

# Prime Numbers

Taylor Letsoaka  
Nando Bingani  
Nkavelo Nxumalo  
Lesetsa Mafisa  
Ziphozonke Mbatha

August 2018

# Contents

<b>1</b>	<b>Purpose</b>	<b>3</b>
<b>2</b>	<b>Description of the code</b>	<b>3</b>
<b>3</b>	<b>Modifications in the way parameters are passed</b>	<b>3</b>
<b>4</b>	<b>Description of the test program :</b>	<b>3</b>
4.1	test case unit testing . . . . .	3
4.1.1	test_primes() . . . . .	3
4.1.2	test_non_primes() . . . . .	3
4.2	Integration testing . . . . .	3
4.2.1	test_primes_of_X() . . . . .	3
4.2.2	test_empty_sets() . . . . .	4
4.3	Visuals Of The tests . . . . .	4
4.3.1	Example when all tests passed . . . . .	4
4.3.2	Example when all tests fail . . . . .	4

## 1 Purpose

The purpose of this document is to show the analysis of a function that takes in a long positive integer number X and then it outputs the number of prime numbers less than or equal to X. It goes into detail and shows how we modified our input and it describes test program such as unit, integration and system testing.

## 2 Description of the code

These prime numbers are then inserted in order in the long integer array `prime[]`. If no prime numbers are possible error codes are returned in N. The use of the algorithm "The sieve of Eratosthenes" to generate our primes.

## 3 Modifications in the way parameters are passed

for the program to execute a list of primes our input must first be considered to be an integer type. We've made the the program to strictly accept integer values, if some sort of input type is entered then the program asks the user to enter the correct input type which is an integer, since only integers have prime values. if the user entered the correct input type then the program checks whether the input is greater than 1 or not, if not an empty list is returned or else the list with values corresponding to the list of primes.

## 4 Description of the test program :

### 4.1 test case unit testing

The unit test was implemented by having two test cases for the `is_prime()` module. The two test cases are `test_primes()` and `test_non_primes()`. The test cases are implemented using a class approach as compared to `testProg()`.

#### 4.1.1 test\_primes()

This test case reads in a standard text file (`primes.txt`) which contains a list of known prime numbers as a string. The `test_primes()` module reads the file as input then splits and maps the entries into an integer array called `numbers`. When `test_primes` iterates through the `numbers` array it is expected to pass.

#### 4.1.2 test\_non\_primes()

This test case reads in a standard text file (`nonprimes.txt`) which contains a list of known non prime numbers as a string. The `test_non_primes()` module reads the file as input then splits and maps the entries into an integer array called `numbers`. When `test_non_primes` iterates through the `numbers` array it is expected to pass.

### 4.2 Integration testing

Integration testing is meant to test the `prime_numbers()` module which invokes `is_prime()`.

#### 4.2.1 test\_primes\_of\_X()

This module reads in a text file named `primes_of_X.txt` which contains X and the first n primes before or including X. X is passed to the `prime_numbers()` module, where the `prime_numbers()` module returns a list of prime numbers before or including X and the result is compared to

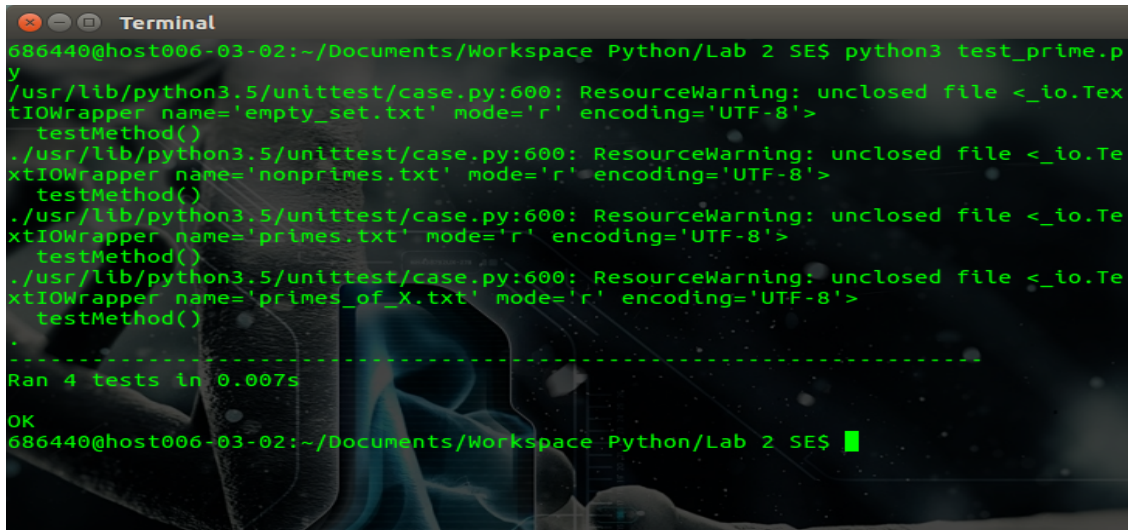
the first n primes before X which is read from the text file primes\_of\_X.txt. This is done for multiple inputs. If the lists are equal then the test passed.

#### 4.2.2 test\_empty\_sets()

This module reads in a text file named empty\_set.txt which contains X. X is passed to the test\_empty\_sets() module, where the test\_empty\_sets() module returns an empty list to indicate that there are no primes before or including X.

### 4.3 Visuals Of The tests

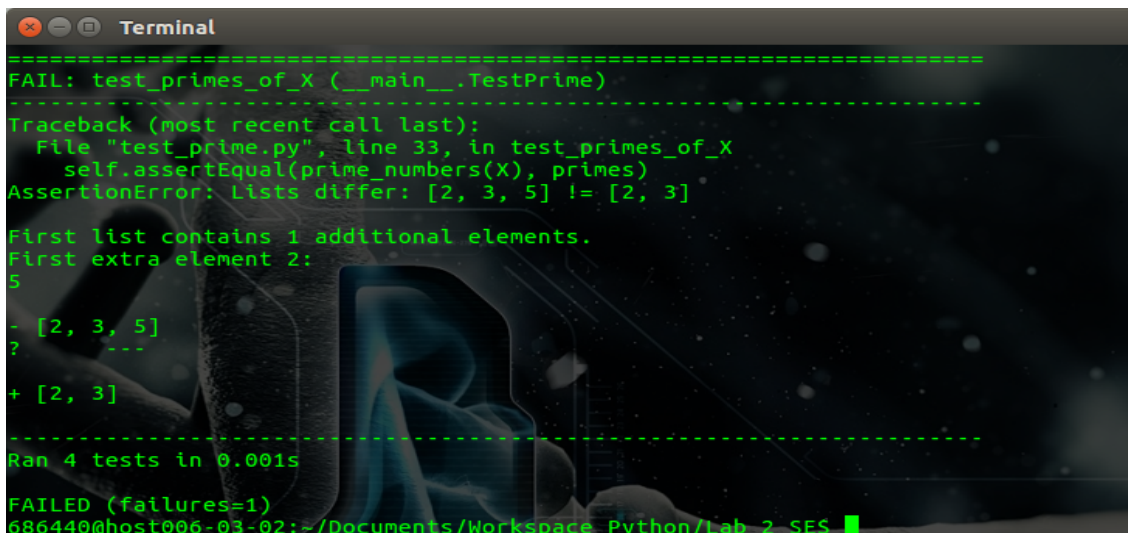
#### 4.3.1 Example when all tests passed



```
Terminal
686440@host006-03-02:~/Documents/Workspace Python/Lab 2 SE$ python3 test_prime.py
/usr/lib/python3.5/unittest/case.py:600: ResourceWarning: unclosed file <_io.TextIOWrapper name='empty_set.txt' mode='r' encoding='UTF-8'>
  testMethod()
./usr/lib/python3.5/unittest/case.py:600: ResourceWarning: unclosed file <_io.TextIOWrapper name='nonprimes.txt' mode='r' encoding='UTF-8'>
  testMethod()
./usr/lib/python3.5/unittest/case.py:600: ResourceWarning: unclosed file <_io.TextIOWrapper name='primes.txt' mode='r' encoding='UTF-8'>
  testMethod()
./usr/lib/python3.5/unittest/case.py:600: ResourceWarning: unclosed file <_io.TextIOWrapper name='primes_of_X.txt' mode='r' encoding='UTF-8'>
  testMethod()
.
-----
Ran 4 tests in 0.007s

OK
686440@host006-03-02:~/Documents/Workspace Python/Lab 2 SE$
```

#### 4.3.2 Example when all tests fail



```
Terminal
=====
FAIL: test_primes_of_X (__main__.TestPrime)
-----
Traceback (most recent call last):
  File "test_prime.py", line 33, in test_primes_of_X
    self.assertEqual(prime_numbers(X), primes)
AssertionError: Lists differ: [2, 3, 5] != [2, 3]

First list contains 1 additional elements.
First extra element 2:
5

- [2, 3, 5]
? ----
+ [2, 3]

-----
Ran 4 tests in 0.001s

FAILED (failures=1)
686440@host006-03-02:~/Documents/Workspace Python/Lab 2 SE$
```

The tests run according to the subsections i.e test first test explained runs first