FORMAN CHRISTIAN COLLEGE (A CHARTERED UNIVERSITY)

CSCS351(A): Software Quality Assurance

Spring 2022

Assignment 1

Unit Testing

Talha Irshad (22-10326)

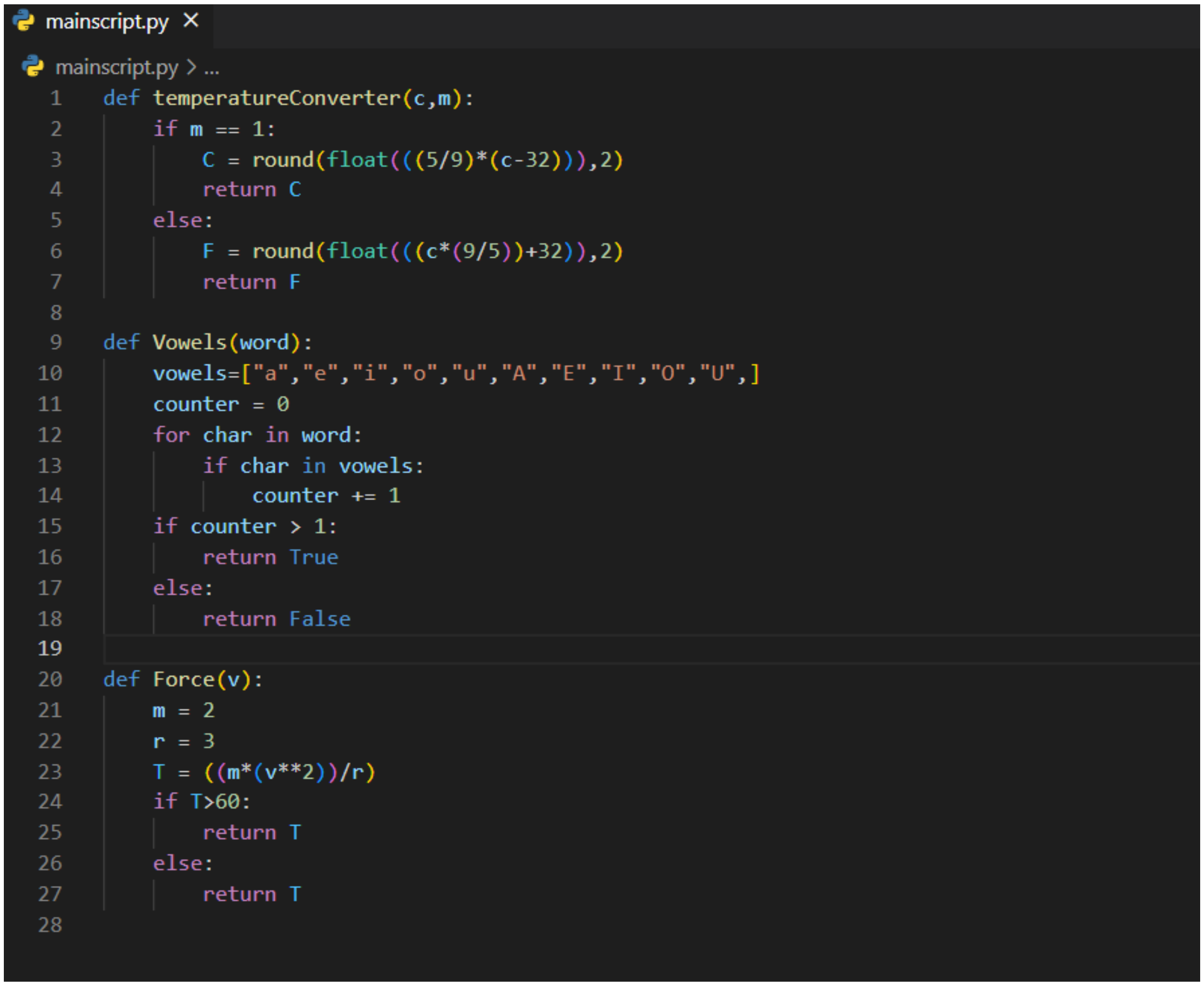
Problem Statement:
You need to write at least five automated test cases using unit testing framework of any language of your choice. You also need to create a testing suite. All the test cases should be executed individually and also under one test suite. You can use any programs / software system of your choice for testing e.g. Junit for Java code, PYunit for Python code, PHPUnit for PHP code, etc.

Introduction

The assignment task is writing test cases using the unit testing framework of a python language program functions. There are two python files: mainscript.py and testcode.py. The mainCode.py is the main program which is tested. It comprises of three functions: temperatureConverter() which is used to convert the temperature from Celsius to Fahrenheit and vice versa, Vowels() which is used to check if a string consists a vowel or not, and Force() which is used to check the force required to break the rope.

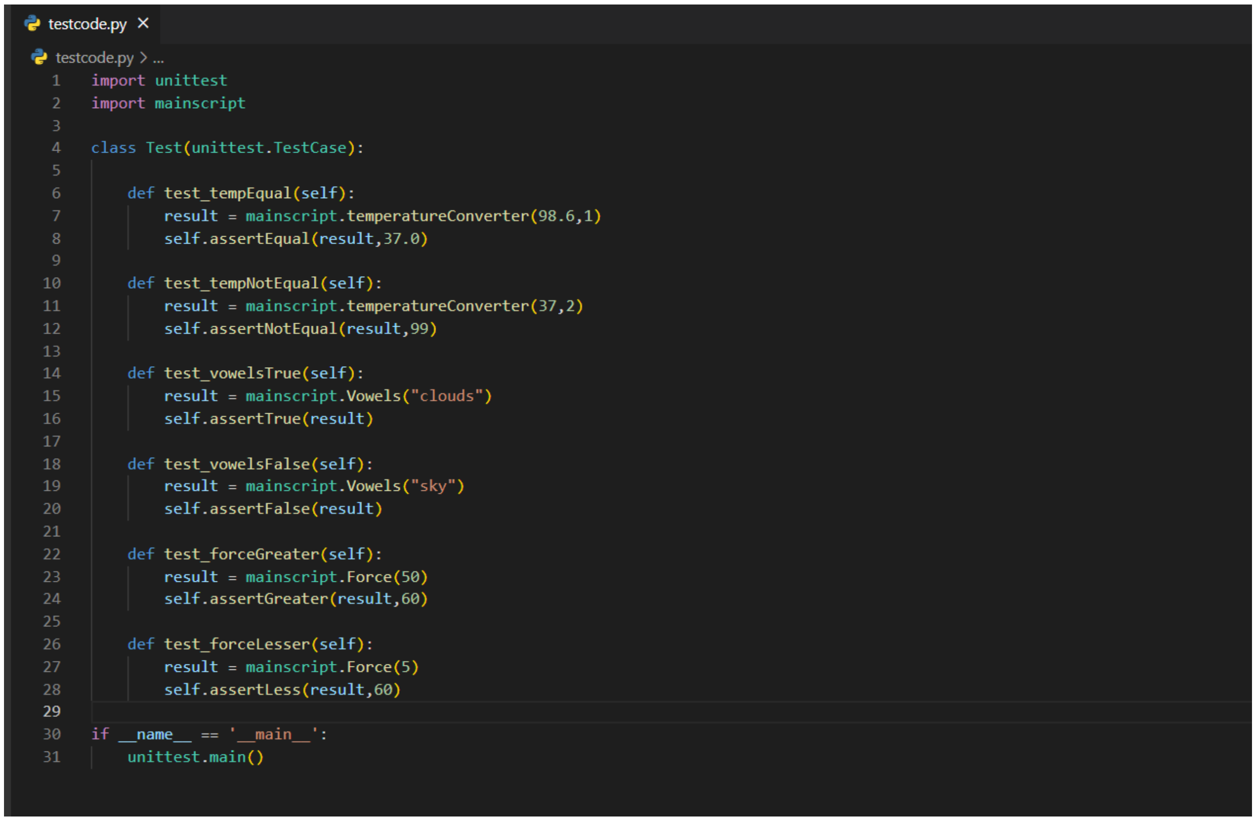
The testcode.py is used to test the mainscript.py python program file and has imported a unittest python framework and mainCode. The unittest framework allows and provides testing tools which comprises of a Test class, a test suite, which contains a total of 6 different types of test cases which includes Equal, NotEqual, True, False, Greater, Less. The Test class provides several assert methods. The test cases are called and used via the ***assert*** methods which check and report for any success or failures. The mainscript allows using the functions and features of the mainCode.py file.

mainscript.py



```python
def temperatureConverter(c,m):
    if m == 1:
        C = round(float(((5/9)*(c-32))),2)
        return C
    else:
        F = round(float(((c*(9/5))+32)),2)
        return F

def Vowels(word):
    vowels=["a","e","i","o","u","A","E","I","O","U",]
    counter = 0
    for char in word:
        if char in vowels:
            counter += 1
    if counter > 1:
        return True
    else:
        return False

def Force(v):
    m = 2
    r = 3
    T = ((m*(v**2))/r)
    if T>60:
        return T
    else:
        return T

```

testcode.py

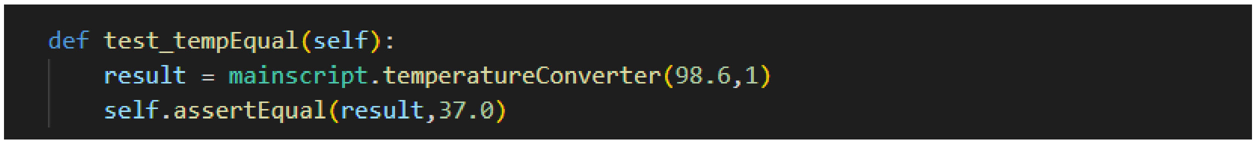

testcode.py
testcode.py > ...

```python
import unittest
import mainscript

class Test(unittest.TestCase):

    def test_tempEqual(self):
        result = mainscript.temperatureConverter(98.6,1)
        self.assertEqual(result,37.0)

    def test_tempNotEqual(self):
        result = mainscript.temperatureConverter(37,2)
        self.assertNotEqual(result,99)

    def test_vowelsTrue(self):
        result = mainscript.Vowels("clouds")
        self.assertTrue(result)

    def test_vowelsFalse(self):
        result = mainscript.Vowels("sky")
        self.assertFalse(result)

    def test_forceGreater(self):
        result = mainscript.Force(50)
        self.assertGreater(result,60)

    def test_forceLesser(self):
        result = mainscript.Force(5)
        self.assertLess(result,60)

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

Test Cases

Test Case 1: temperatureConvertor(a,b)

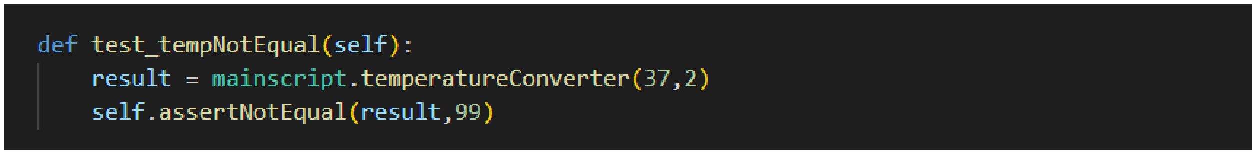
assertEqual is used to test that the arguments are equal. The test fails if the arguments are not equal. Two arguments have been passed to this method, first argument is the actual output of the function being tested and the second argument is the correct expected outcome. It is then tested and the user is notified that the test is successful or did it fail.



```python
def test_tempEqual(self):
    result = mainscript.temperatureConverter(98.6,1)
    self.assertEqual(result,37.0)
```

Test Case 2: temperatureConvertor(a,b)

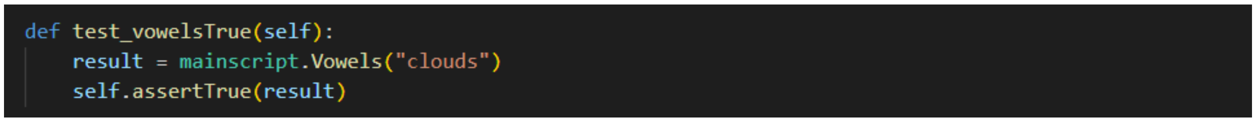
assertNotEqual is used to test that the arguments are not equal. The test fails if the arguments are equal. Two arguments have been passed to this method, first argument is the actual output of the function being tested and the second argument is the incorrect expected outcome. It is then tested and the user is notified that the test is successful or did it fail.



```python
def test_tempNotEqual(self):
    result = mainscript.temperatureConverter(37,2)
    self.assertNotEqual(result,99)
```

Test Case 3: Vowels(c,d)

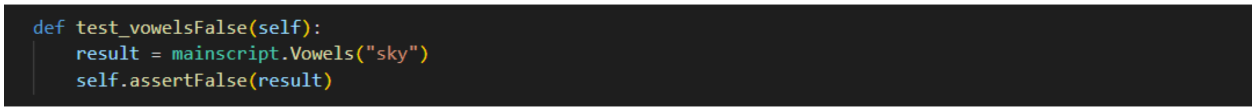
assertTrue is used to test that the argument passed is True or not. The test fails if the argument is not true. Only one argument has been passed to this method which is the actual output of the function being. It is then tested and the user is notified that the test is successful or did it fail.



```python
def test_vowelsTrue(self):
    result = mainscript.Vowels("clouds")
    self.assertTrue(result)
```

Test Case 4: Vowels(c,d)

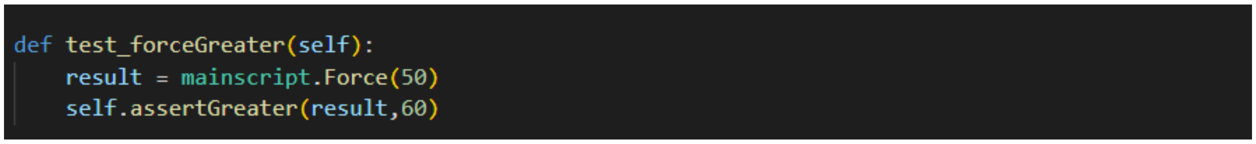
assertFalse is used to test that the argument passed is False or not. The test fails if the argument is true. Only one argument has been passed to this method which is the actual output of the function being. It is then tested, and the user is notified that the test is successful or did it fail.



```python
def test_vowelsFalse(self):
    result = mainscript.Vowels("sky")
    self.assertFalse(result)
```

Test Case 5: Force(e,f)

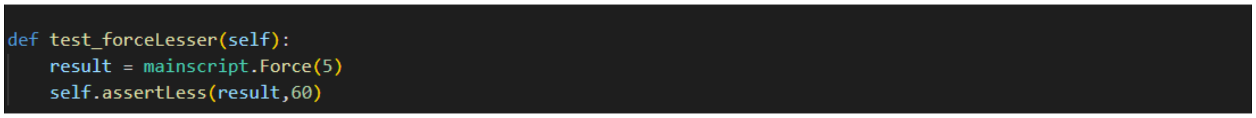
assertGreater is used to test that one argument is greater than the other argument. The test fails if the expected argument is not greater than the other argument. Two arguments have been passed to this method, first argument is the actual output of the function being tested and the second argument is the force limit. It is then tested and the user is notified that the test is successful or did it fail.



```
def test_forceGreater(self):
    result = mainscript.Force(50)
    self.assertGreater(result,60)
```

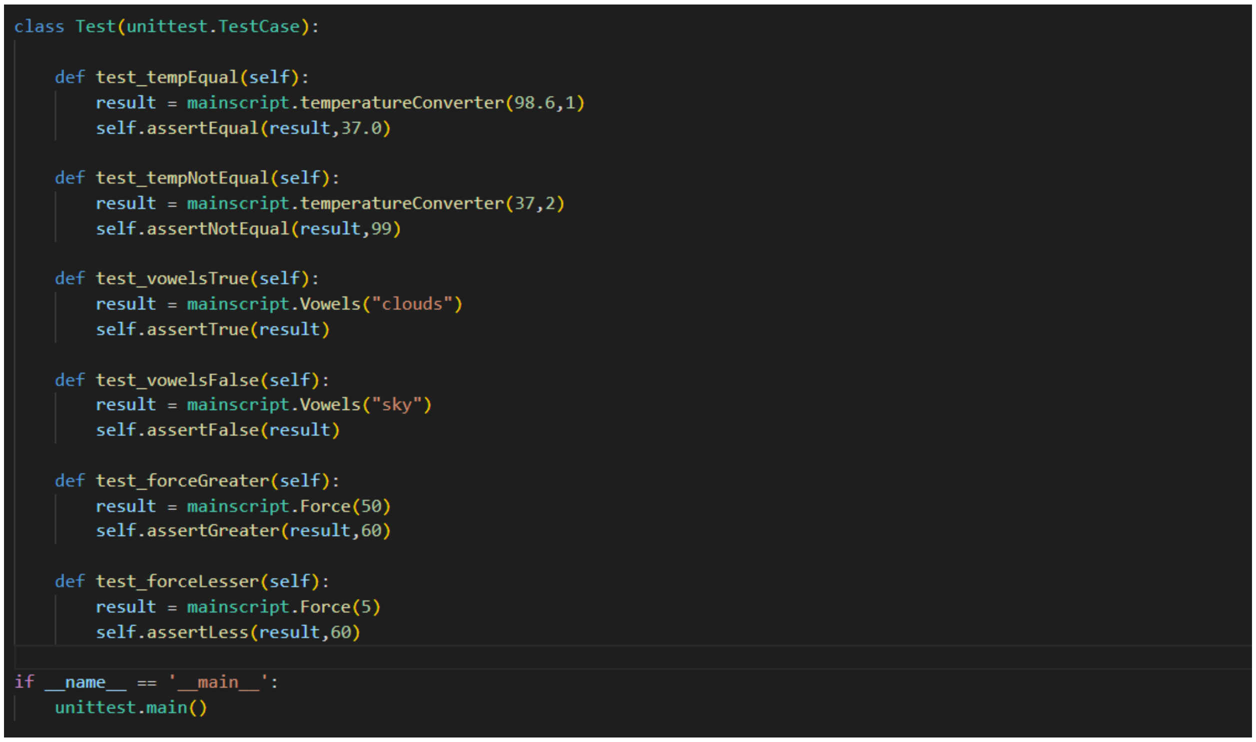
Test Case 6: Force(e,f)

assertLesser is used to test that one argument is smaller than the other argument. The test fails if the expected argument is not smaller than the other argument. Two arguments have been passed to this method, first argument is the actual output of the function being tested and the second argument is the force limit. It is then tested and the user is notified that the test is successful or did it fail.



```
def test_forceLesser(self):
    result = mainscript.Force(5)
    self.assertLess(result,60)
```

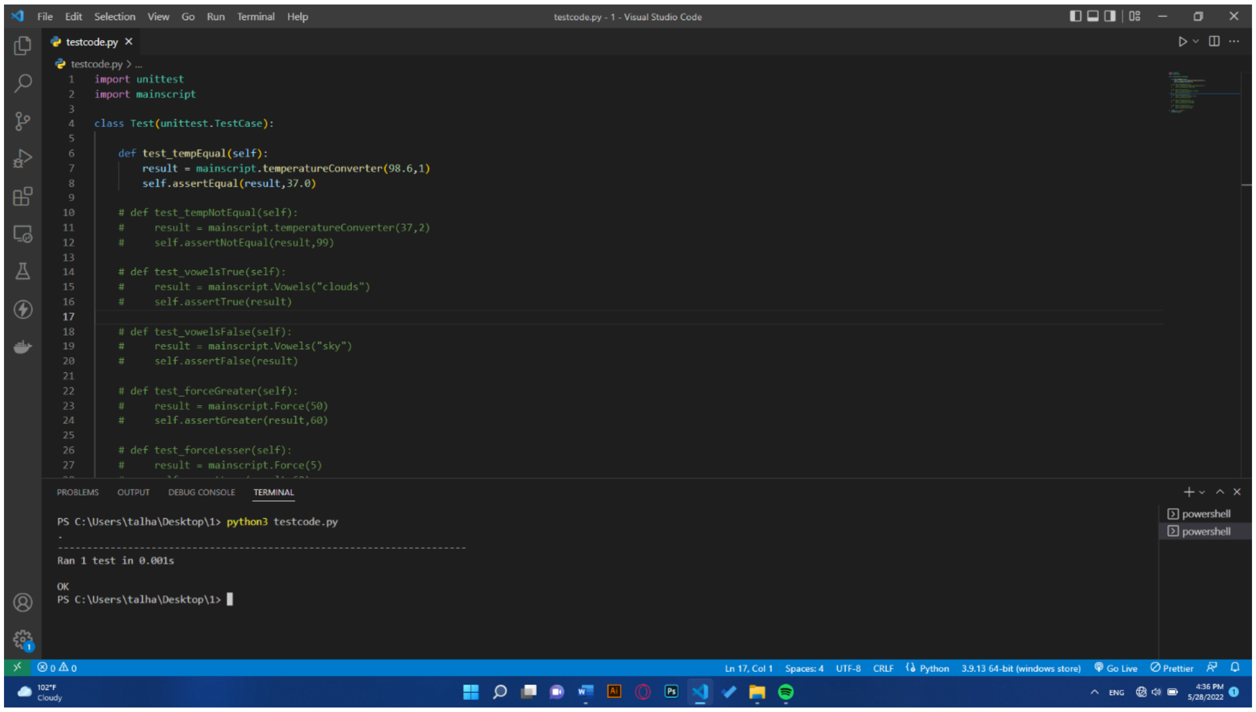
Test Suite



```python
class Test(unittest.TestCase):

    def test_tempEqual(self):
        result = mainscript.temperatureConverter(98.6,1)
        self.assertEqual(result,37.0)

    def test_tempNotEqual(self):
        result = mainscript.temperatureConverter(37,2)
        self.assertNotEqual(result,99)

    def test_vowelsTrue(self):
        result = mainscript.Vowels("clouds")
        self.assertTrue(result)

    def test_vowelsFalse(self):
        result = mainscript.Vowels("sky")
        self.assertFalse(result)

    def test_forceGreater(self):
        result = mainscript.Force(50)
        self.assertGreater(result,60)

    def test_forceLesser(self):
        result = mainscript.Force(5)
        self.assertLess(result,60)

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

Individual Test Cases Results

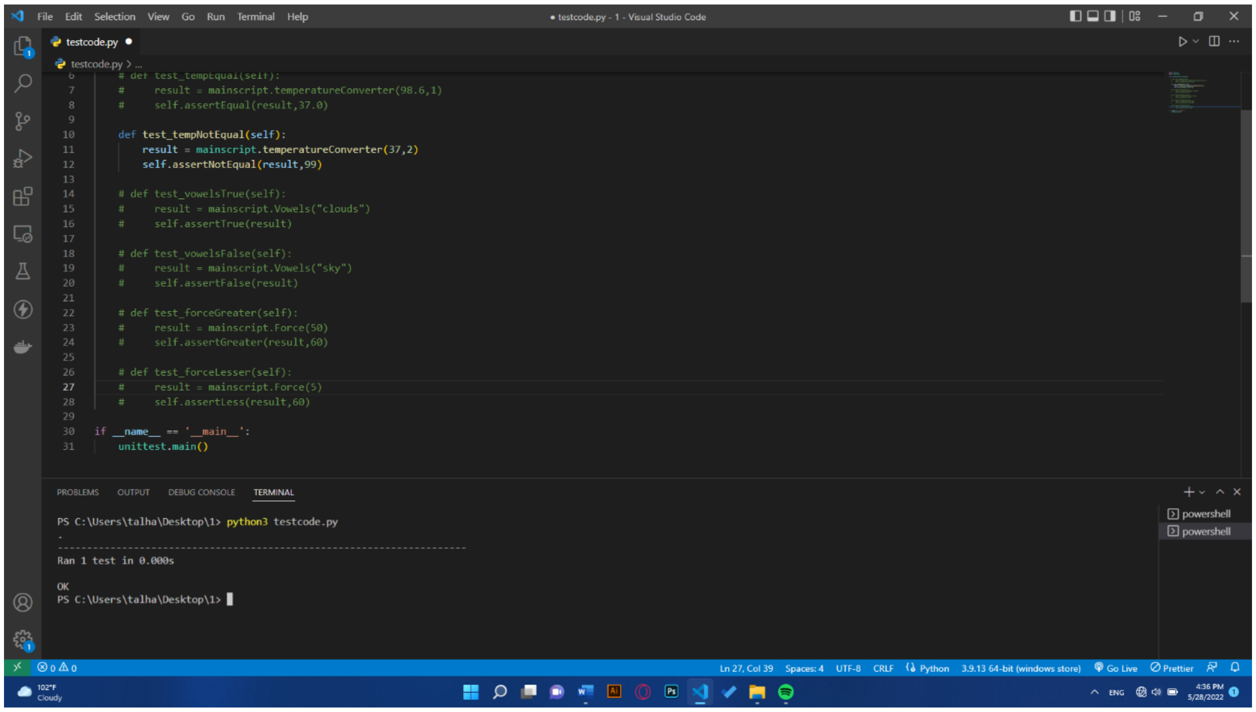
Test Case 1: assertEqual



```python
import unittest
import mainscript

class Test(unittest.TestCase):

    def test_tempEqual(self):
        result = mainscript.temperatureConverter(98.6,1)
        self.assertEqual(result,37.0)

    # def test_tempNotEqual(self):
    #     result = mainscript.temperatureConverter(37,2)
    #     self.assertNotEqual(result,99)

    # def test_vowelsTrue(self):
    #     result = mainscript.Vowels("clouds")
    #     self.assertTrue(result)

    # def test_vowelsFalse(self):
    #     result = mainscript.Vowels("sky")
    #     self.assertFalse(result)

    # def test_forceGreater(self):
    #     result = mainscript.Force(50)
    #     self.assertGreater(result,60)

    # def test_forceLesser(self):
    #     result = mainscript.Force(5)
```

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
PS C:\Users\talha\Desktop\1> 
```

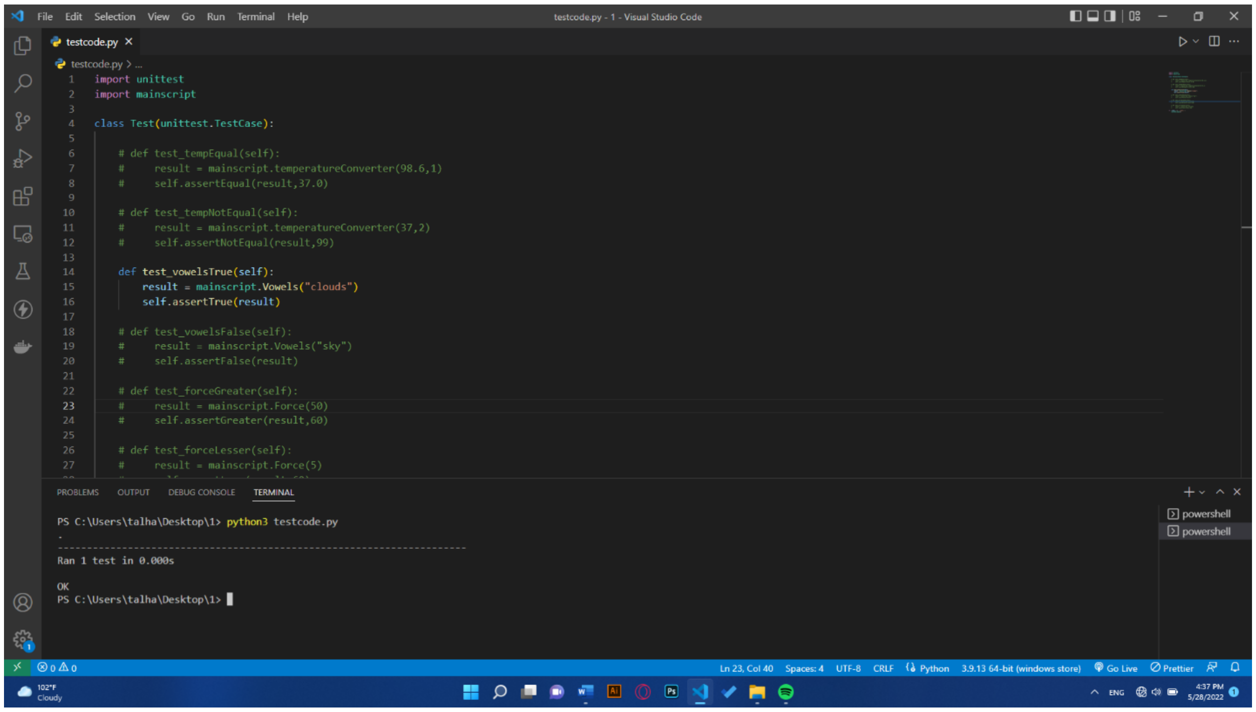
Test Case 2: assertNotEqual



```python
    # def test_tempEqual(self):
    #     result = mainscript.temperatureConverter(98.6,1)
    #     self.assertEqual(result,37.0)

    def test_tempNotEqual(self):
        result = mainscript.temperatureConverter(37,2)
        self.assertNotEqual(result,99)

    # def test_vowelsTrue(self):
    #     result = mainscript.Vowels("clouds")
    #     self.assertTrue(result)

    # def test_vowelsFalse(self):
    #     result = mainscript.Vowels("sky")
    #     self.assertFalse(result)

    # def test_forceGreater(self):
    #     result = mainscript.Force(50)
    #     self.assertGreater(result,60)

    # def test_forceLesser(self):
    #     result = mainscript.Force(5)
    #     self.assertLess(result,60)

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

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\talha\Desktop\1> 
```

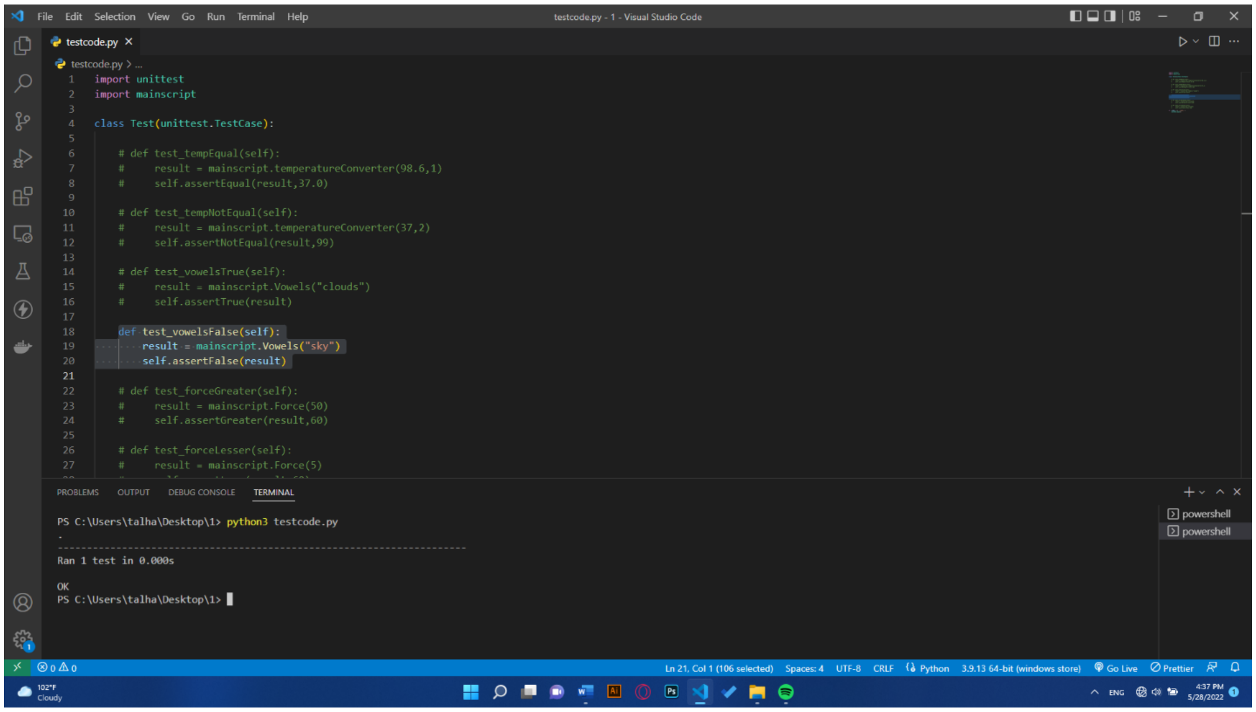
Test Case 3: assertTrue



```python
import unittest
import mainscript

class Test(unittest.TestCase):

    # def test_tempEqual(self):
    #     result = mainscript.temperatureConverter(98.6,1)
    #     self.assertEqual(result,37.0)

    # def test_tempNotEqual(self):
    #     result = mainscript.temperatureConverter(37,2)
    #     self.assertNotEqual(result,99)

    def test_vowelsTrue(self):
        result = mainscript.Vowels("clouds")
        self.assertTrue(result)

    # def test_vowelsFalse(self):
    #     result = mainscript.Vowels("sky")
    #     self.assertFalse(result)

    # def test_forceGreater(self):
    #     result = mainscript.Force(50)
    #     self.assertGreater(result,60)

    # def test_forceLesser(self):
    #     result = mainscript.Force(5)
```

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\talha\Desktop\1>
```

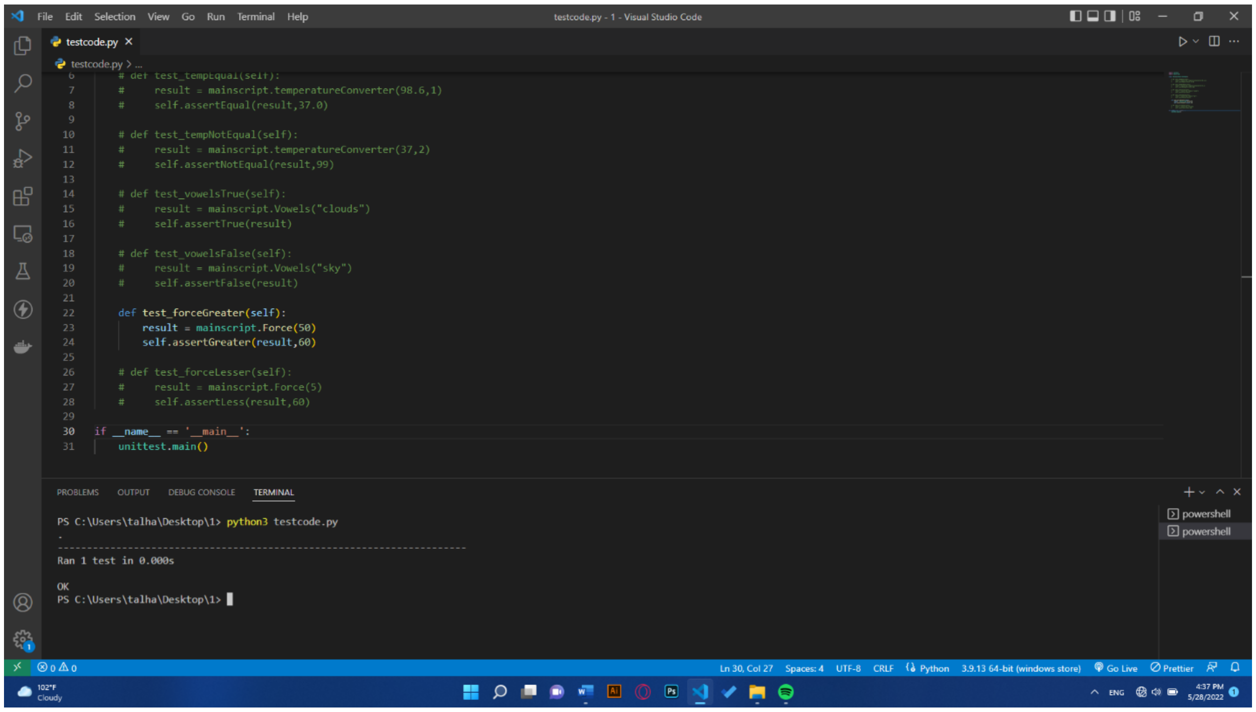
Test Case 4: assertFalse



```python
import unittest
import mainscript

class Test(unittest.TestCase):

    # def test_tempEqual(self):
    #     result = mainscript.temperatureConverter(98.6,1)
    #     self.assertEqual(result,37.0)

    # def test_tempNotEqual(self):
    #     result = mainscript.temperatureConverter(37,2)
    #     self.assertNotEqual(result,99)

    # def test_vowelsTrue(self):
    #     result = mainscript.Vowels("clouds")
    #     self.assertTrue(result)

    def test_vowelsFalse(self):
        result = mainscript.Vowels("sky")
        self.assertFalse(result)

    # def test_forceGreater(self):
    #     result = mainscript.Force(50)
    #     self.assertGreater(result,60)

    # def test_forceLesser(self):
    #     result = mainscript.Force(5)
```

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\talha\Desktop\1>
```

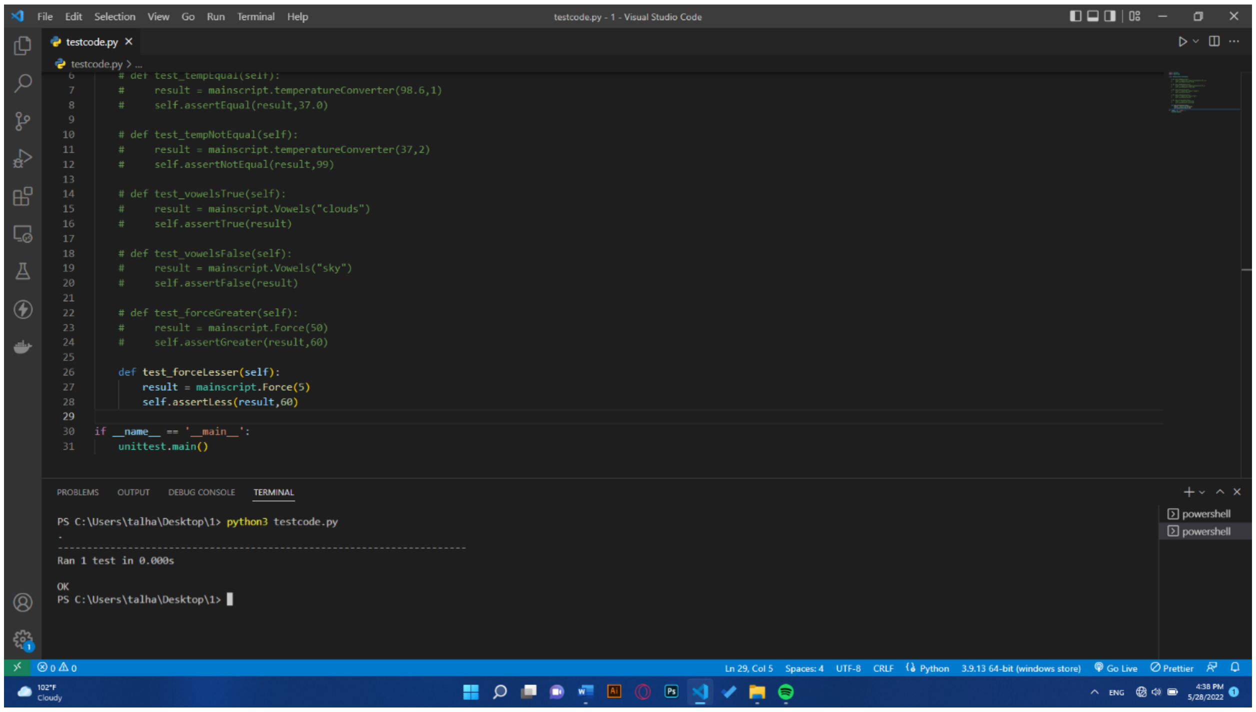
Test Case 5: assertGreater



```
    # def test_tempEqual(self):
    #     result = mainscript.temperatureConverter(98.6,1)
    #     self.assertEqual(result,37.0)

    # def test_tempNotEqual(self):
    #     result = mainscript.temperatureConverter(37,2)
    #     self.assertNotEqual(result,99)

    # def test_vowelsTrue(self):
    #     result = mainscript.Vowels("clouds")
    #     self.assertTrue(result)

    # def test_vowelsFalse(self):
    #     result = mainscript.Vowels("sky")
    #     self.assertFalse(result)

    def test_forceGreater(self):
        result = mainscript.Force(50)
        self.assertGreater(result,60)

    # def test_forceLesser(self):
    #     result = mainscript.Force(5)
    #     self.assertLess(result,60)

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

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\talha\Desktop\1>
```

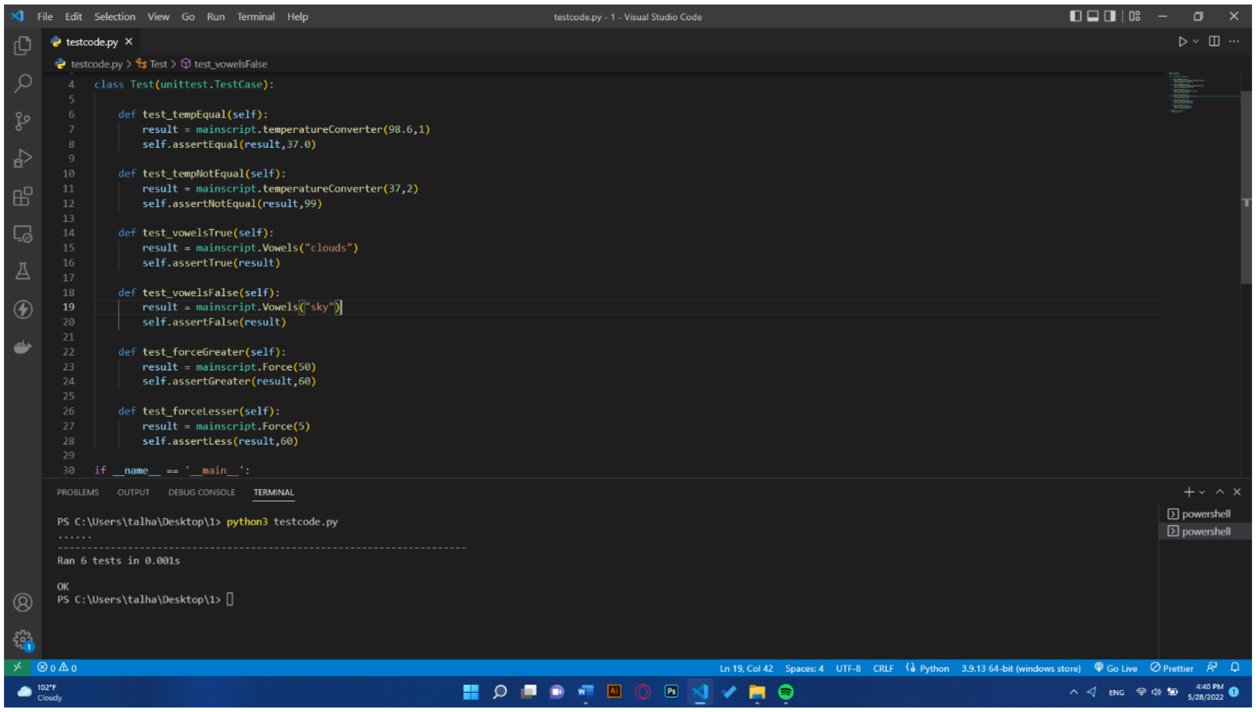
Test Case 6: assertLesser



```
    # def test_tempEqual(self):
    #     result = mainscript.temperatureConverter(98.6,1)
    #     self.assertEqual(result,37.0)

    # def test_tempNotEqual(self):
    #     result = mainscript.temperatureConverter(37,2)
    #     self.assertNotEqual(result,99)

    # def test_vowelsTrue(self):
    #     result = mainscript.Vowels("clouds")
    #     self.assertTrue(result)

    # def test_vowelsFalse(self):
    #     result = mainscript.Vowels("sky")
    #     self.assertFalse(result)

    # def test_forceGreater(self):
    #     result = mainscript.Force(50)
    #     self.assertGreater(result,60)

    def test_forceLesser(self):
        result = mainscript.Force(5)
        self.assertLess(result,60)

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

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\talha\Desktop\1>
```

Test Suite Results



```python
class Test(unittest.TestCase):

    def test_tempEqual(self):
        result = mainscript.temperatureConverter(98.6,1)
        self.assertEqual(result,37.0)

    def test_tempNotEqual(self):
        result = mainscript.temperatureConverter(37,2)
        self.assertNotEqual(result,99)

    def test_vowelsTrue(self):
        result = mainscript.Vowels("clouds")
        self.assertTrue(result)

    def test_vowelsFalse(self):
        result = mainscript.Vowels("sky")
        self.assertFalse(result)

    def test_forceGreater(self):
        result = mainscript.Force(50)
        self.assertGreater(result,60)

    def test_forceLesser(self):
        result = mainscript.Force(5)
        self.assertLess(result,60)

if __name__ == '__main__':
```

```
PS C:\Users\talha\Desktop\1> python3 testcode.py
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s

OK
PS C:\Users\talha\Desktop\1>
```