Name: Kenneth Thomas, Jr.
NetID: Kwt61
Github: Swagatroni

Introduction

The Body Mass Index (BMI) App project aims to provide a simple tool for calculating BMI based on weight and height inputs. BMI is a widely used measure for assessing whether a person has a healthy body weight relative to their height. By implementing this app, users can quickly determine their BMI value and understand their weight status based on established BMI categories.

This report summarizes the efforts involved in developing the BMI App, including the implementation of BMI calculation and classification functions, testing procedures, and the results obtained. Additionally, it highlights the key challenges faced during the development process and suggests potential future improvements to enhance the functionality and usability of the BMI App.

Implementation

The implementation of the BMI calculation and classification functions in the BMI App involves two main components: calculating the BMI value based on weight and height inputs, and classifying the BMI value into predefined categories (underweight, normal weight, overweight, or obese).

1. BMI Calculation Function:
    a. The `calculate_bmi` function takes two parameters: `weight` (in pounds) and `height` (in feet and inches).
    b. It converts the weight from pounds to kilograms by multiplying it by 0.45.
    c. It converts the height from feet and inches to meters by multiplying the total inches (feet * 12 + inches) by 0.025.
    d. It calculates the BMI using the formula: BMI = weight (kg) / height^2 (m^2).
    e. The function returns the calculated BMI value.

2. BMI Classification Function:
    a. The `classify` function takes the calculated BMI value as input and classifies it into one of the predefined categories.
    b. It uses a series of if-elif-else statements to determine the category based on the BMI value ranges specified in the project requirements.
    c. It prints the corresponding weight status message for the calculated BMI value.

These functions work together to calculate the BMI value and classify it into the appropriate weight status category, providing users with valuable information about their weight relative to their height.

Testing

To test the `calculate_bmi` and `classify` functions, I used the pytest framework along with the `pytest.approx` function for approximate value comparisons. Here's how I structured the tests:

1. Test Cases for `calculate_bmi` Function:
    a. I wrote test cases to check the BMI calculation with different sets of input values, including both normal and extreme values.
    b. I used `pytest.approx` to check that the calculated BMI values were approximately equal to the expected values, with a tolerance of 0.01.

2. Test Cases for `classify` Function:
    a. I wrote test cases to check the classification of BMI values into different categories (underweight, normal weight, overweight, and obese).
    b. For each category, I tested with different BMI values that fell within that category to ensure correct classification.
    c. I also tested with invalid BMI values (negative, out-of-range) to ensure that the function raised a `ValueError` with the expected error message.

3. Handling Invalid Inputs:
    a. I wrote test cases to check how the functions handled invalid input values (negative weight, zero height, non-numeric inputs).
    b. I used `pytest.raises` to check that the functions raised the correct exceptions with the expected error messages for these cases.

I ran into an issue where negative values for BMI were classified as “Underweight.” To combat this issue I added a lower bound of ‘0’ to the underweight check and added new test to ensure coverage. Overall, the testing approach focused on ensuring that the BMI calculation and classification functions behaved as expected, handling both valid and invalid inputs correctly and providing accurate results.

Boundary Testing

1. Weight and Height Boundary Testing:
    a. I chose test cases where weight and height values were at the lower and upper bounds of their valid ranges. For example, testing with a weight of 0 pounds, which is the lower bound, and testing with very large weight values to ensure the function handles them correctly.

b. Similarly, for height, I tested with height values of 0 feet and 0 inches, as well as very large height values, to cover the extremes of the input domain.

2. BMI Category Boundary Testing:
    a. I chose test cases where the calculated BMI values were just below and just above the boundaries between BMI categories (e.g., just below 18.5 for underweight, just above 24.9 for normal weight, etc.).
    b. This helps ensure that the function correctly classifies BMI values that are close to the boundaries between categories.

3. Invalid Input Boundary Testing:
    a. I also considered boundary cases for invalid inputs, such as negative weight values, zero height values, and non-numeric inputs.
    b. By testing at the boundaries of invalid input domains, I ensured that the functions raised the correct exceptions with the expected error messages for these cases.

Boundary testing is important because it helps uncover errors that are often found at the edges or boundaries of input domains, where the behavior of the function may change. By testing these boundary cases, I aimed to ensure that the BMI calculation and classification functions were robust and handled a wide range of inputs correctly.

```
def classify(val):
    if 0 <= val < 18.5:
        return "You are Underweight"
    elif 18.6 <= val <= 24.9: # Raised the lower bound by 0.1
        return "You have a Normal Weight"
    elif 25 <= val <= 29.9:
        return "You are Overweight"
    elif val >= 30:
        return "You are Obese"
    else:
        raise ValueError("Invalid BMI value")
```

By shifting the lower boundary of “Normal Weight” by 0.1 a gap in coverage is created between 18.5 and 18.6 where any BMI values in that range will be unclassified. My code succesflully caught the discrepancy due to each category having a test at each of their boundaries.

BMI Calculator User Manual for Windows Users

This user manual provides step-by-step instructions for downloading and running the BMI Calculator application on a Windows 10 computer. The BMI Calculator is a Python application that calculates Body Mass Index (BMI) based on user input for weight and height.

****Downloading the Application:****

1. Download Python: If Python is not already installed on your computer, download and install Python from the official website (https://www.python.org/downloads/). Choose the latest version compatible with your operating system (Windows).

2. Download the Application Files: Download the BMI Calculator application files (bmi_app.py) from the source where the application is hosted
   git clone https://github.com/Swagatroni/BMI-App.git
   Alternatively, you can download the zip file and extract the files in the desired location.

3. Save the Application Files: Save the downloaded bmi_app.py file to a location on your computer where you can easily access it, such as the Desktop or a specific folder.

Running the Application:

1. Open Command Prompt: Press `Win + R`, type `cmd`, and press Enter to open the Command Prompt.

2. Navigate to the Application Folder: Use the `cd` command to navigate to the folder where the bmi_app.py file is saved. For example, if the file is on your Desktop, you would use the following command:

```
cd Desktop
```

3. Run the Application: To run the BMI Calculator application, use the following command:

```
python bmi_app.py
```

4. Enter Weight and Height: Follow the on-screen instructions to enter your weight in pounds and your height in feet and inches.

5. View BMI and Classification: After entering your weight and height, the application will calculate your BMI and display it on the screen. It will also classify your BMI into one of the categories: Underweight, Normal Weight, Overweight, or Obese.

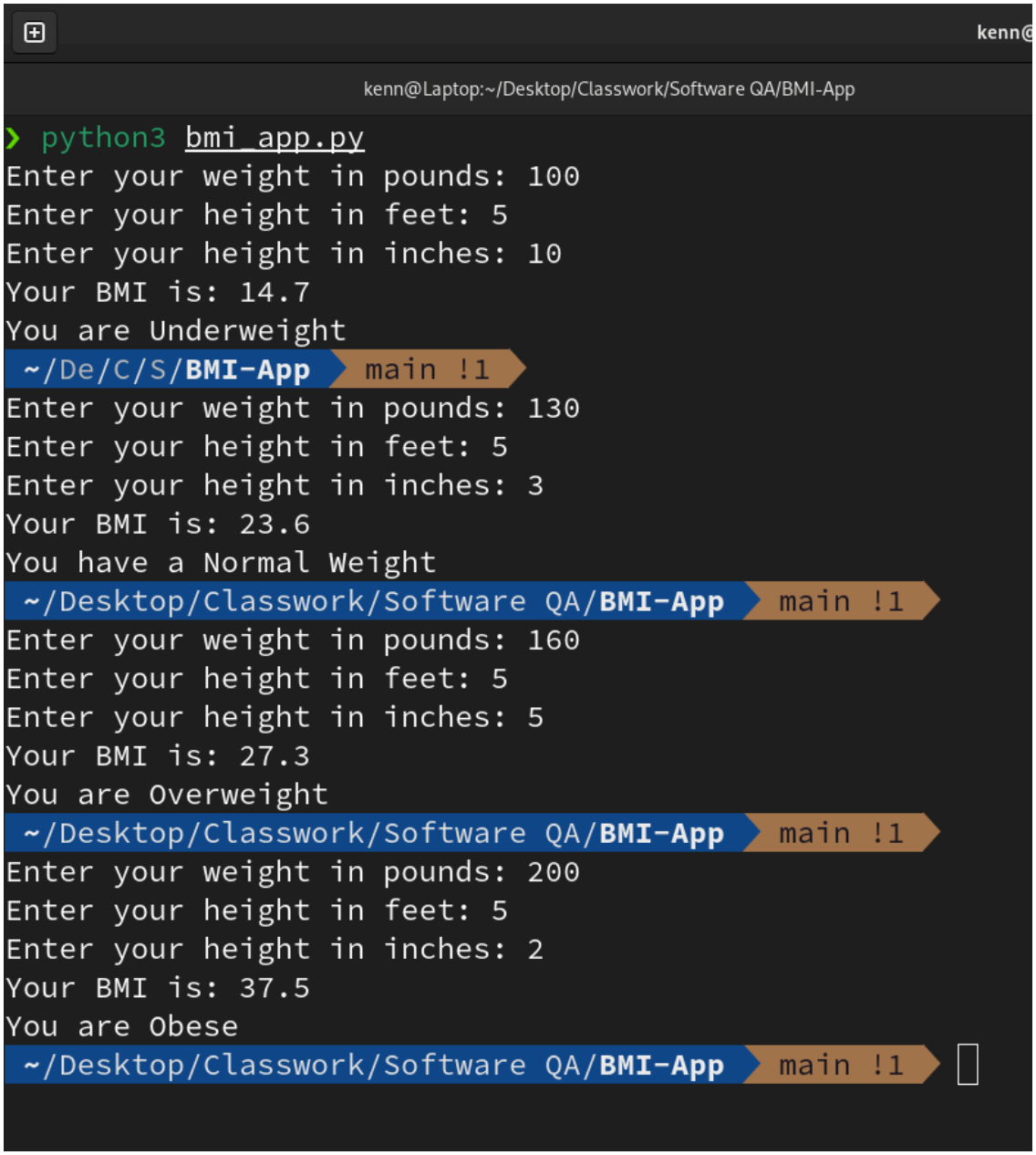
6. Exit the Application: Once you have viewed your BMI and classification, you can exit the application by closing the Command Prompt window.

Note: If you encounter any issues or errors while running the application, ensure that Python is correctly installed and that you have entered the correct commands in the Command Prompt.

Troubleshooting:

- Python Not Found: If you receive an error stating that Python is not recognized as a command, you may need to add Python to your PATH environment variable. You can do this during the installation process or manually through the Control Panel.
-
- Invalid Input Values: If you enter invalid input values (e.g., non-numeric characters), the application will raise a ValueError. Ensure that you enter numeric values for weight and height.
-
- File Not Found: If you receive a "File not found" error when trying to run the application, double-check the file path and make sure the bmi_app.py file is saved in the correct location.

By following these instructions, you can download and run the BMI Calculator application on your Windows 10 computer to calculate your BMI and determine your weight classification.


kenn@
kenn@Laptop:~/Desktop/Classwork/Software QA/BMI-App
python3 bmi_app.py
Enter your weight in pounds: 100
Enter your height in feet: 5
Enter your height in inches: 10
Your BMI is: 14.7
You are Underweight
~/De/C/S/BMI-App main !1
Enter your weight in pounds: 130
Enter your height in feet: 5
Enter your height in inches: 3
Your BMI is: 23.6
You have a Normal Weight
~/Desktop/Classwork/Software QA/BMI-App main !1
Enter your weight in pounds: 160
Enter your height in feet: 5
Enter your height in inches: 5
Your BMI is: 27.3
You are Overweight
~/Desktop/Classwork/Software QA/BMI-App main !1
Enter your weight in pounds: 200
Enter your height in feet: 5
Enter your height in inches: 2
Your BMI is: 37.5
You are Obese
~/Desktop/Classwork/Software QA/BMI-App main !1