Developing Secure Software – CMP 6045B/7038B

Lab -Software Testing – Unit testing

**Objectives:**

The aim of this lab is to learn how to learn how to use Python’s unittest functionality and complete a test plan template. These processes are an essential part of assignment 2’s Technical Design Document.

**Overview:**

You will need to refer to the Week 12 lecture content, which explains about testing. This will enable you to learn how to run unit tests, debug errors and complete a test plan template. It is important you complete this lab as these processes are part of assignment 2.

**Step 1 - Create a new project in Pycharm**

* Call the projectUnitTestingExample
* Right-click on UnitTestingExample, New, Python File and call the file Car
* Type the following code into Car.py

**class** Car:

**def** \_\_init\_\_(self, speed=0):

self.speed = speed

self.odometer = 0

self.time = 0

**def** say\_state(self):

**print**(**"I'm going {} kph!"**.format(self.speed))

**def** accelerate(self):

self.speed += 5

**def** brake(self):

self.speed -= 5

**def** step(self):

self.odometer += self.speed

self.time += 1

**def** average\_speed(self):

**if** self.time != 0:

**return** self.odometer / self.time

**else**:

**pass**

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

my\_car = Car()

**print**(**"I'm a car!"**)

**while** True:

action = input(**"What should I do? [A]ccelerate, [B]rake, "**

**"show [O]dometer, or show average [S]peed?"**).upper()

**if** action **not** **in** **"ABOS"** **or** len(action) != 1:

**print**(**"I don't know how to do that"**)

**continue**

**if** action == **'A'**:

my\_car.accelerate()

**elif** action == **'B'**:

my\_car.brake()

**elif** action == **'O'**:

**print**(**"The car has driven {} kilometers"**.format(my\_car.odometer))

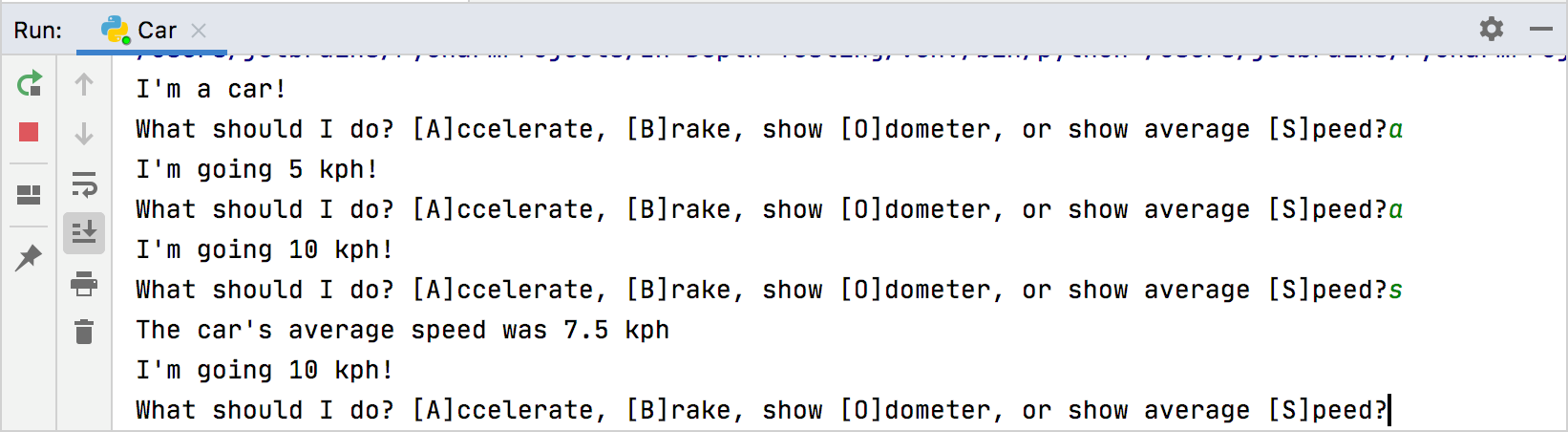
**elif** action == **'S'**:

**print**(**"The car's average speed was {} kph"**.format(my\_car.average\_speed()))

my\_car.step()

my\_car.say\_state()

* Click Run Car. Something similar to below should appear in your console when you input A, B, O and S

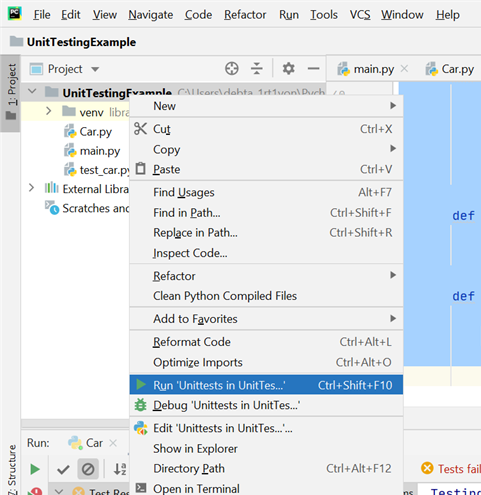


**Step 2 – Writing and running unit tests**

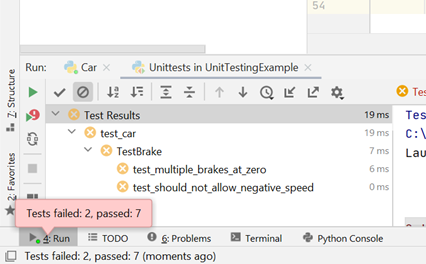
* Right-click UnitTestingExample, New, Python Files and choose Python Unit Test
* Name the test file test\_car
* This will import unittest and create a default test showing MyTestCase class.
* Now type the following information into your test file, *override* everything already showing

import unittest  
  
from Car import Car  
  
  
class TestCar(unittest.TestCase):  
 def setUp(self):  
 self.car = Car()  
  
  
class TestInit(TestCar):  
 def test\_initial\_speed(self):  
 self.assertEqual(self.car.speed, 0)  
  
 def test\_initial\_odometer(self):  
 self.assertEqual(self.car.odometer, 0)  
  
 def test\_initial\_time(self):  
 self.assertEqual(self.car.time, 0)  
  
  
class TestAccelerate(TestCar):  
 def test\_accelerate\_from\_zero(self):  
 self.car.accelerate()  
 self.assertEqual(self.car.speed, 5)  
  
 def test\_multiple\_accelerates(self):  
 for \_ in range(3):  
 self.car.accelerate()  
 self.assertEqual(self.car.speed, 15)  
  
  
class TestBrake(TestCar):  
 def test\_brake\_once(self):  
 self.car.accelerate()  
 self.car.brake()  
 self.assertEqual(self.car.speed, 0)  
  
 def test\_multiple\_brakes(self):  
 for \_ in range(5):  
 self.car.accelerate()  
 for \_ in range(3):  
 self.car.brake()  
 self.assertEqual(self.car.speed, 10)  
  
 def test\_should\_not\_allow\_negative\_speed(self):  
 self.car.brake()  
 self.assertEqual(self.car.speed, 0)  
  
 def test\_multiple\_brakes\_at\_zero(self):  
 for \_ in range(3):  
 self.car.brake()  
 self.assertEqual(self.car.speed, 0)

* Now run the unittest file. Right click class name, then click Run Unittests in UnitTest

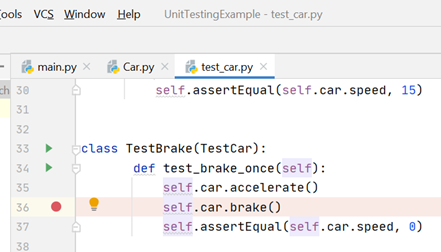


* This will show Tests failed: 2, passed:7 in a pink pop up box.

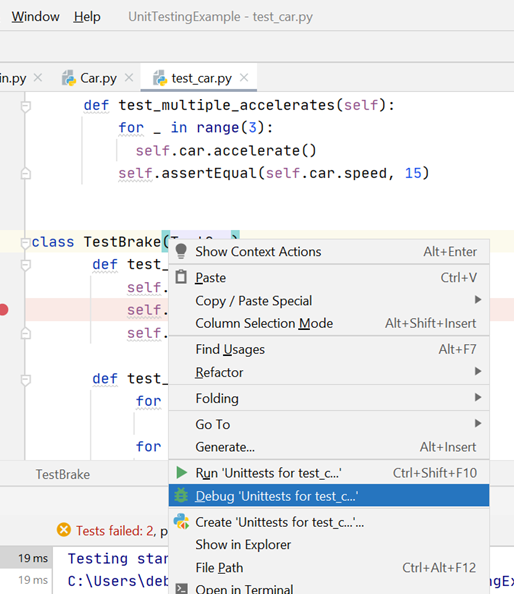


**Step 3 – Debugging a test**

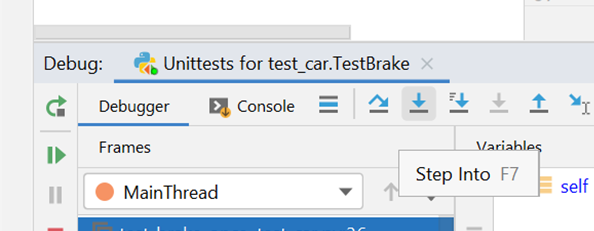
* Debug one of the failed tests TestBrake
* Add a breakpoint in the following place in the main TestBreak code by left-clicking next to the row number e.g. 36



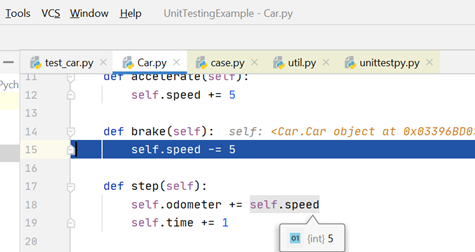
* Now launch a debugger session. Right-click the TestBreak method and click Start debugger Debug “Unittests for test\_c…”



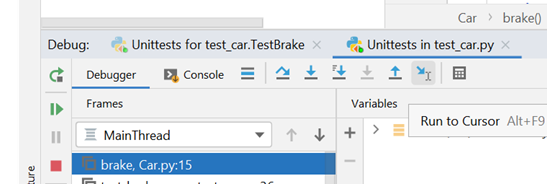
* Click the Step into button to skip the library classes and go straight to class Car



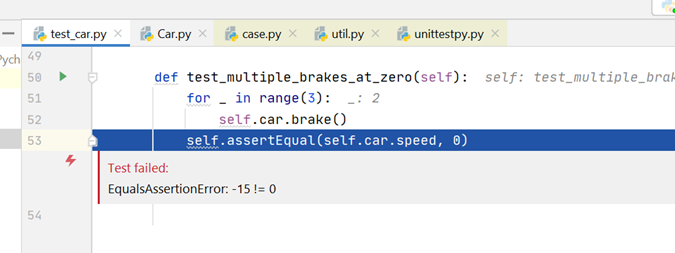
* This will show:



* Now click the Stepping into my code button



* This will show the test failed due to the speed possibly becoming negative:



* Now you know what’s causing the error you need to change brake function in Car.py to:

**def** brake(self):

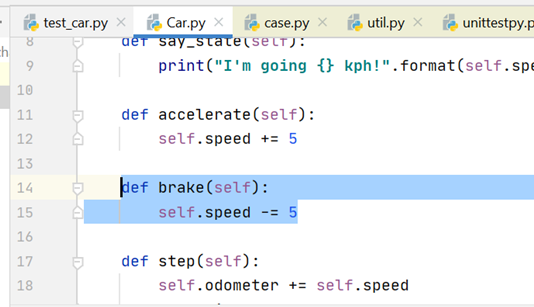
**if** self.speed < 5:

self.speed = 0

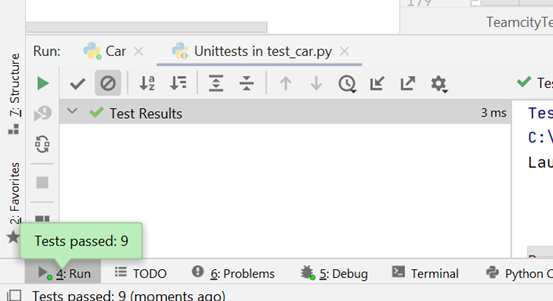
**else**:

self.speed -= 5

* *Instead of the previous code of:*



* Run “Unittests in test\_c…” again. This time the tests should all pass and show a green pop up box confirming this



* You have now successfully completed a full unit test and debug process.

**Step 4 – Fill in a Test Plan**

* If you haven’t already done so, download the agile-test-plan-template from Blackboard
* Fill in the test plan for each of the functions in the Car.py file
* Remember to show you fixed and re-tested the brake function. Should show as both fail and pass
* Show the completed test plan to your lab leaders so you can get formative feedback

**Step 5 – Add comments to all the above code**

* This will help you understand what the code is doing