf{i})\ (c + d\mathbf{i}) = ac + ad\mathbf{i} + bc\mathbf{i} + bd\mathbf{i}2$$

Note: i2 = -1