Command Line BMI Calculator Project Report

Wyatt Shanahan

NetID: wls219

Github: wyattshanahan

Table of Contents

Program Functions

This section details the component functions of the program. Each subsection includes the function name, purpose, and test cases used for testing and evaluating the functions.

calculateBMI

Purpose: Calculates the user's BMI using float inputs weight and height and returning float output BMI.

Test Cases: The testing procedure for this function utilises a series of unit tests to test different types of input to verify the function's ability to calculate the conversions correctly.

The function expects either integers or floats to be passed in and then converts the inputs to floats for doing the metric conversion. This conversion is performed to attempt some error handling to prevent exceptions. Tests will use the format of:

```
assert calculateBMI(inputWeight, inputHeight) == BMI
```

Which will verify that all values are working and can cope with different types of values in the expected manner. The types used as inputs are integer, float, and string. Two different variants of strings are used for testing. "67" is a string which can be converted by Python's float() function while "sixtyseven" cannot be converted and results in an error being thrown.

The table on the next page highlights each value chosen as an input, the expected output, and information regarding the rationale of using each as an input for testing purposes.

Unit Test Cases for calculateBMI Functionality

Name	**Input**	**Input Types**	**Expected Outputs**
calculateBMI_int_int	67, 14	Int, int	4.785714285714286
calculateBMI_int_flt	67, 14.0	Int, float	4.785714285714286
calculateBMI_int_str	67, “14”	Int, string	4.785714285714286
calculateBMI_int_invalid	67, “fourteen”	Int, string	"Error converting your inputs"
calculateBMI_flt_int	67.0, 14	Float, int	4.785714285714286
calculateBMI_flt_flt	67.0, 14.0	Float, float	4.785714285714286
calculateBMI_flt_str	67.0, “14”	Float, string	4.785714285714286
calculateBMI_flt_invalid	67.0, “fourteen”	Float, string	"Error converting your inputs"
calculateBMI_str_int	“67”, 14	String, int	4.785714285714286
calculateBMI_str_flt	“67”, 14.0	String, float	4.785714285714286
calculateBMI_str_str	“67”, “14”	String, string	4.785714285714286
calculateBMI_str_invalid	“67”, “fourteen”	String, string	"Error converting your inputs"
calculateBMI_invalid_int	“sixtyseven”, 14	String, int	"Error converting your inputs"
calculateBMI_invalid_flt	“sixtyseven”, 14.0	String, float	"Error converting your inputs"
calculateBMI_invalid_str	“sixtyseven”, “14”	String, string	"Error converting your inputs"
calculateBMI_invalid_invalid	“sixtyseven”, “fourteen”	String, string	"Error converting your inputs"

categoriseBMI

Purpose: Using the user's BMI, this function categorises the user into the categories of Underweight, Normal Weight, Overweight, or Obese based on the guidelines from the World Health Organisation. This function returns a string containing the category which the user falls into.

Test Cases: The testing procedure for this function utilises the Weak Nx1 boundary testing procedure. Thus, using the formula:

$$(N+1) * b + 1 = 3*3+1 = 10$$

The procedure will use 10 different boundary testing points. For these tests, PyTest uses a simple assertion to check if categoriseBMI(input) is consistent with the expected categories as per the BMI calculations formula. It does so using the following standardized formula:

```
assert categoriseBMI(input) == output
```

The table on the next page highlights each value chosen as an input, the expected output, and information regarding the rationale of using each as an input for testing purposes.

Boundary Test Cases for categoriseBMI Functionality

Name	Input	Expected Output	Point Type	Notes
categorise_9_25	9.25	"Underweight"	Interior point for Underweight	Point calculated as the midpoint between 18.5 and 0
categorise_18_4	18.4	"Underweight"	Off point for Underweight and Normal Weight	
categorise_18_5	18.5	"Normal Weight"	On boundary point for Underweight and Normal Weight	
categorise_21_7	21.7	"Normal Weight"	Interior point for Normal Weight	
categorise_ 24_9	24.9	"Normal Weight"	Off point for Normal Weight and Overweight	
categorise_25_0	25.0	"Overweight"	On boundary point for Normal Weight and Overweight	
categorise_27_45	27.45	"Overweight"	Interior point for Overweight	
categorise_29_9	29.9	"Overweight"	Off point for Overweight and Obese	
categorise_30_0	30.0	"Obese"	On boundary point for Overweight and Obese	
categorise_39_25	39.25	"Obese"	Interior point for Obese	Point calculated by adding Underweight midpoint to lower bound

Several unit tests were used to validate input processing for several different types of input. The table on the next page highlights these tests.

Boundary Test Cases for categoriseBMI Functionality

Name	**Input**	**Input Types**	**Expected Outputs**
categorise_int	30	Int	“Obese”
categorise_flt	30.0	Float	“Obese”
categorise_str	“30”	String	“Obese”
categorise_invalid	“thirty”	String	“Error converting your inputs”

imperialToMetric

Purpose: Converts imperial units into metric units for use in calculating BMI. It takes input for weight (in pounds) and height (in inches) and returns the user’s weight in kilograms and the user’s height in square metres.

Test Cases: The testing procedure for this function utilises a series of unit tests to test different types of input to verify the function’s ability to calculate the conversions correctly.

The function expects either integers or floats to be passed in and then converts the inputs to floats for doing the metric conversion. Tests will use the format of

```
assert categoriseBMI(inputWeight, inputHeight) == outputWeight, outputHeight
```

Which will verify that all values are working and can cope with different types of values in the expected manner. The types used as inputs are integer, float, and string. Two different variants of strings are used for testing. “150” is a string which can be converted by Python’s float() function while “onefifty” cannot be converted and causes a conversion error.

The table on the next page highlights each value chosen as an input, the expected output, and information regarding the rationale of using each as an input for testing purposes.

Unit Test Cases for imperialToMetric Functionality

Name	Input	Input Types	Expected Outputs
imperialToMetric_int_int	150, 150	Int, int	(67.5,14.0625)
imperialToMetric_int_flt	150, 150.0	Int, float	(67.5,14.0625)
imperialToMetric_int_str	150, “150”	Int, string	(67.5,14.0625)
imperialToMetric_int_invalid	150, “onefifty”	Int, string	"Error processing height input"
imperialToMetric_flt_int	150.0, 150	Float, int	(67.5,14.0625)
imperialToMetric_flt_flt	150.0, 150.0	Float, float	(67.5,14.0625)
imperialToMetric_flt_str	150.0, “150”	Float, string	(67.5,14.0625)
imperialToMetric_flt_invalid	150.0, “onefifty”	Float, string	"Error processing height input"
imperialToMetric_str_int	“150”, 150	String, int	(67.5,14.0625)
imperialToMetric_str_flt	“150”, 150.0	String, float	(67.5,14.0625)
imperialToMetric_str_str	“150”, “150”	String, string	(67.5,14.0625)
imperialToMetric_str_invalid	“150”, “onefifty”	String, string	"Error processing height input"
imperialToMetric_invalid_int	“onefifty”, 150	String, int	"Error processing weight input"
imperialToMetric_invalid_flt	“onefifty”, 150.0	String, float	"Error processing weight input"
imperialToMetric_invalid_str	“onefifty”, “150”	String, string	"Error processing weight input"
imperialToMetric_invalid_invalid	“onefifty”, “onefifty”	String, string	"Error processing weight input"

processRawHeight

Purpose: Processes the raw user input to split the string into an array. It then converts the feet to inches to return the total inches in the user's height. This function expects the two values to be split by a comma, but will attempt to split at a space. In the event that more than two values are entered, the first two values will be used.

Test Cases: The testing procedure for this function utilises a series of unit tests to test different types of input to verify the function's ability to process the input, or to reject it if invalid, and to return a correct calculation. Several tests are used to validate different data types and formatting issues which may be present in user entered inputs. Tests will use the format of:

assert processRawHeight(input) == output

Which will verify the behaviour of the function matches the intended behaviour.

Unit Test Cases for processRawHeight Functionality

Name	**Input**	**Expected Output**	**Notes**
processRawHeight_comma	"12, 6"	150.0	Parses using comma
processRawHeight_space	"12 6"	150.0	Parses using space
processRawHeight_invalid	"twelvefootsix"	"Error parsing height"	Error thrown due to invalid input
processRawHeight_threevals	"12, 6, 12"	150.0	Function uses index values 0 and 1
processRawHeight_oneval	"12"	"Error processing your height, please validate your input"	Error thrown because only one value was entered

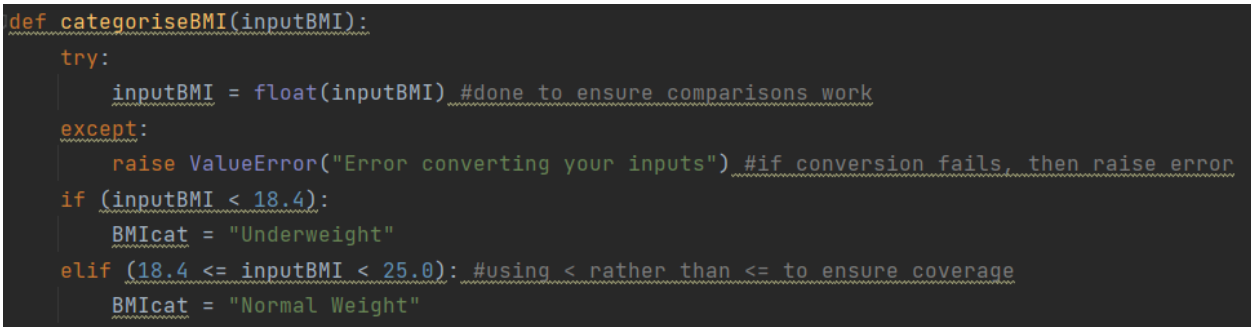
Boundary Testing Technique

For boundary testing, the Weak Nx1 boundary testing methodology was chosen. It was utilised because of its ability to detect any boundary shifts in addition to any boundary closure issues and missing boundaries. It was chosen over a strong testing method as the boundaries are clear and are not irregular. It was chosen over EPC to ensure that boundary shifts are detectable and due to its strong reliability when performing boundary testing. Using Weak Nx1 ensures that there is adequate coverage of all boundaries, which in turn ensures that no faults are present and that the program is behaving around the boundaries as intended.

Boundary Shift

This section contains information about the induction of a boundary shift and why it was successfully detected by the boundary testing procedure.

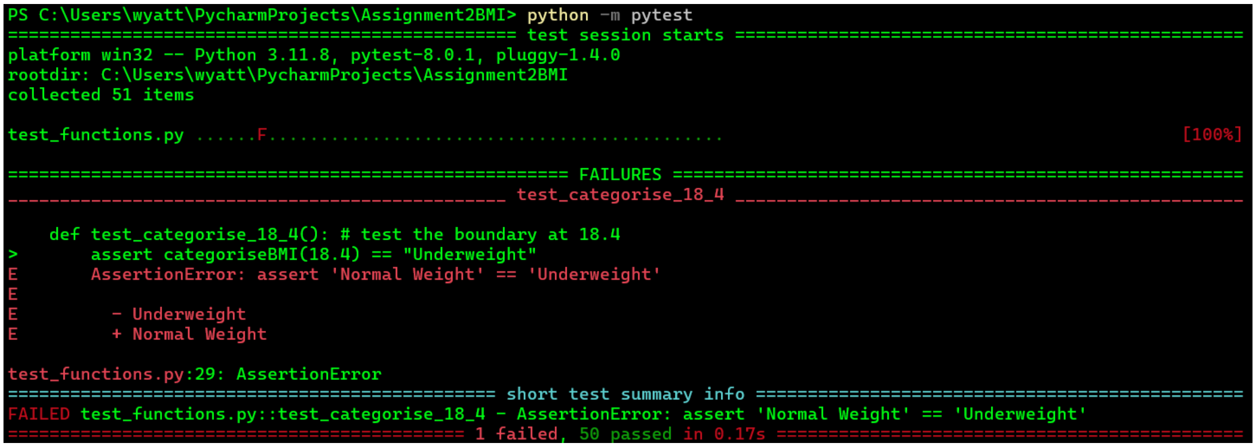
Boundary Shift Induction



```
def categoriseBMI(inputBMI):
    try:
        inputBMI = float(inputBMI) #done to ensure comparisons work
    except:
        raise ValueError("Error converting your inputs") #if conversion fails, then raise error
    if (inputBMI < 18.4):
        BMIcat = "Underweight"
    elif (18.4 <= inputBMI < 25.0): #using < rather than <= to ensure coverage
        BMIcat = "Normal Weight"
```

A boundary shift of 0.1 has been induced at the lower end of the “Normal Weight” section by adjusting the boundary between “Normal Weight” and “Underweight” from 18.5 to 18.4 in the code snippet above.

Boundary Shift Test Results



```
PS C:\Users\wyatt\PycharmProjects\Assignment2BMI> python -m pytest
================================================= test session starts =================================================
platform win32 -- Python 3.11.8, pytest-8.0.1, pluggy-1.4.0
rootdir: C:\Users\wyatt\PycharmProjects\Assignment2BMI
collected 51 items

test_functions.py ......F............................................                                            [100%]

====================================================== FAILURES =======================================================
________________________________________________ test_categorise_18_4 _________________________________________________

    def test_categorise_18_4(): # test the boundary at 18.4
>       assert categoriseBMI(18.4) == "Underweight"
E       AssertionError: assert 'Normal Weight' == 'Underweight'
E
E         - Underweight
E         + Normal Weight

test_functions.py:29: AssertionError
=============================================== short test summary info ===============================================
FAILED test_functions.py::test_categorise_18_4 - AssertionError: assert 'Normal Weight' == 'Underweight'
============================================ 1 failed, 50 passed in 0.17s =============================================
```

As shown in the testing output screenshot above, the tests caught the boundary shift successfully. The tests caught the problem because one of the boundary points being tested is 18.4, which under standard conditions should be processed and returns “Underweight”. However, with the boundary shift, it is processed and returns “Normal Weight”. The tests are checking for this return value and, due to the boundary shift, they detected the shifted boundary and reported the issue. This functionality is present because Weak Nx1 was utilised for boundary testing.

Download and Execution

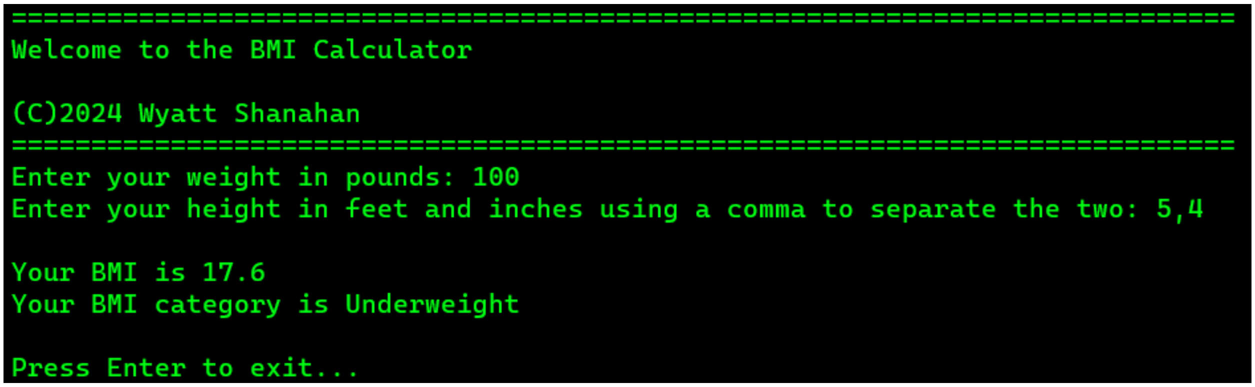
To execute this program, an installation of Python 3 is required. This program is optimised to run using Python version 3.11, which can be downloaded by clicking here. Alternatively, it can be found by going to the Microsoft Store application and searching for Python 3.11. To download the software, simply click on the download button. Please note that while other versions of Python 3 may work, this program runs best using version 3.11.

To download the program files, please go to the GitHub repository by clicking here and downloading calculator.py and functions.py. To do this, click on 'Code' to the top right of the centre of the screen and click 'Download ZIP'.

Once the file has been downloaded, navigate to it using Windows Explorer and extract the zip archive by right clicking it and then selecting 'Extract All…'. Once the zip archive has been extracted, navigate to the extracted files and ensure that calculator.py and functions.py are in the same directory. Once in the correct directory, right click on calculator.py. Select 'Open with' and choose Python 3.11 from the listed programs to execute the program and start the calculator. Then, simply follow the instructions provided in the program to calculate your BMI.
Download links from above can also be found below for convenience.
*Python 3.11***:**
https://apps.microsoft.com/detail/9nrwmjp3717k?ocid=pdpshare&hl=en-us&gl=US
*Calculator Files (Github)***:** https://github.com/wyattshanahan/BMI_CLI

Output Screenshots

This section contains screenshots for each of the possible BMI states during a standard program execution. Each uses a height of ‘5,4’ meaning 5 feet and 4 inches. The weight is adjusted to calculate each of the four states.



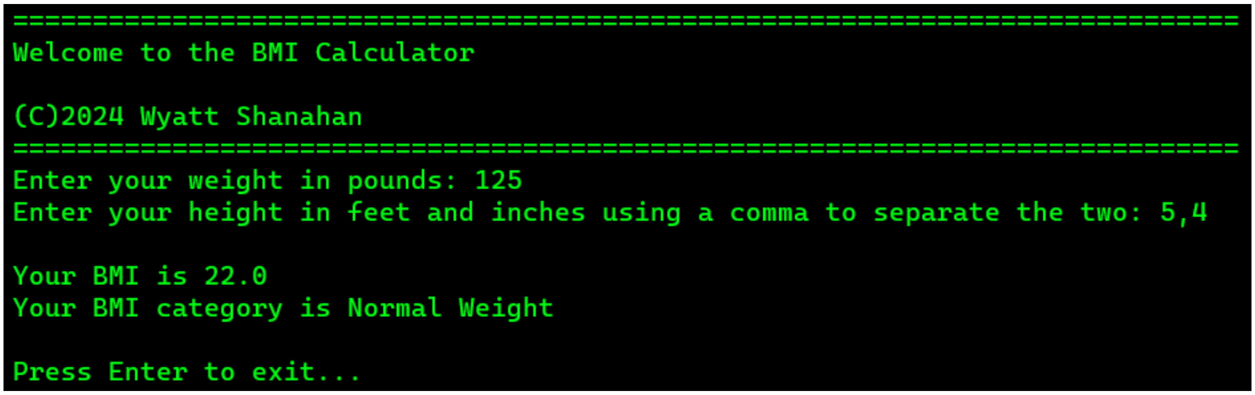
```
=============================================================================
Welcome to the BMI Calculator

(C)2024 Wyatt Shanahan
=============================================================================
Enter your weight in pounds: 100
Enter your height in feet and inches using a comma to separate the two: 5,4

Your BMI is 17.6
Your BMI category is Underweight

Press Enter to exit...
```

A standard execution resulting in an underweight BMI



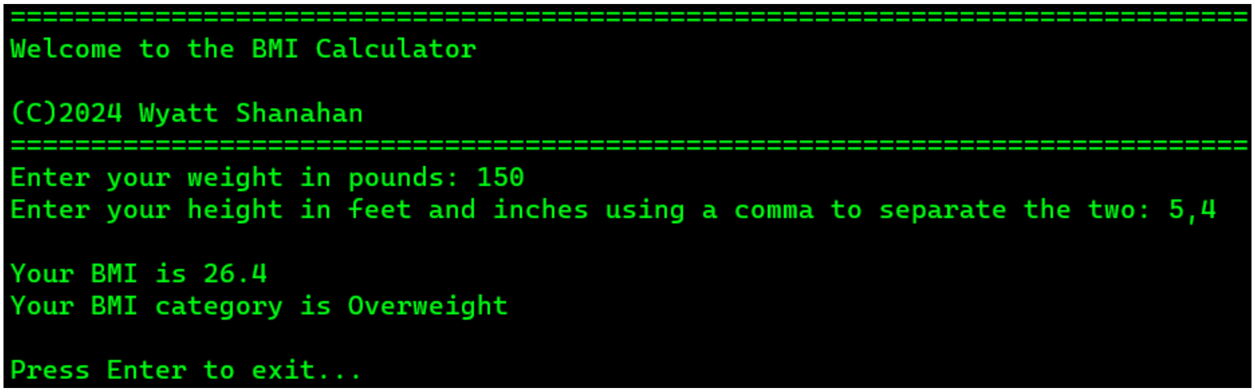
```
=============================================================================
Welcome to the BMI Calculator

(C)2024 Wyatt Shanahan
=============================================================================
Enter your weight in pounds: 125
Enter your height in feet and inches using a comma to separate the two: 5,4

Your BMI is 22.0
Your BMI category is Normal Weight

Press Enter to exit...
```

A standard execution resulting in a normal BMI



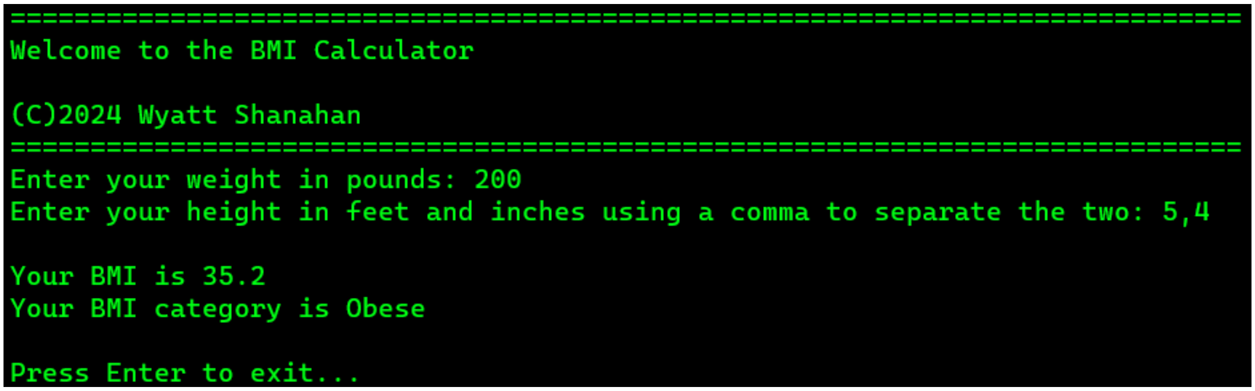
```
=============================================================================
Welcome to the BMI Calculator

(C)2024 Wyatt Shanahan
=============================================================================
Enter your weight in pounds: 150
Enter your height in feet and inches using a comma to separate the two: 5,4

Your BMI is 26.4
Your BMI category is Overweight

Press Enter to exit...
```

A standard execution resulting in an overweight BMI



```
=============================================================================
Welcome to the BMI Calculator

(C)2024 Wyatt Shanahan
=============================================================================
Enter your weight in pounds: 200
Enter your height in feet and inches using a comma to separate the two: 5,4

Your BMI is 35.2
Your BMI category is Obese

Press Enter to exit...
```

A standard execution resulting in an obese BMI