Assignment 1:

Zaq Widdows – zvw3 – ZAQ922
Karl Smink – ks2825 – krlsmnk
Joseph Sommer – jjs451 – sommerjj
Michael Johnston – mlj328 – MichaelJohnston92

NOTICE FOR GRADER/TEACHER:

Github does not have a "delete folder" option. Please ignore all folders within "MMChoiceInputTests" as they are outdated and I cannot get rid of them. The folder for the main menu coverage tests is "MainMenuTests". Sorry for the inconvenience.

Discussion:

I think TDD is wonderful, but not in this assignment. Keeping up with all the test cases becomes really easy after you just think about what all you need to be testing and how they should be tested. Once the tests are implemented, running them is easy and well documented. However, it freaking sucks to automate such testing methods on such a small assignment. Testing each individual application is very tedious and not very rewarding. Testing all of the applications together with the "main menu" interface is more of the scale necessary for automated testing.

I don't really want to write more than that first paragraph because that's really all my thoughts on it. So here are some examples. The BMI application was 25 lines of code, the Distance was 2 lines, the Retirement was 35 lines and the email verifier <UNKNOWN> lines long. The smallness of these programs makes documenting TDD a pain. However, the MainMenu application is closer to 100 lines and has multiple aspects to test. This makes the documentation more necessary to avoid repetitive test cases and general silliness.

I think the larger and more complex the project, the more important documentation of testing applications becomes while tiny ones do not necessitate it. The TDD just requires the platform of testing and upkeep overhead to the application actually being tested. This just makes the platform itself unnecessary to maintain when the programmer could easily remember what needs to be tested within 50 lines of code.

Naming Conventions:

The main naming convention is to name the tests "blahTest.py". In the actual code the convention, loosely, follows "test_blahfunctionality()". At least this is how Zaq, the guy writing this, did it.

GitHub Link:

https://github.com/ZAQ922/Assignment-1

Setup Instructions:

Download:
(from Github)

-BMIhtmlcov (proof of BMI coverage)
-MainMenuApplicationsCoverage (proof of Apps and Main Menu coverage)
-MainMenuTests (proof of Main Menu coverage)
-__pycache__ (proof of distance coverage)
- htmlcov (proof of distance coverage)
-Main Menu.mp4 (ScreenCast video)
-MainMenu.py (program to run)
-TDD.7z (ScreenCast video)
-TestRetirement.py (tests the retirement app)
-bmiTests.py (tests the bmi application)
-distanceFormulaTests.py (tests the distance app)

-retirement.py (tests the retirement app)

(from the internet)

-python3.xx(https://www.python.org/downloads/ program language) use the latest python shell to run the code

NOTE

Use pycharm(https://www.jetbrains.com/pycharm/download/#section=windows) OR
sublime text editor(https://www.sublimetext.com/3) OR
IDLE(Comes installed in the python 3.xx download) for running/debugging/editing the code

They should all run in a windows 10 environment.
Download the required software above
Run MainMenu.py in python shell
Follow the onscreen instructions.

Testing Output Picture: