Cpython Testing

Python is one of the numerous Python runtimes, kept up with and composed by various groups of engineers. Some other runtimes I might have heard are PyPy, Cython, and Jython.

The remarkable thing about CPython is that it contains both a runtime and the common language determination that all Python runtimes use. CPython is the "official," or reference execution of Python.

The Python language determination is the report that the portrayal of the Python language. For instance, it says that state is a held watchword, and that [] is utilized for ordering, cutting, and making void records

Testing

I write some cpython test cases and checked it and it gives me the following results. I used coverage python library for finding code and file coverages.

Write the following test cases

import random

import unittest

class TestSequenceFunctions(unittest.TestCase):

    def setUp(self):

        self.seq = range(10)

    def test\_choice(self):

        element = random.choice(self.seq)

        self.assertTrue(element in self.seq)

    def test\_sample(self):

        with self.assertRaises(ValueError):

            random.sample(self.seq, 20)

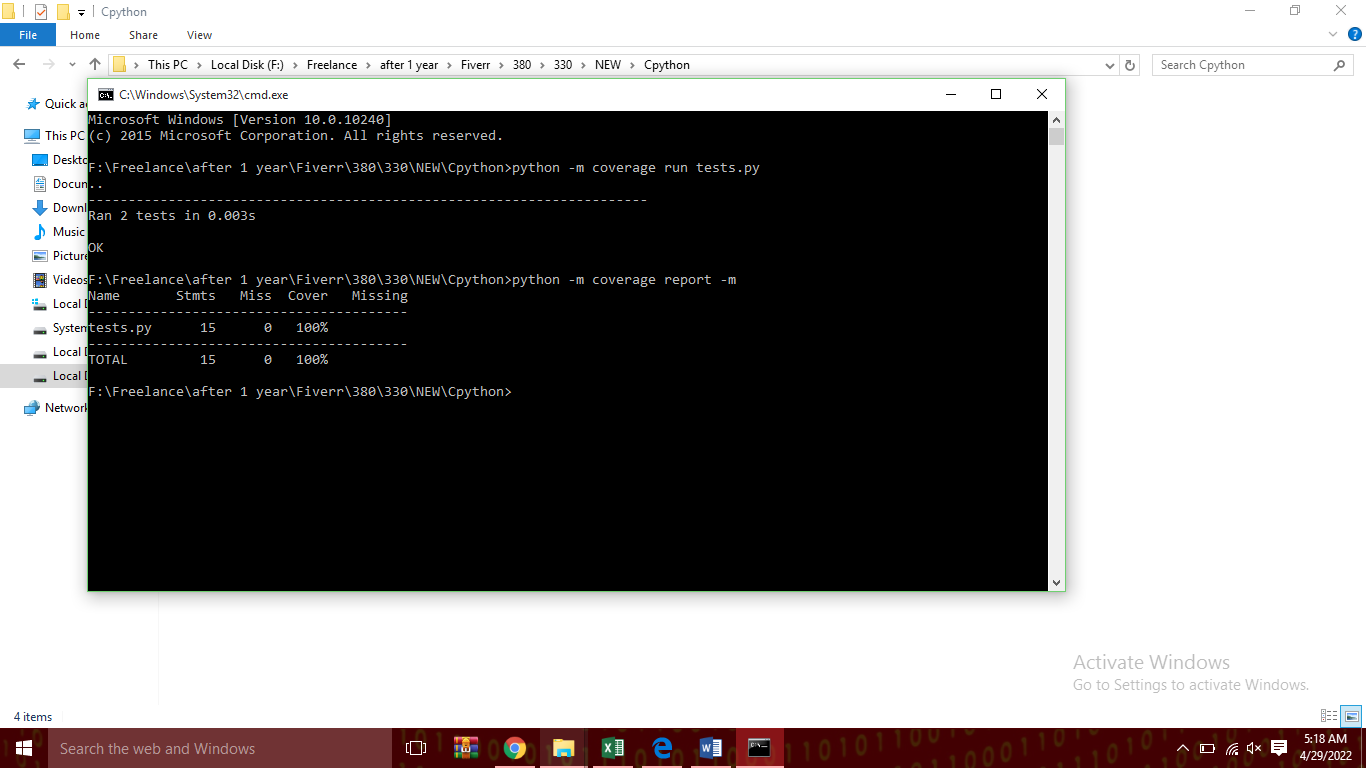
        for element in random.sample(self.seq, 5):

            self.assertTrue(element in self.seq)

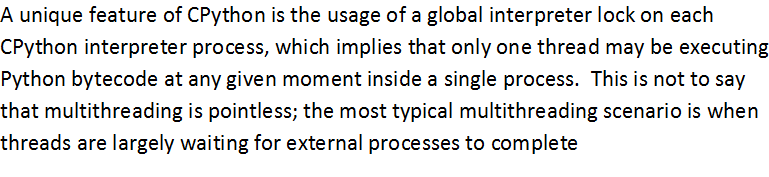
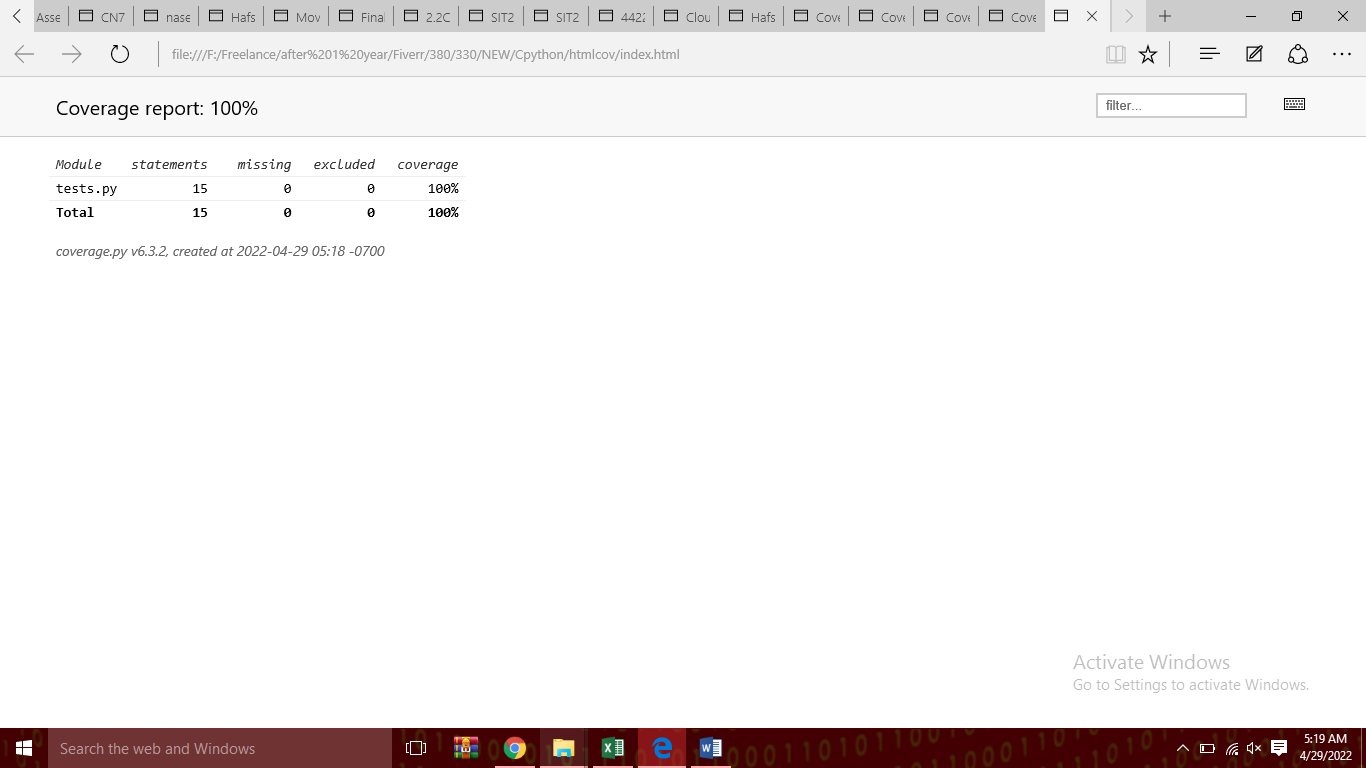
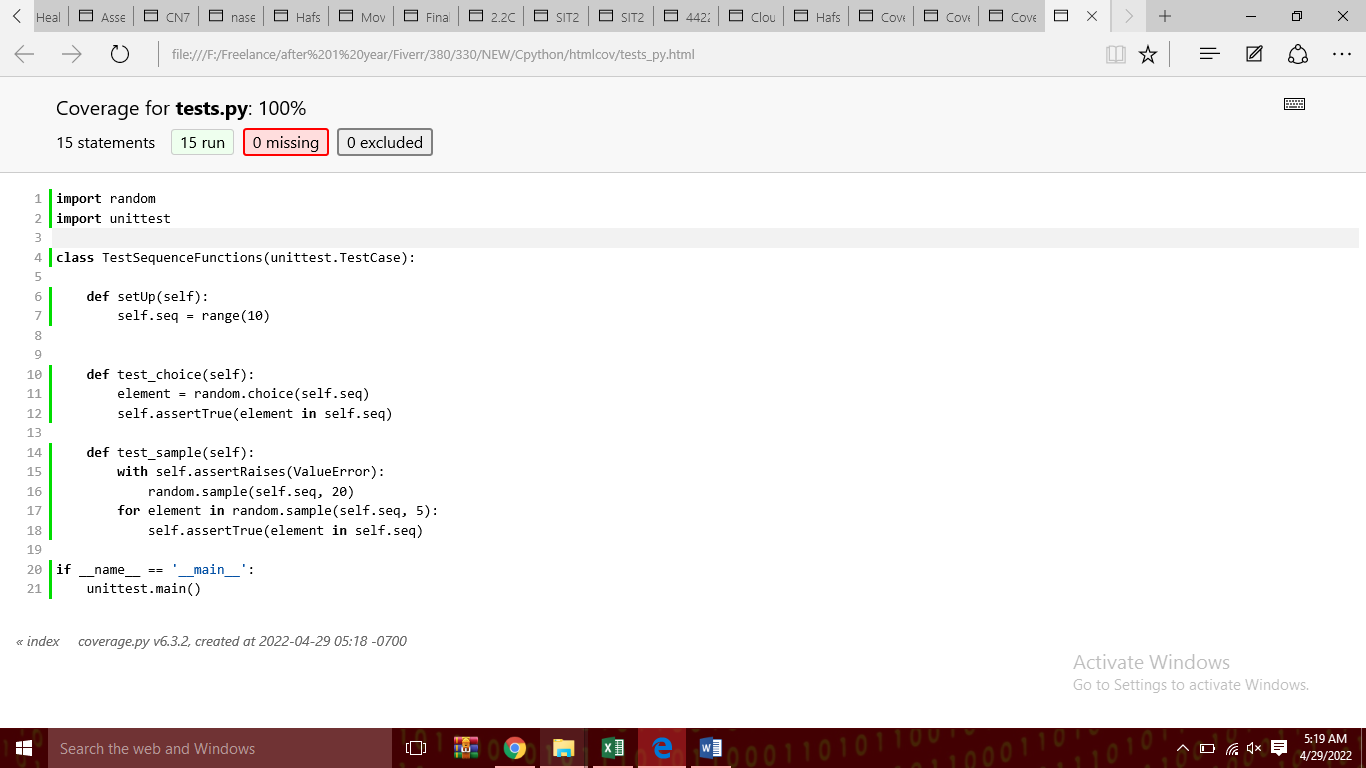
if \_\_name\_\_ == '\_\_main\_\_':

    unittest.main()

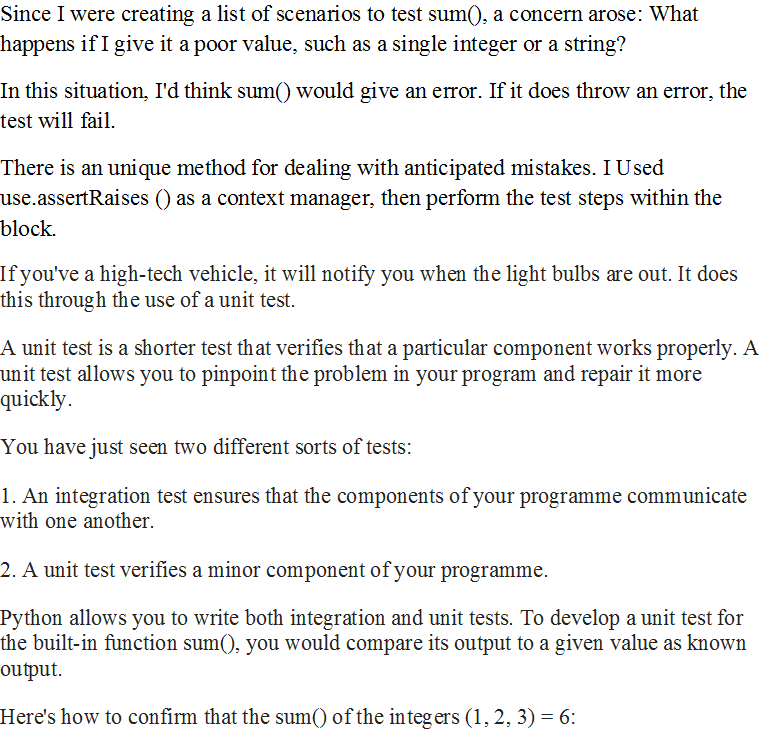
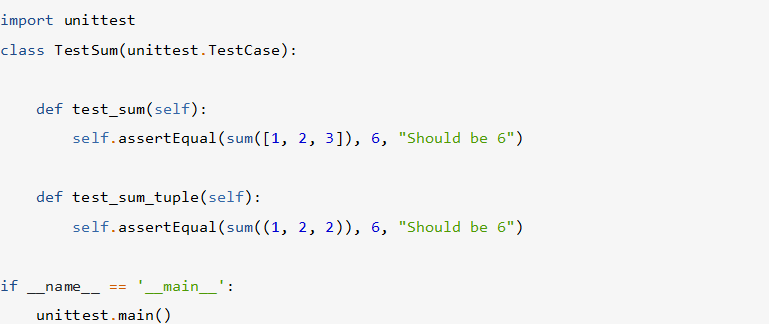
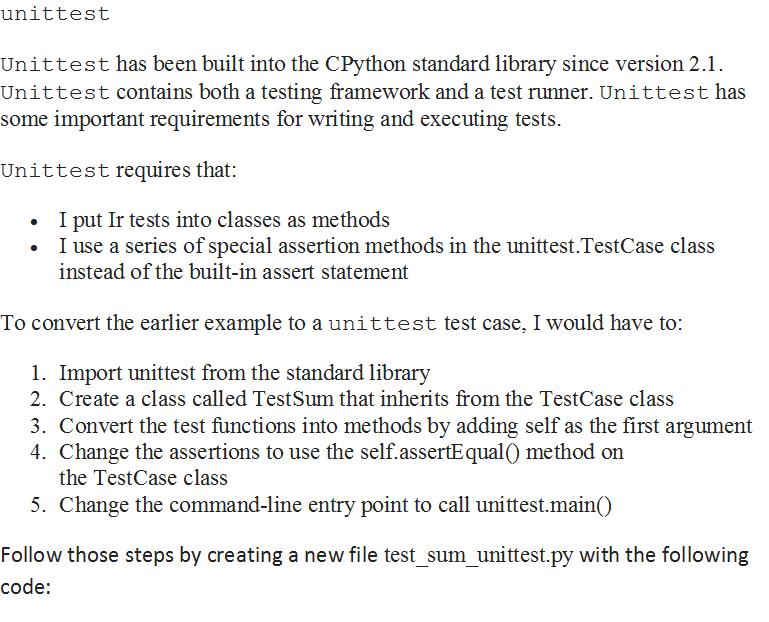
The running it gives me the following.



The result of the coverage are following.



**Testing Method**



Now for simulating all these tests I used a built in python module called coverage.

Following is the way to configure Coverage.py

To install it:

Pip install coverage

Coverage searches for a.coverage file to read and prepare a report for you. Py.test does not generate one on its own. For coverage, you'll require the py.test plugin:

install pytest-cov with pip

I already have it, So I run both at the same time as follows:

test.py —cov=mytests.py test.py

That is, execute test.py and record/display a coverage report on mytests.py.

It also generates the html report too as I generates it, the following way.

For a comprehensive report, you may now execute the coverage command separately:

report on coverage -m

The preceding programme simply shows a structured coverage report based on the gathered data.

data file containing coverage data from prior test runs -m indicates that lines that were missed, i.e. lines that were not covered by tests, should be displayed:

Name Stmts Miss Cover

-----------------------------------------

sample.py 6 0 100%