Software Testing Q&A: Assignment 2

Name: Tanner Hess | NetID: tmh648 | GitHub: TanHess

Bagley College of Engineering, Mississippi State University

Professor Tanmay Bhowmik

March 10, 2022

Overview:

This report is a review of the testing of the BMI app built for assignment 2 of Software Testing and Q/A. In this app there is 1 function to be tested: the BMI function. The exact details of the testing will be described throughout this report along with the results of the testing procedures.

Functions

BMI(feet, inches, pounds):

Description:

This function takes 3 inputs as shown: feet, inches, pounds. The function first converts the input pounds into kilograms, then combines the feet and inches into just inches and converts this number into meters. After this, the function uses the BMI equation ($bmi=kg/meters^2$) to compute the BMI of the given parameters rounding to the nearest 0.1. The function then uses the calculated BMI to calculate weight class based on these parameters: BMI = underweight < 18.5 >= normal <= 24.9 > overweight <= 29.9 > obese. After weight class is determined, the function returns a tuple containing the calculated BMI value and weight class.

Tests:

This function is tested for each of the 4 weight categories it can return: underweight, normal, overweight, or obese. For testing, the Pytest framework is used (external Python testing library). Each of the weight categories has it's own test function

parametrized with the various test cases used. Test cases were chosen according to the Nx1 boundary testing technique (as that is what I used to test the function). The reason I chose the Nx1 (1x1 for these tests) testing technique is so that my test cases could catch a boundary shift issue if it arose. Figure 1 shows the test cases chosen for each category using the Nx1 method. Also note that epsilon is .1 for this function and these test cases so the boundary points are selected accordingly.

Test Cases	1	2	3	4	5
Underweight	0	0.1	12	18.4	18.5
Normal	18.4	18.5	21	24.9	25
Overweight	24.9	25	28	29.9	30
Obese	29.9	30	50	N/A	N/A

Figure 1

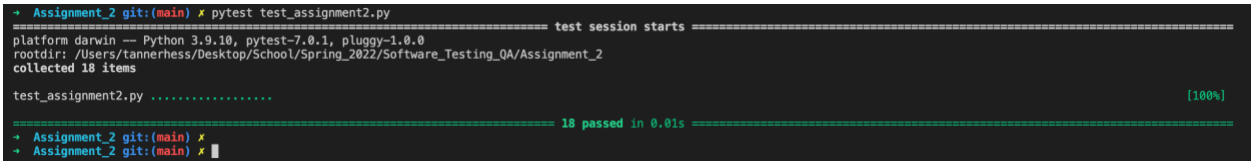
Execution:

Example of a full execution

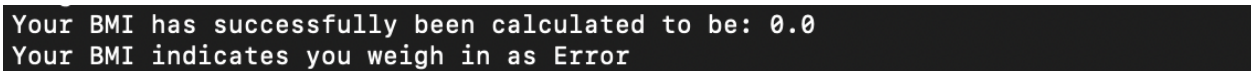
```
Please enter your body metrics below (Height is presented as feet then inches):
Height (Feet): 5
Height (Inches): 0
Weight: 250
Your BMI has successfully been calculated to be: 50.0
Your BMI indicates you weigh in as Obese
Would you like to continue?
1) Yes
2) No
Selection:
```

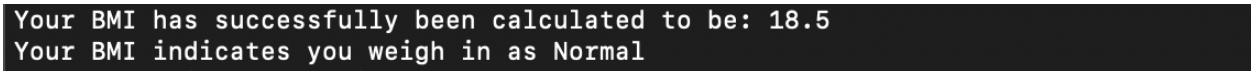
This section contains primarily screenshots demonstrating 2 things: the execution of the test file, as well as the tests running manually through the app interface along with the expected results. For the manual execution of the test cases, I will omit the redundant tests (where the test cases overlap: i.e. underweight: 18.4 and normal 18.4).

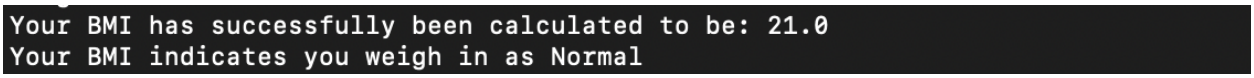
Test file execution

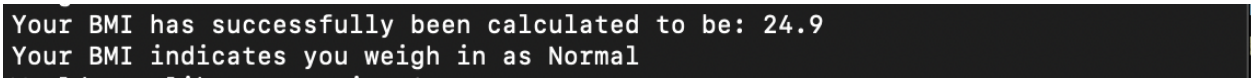


Manual execution









```
Your BMI has successfully been calculated to be: 29.9
Your BMI indicates you weigh in as Overweight
```

```
Your BMI has successfully been calculated to be: 30.0
Your BMI indicates you weigh in as Obese
```

```
Your BMI has successfully been calculated to be: 50.0
Your BMI indicates you weigh in as Obese
```

Boundary Shift:

For one execution of test cases, I introduced a boundary shift in my code. I changed the boundary at the lower end of normal and the higher end of underweight. The boundary shift changes the boundary bordering Underweight and normal from 29.9 to 29.8. The change in code can be seen in the next two screenshots.

```python
# Impossible to have bmi 0 or less
if bmi < 0.1:
    weight_class = "Error"
elif bmi < 18.5:
    weight_class = "Underweight"
elif bmi <= 24.9:
    weight_class = "Normal"
elif bmi <= 29.9:
    weight_class = "Overweight"
else:
    weight_class = "Obese"

return bmi, weight_class
```

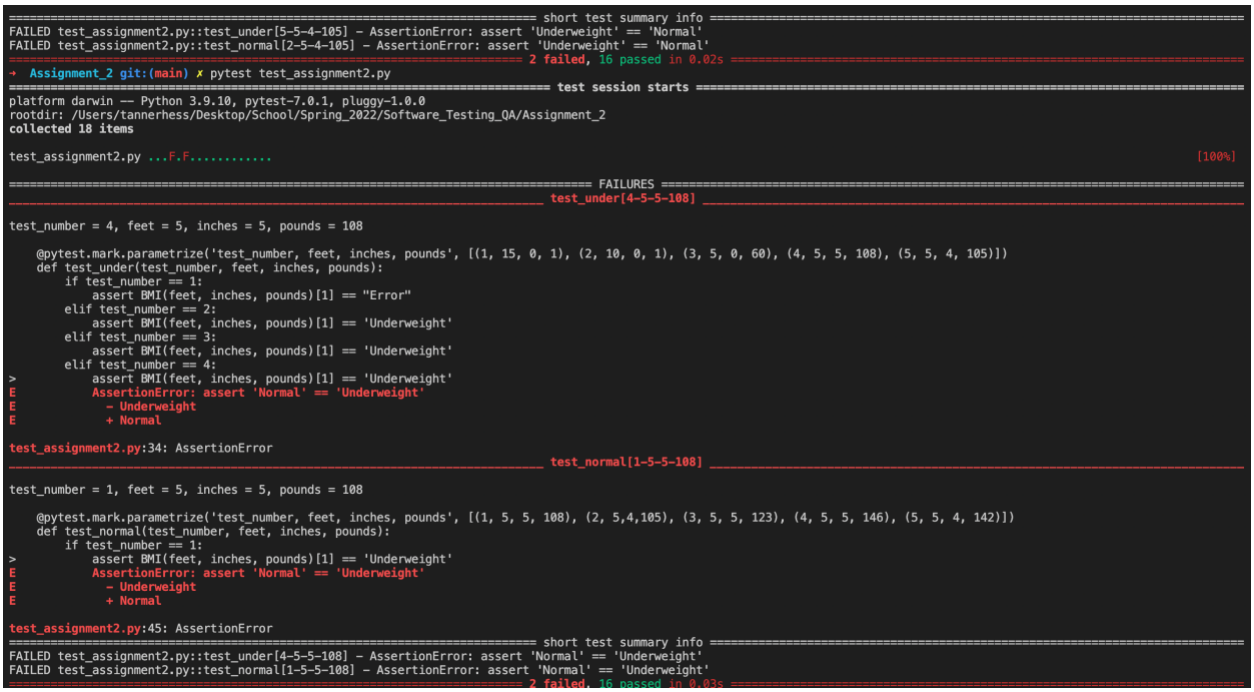
Original Code

```
# Impossible to have bmi 0 or less
if bmi < 0.1:
    weight_class = "Error"
elif bmi < 18.4:
    weight_class = "Underweight"
elif bmi <= 24.9:
    weight_class = "Normal"
elif bmi <= 29.9:
    weight_class = "Overweight"
else:
    weight_class = "Obese"

return bmi, weight_class
```

Code Introducing Boundary Shift

Below is the execution of the test cases after the boundary shift was introduced. As can be seen, the boundary shift is detected by the code. This is due to the implementation of Nx1 technique. This technique can determine a boundary shift problem where EPC would not have. The reason Nx1 was able to detect a boundary shift problem was because we used 1 point “on” the boundary, and one point “off” the boundary. The “off” point guarantees that we will be able to catch a boundary shift problem as regardless of the way the boundary shifts, either the “on” OR the “off” point will catch the mistake in code.



Instructions for Setup

IMPORTANT NOTE

If the python/pip commands listed within these instructions are not working, replace “python” and “pip” with “python3” and “pip3” respectively.

To download and setup this project, you must first have python along with pip (the package manager for python) installed on your device. If you do not have python, please follow this link to download it https://www.python.org/downloads/. After installing python, ensure pip installed along with your package and is working properly by running the command “pip --version” in the command line (most IDE’s have a built-in command line, if not, use your system’s command line).

After ensuring python and pip are installed, follow these step-by-step instructions to download this project:

1. Create a directory to store the project files in wherever suites you.
2. Go to https://github.com/TanHess/Software_Testing_QA_Assignment2
3. Click the green “Code” button on the page
4. Click the “Download Zip” button
5. Move the downloaded zip file to your directory you created earlier
6. Unzip the project
7. Use either the command line or an IDE with a command line to navigate to the project directory.
8. Once in the project directory within the command line, run the command “pip install -r requirements.txt”
9. After installing the dependencies, run the command “python assignment2.py” to run the main program.
10. Run the command “pytest test_assignment2.py” to run the test program.

After following these instructions, you should have this project fully functional on your device. If for some reason errors are arising, please email me with questions concerning this project: tmh648@msstate.edu