**pytTitle:** Unit Testing

* Using unittest module for Unit-Testing
* Writing Unit Test Function
* Execute the Test Script

**Objective(s):**

Upon finishing the lab, student will attain mastery of the following:

* Writing Unit-Testing Program.
* Executing the Test Script.

**Tools, Equipment and Materials:**

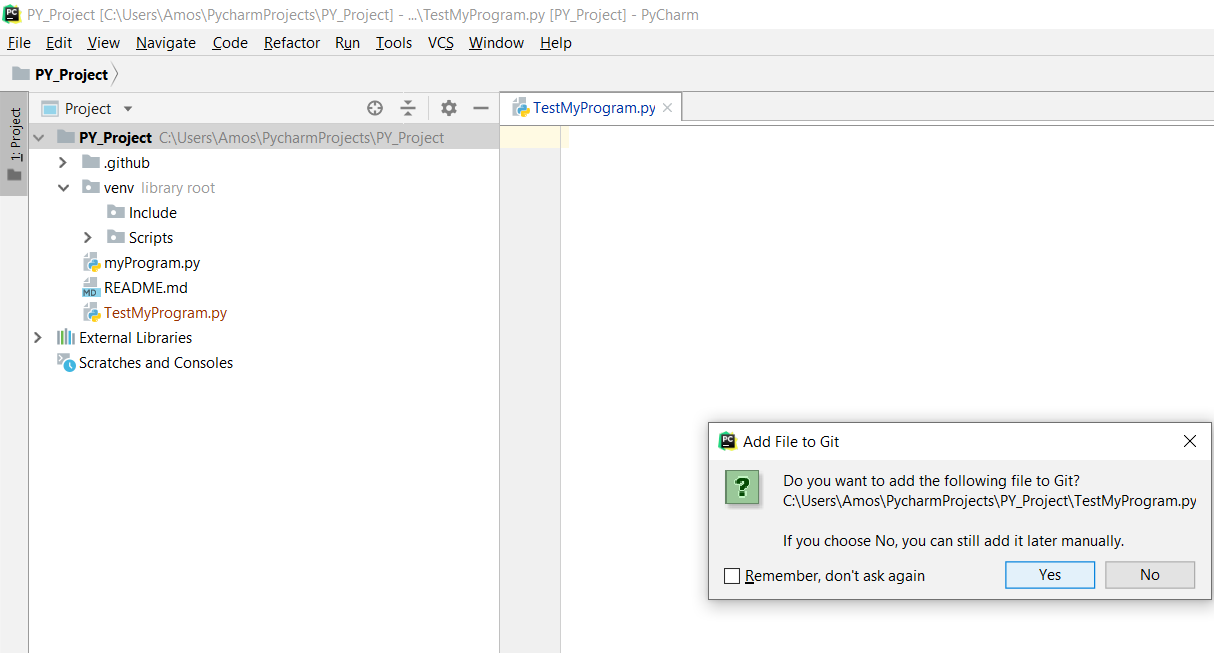
1. Personal Computer with Internet Access
2. Operation System with Installation of Python Software / Python IDE (e.g. PyCharm)

**Instructions:**

1. In this lab, you'll learn how to create a unit test program.

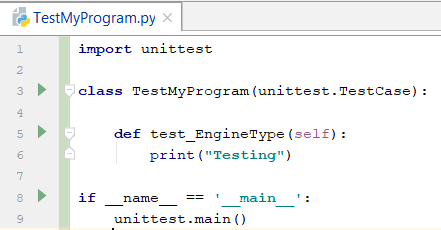
**Open PyCharm Project**

1. Start PyCharm IDE and open the previous created project (PY\_Project) in PyCharm. Add a new Python file named **TestMyProgram.py**. A window will prompt you to Add to Git if you have setup VCS. Click ‘Yes’ to continue.

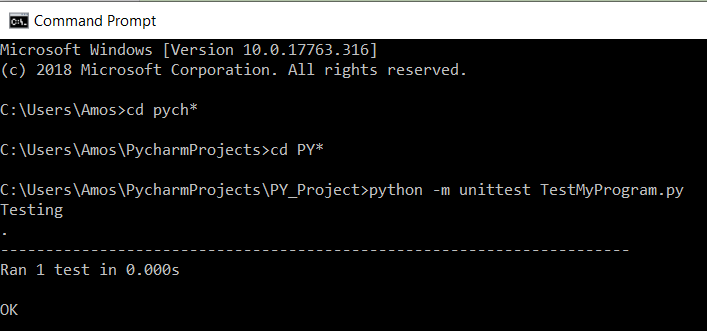


**Write a simple Unit Testing Program (with no testing) and executing the test script**

1. Write a simple UnitTest program with only 1 test case as shown below:



1. Open command prompt and change the path to the location of the TestMyProgram.py file. Type the following command as shown below to run the test case. The result is shown below that it has run successfully with 1 test pass because there is no Assert function.



NOTE:

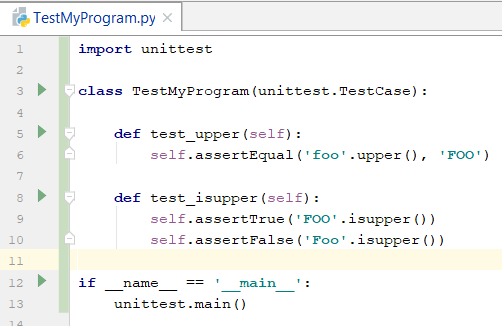
1. Right click on the file, TestMyProgram àopen in Terminal
2. Typeà python -m unittest TestMyProgram

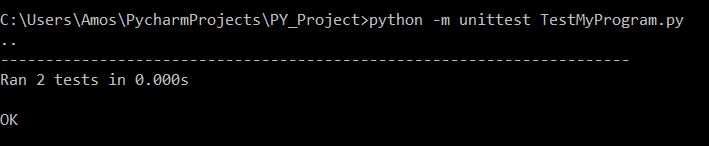
**A screenshot of a cell phone

Description automatically generated**

**Write a simple Unit Testing and executing the test script**

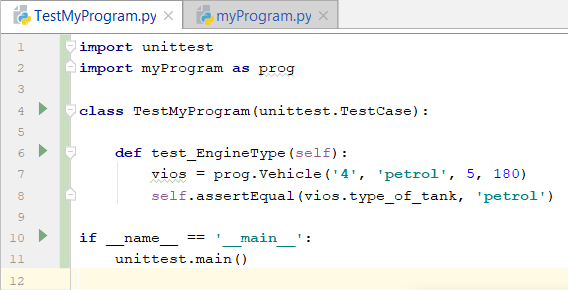
1. Change the program to the code below to demonstrate unit-testing Assert Function. With 2 test function written below, unittest will test the 2 function.





**Write a simple Unit Testing myProgram and executing the test script**

1. Write another test function which test myProgram Vehicle class as shown below:



Create myProgram.py as below:

import unittest  
import myVehicle as prog  
#from myVehicle import Vehicle  
class TestMyProgram(unittest.TestCase):  
 def test\_EngineTYpe(self):  
 vios = prog.Vehicle('4','petrol', 5, 180)  
 self.assertEqual(vios.type\_of\_tank, 'petrol')  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

NOTE: change import myProgram as prog to import myVehicle as prog

**Create the follow python file, save as myVehicle.py**

class Vehicle:  
 def \_\_init\_\_(self, number\_of\_wheels, type\_of\_tank, seating\_capacity, maximum\_velocity):  
 self.number\_of\_wheels = number\_of\_wheels  
 self.type\_of\_tank = type\_of\_tank  
 self.seating\_capacity = seating\_capacity  
 self.maximum\_velocity = maximum\_velocity

def drive(self):  
 print(**"The vehicle is in driving mode now"**)

vios = Vehicle (4, **'petrol'**, 5, 180)

vios.drive()

Test Result:

Text

Description automatically generated

1. Create a Python file, named as operationMaths.py with the codes below:

Text

Description automatically generated with medium confidence

1. Create a unit test Python file names as UnitTestMaths.py with the code below:

Text

Description automatically generated

data=[…......]

result=…......(data)

1. Run the UnitTestMaths in Terminal with the command below:

python -m unittest UnitTestMaths.py

Result:

Text

Description automatically generated

1. Challenge Question
2. Write a unit test function called testMark.py to test function in the marks.py
3. You are given a list of 10 marks: [45,60,70,40,80,90,55,75,20,65]
4. In the mark.py file, create a class called EvaluateMarks.
5. In this class, create 4 functions, total(), max(), min() & mean() and return their values to the testMark.py unit test file for the assertion test. The list will be passed to the 4 functions.
6. Execute the test script. You should do the following test options:
   1. All 4 correct results.
   2. First 2 tests (total & max) wrong answer.

|  |  |
| --- | --- |
|  |  |

- END