BMI Calculator Application

Cover Page

BMI CALCULATOR APPLICATION
A Health Monitoring System

- **Course:** BTech Computer Science Engineering
- **Subject:** Introduction to Problem solving using python
- **Academic Year:** 2025-26
- **Submitted By:** Shubham Raj
- **Roll Number:** 25BCE11082
- **Institution:** VIT BHOPAL
- **Submission Date:** November 2025

Introduction

Body Mass Index (BMI) is a globally accepted metric for assessing whether an individual maintains a healthy body weight relative to their height. This BMI Calculator Application is a console-based Python program that automates BMI computations and provides health risk assessments according to the latest WHO standards[1]. The application takes the user's height and weight as input, computes the BMI, and provides category-based health feedback using a reference chart.

Problem Statement

Many individuals do not have ready access to digital health tools for instant BMI calculation, leading to decreased health awareness. Manual calculation is error-prone and may cause confusion regarding health risks associated with weight. There is a clear need for an accurate, user-friendly, and accessible tool that automates BMI calculation while offering category-based health information.

Objective:
Develop a Python-based command-line BMI calculator that computes BMI values, categorizes results according to international health standards, and gives immediate health risk feedback via a simple and intuitive interface[2].

Functional Requirements

1. The system shall accept user input for height in meters (float).
2. The system shall accept user input for weight in kilograms (float).
3. The system shall validate that height is greater than zero.
4. The system shall calculate BMI using the formula: BMI = weight(kg) / height²(m²).
5. The system shall round BMI results to two decimal places.
6. The system shall display a BMI reference chart showing six categories.
7. The system shall assign each BMI to its correct category.
8. The system shall provide feedback on the user's weight status.
9. The system shall handle invalid values with error messages.

Non-functional Requirements

- **Performance:** Calculations and outputs must complete within one second.
- **Usability:** User-friendly text prompts and tables for clarity.
- **Accuracy:** Precision up to two decimal places.
- **Reliability:** Catches invalid/edge case inputs.
- **Portability:** Runs on all Python 3.x environments.
- **Maintainability:** Modular function-based code structure.
- **Error Handling:** Graceful reporting of wrong input types.

System Architecture

This application uses a **modular procedural architecture**.

- **Presentation Layer:** Displays the reference chart
- **Business Logic Layer:** Performs BMI calculation
- **Interpretation Layer:** Analyzes results and generates feedback
- **Controller Layer:** Manages program flow and user input handling

---

Design Decisions & Rationale

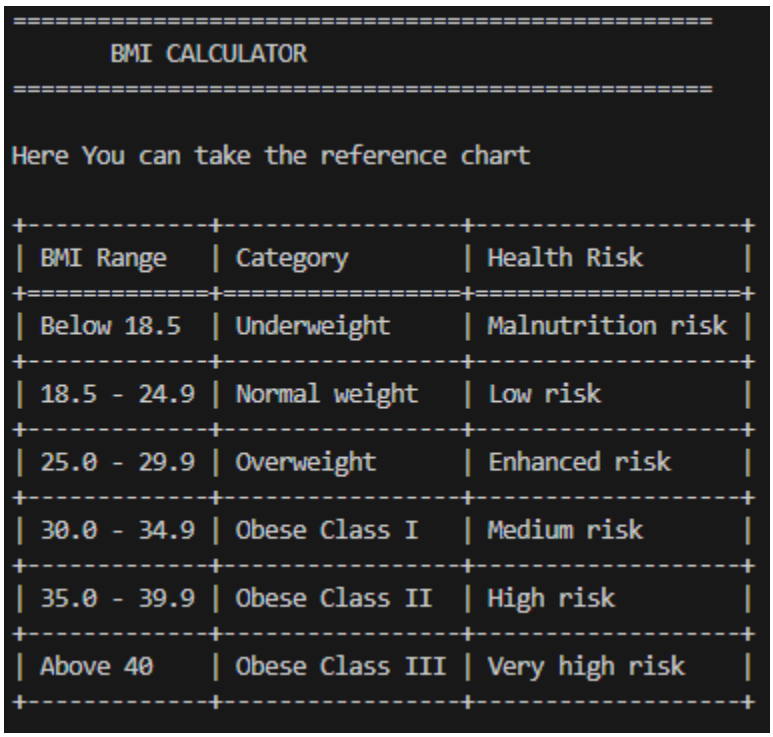
- **Modular Functions:** Each major operation is a separate function for maintainability.
- **Console Interface:** Ensures usability on any operating system without a GUI.

- **Metric Inputs:** Meters/kilograms used for international standardization.
- **Table Formatting:** Programmed dynamic width for readable reference table.
- **Error Handling:** Try-except blocks for robust handling of invalid inputs.

Implementation Details

- **Language Used:** Python 3.x
- **Editor:** VS Code / Jupyter Notebook
- **Structure:**
  - `reference_chart()` - Dynamic, formatted BMI reference chart.
  - `calculate_bmi(height, weight)` - Core BMI math.
  - `interpret_bmi(bmi)` - Returns readable assessment based on BMI.
  - `main()` - Input/output orchestration and flow control.

Screenshots / Results



```
==================================================
       BMI CALCULATOR
==================================================

Here You can take the reference chart

+-------------+-----------------+-------------------+
| BMI Range   | Category        | Health Risk       |
+=============+=================+===================+
| Below 18.5  | Underweight     | Malnutrition risk |
+-------------+-----------------+-------------------+
| 18.5 - 24.9 | Normal weight   | Low risk          |
+-------------+-----------------+-------------------+
| 25.0 - 29.9 | Overweight      | Enhanced risk     |
+-------------+-----------------+-------------------+
| 30.0 - 34.9 | Obese Class I   | Medium risk       |
+-------------+-----------------+-------------------+
| 35.0 - 39.9 | Obese Class II  | High risk         |
+-------------+-----------------+-------------------+
| Above 40    | Obese Class III | Very high risk    |
+-------------+-----------------+-------------------+
```

```
==============================================
       BMI CALCULATOR
==============================================

Here You can take the reference chart

+-------------+-----------------+-------------------+
| BMI Range   | Category        | Health Risk       |
+=============+=================+===================+
| Below 18.5  | Underweight     | Malnutrition risk |
+-------------+-----------------+-------------------+
| 18.5 - 24.9 | Normal weight   | Low risk          |
+-------------+-----------------+-------------------+
| 25.0 - 29.9 | Overweight      | Enhanced risk     |
+-------------+-----------------+-------------------+
| 30.0 - 34.9 | Obese Class I   | Medium risk       |
+-------------+-----------------+-------------------+
| 35.0 - 39.9 | Obese Class II  | High risk         |
+-------------+-----------------+-------------------+
| Above 40    | Obese Class III | Very high risk    |
+-------------+-----------------+-------------------+

Enter your height in meters: 1.72
Enter your weight in kilograms: 75

==============================================
Your BMI is 25.35, you are overweight.
==============================================
```

Testing Approach

Test Scenarios:

1. Normal Input: Height=1.75, Weight=70 → Normal weight
2. Boundary Value: Height=1.50, Weight=45 → Normal weight
3. Zero Height: Height=0, Weight=60 → Invalid input error
4. Alphabetic Input: Height=`abc`, Weight=60 → Value error handled
5. Underweight: Height=1.80, Weight=55 → Underweight
6. Obese: Height=1.65, Weight=100 → Obese Category

Tested interactively with varied inputs to ensure all requirements are met.

Challenges Faced

- Ensuring strict two-decimal rounding precision in BMI output.
- Dynamic column width calculations for reference table alignment.
- Managing invalid and unexpected inputs without program crashes.
- Correct category assignment at boundary BMI values.

Learnings & Key Takeaways

- Effective modular programming in Python with function design.
- Robust input validation and exception handling.
- Text-based interface formatting for clear presentation.
- Reinforced concepts of mathematical programming for real-world health apps.

Future Enhancements

- Add support for imperial units (inches/pounds).
- GUI version with Tkinter or web interface.
- Store BMI history over time for users.
- Provide health tips and recommendations.
- Multi-language and accessibility improvements.
- Export feature for BMI records as PDF or CSV.

References

1. VITyarthi's Python Essesntials Course - https://vityarthi.com/course/python-essentials-2
2. Python Documentations