PROJECT REPORT: Body Mass Index (BMI) Calculator

Subject: Python Programming
Project Title: Simple BMI Calculator Utility

1. Abstract

The objective of this project is to develop a software utility that calculates a user's Body Mass Index (BMI) based on weight and height inputs. The program aims to provide a quick, accessible, and automated screening tool to categorize body weight status (Underweight, Normal weight, Overweight, Obesity) according to standard medical guidelines. The solution is implemented using the Python programming language, focusing on code modularity, input validation, and user-friendly interaction.

2. Introduction

2.1 Background

Body Mass Index (BMI) is a simple index of weight-for-height that is commonly used to classify underweight, overweight, and obesity in adults. It is defined as the weight in kilograms divided by the square of the height in meters (kg/m^2).

2.2 Problem Statement

Manual calculation of BMI can be prone to arithmetic errors. Furthermore, remembering the specific threshold values for different weight categories (e.g., 18.5, 25, 30) is not convenient for the average user. There is a need for a lightweight, command-line tool that performs this calculation and interpretation instantly.

2.3 Objectives

- To accept user input for weight (kg) and height (m).
- To implement the mathematical formula for BMI.
- To strictly validate user inputs to prevent errors (e.g., negative numbers or text inputs).
- To output the calculated BMI and the corresponding health category.

3. System Analysis and Requirements

3.1 Software Requirements

- **Operating System:** Platform Independent (Windows, macOS, Linux).
- **Language:** Python 3.x.
- **Libraries:** Standard Python Input/Output libraries (no external dependencies required).

3.2 Feasibility Study

The project is highly feasible as it requires minimal computational resources and relies on basic arithmetic operations and conditional logic, which are core strengths of Python.

4. System Design

4.1 Algorithm

1. **Start** the program.
2. **Prompt** user for Weight in Kilograms.
3. **Validate** input:
    - If input is non-numeric or ≤ 0, display error and ask again.
4. **Prompt** user for Height in Meters.
5. **Validate** input:
    - If input is non-numeric or ≤ 0, display error and ask again.
6. **Calculate** BMI using formula: $BMI = Weight / (Height^2)$.
7. **Compare** BMI value against categories:
    - BMI < 18.5: Underweight
    - 18.5 $\le$ BMI < 25: Normal Weight
    - 25 $\le$ BMI < 30: Overweight
    - BMI $\ge$ 30: Obesity
8. **Display** Weight, Height, Calculated BMI, and Category.
9. **End**.

4.2 Modular Structure

The code is organized into functions to improve readability and maintainability:

- get_user_input(): Handles input and exception handling.
- calculate_bmi(): Performs the math.
- interpret_bmi(): Handles the logic for categorization.
- main(): Driver code.

5. Implementation Details

The core logic rests on the interpret_bmi function, which uses conditional statements (if-elif-else) to map the numerical BMI value to a string category.

Key Code Snippet (Logic):

```
if bmi < 18.5:
   return "Underweight"
elif 18.5 <= bmi < 25:
   return "Normal weight"
elif 25 <= bmi < 30:
```

```
    return "Overweight"
else:
    return "Obesity"
```

6. Testing and Output Results

The system was tested with various inputs to ensure accuracy and robustness.

Test Case 1: Normal Scenario

- **Input:** Weight = 70 kg, Height = 1.75 m
- **Expected Output:** BMI ~ 22.86, Category = Normal weight
- **Actual Output:** Pass

Test Case 2: Boundary Value

- **Input:** Weight = 85 kg, Height = 1.75 m
- **Calculation:** $85 / (1.75 * 1.75) = 27.75$
- **Expected Output:** Overweight
- **Actual Output:** Pass

Test Case 3: Invalid Input (Error Handling)

- **Input:** Weight = -5, Height = "abc"
- **Expected Output:** Error message prompts, program does not crash.
- **Actual Output:** Pass (Program successfully caught ValueError and logical errors).

7. Limitations and Future Scope

7.1 Limitations

- The current version only accepts Metric units (kg/m).
- It is a command-line interface (CLI) application, which may not be visually appealing to all users.

7.2 Future Scope

- **Unit Conversion:** Add support for Imperial units (pounds and inches).
- **GUI:** Develop a Graphical User Interface using Tkinter or PyQt.
- **Data Persistence:** Allow users to save their history to a file to track progress over time.

8. Conclusion

The "BMI Calculator" project successfully demonstrates the use of Python for solving real-world health calculation problems. It meets all specified objectives, providing a robust, error-free method for users to determine their BMI category. The modular design allows for

easy expansion in future versions.

End of Report