**Problem solving discussion and documentation**

**1. Problem Identification:**

* The task was to create an application that predicts the likelihood of a heart condition based on user inputs, such as BMI, smoking status, alcohol consumption status, and physical activity status. The problem was to implement a solution that efficiently calculates BMI, categorizes it, predicts heart condition risk, and stores the data for further analysis and visualization.

**2. Problem Analysis:**

* **Key Inputs:**
  + Height (m)
  + Weight (kg)
  + Smoking Status (Yes/No)
  + Alcohol Consumption Status (Yes/No)
  + Physical Activity Status (Yes/No)
* **Key Outputs:**
  + BMI Category
  + Predicted Heart Condition Risk
  + Visualization options
* **Constraints:**
  + The application should use only Python’s standard libraries, Pandas, NumPy, and Matplotlib.
  + Data needs to be saved in CSV format.
  + The application must be user-friendly, handle edge cases, and ensure correct data validation.

**3. Strategy Formulation:**

* **Plan:**
  + Gather user inputs for health and lifestyle factors.
  + Calculate BMI and categorize it based on predefined ranges.
  + Use a risk matrix (from the provided table) to predict heart condition risk.
  + Store the data in a CSV file for future reference.
  + Provide options for data visualization to make the output more insightful.
* **Design Choices:**
  + Used Python functions to modularize code and improve readability.
  + Used Pandas for efficient data manipulation and saving data in CSV format.
  + Used Matplotlib for visualizations.
  + Implemented proper input validation to handle incorrect data input.

**4. Solution Implementation:**

* **Data Structures:**
  + Used a Pandas DataFrame to represent the heart condition risk matrix as it provides an easy way to manipulate and filter data.
  + Used lists for storing user input values temporarily before writing them to the CSV file.
  + Used CSV for data persistence, which allows easy storage and retrieval of user data.
* **Algorithm:**
  1. Prompt the user for input (height, weight, smoking status, etc.).
  2. Calculate the BMI using the formula: BMI = weight\_kg / (height\_m \*\* 2).
  3. Categorize the BMI into one of the categories: Underweight, Normal weight, Overweight, or Obesity.
  4. Predict the heart condition risk by filtering the Pandas DataFrame based on the user's input.
  5. Save the user data, along with the predicted risk, into a CSV file.
  6. Offer the user an option to visualize the collected data using Matplotlib.
* **Code Comments:**
  + Each function was thoroughly commented to explain its purpose, input parameters, and expected output.

**5. Testing and Debugging:**

* **Debugging:**
  + Encountered some issues with data validation, where invalid inputs (e.g., non-numeric height or weight) would crash the program. This was fixed by adding try-except blocks to handle input validation.
  + Handled edge cases where the combination of user inputs did not match any entry in the risk matrix by returning a default message ("Risk not found").
  + Tested the application with a variety of inputs to ensure that the predicted risk and stored data were accurate.
* **Testing:**
  + Ran multiple tests to ensure that different user inputs produced correct BMI categories and heart condition risk predictions.
  + Verified that the CSV file was correctly written to and that the saved data matched the user inputs.
  + Tested the visualization options to ensure that they were correctly displaying data from the CSV file.

**6. Solution Evaluation:**

* **Successes:**
  + The application successfully predicts the heart condition risk based on user inputs and saves the data in a CSV file.
  + It provides intuitive visualizations that help users understand the distribution of risks among different categories.
  + The modularized code structure makes it easier to maintain and update.
* **Challenges:**
  + One of the challenges was handling cases where the user input did not exactly match the data in the risk matrix. This was mitigated by clearly defining possible inputs and adding fallback conditions.
  + Another challenge was ensuring that the application remained user-friendly while incorporating robust error handling and input validation.
* **Improvements:**
  + In the future, I could enhance the application by integrating more advanced machine learning models to predict heart conditions based on larger datasets and more complex factors.