

Testing our code

- Central to the task of computer programmers is writing code free of errors.
- Because of that, testing becomes an essential part of writing code.
- One way of testing code is by checking that each method we write does what we intend it to do.
 - We test the methods one at a time
 - This is called unit testing.

```
if __name__ == "__main__":  
    employee1 = Employee("Jaime", "Amherst", "111-111-1111", "CICS", 15)  
    sample = Employees()  
    sample.add(employee1)
```

If what we see being printed contains the same data we put into the Employees object, we have a sense that things are working correctly.

```
if __name__ == "__main__":  
    employee1 = Employee("Jaime", "Amherst", "111-111-1111", "CICS", 15)  
    sample = Employees()  
    sample.add(employee1)  
    sample.add(Employee("Maria", "Stow", "222-222-2222", "somewhere", 50))  
    sample.add(Employee("Jaime", "Amherst", "111-111-1111", "CICS", 15))
```

The more we test, the more confident we can be.

Testing with unittest

- Central to the task of computer programmers is writing code free of errors.
- Because of that, testing becomes an essential part of writing code.
- Many languages offer tools to facilitate testing.
- We will now see one such tool for Python.

unittest

- A module with a number of methods useful during testing
- The name comes from the type of testing it facilitates:
 - It helps us test units of code, each of the separately.
- The part of unittest that we will use the most often is the TestCase class
- We will build our own test cases by creating classes that inherit from `unittest.TestCase`
- Let's see an example
 - This example is based on the actual autograder we used for assignment #1 this semester

unittest

This is the Python module that will provide testing tools we want to use

```
import unittest
```

```
class TestFirstAssignment(unittest.TestCase):  
    def test_isPrimeGivesCorrectAnswerForNEqualTwo(self):  
        self.assertEqual(True, is_prime(2))
```

We're creating a class that inherits from TestCase

The name of this method needs to start with the letters "test"

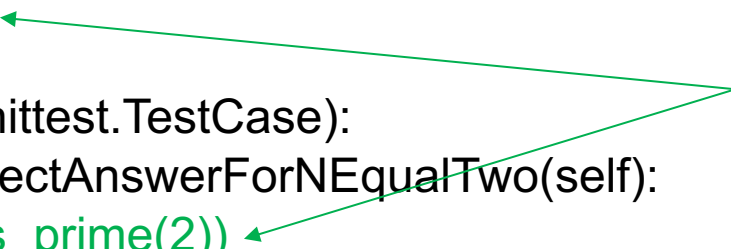
unittest

```
import unittest
from primes import is_prime

class TestFirstAssignment(unittest.TestCase):
    def test_isPrimeGivesCorrectAnswerForNEqualTwo(self):
        self.assertEqual(True, is_prime(2))

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

Because we are calling the function we are testing, we have to provide this testing file access to it.



unittest

```
import unittest
from primes import is_prime, are_relatively_prime, primes, prime_decomposition

class TestFirstAssignment(unittest.TestCase):
    def test_isPrimeGivesCorrectAnswerForNEqualTwo(self):
        self.assertEqual(True, is_prime(2))

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

unittest

```
class TestFirstAssignment(unittest.TestCase):  
    def test_isPrimeGivesCorrectAnswerForNEqualTwo(self):  
        self.assertEqual(True, is_prime(2))
```

- This test checks one particular condition: does the function return the correct output with an input of 2?
- We should do further testing, with different inputs.

unittest

```
def test_1(self):  
    self.assertTrue(is_prime(2))  
  
def test_2(self):  
    self.assertTrue(is_prime(23))  
  
def test_3(self):  
    self.assertFalse(is_prime(46))  
  
def test_4(self):  
    self.assertFalse(is_prime(49))
```

Exercise: write tests for function *prime_decomposition* from assignment 1

unittest

A couple of ways of executing these tests from outside VSCode:

1. `python3 -m unittest test_assignment1.py`

```
python3 -m unittest test_assignment1.py
```

```
.
```

```
-----  
Ran 1 test in 0.000s
```

```
OK
```

unittest

A couple of ways of executing these tests from outside VSCode:

2. Add the following main block to your testing file:

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