Jacob Adams
NetID & GitHub: jra457

Discuss each function and associated tests: For each function/method in your code: (i) describe what the function does and (ii) write all the test cases for the function.

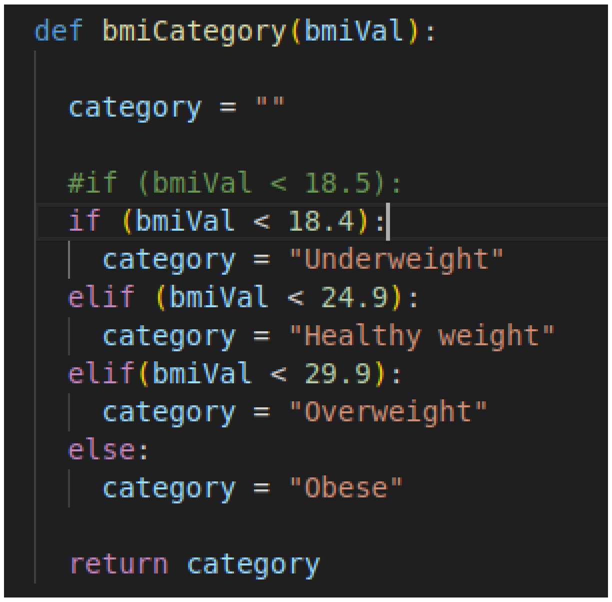
- Function 1: Weight
  1. i) Takes the input weight in pounds and multiplies by 0.45 to convert to kilograms. Returns the weight in kilograms.
  2. ii) Testing at the boundary of underweight (122.1 lbs.), healthy weight (122.4 lbs. & 164.6 lbs.), overweight (164.7 lbs. & 197.7 lbs.), and obese (197.8 lbs.).
- Function 2: Height
  1. i) Takes the input in inches and multiplies by 0.025 to convert to meters. Returns height in meters.
  2. ii) All tests conducted at average male height (5'9", 69").
- Function 3: BMI
  1. i) Takes the converted weight and divides by the converted height to calculate BMI, returns the result.
  2. ii) Testing at the boundary of underweight (122.1 lbs.), healthy weight (122.4 lbs. & 164.6 lbs.), overweight (164.7 lbs. & 197.7 lbs.), and obese (197.8 lbs.). All tests conducted at a height of 69".
- Function 4: BMI Category
  1. i) Takes the calculated BMI and applies it to the scale to determine if it underweight, healthy weight, overweight, or obese. Returns the category as a string.
  2. ii) Testing at BMI at underweight (<18.5), healthy weight (<24.9), overweight (<29.9), and obese (>=30.0).

Discuss which boundary testing technique you followed to choose your test cases and why.

To confirm the success of the system, Test Driven Development was implemented using Extreme Point Combination. The domain was defined as the minimum and maximum acceptable values for each category: underweight, healthy weight, overweight, and obese. Test cases were set to be boundary points for each category and ensured that each boundary point on the domain returned the correct category.

Include boundary shift by 0.1 at the lower boundary of "Normal Weight" and give a screenshot of the snippet with this boundary shift problem.

Code:



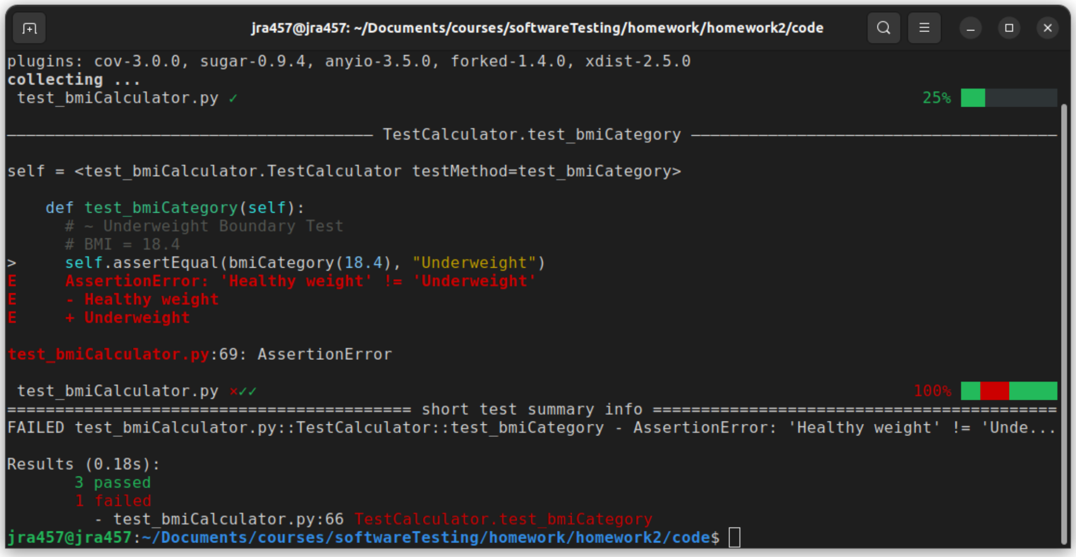
```
def bmiCategory(bmiVal):

    category = ""

    #if (bmiVal < 18.5):
    if (bmiVal < 18.4):
        category = "Underweight"
    elif (bmiVal < 24.9):
        category = "Healthy weight"
    elif(bmiVal < 29.9):
        category = "Overweight"
    else:
        category = "Obese"

    return category
```

Output:



```
jra457@jra457: ~/Documents/courses/softwareTesting/homework/homework2/code
plugins: cov-3.0.0, sugar-0.9.4, anyio-3.5.0, forked-1.4.0, xdist-2.5.0
collecting ...
 test_bmiCalculator.py ✓                                                     25% 

―――――――――――――――――――――――――――――――――――――― TestCalculator.test_bmiCategory ――――――――――――――――――――――――――――――――――――――

self = <test_bmiCalculator.TestCalculator testMethod=test_bmiCategory>

    def test_bmiCategory(self):
      # ~ Underweight Boundary Test
      # BMI = 18.4
>     self.assertEqual(bmiCategory(18.4), "Underweight")
E     AssertionError: 'Healthy weight' != 'Underweight'
E     - Healthy weight
E     + Underweight

test_bmiCalculator.py:69: AssertionError

 test_bmiCalculator.py ×✓✓                                                   100% 
========================================== short test summary info ==========================================
FAILED test_bmiCalculator.py::TestCalculator::test_bmiCategory - AssertionError: 'Healthy weight' != 'Unde...

Results (0.18s):
       3 passed
       1 failed
         - test_bmiCalculator.py:66 TestCalculator.test_bmiCategory
jra457@jra457:~/Documents/courses/softwareTesting/homework/homework2/code$ 
```

Did your test cases catch this boundary shift problem? Why or why not?

Yes, my test case for “Healthy Weight” caught the error, as the output produced was “Underweight,” which did not match the expected output. The test case caught the error, as the boundary point was set to be the minimum possible value.

Detailed setup and execution instructions: The application should be compatible with Windows 10 OS. Include links to download the application, instructions to setup and execute the application.

Download the bmiCalculator.py and test_bmiCalculator.py files, save them into their own directory. Open the directory through PowerShell and run:

```
$ pip install –U pytest
$ pip install –U numpy
```

Run Code:

```
$ python3 bmiCalculator.py
```

Run Tests:

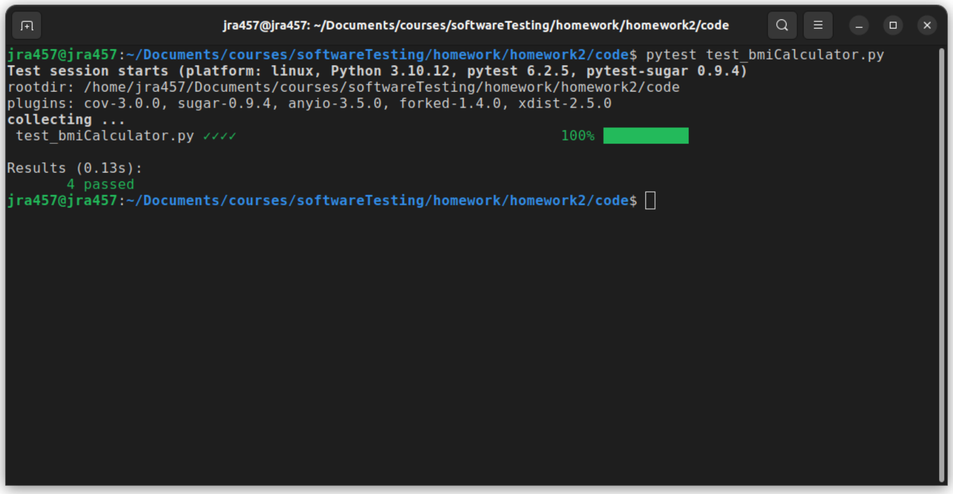
```
$ pytest test_bmiCalculator.py
```

Code Repository:

https://github.com/jra457/softwareTesting

Provide a screenshot showing that the application returns the correct output for all four BMI categories.

All tests passed, as shown in the screenshot below:



jra457@jra457: ~/Documents/courses/softwareTesting/homework/homework2/code

```
jra457@jra457:~/Documents/courses/softwareTesting/homework/homework2/code$ pytest test_bmiCalculator.py
Test session starts (platform: linux, Python 3.10.12, pytest 6.2.5, pytest-sugar 0.9.4)
rootdir: /home/jra457/Documents/courses/softwareTesting/homework/homework2/code
plugins: cov-3.0.0, sugar-0.9.4, anyio-3.5.0, forked-1.4.0, xdist-2.5.0
collecting ...
 test_bmiCalculator.py ✓✓✓✓                                        100% ██████████

Results (0.13s):
       4 passed
jra457@jra457:~/Documents/courses/softwareTesting/homework/homework2/code$ 
```