Assignment 2 Project Report
CSE 4283: Software Testing & QA

Jade Thompson

NetID: jet475
GitHub Username: jadethomp

Set Up and Execution

This program is written in Python. To run the program, Python must be installed on your system. The latest version of Python can be downloaded at this link:
https://www.python.org/downloads/

Additionally, this program uses the pytest package for unit testing. Instructions for setting up this dependency can be found at this link:
https://docs.pytest.org/en/7.1.x/getting-started.html

Once Python and dependencies are installed, navigate to the program's repository, which can be found at this link:
https://github.com/jadethomp/bmiCalculator

Clone the repository or download the source files to your system. To run the program, open a command line window and navigate to the program's directory. Run the following command:

```
python3 main.py
```

During the program's run, follow displayed instructions to input required information. After inputting your height and weight, the program will output your corresponding BMI and associated category.

Functions and Test Cases

This program uses three main functions for executing the requirements associated with calculating and displaying BMI. The first function converts user-inputted height into total inches, from the original feet and inches value. The second function calculates BMI using adjusted height and user-inputted weight. The third function uses calculated BMI to categorize the user into one of the four defined BMI categories.

Unit tests for this program were created using the Pytest testing framework. The unit tests are integrated within a GitHub Actions workflow that runs each test when changes are pushed to the program's repository.

I. Height Conversion

A. Function Behavior

This function takes two parameters, which are the separated foot and inch values for the user's height. This function converts the foot component of the height into inches, and returns this value combined with the original inch component of the user's inputted height. This results in a single height value in inches that can be used to calculate BMI.

B. Test Cases

1. **Positive Feet and Inches:** This test case uses the values 5 (feet) and 6 (inches), and expects a result of 66 inches.
2. **Positive Feet and Zero Inches:** This test case uses the values 4 (feet) and 0 (inches), and expects a result of 48 inches.
3. **Zero Feet and Inches:** This test case uses the values 0 (feet) and 0 (inches), and expects a result of -1, which indicates error.
4. **Negative Feet and Inches:** This test case uses the values -4 (feet) and -5 (inches), and expects a result of -1, which indicates error.

II. BMI Calculation

A. Function Behavior

This function takes two parameters: total height in inches and weight in pounds. Within the function, the provided BMI formula (viewable here) is executed in three main steps. Weight is converted to kilograms, inches are converted to square meters, and then the quotient of these two values is returned. The returned value is the user's BMI based on input height and weight.

It is important to note that for this function, height is not manipulated as a tested variable. Height input is tested in the Height Conversion function and respective test cases, so in these test cases, height is assumed to be accurate and usable. Thus, test coverage is maintained.

B. Test Cases

1. **Positive Weight:** This test case uses the values 65 (inches) and 160 (pounds), and expects a BMI of 27.27.
2. **Zero Weight:** This test case uses the values 65 (inches) and 0 (pounds), and expects a result of -1, which indicates error.

3. **Negative Weight:** This test case uses the values 65 (inches) and -160 (pounds), and expects a result of -1, which indicates error.

III. BMI Categorization

A. Function Behavior

This function takes a single parameter, the BMI calculated by the BMI Calculation function, and directly determines and prints the corresponding BMI category. This determination uses the defined boundaries provided in the assignment description.

B. Test Cases

1. **Underweight Interior:** This test case uses the BMI 16.00, and expects the program to output “Category: Underweight”.
2. **Underweight ON:** This test case uses the BMI 0.00, and expects the program to output “Invalid BMI”.
3. **Underweight OFF:** This test case uses the BMI 0.50, and expects the program to output “Category: Underweight”.
4. **Normal Weight Interior:** This test case uses the BMI 21.00, and expects the program to output “Category: Normal Weight”.
5. **Normal Weight ON:** This test case uses the BMI 18.50, and expects the program to output “Category: Normal Weight”.
6. **Normal Weight OFF:** This test case uses the BMI 24.9, and expects the program to output “Category: Underweight”.
7. **Overweight Interior:** This test case uses the BMI 27.00, and expects the program to output “Category: Overweight”.
8. **Overweight ON:** This test case uses the BMI 25.00, and expects the program to output “Category: Overweight”.
9. **Overweight OFF:** This test case uses the BMI 24.90, and expects the program to output “Category: Normal Weight”.
10. **Obese Interior:** This test case uses the BMI 35.00, and expects the program to output “Category: Obese”.
11. **Obese ON:** This test case uses the BMI 30.00, and expects the program to output “Category: Obese”.

12. **Obese OFF:** This test case uses the BMI 29.90, and expects the program to output “Category: Overweight”.

Boundary Testing Technique

For this program, the Weak Nx1 technique was chosen for boundary testing. This technique was chosen because of its wider ability to detect boundary errors compared to Extreme Point Combination (EPC). Thus, for each of the three boundaries separating the four major BMI categories, an on and off point was tested, as well as four interior test points within each of the categories. These tests are listed within the BMI Categorization test cases..

Boundary Shift Testing

Boundary shift was induced in line 29 of the program by changing the conditional from (final_bmi >= 18.50) to (final_bmi >= 18.40). The manipulated code can be seen below, before and after this change.

```
19  def print_category(final_bmi):
20      if(final_bmi <= 0.00):
21          print("Invalid BMI")
22          return
23      if(final_bmi >= 30.00):
24          # obese
25          print("Category: Obese")
26      elif(final_bmi >= 25.00):
27          # overweight
28          print("Category: Overweight")
29      elif(final_bmi >= 18.50):
30          # normal weight
31          print("Category: Normal Weight")
32      else:
33          # underweight
34          print("Category: Underweight")
```

```
19  def print_category(final_bmi):
20      if(final_bmi <= 0.00):
21          print("Invalid BMI")
22          return
23      if(final_bmi >= 30.00):
24          # obese
25          print("Category: Obese")
26      elif(final_bmi >= 25.00):
27          # overweight
28          print("Category: Overweight")
29      elif(final_bmi >= 18.40):
30          # normal weight
31          print("Category: Normal Weight")
32      else:
33          # underweight
34          print("Category: Underweight")
```

Following this boundary shift, unit tests were rerun. The output from the testing framework can be seen below.

```
C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator>python -m pytest
============================================= test session starts =============================================
platform win32 -- Python 3.11.5, pytest-8.0.2, pluggy-1.4.0
rootdir: C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator
collected 19 items

test_main.py ............F......                                                                      [100%]

================================================== FAILURES ==================================================
_____________________________ test_print_category[18.49-Category: Underweight] _____________________________

bmi = 18.49, statement = 'Category: Underweight', capsys = <_pytest.capture.CaptureFixture object at 0x000001632D5D21D0>

    @pytest.mark.parametrize("bmi, statement", [(16.00, "Category: Underweight"), (0.00, "Invalid BMI"), (0.50, "Category: Underweight"),
                                                (21.00, "Category: Normal Weight"), (18.50, "Category: Normal Weight"), (18.49, "Category: Underweight"),
                                                (27.00, "Category: Overweight"), (25.0, "Category: Overweight"), (24.90, "Category: Normal Weight"),
                                                (35.00, "Category: Obese"), (30.00, "Category: Obese"), (29.90, "Category: Overweight")])
    def test_print_category(bmi, statement, capsys):
        print_category(bmi)
        captured_stdout, captured_stderr = capsys.readouterr()
>       assert captured_stdout.strip() == statement
E       AssertionError: assert 'Category: Normal Weight' == 'Category: Underweight'
E
E         - Category: Underweight
E         + Category: Normal Weight

test_main.py:19: AssertionError
========================================== short test summary info ==========================================
FAILED test_main.py::test_print_category[18.49-Category: Underweight] - AssertionError: assert 'Category: Normal Weight' == 'Category: Underweight'
======================================== 1 failed, 18 passed in 0.18s ========================================
```

As shown by the test report, the boundary shift was detected by the program's unit tests. This is because of the Weak Nx1 technique used to create the boundary-based unit tests. Each boundary is addressed in a way that checks for small discrepancies, such as boundary shifts, missing boundaries, and closure problems.

Outputs

I. Underweight Category Output

```
C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator>python3 main.py
Please input your height in feet and inches, separating the two values with a space.
For instance, if you are 5'0", input 5 0.
>> 5 10
Please input your weight in pounds.
>> 120
Your BMI is 17.63.
Category: Underweight
```

II. Normal Weight Category Output

```
C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator>python3 main.py
Please input your height in feet and inches, separating the two values with a space.
For instance, if you are 5'0", input 5 0.
>> 5 4
Please input your weight in pounds.
>> 130
Your BMI is 22.85.
Category: Normal Weight
```

III. Overweight Category Output

```
C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator>python3 main.py
Please input your height in feet and inches, separating the two values with a space.
For instance, if you are 5'0", input 5 0.
>> 4 10
Please input your weight in pounds.
>> 140
Your BMI is 29.96.
Category: Overweight
```

IV. Obese Category Output

```
C:\Users\jadet\OneDrive\Documents\GitHub\bmiCalculator>python3 main.py
Please input your height in feet and inches, separating the two values with a space.
For instance, if you are 5'0", input 5 0.
>> 5 0
Please input your weight in pounds.
>> 210
Your BMI is 42.0.
Category: Obese
```