INFO5002: Intro to Python for Info Sys

Testing

PCC 209-223

Northeastern University

Slides created by: Zachary Doucet

Unit tests

- Test the smallest functional unit.
- Tests should be independent.
- Sum of tests should cover as many lines of code (have high coverage).

Python unittest

- assertEqual(a, b): see if a == b
- assertTrue(x): see if bool(x) == True
- assertIsInstance(a, b): see if a is instance of b
- assertIsNone(x): see if x == None
- assertFalse(x): see if bool(x) == False
- assertIs(a, b): see if a is b
- assertIn(a, b): see if a in b

Concepts

- Test case: individual unit of testing.
- Test suite: group of test cases and test suites.
- Test fixture: any code that runs before or after tests to prepare or cleanup.
- Test runner: orchestrates the execution of tests and returns result to user.

```python
import unittest

def add(a, b):
    return a + b

class TestAddFunction(unittest.TestCase):
    def test_add_zeroes(self):
        self.assertEqual(add(0, 0), 0)

    def test_add_negative_numbers(self):
        self.assertEqual(add(-5, -6), -11)

    def test_add_mixed_numbers(self):
        self.assertEqual(add(5, -6), -1)
        self.assertEqual(add(-9, 3), -6)

if __name__ == '__main__':
    unittest.main()
```

A test case named TestAddFunction with 3 unit tests.

Individual unit test must start with the letters test_ and have at least 1 assert.

Will run all test cases that inherit unittest.TestCase

```python
import unittest

class CompoundInterest:
    def __init__(self, start, rate):
        self.curr = start
        self.rate = rate

    def compound(self):
        self.curr += self.curr * self.rate
```

Code runs before each test

Code runs after each test

```python
class TestCompoundInterest(unittest.TestCase):
    def setUp(self):
        self.ci = CompoundInterest(100, 0.1)

    def tearDown(self):
        pass

    def test_first_compound(self):
        self.ci.compound()
        self.assertEqual(self.ci.curr, 110)

    def test_two_compound(self):
        self.ci.compound()
        self.ci.compound()
        self.assertEqual(self.ci.curr, 121)

if __name__ == "__main__":
    unittest.main()
```

Testing for exceptions

```
import unittest

def div(x, y):
    return x / y

class TestDiv(unittest.TestCase):
    def test_div_0_exception(self):
        with self.assertRaises(ZeroDivisionError):
            div(10, 0)

if __name__ == "__main__":
    unittest.main()
```

Create a block of **self.assertRaises(ExceptionType)** to test if the block throws the exception

Let’s practice

In canvas you will find a file with some code written. I want you to create a new file **test.py** which will have unit tests that together have 100% coverage of the file’s functionalities.

Why bother?

- If we test out code as we go why create unit tests?
- Because if we change it later we may break it.
  - Unit tests allow us to detect if new code breaks existing behaviour.