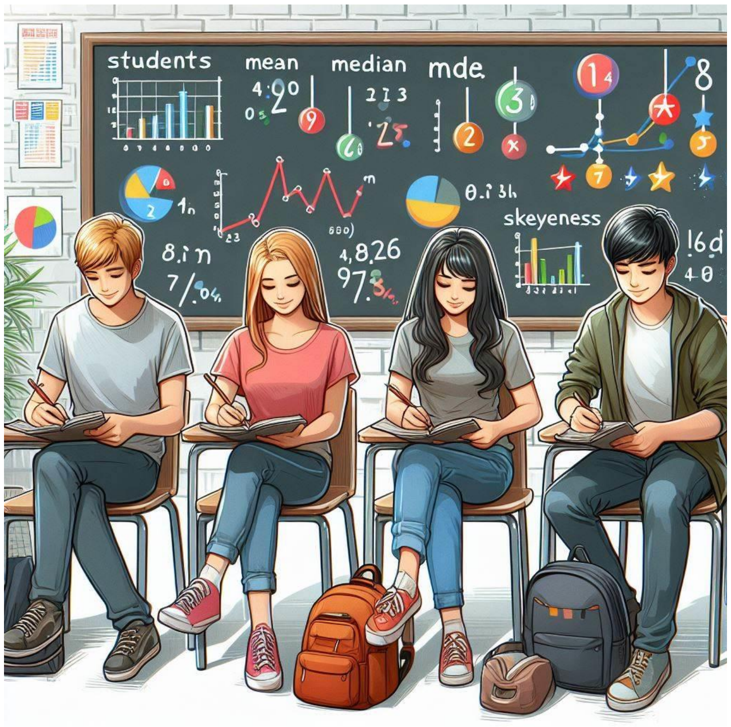
STUDENTS MARKS STATISTICS APPLICATION V 1.0

CS4051 23-24 Spring Coursework



By

Emilio Antonio Barrera Sepúlveda
22047090

London Metropolitan University
BSc Computing
CS4051 Fundamentals of Computing
LONDON
Spring 2023-24

Table of Content

1. INTRODUCTION

This student marks calculation application follows the Model-View-Controller (MVC) architectural pattern, providing a structured approach to managing data, user interfaces, and application logic. The Model (backend) handles data processing, including statistical calculations such as mean, median, mode, and skewness. The View (frontend) presents a user-friendly interface, allowing users to input marks individually or from a file and view calculated statistics. The Controller acts as an intermediary, facilitating communication between the Model and View, ensuring seamless interaction and enhancing the application's modularity and maintainability. With MVC, this program offers an organized, scalable, and efficient solution for managing and analyzing student assessment

2. Aim

The purpose of the program is to create a student marks calculation application that allows users to perform various operations on a list of marks entered by the user. The program implements the Model-View-Controller (MVC) design pattern to organize its components efficiently.

The main functionalities of the program include:

- Allowing users to enter individual marks one at a time or as a string with numbers separated by commas.

- Providing a menu with options to

    - Calculate the mean, median, mode, and skewness of the entered marks.
    - Add more numbers to the existing list of marks.
    - Load numbers from a file on the computer's hard drive.
    - Exit the application.

- Performing appropriate calculations based on user input and displaying the results.

Overall, the program aims to provide a user-friendly interface for managing and analyzing student marks data efficiently.

3. Literature Review

A comprehensive grasp of statistical measures is indispensable for comprehending data distributions. Delving into each statistical measure step by step offers profound insights into the central tendency and distribution shape of a dataset. Among the fundamental measures are the mean, median, mode, and skewness, each offering unique perspectives crucial for data analysis and interpretation.

3.1 Mean

The mean, also known as the average, represents the central value of a dataset. It is calculated by summing up all the values in the dataset and dividing the sum by the total number of values.

- Calculation
  - Add up all the values in the dataset.
  - Divide the sum by the total number of values in the dataset.

- Purpose

  Provides a measure of the central tendency of the dataset, representing the typical value.

- Sensitivity

  The mean is sensitive to extreme values or outliers, as it considers all values in the dataset equally.

3.2 Median

The median is the middle value of a dataset when the values are arranged in ascending or descending order. It divides the dataset into two equal halves.

- Calculation
  - Arrange the values in the dataset in ascending or descending order.

    - If the number of values is odd, the median is the middle value. If the number of values is even, the median is the average of the two middle values.

- Purpose

    - Provides a measure of central tendency that is not influenced by extreme values or outliers.

- Robustness

    - The median is robust to outliers, making it useful for skewed datasets.

3.3 Mode

The mode is the value that appears most frequently in a dataset.

- Calculation
    - Count the frequency of each unique value in the dataset.
    - Identify the value(s) with the highest frequency.

- Purpose
    - Indicates the most common value(s) in the dataset.

- Applicability
    - Can be applied to both numerical and categorical data.

3.4 Skewness

Skewness measures the asymmetry of the distribution of values in a dataset.

- Calculation:
  - Calculate the mean, median, and standard deviation of the dataset.
  - Use these values to compute the skewness coefficient, which quantifies the degree and direction of skewness.
- Interpretation
  - A positive skewness value indicates right skewness, meaning the distribution has a longer tail on the right side.
  - A negative skewness value indicates left skewness, meaning the distribution has a longer tail on the left side.
  - A skewness value of 0 indicates a symmetrical distribution.
- Purpose
  - Provides insight into the shape and symmetry of the dataset, helping analysts understand the distribution's characteristics.

These statistical measures are essential tools in descriptive statistics for summarizing and analyzing datasets, each offering unique perspectives on the dataset's central tendency and distributional properties. Understanding their calculations and interpretations is fundamental for data analysis and interpretation.

4. The GitHub links

https://github.com/emiliobs/CS405StudentMarksStatisticsCoursework

5. Model

The Model-View-Controller (MVC) is a software architectural pattern commonly used in the development of applications. It divides an application into three interconnected components:

5.1 Model:

- Functions are defined to calculate statistical measures such as mean, median, mode, and skewness of marks.
- A function to read data from a file and extract marks into a list is defined.

5.2 View

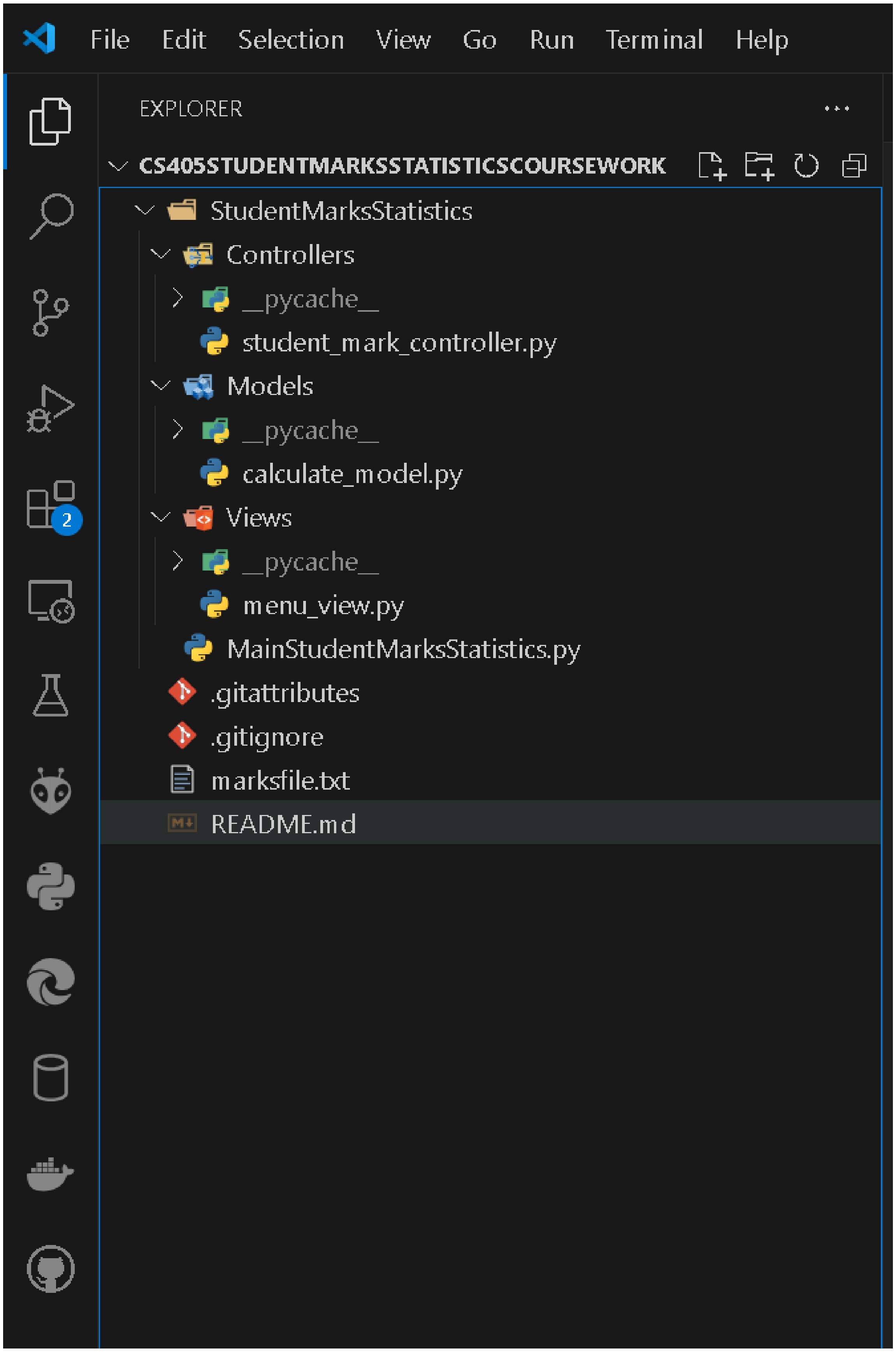
- Functions are defined to display a user interface (menu), prompt the user to input marks, and display calculated statistics or error messages.

5.3 Controller

- An empty list is initialized to store marks.
- A loop is implemented to present menu options and handle user input.
- Based on the user's choice, appropriate model functions are called, and results or error messages are displayed using view functions.
- Error handling and validation are ensured within the loop.

5.4 Main Program

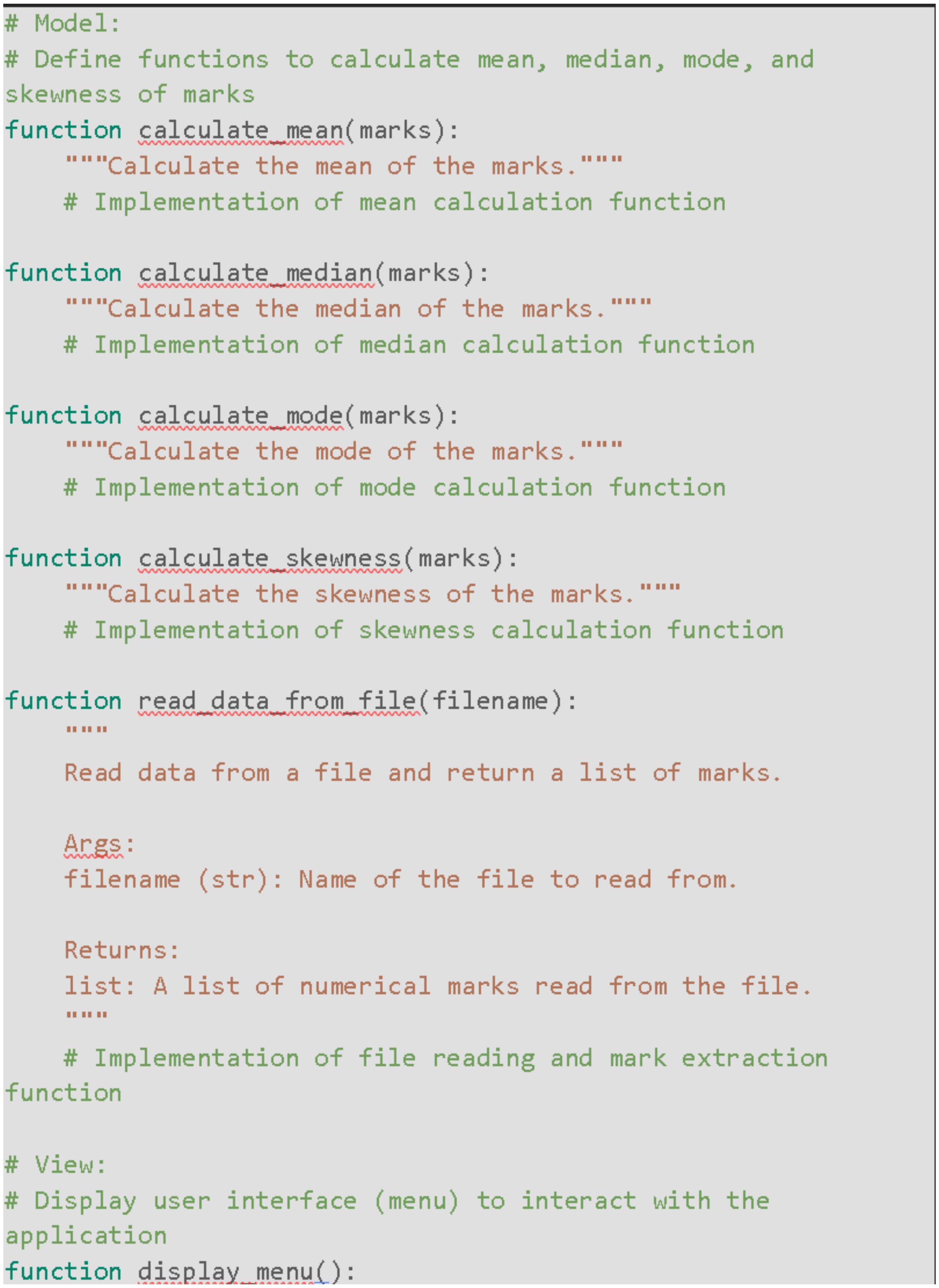
The main program instantiates the controller object and calls the main controller function to start the application.



```
CS405STUDENTMARKSSTATISTICSCOURSEWORK
  StudentMarksStatistics
    Controllers
      __pycache__
      student_mark_controller.py
    Models
      __pycache__
      calculate_model.py
    Views
      __pycache__
      menu_view.py
    MainStudentMarksStatistics.py
  .gitattributes
  .gitignore
  marksfile.txt
  README.md
```

6. Pseudocode

This pseudocode outlines the structure and functionality of a program following the Model-View-Controller (MVC) architectural pattern.



```
# Model:
# Define functions to calculate mean, median, mode, and
skewness of marks
function calculate_mean(marks):
    """Calculate the mean of the marks."""
    # Implementation of mean calculation function

function calculate_median(marks):
    """Calculate the median of the marks."""
    # Implementation of median calculation function

function calculate_mode(marks):
    """Calculate the mode of the marks."""
    # Implementation of mode calculation function

function calculate_skewness(marks):
    """Calculate the skewness of the marks."""
    # Implementation of skewness calculation function

function read_data_from_file(filename):
    """

    Read data from a file and return a list of marks.

    Args:
    filename (str): Name of the file to read from.

    Returns:
    list: A list of numerical marks read from the file.
    """

    # Implementation of file reading and mark extraction
function

# View:
# Display user interface (menu) to interact with the
application
function display_menu():
    """Display menu options for the user."""
    # Implementation of menu display function

# Prompt user to input marks individually or from a file
function prompt_user_for_marks():
    """Prompt the user to input marks."""
    # Implementation of user input prompt function

# Display calculated statistics and error messages as needed
function display_statistics(statistics):
    """Display calculated statistics."""
    # Implementation of statistics display function

function display_error_message(message):
    """Display error messages."""
    # Implementation of error message display function

# Controller:
# Initialize empty list to store marks
marks = []

# Loop to present menu options and handle user input
while True:
    # Display menu options
    display_menu()
    # Prompt user for input
    user_input = input("Enter your choice: ")
    # Handle user input
    if user_input == '1':
        # Call appropriate model functions based on user choice
        # Display results or error messages using view
functions
        pass  # Placeholder for user choice 1 handling
    elif user_input == '2':
        pass  # Placeholder for user choice 2 handling
    # Repeat for other menu options
    # Ensure appropriate error handling and validation
    # Handle exceptions, input validation, etc.

# Main Program:
# Instantiate controller object
# Call main controller function to start the application
```

7. Data (Input-Output Data Structures)

Describing input-output data structures typically involves explaining how data is formatted for input into a system or program and how it is structured for output or presentation to users. Here's how this section might be detailed

7.1 Input Data Structure

- Format
    - The input data structure for the student marks calculation application involves receiving individual student marks entered by the user.
- Structure
    - Marks are entered sequentially, with each mark provided as a separate input.
    - Each mark is expected to be a numerical value representing the score achieved by a student.
- Method of Entry
    - Users input marks one at a time via the keyboard.
    - A specific keyword (e.g., "done") is used to indicate the end of mark entry.

7.2 Output Data Structure

- Format

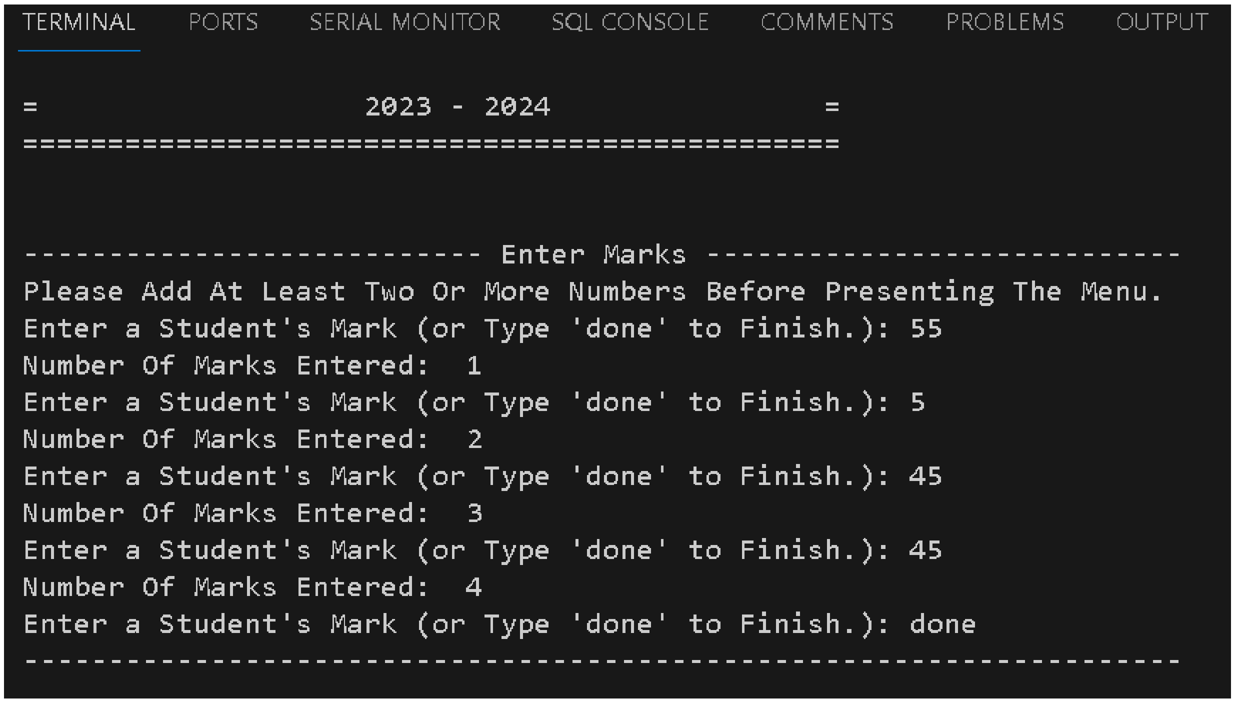
  The output data structure includes displaying the calculated mean of the entered marks to the user.

- Structure
  - The mean value is typically presented as a floating-point number.
  - Additional contextual information, such as the total number of marks entered, may also be included.

- Method of Presentation
  - The calculated mean is displayed as formatted text output on the console or terminal.
  - The output is presented in a clear and understandable manner for the user.

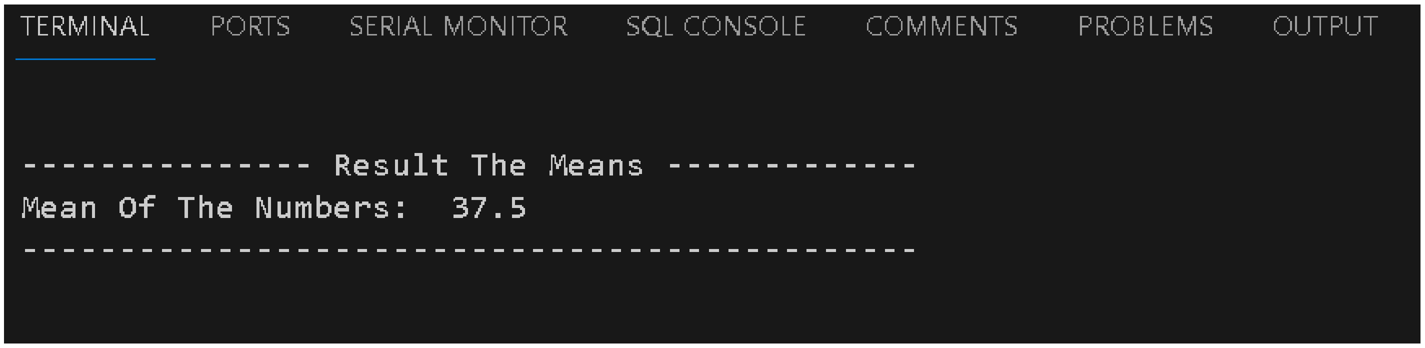
7.3 Example

- Input Data
  - User enters marks sequentially: 55, 5, 45,45 ...
  - Marks are provided as individual inputs, with "done" indicating the end of entry.



```
=                    2023 - 2024                =
================================================



--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.): 55
Number Of Marks Entered:  1
Enter a Student's Mark (or Type 'done' to Finish.): 5
Number Of Marks Entered:  2
Enter a Student's Mark (or Type 'done' to Finish.): 45
Number Of Marks Entered:  3
Enter a Student's Mark (or Type 'done' to Finish.): 45
Number Of Marks Entered:  4
Enter a Student's Mark (or Type 'done' to Finish.): done
--------------------------------------------------------------------
```

- Output Data:
  - Mean of the entered marks is calculated: 37.5
  - The mean value is displayed along with the total number of marks entered: "You have entered 3 mark(s). The mean of the entered marks is: 37.5"

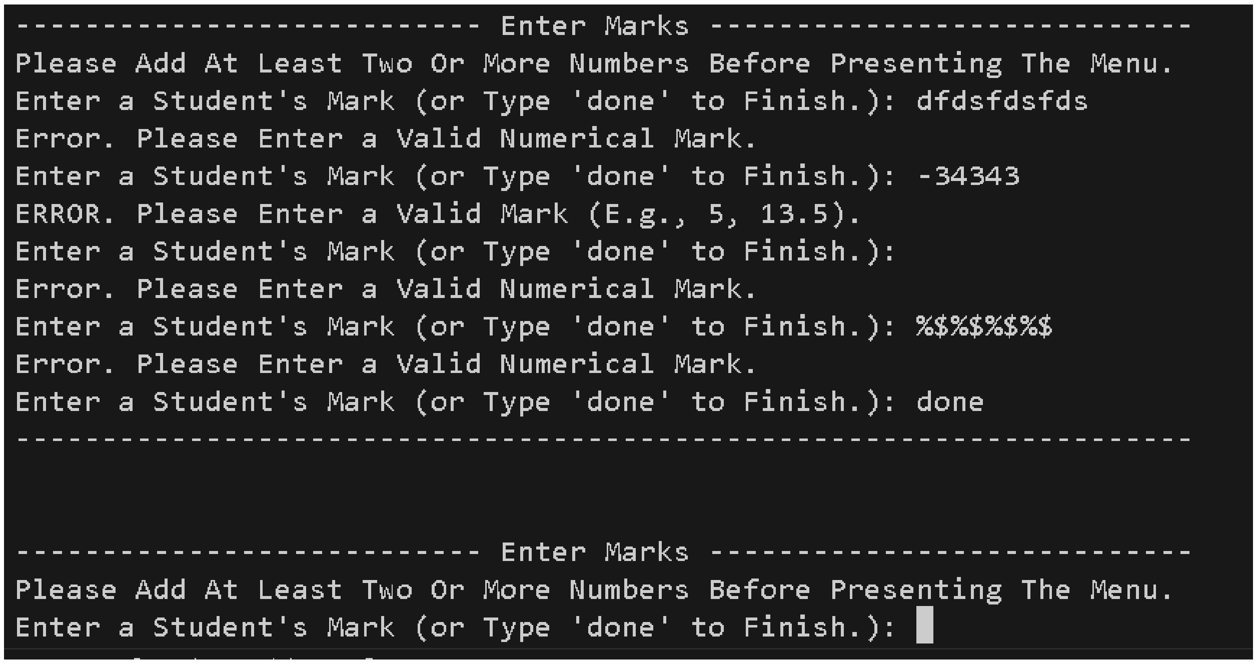


```
--------------- Result The Means -------------
Mean Of The Numbers:  37.5
----------------------------------------------
```

7.4 Data Validation

Input data may be validated to ensure it meets certain criteria (e.g., numeric format, within a specified range).

Error handling mechanisms are employed to address invalid inputs and provide feedback to the user.



```
--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.): dfdsfdsfds
Error. Please Enter a Valid Numerical Mark.
Enter a Student's Mark (or Type 'done' to Finish.): -34343
ERROR. Please Enter a Valid Mark (E.g., 5, 13.5).
Enter a Student's Mark (or Type 'done' to Finish.):
Error. Please Enter a Valid Numerical Mark.
Enter a Student's Mark (or Type 'done' to Finish.): %$%$%$%$
Error. Please Enter a Valid Numerical Mark.
Enter a Student's Mark (or Type 'done' to Finish.): done
--------------------------------------------------------------------



--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.):
```

8. Description

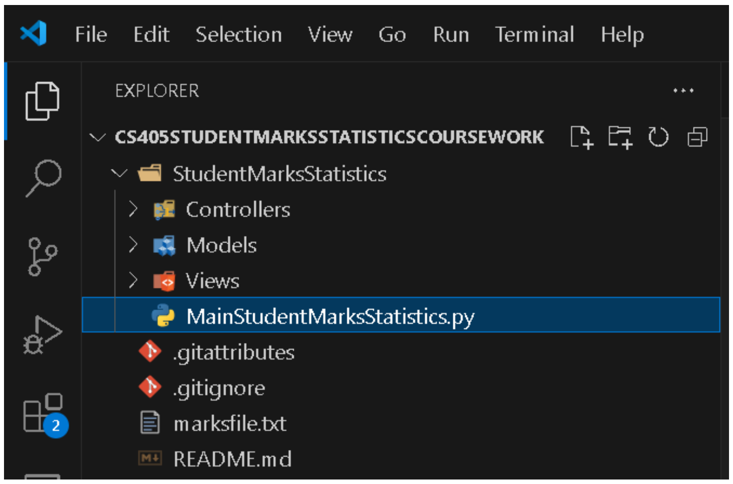
The student marks calculation application is designed to facilitate the efficient management and analysis of student marks data. It follows the Model-View-Controller (MVC) architectural pattern to maintain a clear separation of concerns and ensure modularity and maintainability. The program offers various input and output operations to enable users to interact with the system seamlessly.

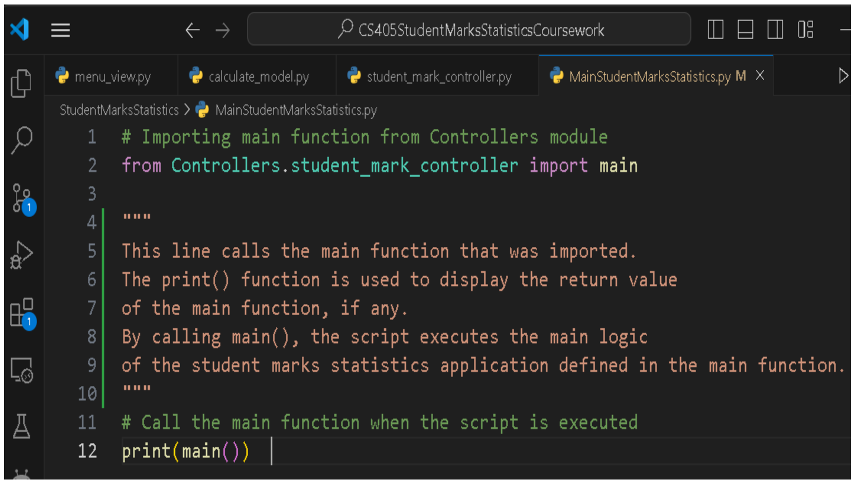
8.1 Provides an overview of the program

The program begins by prompting the user to enter student marks either individually or as a string with numbers separated by commas. The user can add as many marks as needed, and the application validates the input to ensure only numeric values are accepted. Once the marks are entered, the program presents a menu with several options for analysis and manipulation of the data.

8.1.1 Main entry point of the application

This code imports the main function from student_mark_controller in the Controllers package and immediately calls it. It acts as the entry point for executing the student marks calculation application. The main function orchestrates tasks such as initializing components, displaying the interface, processing input, performing calculations, and presenting results, ensuring seamless execution.




CS405StudentMarksStatisticsCoursework
menu_view.py
calculate_model.py
student_mark_controller.py
MainStudentMarksStatistics.py M
StudentMarksStatistics > MainStudentMarksStatistics.py

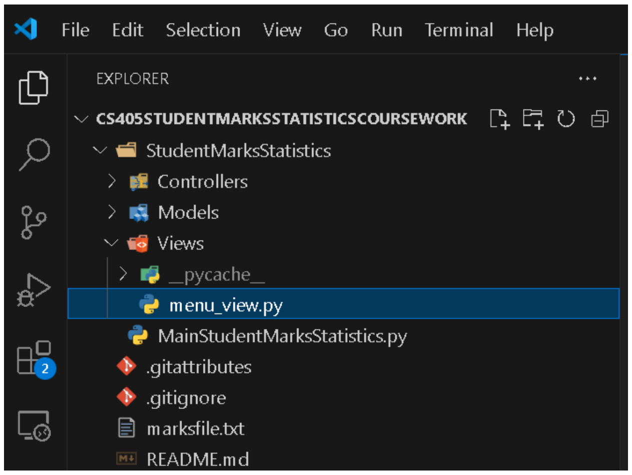
```
# Importing main function from Controllers module
from Controllers.student_mark_controller import main

"""
This line calls the main function that was imported.
The print() function is used to display the return value
of the main function, if any.
By calling main(), the script executes the main logic
of the student marks statistics application defined in the main function.
"""
# Call the main function when the script is executed
print(main())
```

8.1.2 Views

- Represents the user the Command Prompt interface and you'll find the Main Menu of the program, encompassing all the functionalities of StudentMarksStatistics program.

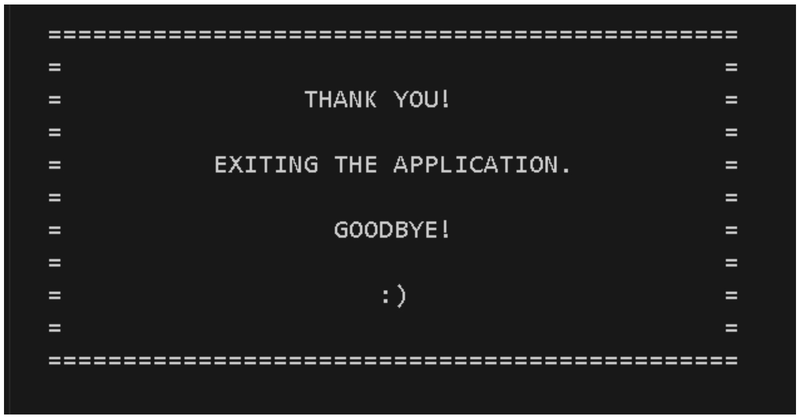
- Listens for changes in the Model and updates the UI accordingly.

- Should not contain business logic.

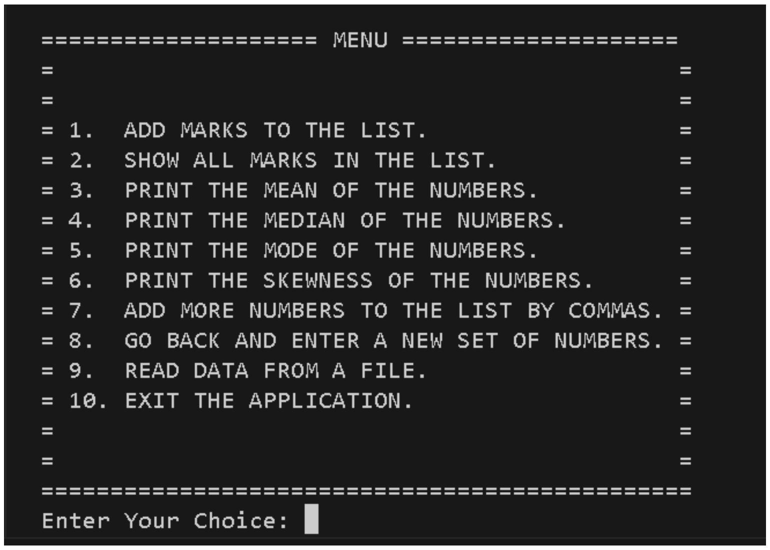


8.1.2.1 menu_view.py

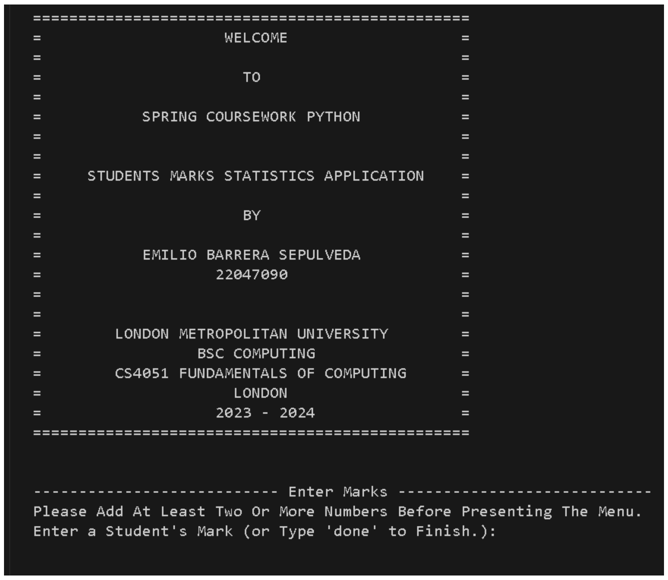
The provided code defines functions for a student marks statistics application. autor_welcome() displays program information and a welcome message. print_menu() presents menu options for the application. goodbye_user() prints a farewell message and exits the program gracefully, enhancing user interaction and experience.



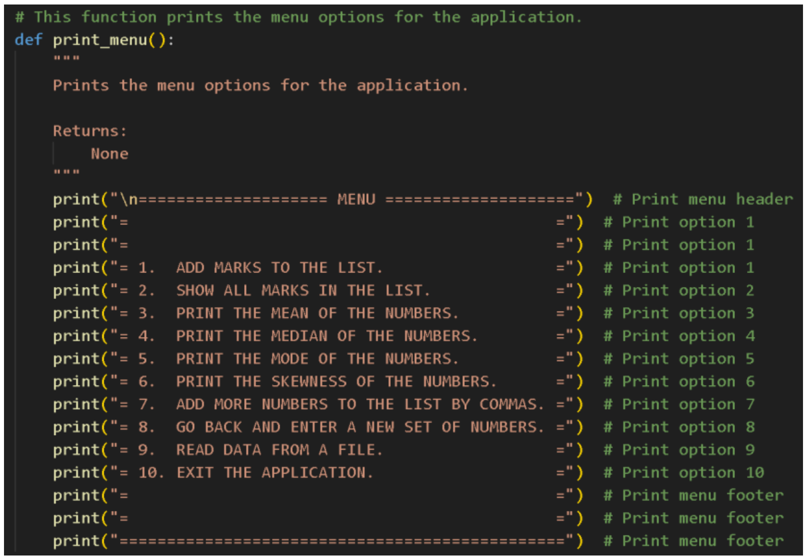
```
================================================
=                                              =
=                 THANK YOU!                   =
=                                              =
=          EXITING THE APPLICATION.            =
=                                              =
=                  GOODBYE!                    =
=                                              =
=                     :)                       =
=                                              =
================================================
```



```
==================== MENU ====================
=                                            =
=                                            =
= 1.  ADD MARKS TO THE LIST.                 =
= 2.  SHOW ALL MARKS IN THE LIST.            =
= 3.  PRINT THE MEAN OF THE NUMBERS.         =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.       =
= 5.  PRINT THE MODE OF THE NUMBERS.         =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.     =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                 =
= 10. EXIT THE APPLICATION.                  =
=                                            =
=                                            =
==============================================
Enter Your Choice: █
```



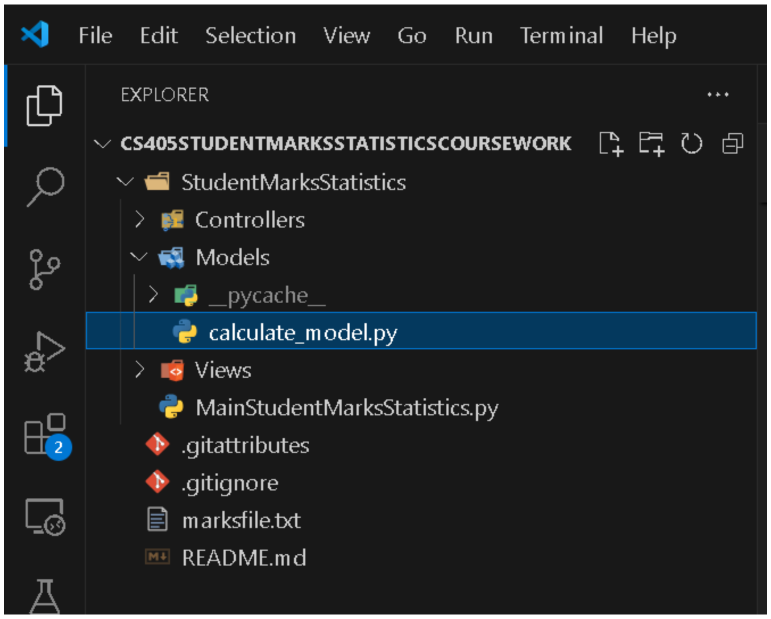
```
===============================================
=                   WELCOME                   =
=                                             =
=                     TO                      =
=                                             =
=           SPRING COURSEWORK PYTHON          =
=                                             =
=                                             =
=     STUDENTS MARKS STATISTICS APPLICATION   =
=                                             =
=                     BY                      =
=                                             =
=          EMILIO BARRERA SEPULVEDA           =
=                  22047090                   =
=                                             =
=                                             =
=        LONDON METROPOLITAN UNIVERSITY       =
=               BSC COMPUTING                 =
=        CS4051 FUNDAMENTALS OF COMPUTING     =
=                   LONDON                    =
=                 2023 - 2024                 =
===============================================


-------------------------- Enter Marks ---------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.):
```



```python
# This function prints the menu options for the application.
def print_menu():
    """
    Prints the menu options for the application.

    Returns:
        None
    """
    print("\n=================== MENU ====================")  # Print menu header
    print("=                                             =")  # Print option 1
    print("=                                             =")  # Print option 1
    print("= 1.  ADD MARKS TO THE LIST.                  =")  # Print option 1
    print("= 2.  SHOW ALL MARKS IN THE LIST.             =")  # Print option 2
    print("= 3.  PRINT THE MEAN OF THE NUMBERS.          =")  # Print option 3
    print("= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =")  # Print option 4
    print("= 5.  PRINT THE MODE OF THE NUMBERS.          =")  # Print option 5
    print("= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =")  # Print option 6
    print("= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =")  # Print option 7
    print("= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =")  # Print option 8
    print("= 9.  READ DATA FROM A FILE.                  =")  # Print option 9
    print("= 10. EXIT THE APPLICATION.                   =")  # Print option 10
    print("=                                             =")  # Print menu footer
    print("=                                             =")  # Print menu footer
    print("===============================================")  # Print menu footer
```

8.1.3 Models

- Represents the application's data and business logic.
- Manages the data, logic, and rules of the application.
- Notifies the View when the data changes.



8.1.3.1 calculate_model.py

The code deals with analyzing student marks. Here's a breakdown of the functionalities:

8.1.3.1.1 Entering Marks

Two functions exist for inputting marks:

enter_marks_to_list: Lets you enter marks one by one until you type "done". It validates each entry to ensure it's a positive number.

enter_marks_to_list_by_comma: Allows entering comma-separated marks, converting them to a list of numbers after validation (positive numbers only).

8.1.3.1.2 Displaying Marks

The Show_All_Marks function displays the number of entered marks followed by each mark on a new line.

8.1.3.1.3 Statistical Calculations

The code offers various functions to calculate statistics on the marks:

calculate_mean: Computes the average of the marks (returns 0 for an empty list).

calculate_median: Finds the middle value after sorting the marks (returns 0 for an empty list). It can handle both even and odd-numbered lists.

calculate_mode: Identifies the most frequent mark(s). It returns None if there are no marks or multiple modes with the same frequency.

calculate_standard_deviation: Calculates the standard deviation, which indicates how spread out the marks are (returns 0 for an empty list).

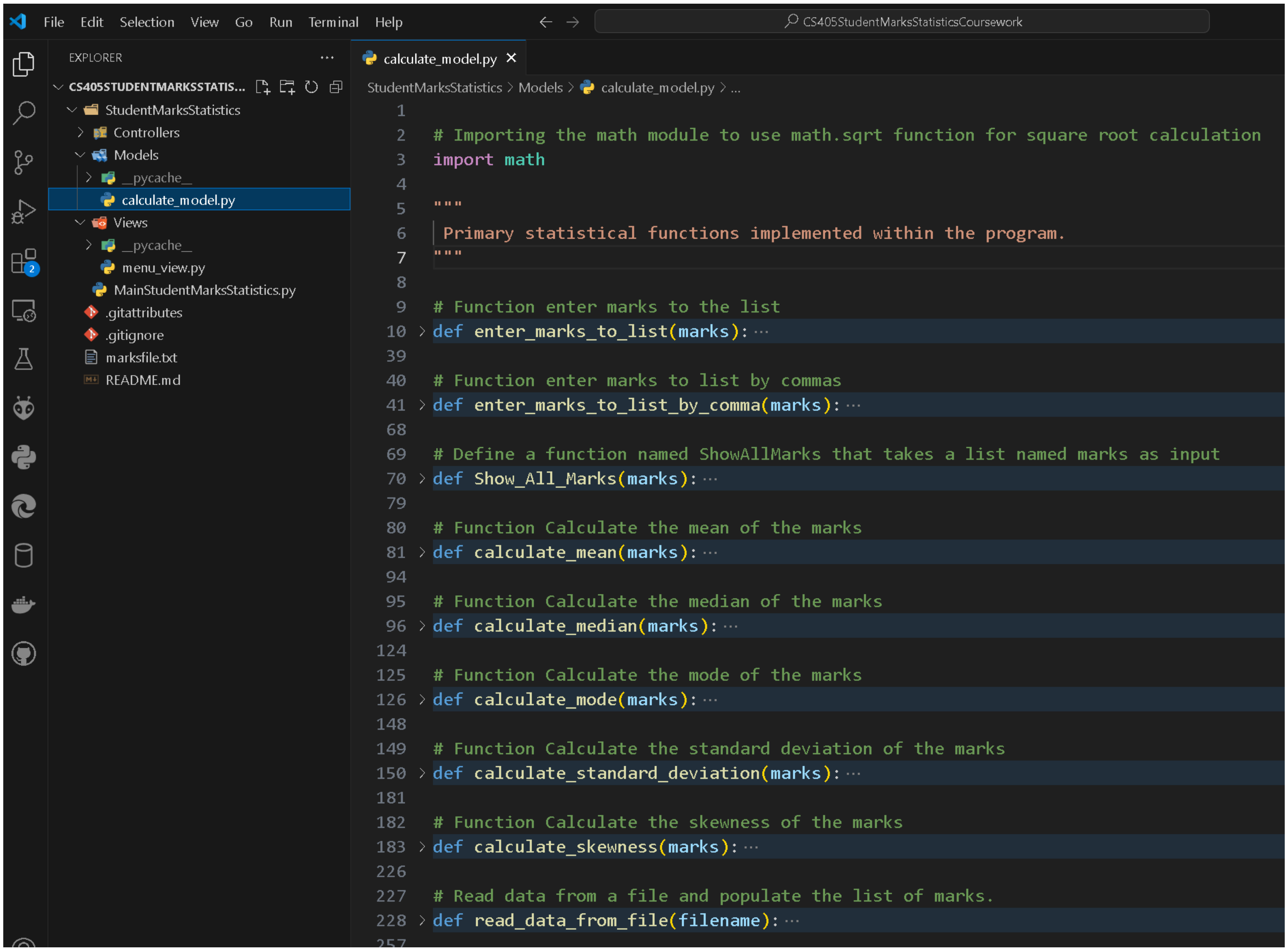
calculate_skewness: Calculates the skewness, which reflects the asymmetry of the data distribution. It requires at least 3 marks and at least one mark greater than zero. It raises an error for insufficient data or invalid data (like all marks being zero).

8.1.3.1.4 Reading Data from File

The read_data_from_file function retrieves marks from a CSV (comma-separated values) file. It returns a list of numerical marks after processing the file. The function handles errors like the file not being found or containing invalid data (non-numerical values).

In essence, this code provides a versatile toolset for managing student marks. You can enter marks, view them, and calculate various statistics to understand the overall performance and distribution of the marks.

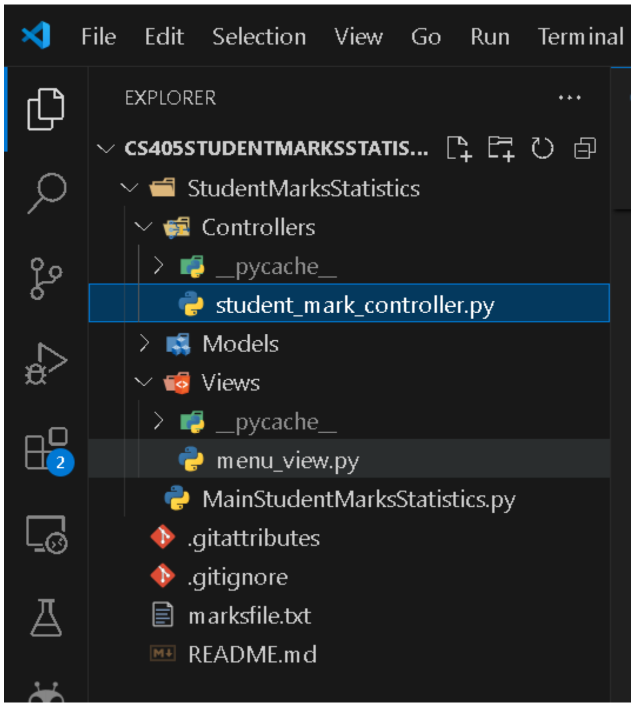
8.1.3.1.5 Code



```python

# Importing the math module to use math.sqrt function for square root calculation
import math

"""
 Primary statistical functions implemented within the program.
"""

# Function enter marks to the list
def enter_marks_to_list(marks): ···

# Function enter marks to list by commas
def enter_marks_to_list_by_comma(marks): ···

# Define a function named ShowAllMarks that takes a list named marks as input
def Show_All_Marks(marks): ···

# Function Calculate the mean of the marks
def calculate_mean(marks): ···

# Function Calculate the median of the marks
def calculate_median(marks): ···

# Function Calculate the mode of the marks
def calculate_mode(marks): ···

# Function Calculate the standard deviation of the marks
def calculate_standard_deviation(marks): ···

# Function Calculate the skewness of the marks
def calculate_skewness(marks): ···

# Read data from a file and populate the list of marks.
def read_data_from_file(filename): ···
```

8.1.4 Controllers

- Acts as an intermediary between the Model and the View.
- Receives user input from the View and updates the Model accordingly.
- Listens for changes in the Model and updates the View.



8.1.4.1 student_mark_controller.py

This Python code represents a statistical tool that interacts with users through a menu-driven interface. Here's a description of its key components and functionality:

8.1.4.1.1 Imports

The code imports necessary functions from two modules: Models.calculate_model and Views.menu_view. These functions include statistical calculations (calculate_mean, calculate_median, calculate_mode, calculate_skewness) and user interface elements (print_menu, autor_welcome, goodbye_user).

8.1.4.1.2 Main Function (main ())

The main function orchestrates the interaction between the model (statistical calculations) and the view (user interface).

It starts by displaying a welcome message and initializes an empty list named marks to store numerical data.

8.1.4.1.3 Menu Interaction

Inside a while loop, the program continuously displays a menu of options and prompts the user for input.

If the user hasn't entered at least two numerical values, the program prompts them to add more data before displaying the menu.

The menu options include functionalities such as adding marks, displaying entered marks, calculating mean, median, mode, and skewness of the data, entering marks separated by commas, clearing the data list to enter a new set of numbers, reading data from a file, and exiting the program.

8.1.4.1.4 Statistical Calculations

When the user selects options to calculate statistical measures (mean, median, mode, skewness), the program calls corresponding functions from the Models.calculate_model module.

Error handling is implemented to handle cases where insufficient data points are available for skewness calculation or when invalid data is encountered.

8.1.4.1.5 Input Validation

The program ensures that user inputs are valid and handles errors gracefully by providing informative messages.

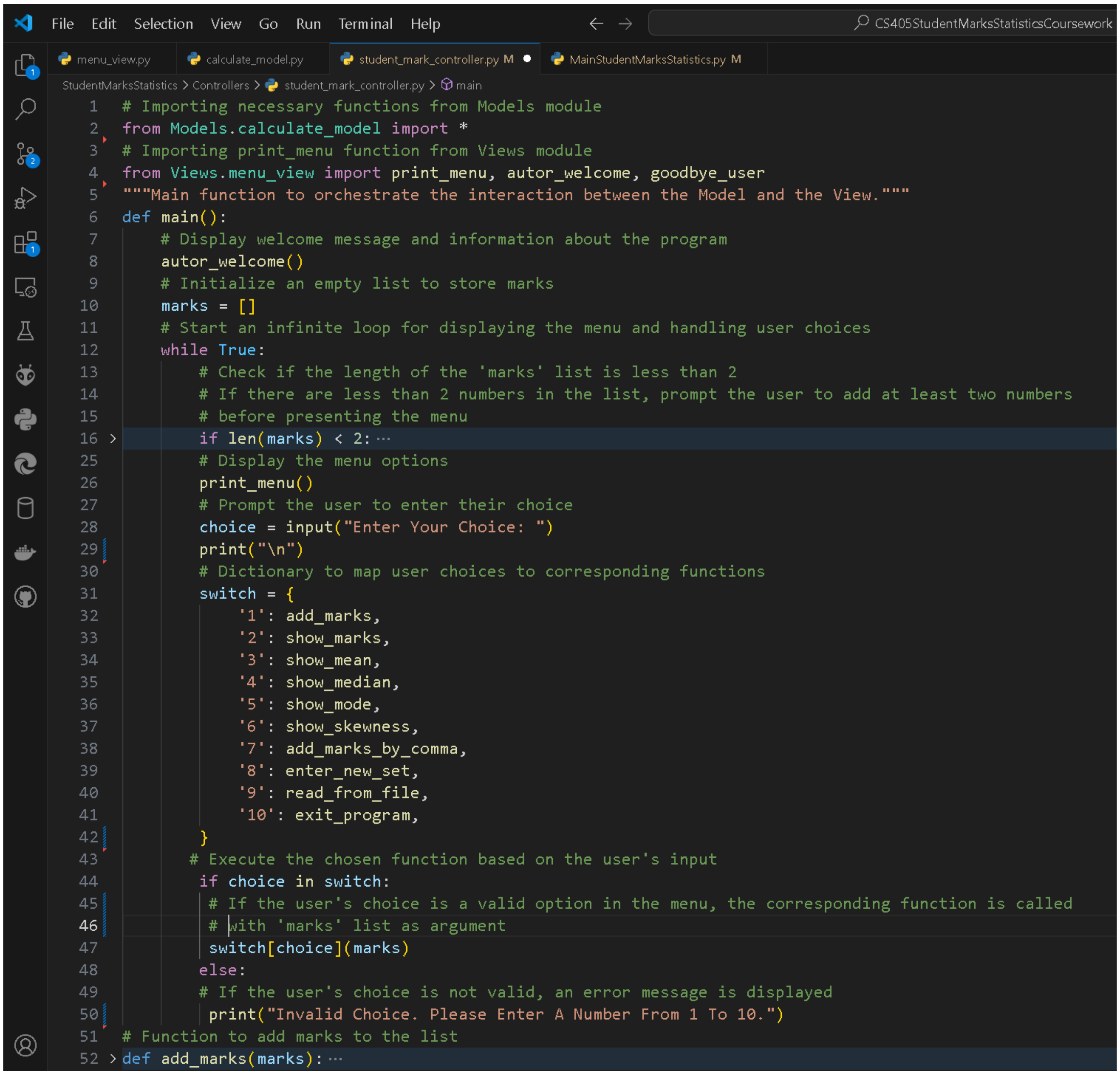
8.1.4.1.6 File Handling

Users can input numerical data from a file by selecting the appropriate menu option. The program reads the data from the file and performs operations accordingly.

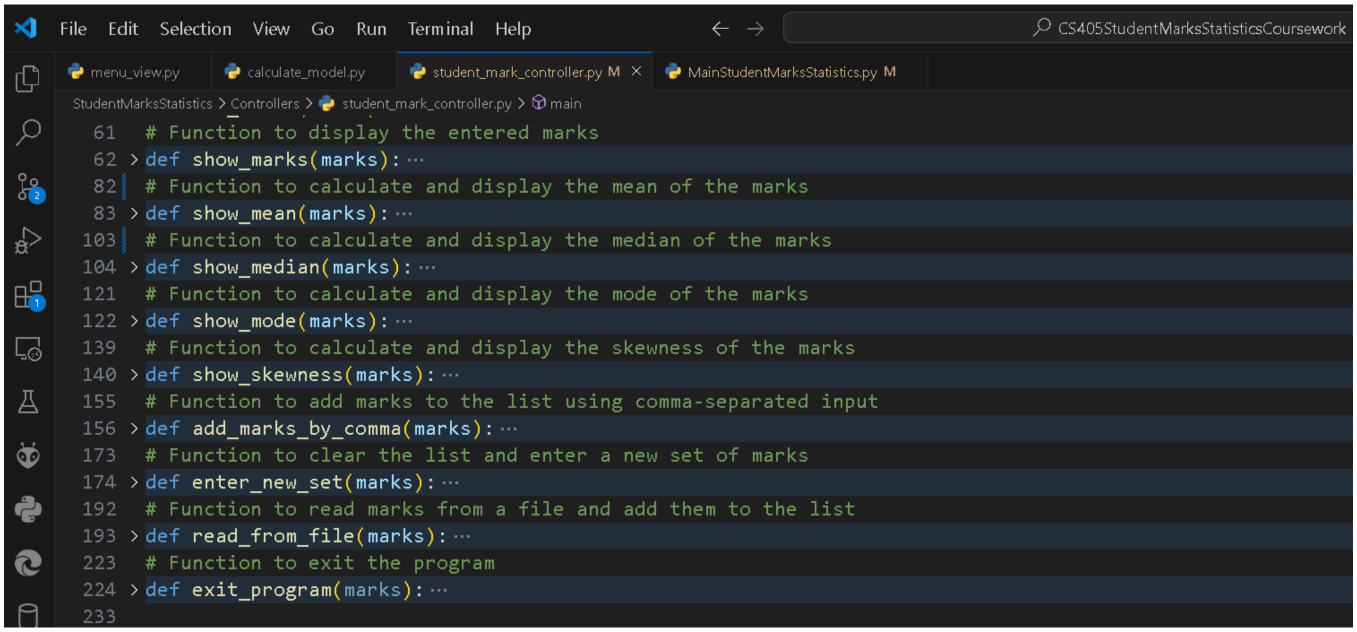
8.1.4.1.7 Exit Option

When the user chooses to exit the program, a farewell message is displayed, and the program terminates.

8.1.4.1.8 Code


File Edit Selection View Go Run Terminal Help
CS405StudentMarksStatisticsCoursework
menu_view.py
calculate_model.py
student_mark_controller.py M
MainStudentMarksStatistics.py M
StudentMarksStatistics > Controllers > student_mark_controller.py > main

```python
# Importing necessary functions from Models module
from Models.calculate_model import *
# Importing print_menu function from Views module
from Views.menu_view import print_menu, autor_welcome, goodbye_user
"""Main function to orchestrate the interaction between the Model and the View."""
def main():
    # Display welcome message and information about the program
    autor_welcome()
    # Initialize an empty list to store marks
    marks = []
    # Start an infinite loop for displaying the menu and handling user choices
    while True:
        # Check if the length of the 'marks' list is less than 2
        # If there are less than 2 numbers in the list, prompt the user to add at least two numbers
        # before presenting the menu
        if len(marks) < 2:…
        # Display the menu options
        print_menu()
        # Prompt the user to enter their choice
        choice = input("Enter Your Choice: ")
        print("\n")
        # Dictionary to map user choices to corresponding functions
        switch = {
            '1': add_marks,
            '2': show_marks,
            '3': show_mean,
            '4': show_median,
            '5': show_mode,
            '6': show_skewness,
            '7': add_marks_by_comma,
            '8': enter_new_set,
            '9': read_from_file,
            '10': exit_program,
        }
       # Execute the chosen function based on the user's input
        if choice in switch:
         # If the user's choice is a valid option in the menu, the corresponding function is called
         # with 'marks' list as argument
         switch[choice](marks)
        else:
        # If the user's choice is not valid, an error message is displayed
         print("Invalid Choice. Please Enter A Number From 1 To 10.")
# Function to add marks to the list
def add_marks(marks):…
```


File Edit Selection View Go Run Terminal Help

CS405StudentMarksStatisticsCoursework

menu_view.py | calculate_model.py | student_mark_controller.py M | MainStudentMarksStatistics.py M

StudentMarksStatistics > Controllers > student_mark_controller.py > main

```
# Function to display the entered marks
def show_marks(marks): ···
# Function to calculate and display the mean of the marks
def show_mean(marks): ···
# Function to calculate and display the median of the marks
def show_median(marks): ···
# Function to calculate and display the mode of the marks
def show_mode(marks): ···
# Function to calculate and display the skewness of the marks
def show_skewness(marks): ···
# Function to add marks to the list using comma-separated input
def add_marks_by_comma(marks): ···
# Function to clear the list and enter a new set of marks
def enter_new_set(marks): ···
# Function to read marks from a file and add them to the list
def read_from_file(marks): ···
# Function to exit the program
def exit_program(marks): ···
```

Overall, this code provides a comprehensive tool for users to input, manipulate, and analyze numerical data through a user-friendly menu interface. It demonstrates modular design principles by separating concerns between the model (statistical calculations) and the view (user interface).

8.2 Covers input and output operations

Overall, the codes encompass input operations through user interaction with the application and output operations by providing feedback and results to the user via the console.

- The main functionality of the program involves receiving input from the user through the console and providing output based on user interactions and calculations.

- The user interacts with the program by entering marks, selecting menu options, and potentially providing file names for data input.

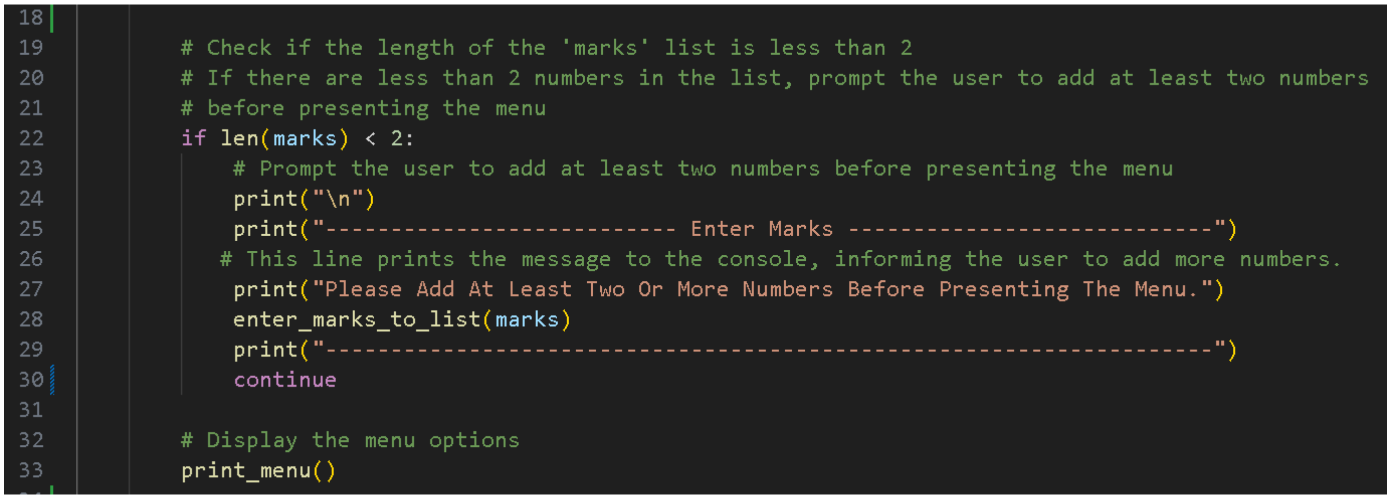
- The program then processes this input, performs calculations or file operations as required, and presents the results or feedback to the user through print statements.

8.2.1 Input Operations

The code snippet ensures at least two marks are entered before displaying menu options. Users input numerical marks using enter_marks_to_list(marks), validating each input. Once sufficient marks are entered, the menu displays. add_marks(marks) facilitate mark addition, indicating the process. Comments enhance code clarity. Likely part of a program managing student marks, users interact through inputting marks and selecting menu options. The program calculates statistics based on user inputs and displays results. Overall, the code ensures adequate data for statistical analysis, improving user experience and program functionality.

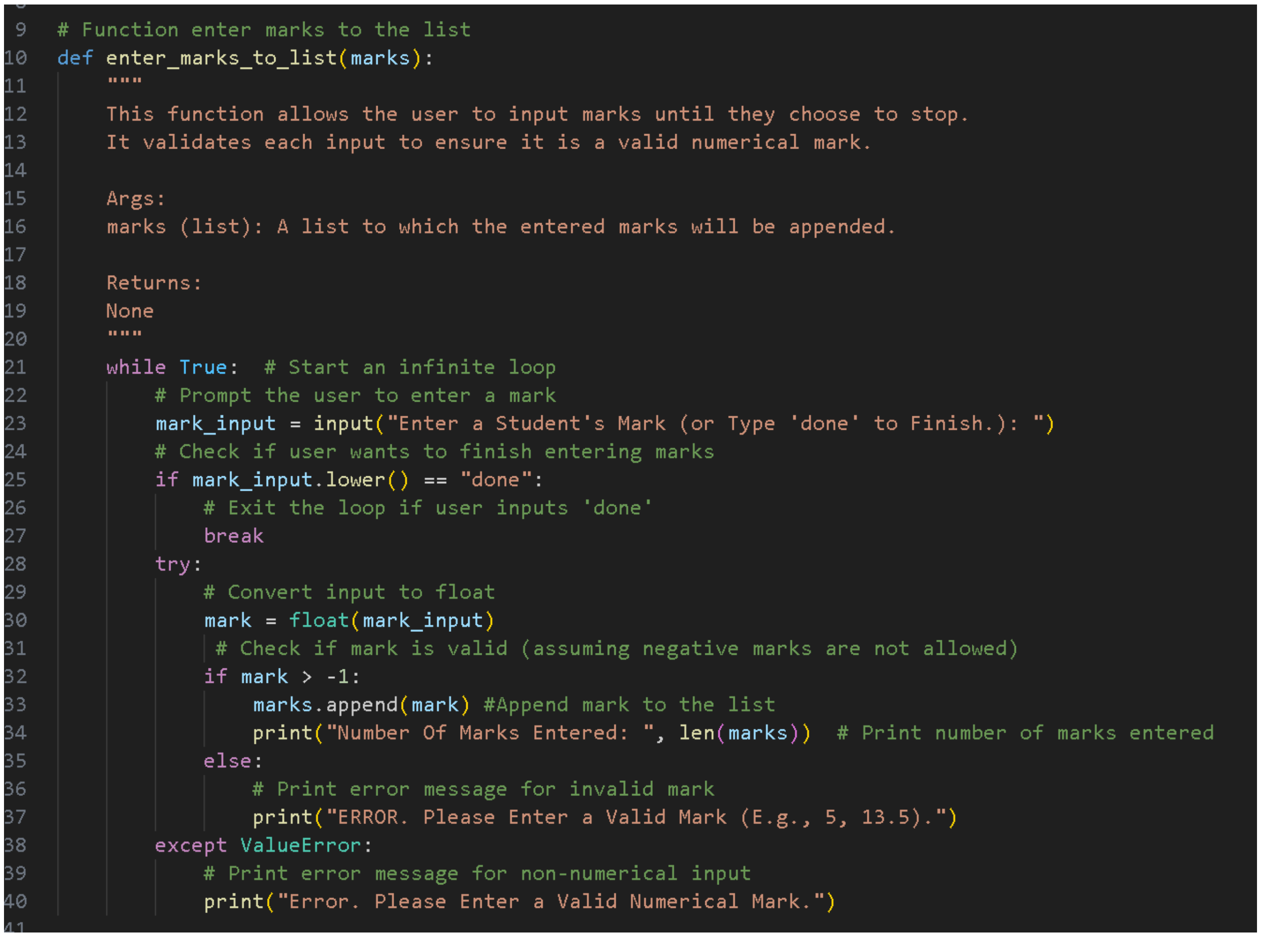
- The enter_marks_to_list(marks) function prompts the user to enter marks until they choose to stop. This function validates each input to ensure it is a valid numerical mark:

- At least two marks are entered before code



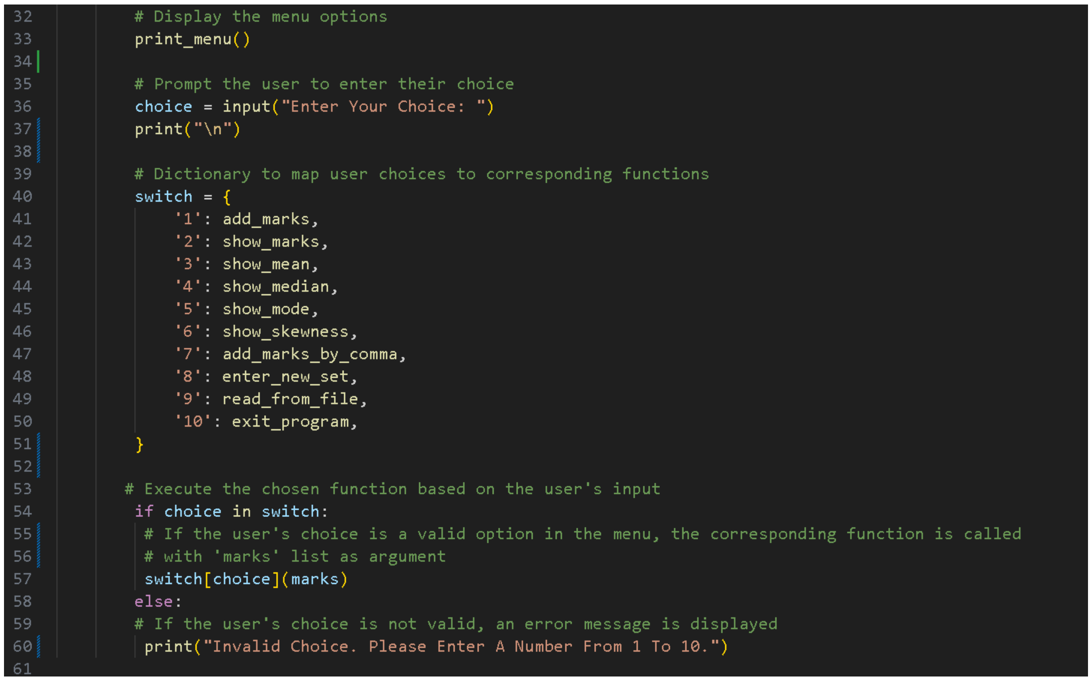
```python

    # Check if the length of the 'marks' list is less than 2
    # If there are less than 2 numbers in the list, prompt the user to add at least two numbers
    # before presenting the menu
    if len(marks) < 2:
        # Prompt the user to add at least two numbers before presenting the menu
        print("\n")
        print("--------------------------- Enter Marks ----------------------------")
       # This line prints the message to the console, informing the user to add more numbers.
        print("Please Add At Least Two Or More Numbers Before Presenting The Menu.")
        enter_marks_to_list(marks)
        print("-------------------------------------------------------------------")
        continue

    # Display the menu options
    print_menu()
```

- enter_marks_to_list(marks) code



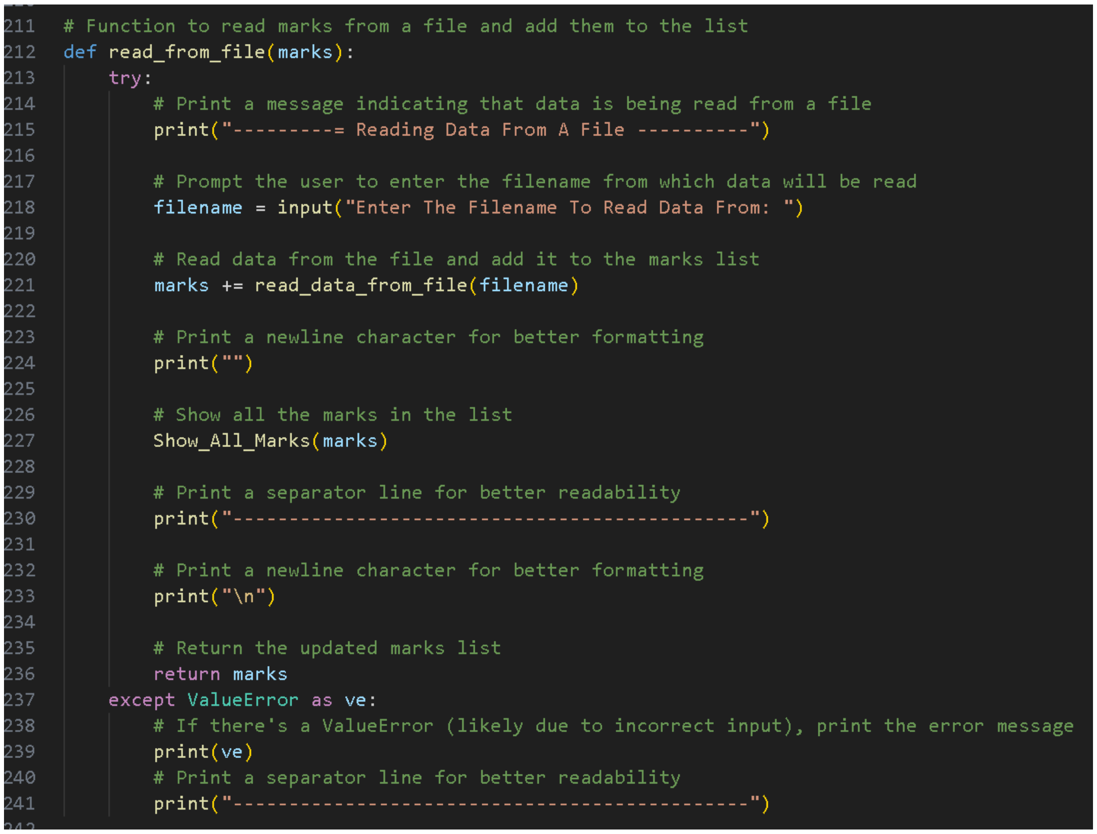
```python
# Function enter marks to the list
def enter_marks_to_list(marks):
    """
    This function allows the user to input marks until they choose to stop.
    It validates each input to ensure it is a valid numerical mark.

    Args:
    marks (list): A list to which the entered marks will be appended.

    Returns:
    None
    """
    while True:  # Start an infinite loop
        # Prompt the user to enter a mark
        mark_input = input("Enter a Student's Mark (or Type 'done' to Finish.): ")
        # Check if user wants to finish entering marks
        if mark_input.lower() == "done":
            # Exit the loop if user inputs 'done'
            break
        try:
            # Convert input to float
            mark = float(mark_input)
             # Check if mark is valid (assuming negative marks are not allowed)
            if mark > -1:
                marks.append(mark) #Append mark to the list
                print("Number Of Marks Entered: ", len(marks))  # Print number of marks entered
            else:
                # Print error message for invalid mark
                print("ERROR. Please Enter a Valid Mark (E.g., 5, 13.5).")
        except ValueError:
            # Print error message for non-numerical input
            print("Error. Please Enter a Valid Numerical Mark.")
```

- The user inputs their choice in the menu options through the input () function, prompting the user to enter their choice code.



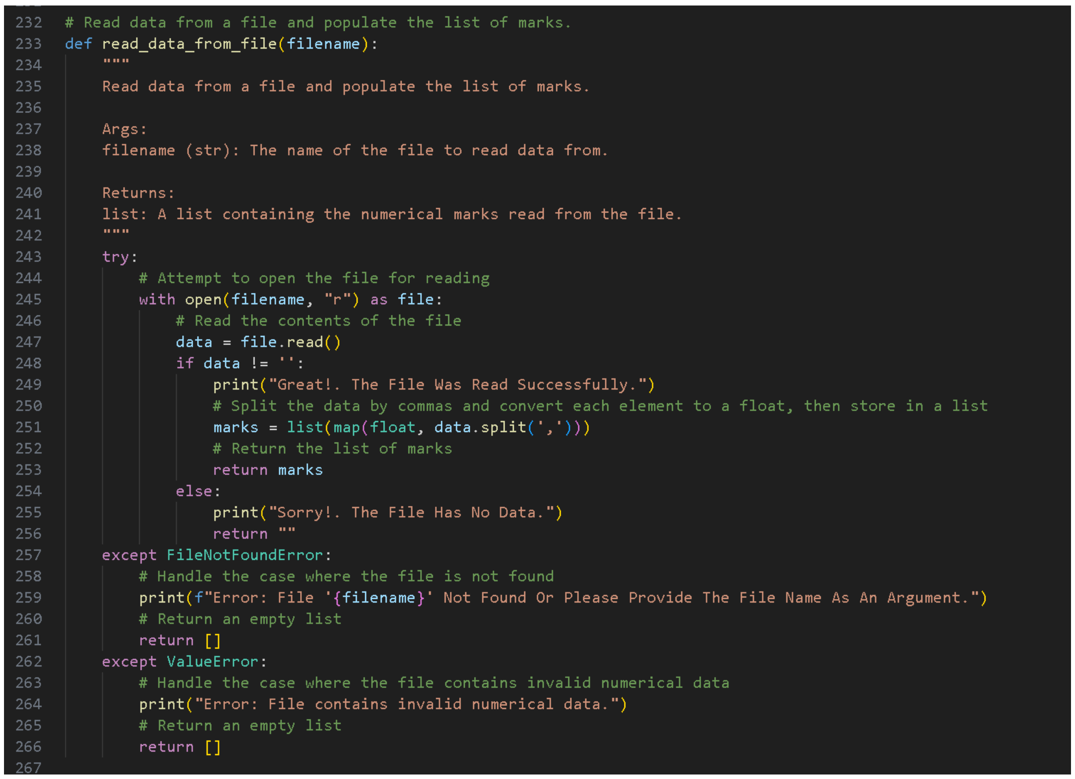
```
        # Display the menu options
        print_menu()

        # Prompt the user to enter their choice
        choice = input("Enter Your Choice: ")
        print("\n")

        # Dictionary to map user choices to corresponding functions
        switch = {
            '1': add_marks,
            '2': show_marks,
            '3': show_mean,
            '4': show_median,
            '5': show_mode,
            '6': show_skewness,
            '7': add_marks_by_comma,
            '8': enter_new_set,
            '9': read_from_file,
            '10': exit_program,
        }

       # Execute the chosen function based on the user's input
        if choice in switch:
         # If the user's choice is a valid option in the menu, the corresponding function is called
         # with 'marks' list as argument
         switch[choice](marks)
        else:
        # If the user's choice is not valid, an error message is displayed
         print("Invalid Choice. Please Enter A Number From 1 To 10.")
```

- **add_marks_by_comma(marks)**



```
# Function to read marks from a file and add them to the list
def read_from_file(marks):
    try:
        # Print a message indicating that data is being read from a file
        print("---------= Reading Data From A File ----------")

        # Prompt the user to enter the filename from which data will be read
        filename = input("Enter The Filename To Read Data From: ")

        # Read data from the file and add it to the marks list
        marks += read_data_from_file(filename)

        # Print a newline character for better formatting
        print("")

        # Show all the marks in the list
        Show_All_Marks(marks)

        # Print a separator line for better readability
        print("----------------------------------------------")

        # Print a newline character for better formatting
        print("\n")

        # Return the updated marks list
        return marks
    except ValueError as ve:
        # If there's a ValueError (likely due to incorrect input), print the error message
        print(ve)
        # Print a separator line for better readability
        print("----------------------------------------------")
```

- The read_from_file(marks) function reads data from a file, which could be considered an input operation as it retrieves data from an external source.



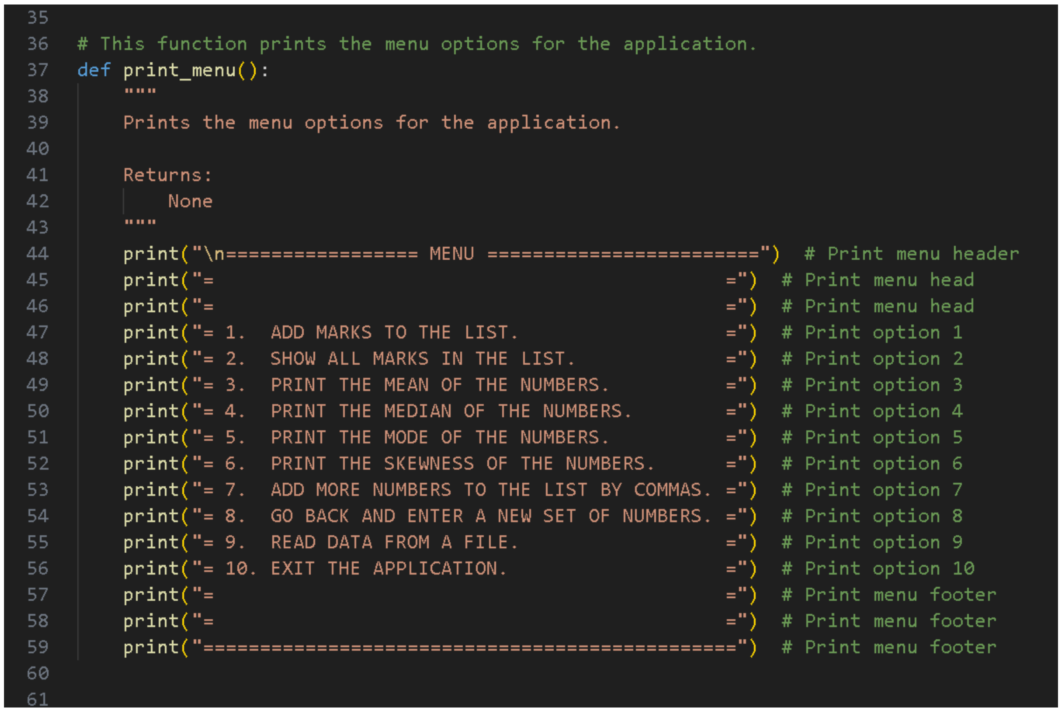
```python
# Read data from a file and populate the list of marks.
def read_data_from_file(filename):
    """
    Read data from a file and populate the list of marks.

    Args:
    filename (str): The name of the file to read data from.

    Returns:
    list: A list containing the numerical marks read from the file.
    """
    try:
        # Attempt to open the file for reading
        with open(filename, "r") as file:
            # Read the contents of the file
            data = file.read()
            if data != '':
                print("Great!. The File Was Read Successfully.")
                # Split the data by commas and convert each element to a float, then store in a list
                marks = list(map(float, data.split(',')))
                # Return the list of marks
                return marks
            else:
                print("Sorry!. The File Has No Data.")
                return ""
    except FileNotFoundError:
        # Handle the case where the file is not found
        print(f"Error: File '{filename}' Not Found Or Please Provide The File Name As An Argument.")
        # Return an empty list
        return []
    except ValueError:
        # Handle the case where the file contains invalid numerical data
        print("Error: File contains invalid numerical data.")
        # Return an empty list
        return []
```

8.2.2 Output Operations

Output operations in programming involve displaying information to the user through various means such as printing to the console, writing to files, or rendering graphical interfaces. In the context of the provided code snippet, printing to the console includes messages, prompts, and statistical results. Feedback to users includes informing them about successful mark addition, displaying error messages for invalid inputs, and providing instructions for menu interactions. Statistical results, such as mean, median, mode, and skewness, are displayed, allowing users to make informed decisions. While file output isn't shown, it's a common output operation for saving data or results. Overall, output operations facilitate user interaction and information dissemination.

- **Code the print_menu() function prints the menu options for the user to choose from.**

  This Python function print_menu() displays a menu with options to add, show, or perform calculations on numerical data, enhancing user interaction in an application.

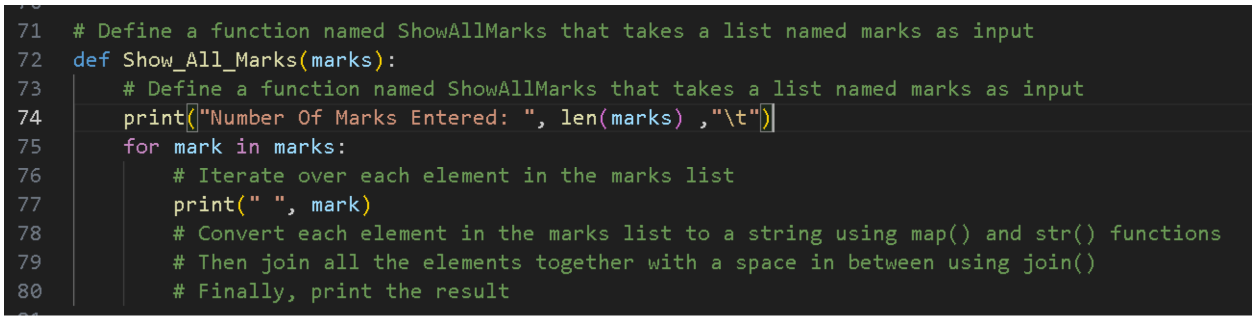


```python

# This function prints the menu options for the application.
def print_menu():
    """
    Prints the menu options for the application.

    Returns:
        None
    """
    print("\n================= MENU ========================")  # Print menu header
    print("=                                             =")  # Print menu head
    print("=                                             =")  # Print menu head
    print("= 1.  ADD MARKS TO THE LIST.                  =")  # Print option 1
    print("= 2.  SHOW ALL MARKS IN THE LIST.             =")  # Print option 2
    print("= 3.  PRINT THE MEAN OF THE NUMBERS.          =")  # Print option 3
    print("= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =")  # Print option 4
    print("= 5.  PRINT THE MODE OF THE NUMBERS.          =")  # Print option 5
    print("= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =")  # Print option 6
    print("= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =")  # Print option 7
    print("= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =")  # Print option 8
    print("= 9.  READ DATA FROM A FILE.                  =")  # Print option 9
    print("= 10. EXIT THE APPLICATION.                   =")  # Print option 10
    print("=                                             =")  # Print menu footer
    print("=                                             =")  # Print menu footer
    print("===============================================")  # Print menu footer


```

- **Various functions Statistical results such as:**

  - **Show_All_Marks(marks)**

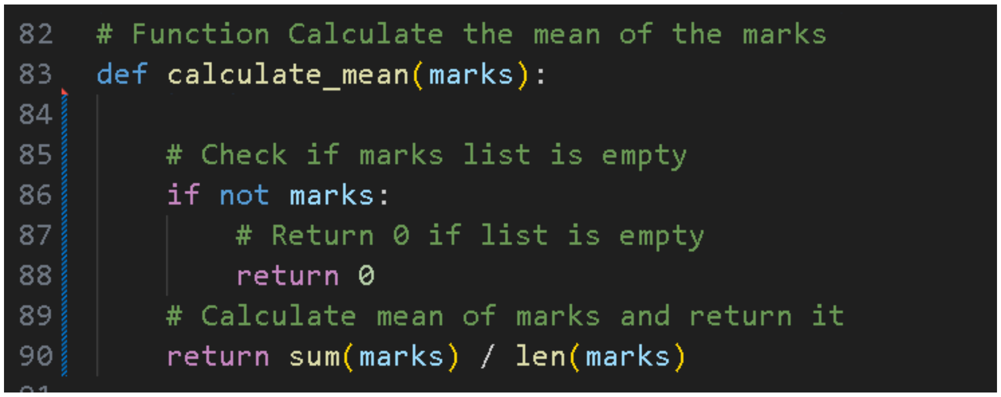
    The Show_All_Marks(marks) function takes a list named marks as input and displays the number of marks entered followed by each mark in the list. Each mark is printed on a new line.



```python
# Define a function named ShowAllMarks that takes a list named marks as input
def Show_All_Marks(marks):
    # Define a function named ShowAllMarks that takes a list named marks as input
    print("Number Of Marks Entered: ", len(marks) ,"\t")
    for mark in marks:
        # Iterate over each element in the marks list
        print(" ", mark)
        # Convert each element in the marks list to a string using map() and str() functions
        # Then join all the elements together with a space in between using join()
        # Finally, print the result
```

- **show_mean(marks).**

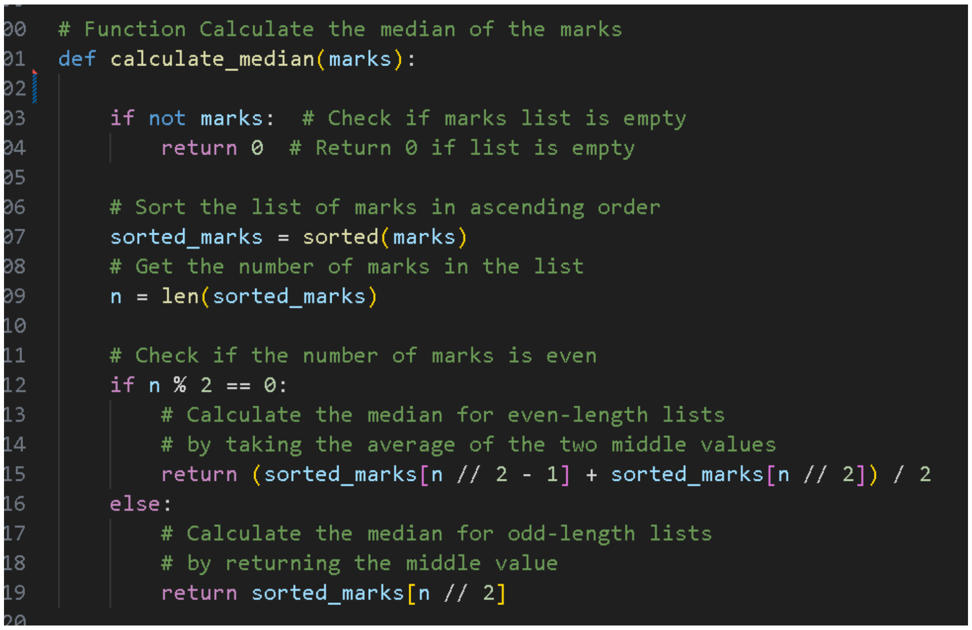
  The calculate_mean(marks) function computes the mean of the marks entered by summing up all the marks and dividing by the total number of marks. If the list of marks is empty, it returns 0.



```python
# Function Calculate the mean of the marks
def calculate_mean(marks):

    # Check if marks list is empty
    if not marks:
        # Return 0 if list is empty
        return 0
    # Calculate mean of marks and return it
    return sum(marks) / len(marks)
```

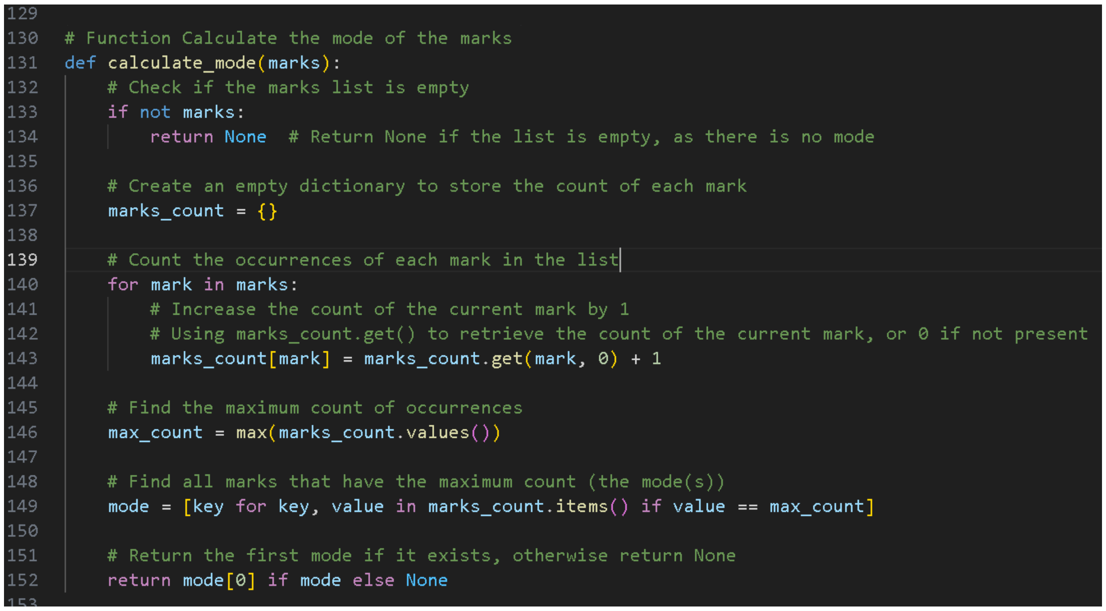
- **show_median()**

  The calculate_median(marks) function calculates the median of a list of numerical marks. It first checks if the list is empty and returns 0 if so. Then, it sorts the list of marks in ascending order and determines whether the number of marks is even or odd. For an even number of marks, it calculates the median by averaging the two middle values. For an odd number of marks, it simply returns the middle value.


```python
# Function Calculate the median of the marks
def calculate_median(marks):

    if not marks:  # Check if marks list is empty
        return 0  # Return 0 if list is empty

    # Sort the list of marks in ascending order
    sorted_marks = sorted(marks)
    # Get the number of marks in the list
    n = len(sorted_marks)

    # Check if the number of marks is even
    if n % 2 == 0:
        # Calculate the median for even-length lists
        # by taking the average of the two middle values
        return (sorted_marks[n // 2 - 1] + sorted_marks[n // 2]) / 2
    else:
        # Calculate the median for odd-length lists
        # by returning the middle value
        return sorted_marks[n // 2]
```

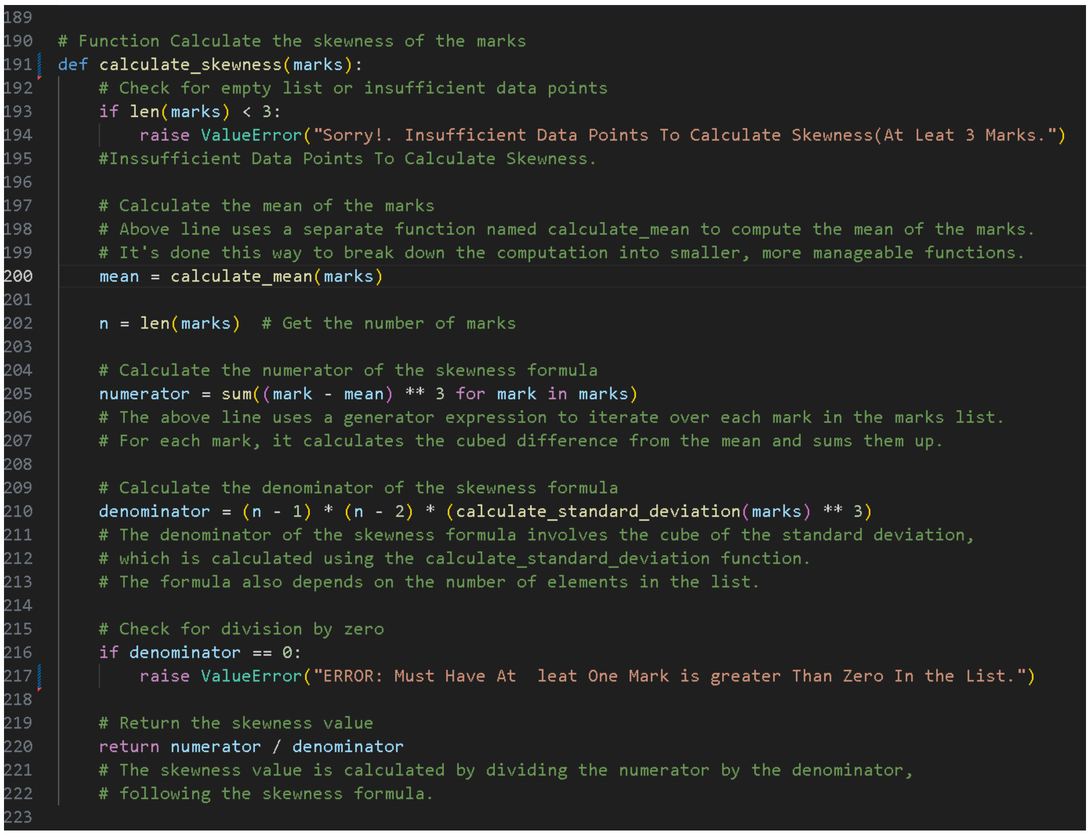
- **show_mode(marks)**

The calculate_mode(marks) function determines the mode of a list of marks, representing the value(s) that occur most frequently. It starts by creating an empty dictionary to count the occurrences of each mark. Then, it iterates through the list of marks, incrementing the count for each mark in the dictionary. After counting, it finds the maximum count of occurrences. Finally, it retrieves all marks that have this maximum count, which represents the mode(s). If multiple modes exist, it returns the first one; otherwise, it returns None if there is no mode.


```python
# Function Calculate the mode of the marks
def calculate_mode(marks):
    # Check if the marks list is empty
    if not marks:
        return None  # Return None if the list is empty, as there is no mode

    # Create an empty dictionary to store the count of each mark
    marks_count = {}

    # Count the occurrences of each mark in the list
    for mark in marks:
        # Increase the count of the current mark by 1
        # Using marks_count.get() to retrieve the count of the current mark, or 0 if not present
        marks_count[mark] = marks_count.get(mark, 0) + 1

    # Find the maximum count of occurrences
    max_count = max(marks_count.values())

    # Find all marks that have the maximum count (the mode(s))
    mode = [key for key, value in marks_count.items() if value == max_count]

    # Return the first mode if it exists, otherwise return None
    return mode[0] if mode else None
```

- **show_skewness(marks)**

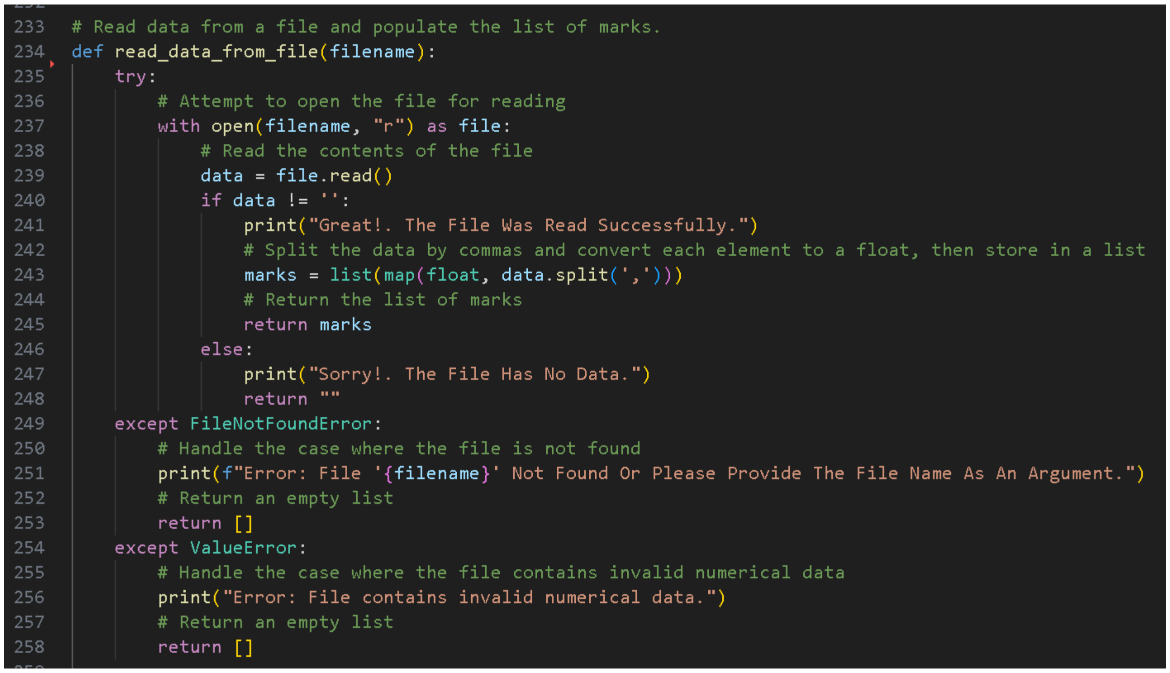
  The calculate_skewness(marks) function computes the skewness of a list of marks, which measures the asymmetry of the distribution of values. It first checks for an insufficient number of data points, throwing a ValueError if there are fewer than three marks. Then, it calculates the mean of the marks and initializes variables for the numerator and denominator of the skewness formula. The numerator involves summing the cubed differences between each mark and the mean, while the denominator involves the cube of the standard deviation. Finally, it returns the skewness value by dividing the numerator by the denominator.



```python

# Function Calculate the skewness of the marks
def calculate_skewness(marks):
    # Check for empty list or insufficient data points
    if len(marks) < 3:
        raise ValueError("Sorry!. Insufficient Data Points To Calculate Skewness(At Leat 3 Marks.")
    #Inssufficient Data Points To Calculate Skewness.

    # Calculate the mean of the marks
    # Above line uses a separate function named calculate_mean to compute the mean of the marks.
    # It's done this way to break down the computation into smaller, more manageable functions.
    mean = calculate_mean(marks)

    n = len(marks)  # Get the number of marks

    # Calculate the numerator of the skewness formula
    numerator = sum((mark - mean) ** 3 for mark in marks)
    # The above line uses a generator expression to iterate over each mark in the marks list.
    # For each mark, it calculates the cubed difference from the mean and sums them up.

    # Calculate the denominator of the skewness formula
    denominator = (n - 1) * (n - 2) * (calculate_standard_deviation(marks) ** 3)
    # The denominator of the skewness formula involves the cube of the standard deviation,
    # which is calculated using the calculate_standard_deviation function.
    # The formula also depends on the number of elements in the list.

    # Check for division by zero
    if denominator == 0:
        raise ValueError("ERROR: Must Have At  leat One Mark is greater Than Zero In the List.")

    # Return the skewness value
    return numerator / denominator
    # The skewness value is calculated by dividing the numerator by the denominator,
    # following the skewness formula.

```

- **displays read_from_file(marks)**

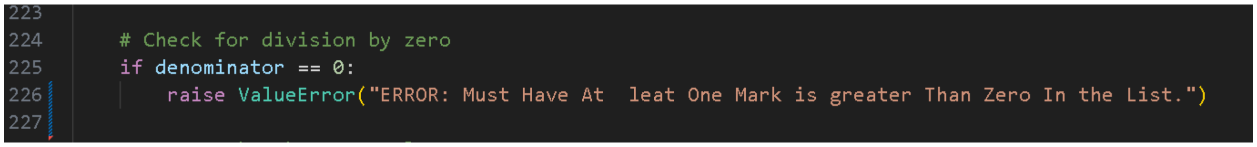
  The read_data_from_file(marks) function reads numerical data from a specified file and returns a list containing the marks. It first attempts to open the file for reading, handling the FileNotFoundError if the file does not exist. If the file exists, it reads the contents, splits them by commas, and converts each element to a float, creating a list of marks. If the file is empty, it prints a message indicating this. If the file contains invalid numerical data, it prints an error message. Finally, it returns the list of marks or an empty list if an error occurs.



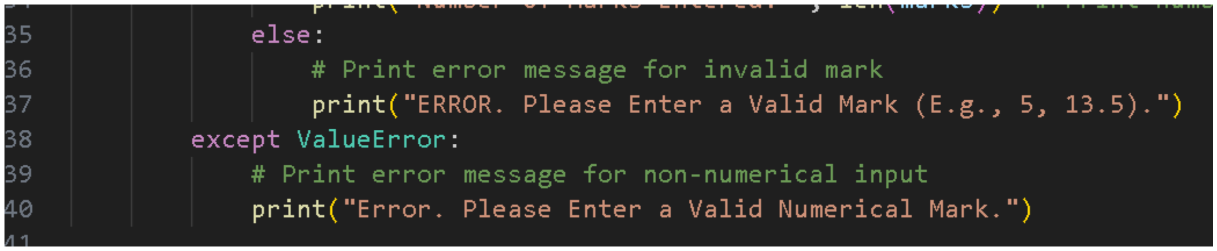
```python
# Read data from a file and populate the list of marks.
def read_data_from_file(filename):
    try:
        # Attempt to open the file for reading
        with open(filename, "r") as file:
            # Read the contents of the file
            data = file.read()
            if data != '':
                print("Great!. The File Was Read Successfully.")
                # Split the data by commas and convert each element to a float, then store in a list
                marks = list(map(float, data.split(',')))
                # Return the list of marks
                return marks
            else:
                print("Sorry!. The File Has No Data.")
                return ""
    except FileNotFoundError:
        # Handle the case where the file is not found
        print(f"Error: File '{filename}' Not Found Or Please Provide The File Name As An Argument.")
        # Return an empty list
        return []
    except ValueError:
        # Handle the case where the file contains invalid numerical data
        print("Error: File contains invalid numerical data.")
        # Return an empty list
        return []
```

- Code Error messages are printed when invalid input is provided, ensuring the user is informed about any issues.

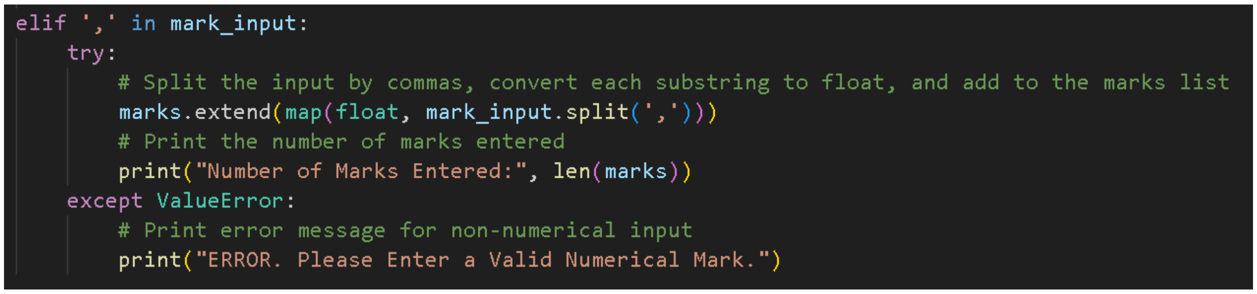
In the provided code, error handling is implemented to ensure that users are informed about any issues with the input they provide. This includes scenarios such as entering non-numerical values when numerical input is expected, providing invalid file names, or encountering file reading errors. By printing descriptive error messages, such as "Please Enter a Valid Numerical Mark" or "File Not Found," users can easily understand and address the issues encountered during the execution of the program. This approach enhances the user experience and helps in troubleshooting potential errors effectively.



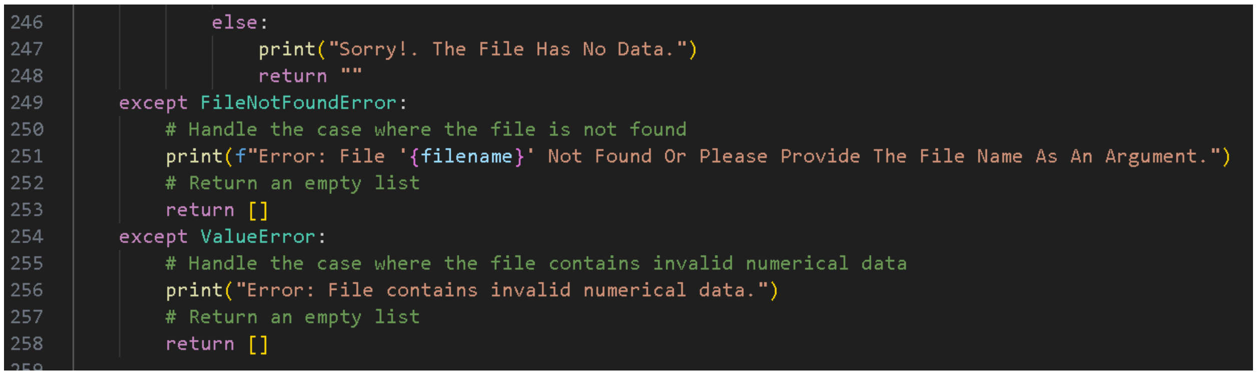
```
    # Check for division by zero
    if denominator == 0:
        raise ValueError("ERROR: Must Have At  leat One Mark is greater Than Zero In the List.")
```



```
            else:
                # Print error message for invalid mark
                print("ERROR. Please Enter a Valid Mark (E.g., 5, 13.5).")
        except ValueError:
            # Print error message for non-numerical input
            print("Error. Please Enter a Valid Numerical Mark.")
```



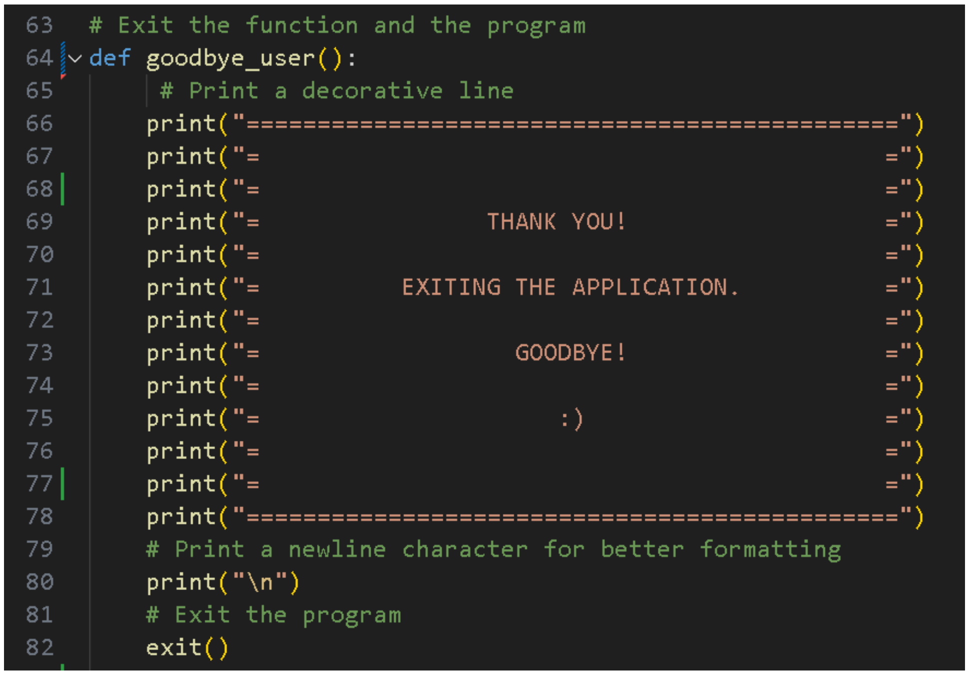
```
elif ',' in mark_input:
    try:
        # Split the input by commas, convert each substring to float, and add to the marks list
        marks.extend(map(float, mark_input.split(',')))
        # Print the number of marks entered
        print("Number of Marks Entered:", len(marks))
    except ValueError:
        # Print error message for non-numerical input
        print("ERROR. Please Enter a Valid Numerical Mark.")
```



```
            else:
                print("Sorry!. The File Has No Data.")
                return ""
    except FileNotFoundError:
        # Handle the case where the file is not found
        print(f"Error: File '{filename}' Not Found Or Please Provide The File Name As An Argument.")
        # Return an empty list
        return []
    except ValueError:
        # Handle the case where the file contains invalid numerical data
        print("Error: File contains invalid numerical data.")
        # Return an empty list
        return []
```

- Code the goodbye_user() function prints a farewell message when the user exits the application.

  The provided code defines a function called goodbye_user() which prints a farewell message and exits the application. The function displays a decorative line followed by a message thanking the user and indicating the program's termination. It then prints a smiley face for a friendly touch. Finally, it exits the program using the exit() function. This function enhances the user experience by providing a clear indication of program termination and expresses gratitude towards the user for their interaction.



```python
# Exit the function and the program
def goodbye_user():
     # Print a decorative line
    print("==============================================")
    print("=                                            =")
    print("=                                            =")
    print("=                THANK YOU!                  =")
    print("=                                            =")
    print("=          EXITING THE APPLICATION.          =")
    print("=                                            =")
    print("=                  GOODBYE!                  =")
    print("=                                            =")
    print("=                     :)                     =")
    print("=                                            =")
    print("=                                            =")
    print("==============================================")
    # Print a newline character for better formatting
    print("\n")
    # Exit the program
    exit()
```

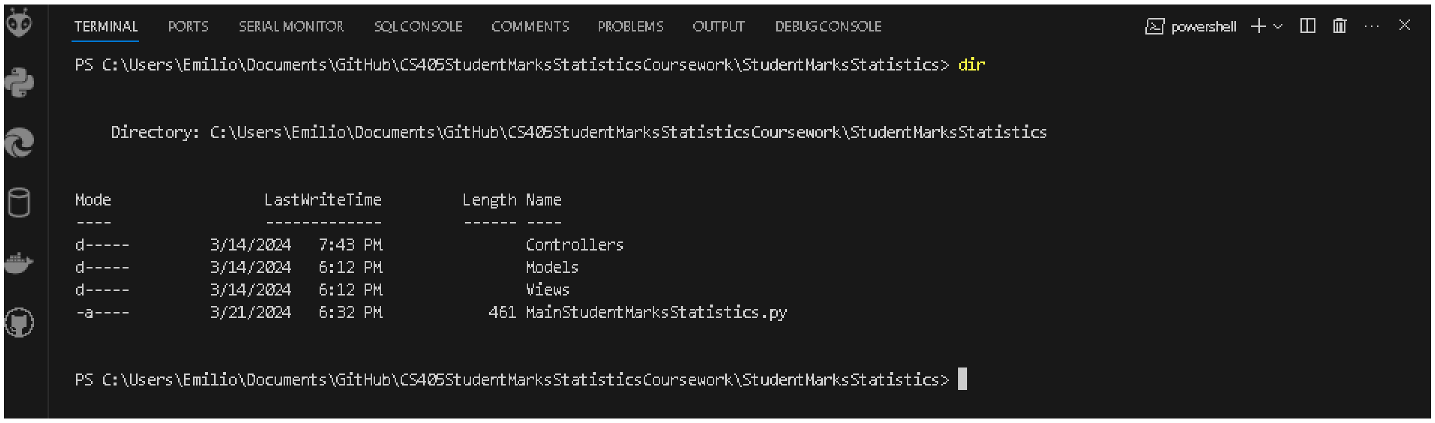
9. Testing

The testing plan for the Students Mark Statistics Application encompasses various tests to ensure functionality and usability. It includes unit testing for individual functions, integration testing to verify interactions, and user interface testing for ease of use. Boundary testing evaluates edge cases, error handling ensures proper responses to invalid input, and file I/O testing confirms data handling. Usability testing gathers user experience feedback, regression testing checks for unintended effects, and documentation testing verifies instructions. Through screenshots and descriptions in the Python prompt interface, all tests will be documented meticulously to ensure comprehensive validation and user comprehension, fostering correctness, reliability, and user satisfaction.

9.1 Test 1: Running my program from Terminal

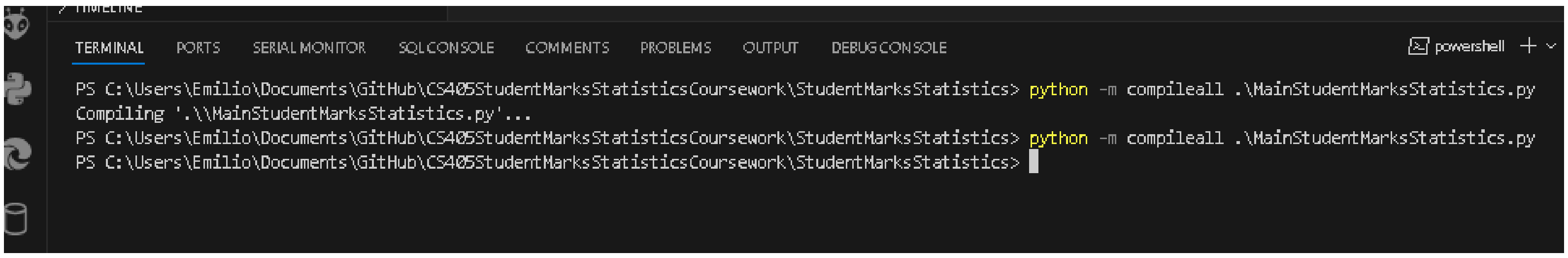
Test that the program can be compiled and run using the command prompt, including a screenshot similar to Figure 1 in the command prompt learning aid.

9.1.1 My first step was changing the directory to the location of my project.


TERMINAL PORTS SERIAL MONITOR SQL CONSOLE COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE
powershell

```
PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics> dir


    Directory: C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/14/2024   7:43 PM                Controllers
d-----         3/14/2024   6:12 PM                Models
d-----         3/14/2024   6:12 PM                Views
-a----         3/21/2024   6:32 PM            461 MainStudentMarksStatistics.py


PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics>
```

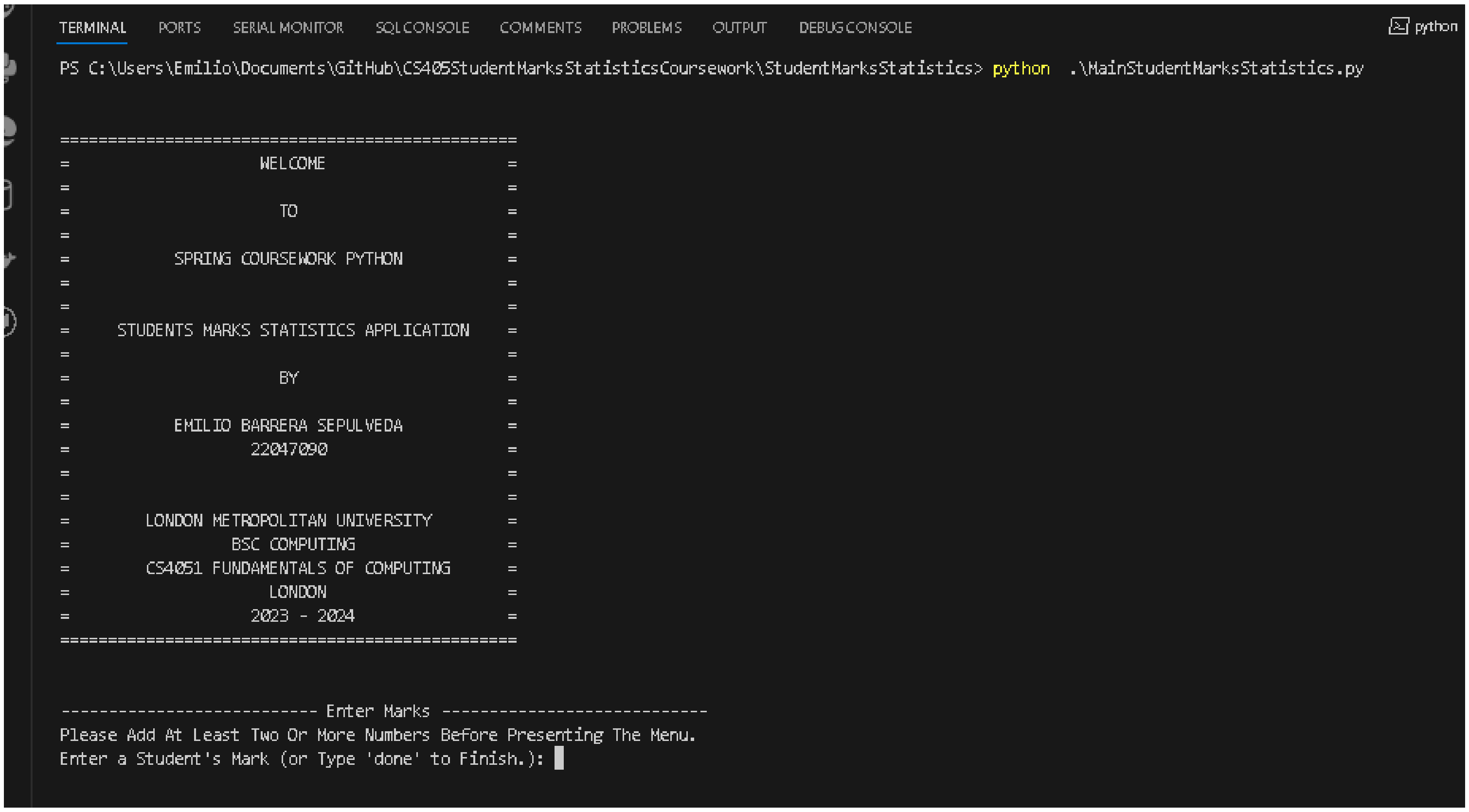
9.1.2 My next step was compiling my main Python file.


TERMINAL PORTS SERIAL MONITOR SQL CONSOLE COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE
powershell

```
PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics> python -m compileall .\MainStudentMarksStatistics.py
Compiling '.\\MainStudentMarksStatistics.py'...
PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics> python -m compileall .\MainStudentMarksStatistics.py
PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics>
```

9.1.3 My final step was running the file.

Program ran successfully.

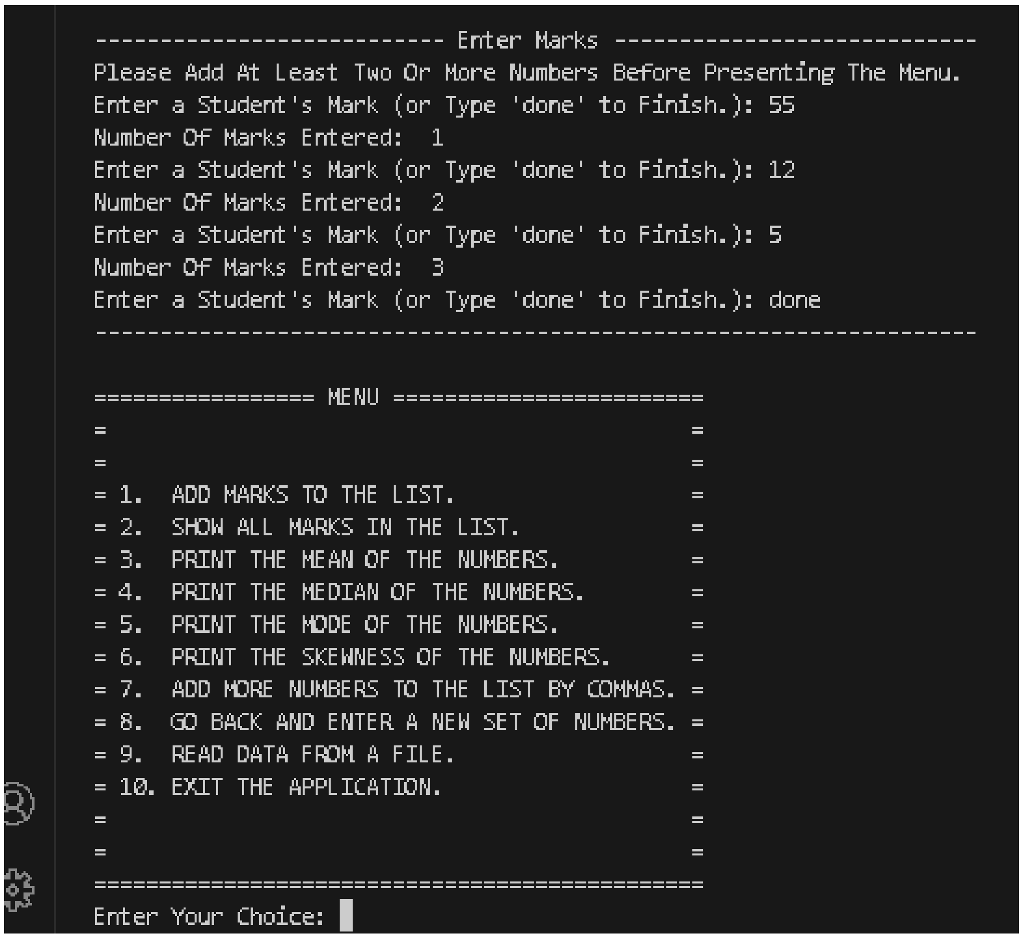


TERMINAL PORTS SERIAL MONITOR SQL CONSOLE COMMENTS PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\Emilio\Documents\GitHub\CS405StudentMarksStatisticsCoursework\StudentMarksStatistics> python  .\MainStudentMarksStatistics.py


================================================
=                    WELCOME                   =
=                                              =
=                      TO                      =
=                                              =
=            SPRING COURSEWORK PYTHON          =
=                                              =
=                                              =
=      STUDENTS MARKS STATISTICS APPLICATION   =
=                                              =
=                      BY                      =
=                                              =
=            EMILIO BARRERA SEPULVEDA          =
=                    22047090                  =
=                                              =
=                                              =
=         LONDON METROPOLITAN UNIVERSITY       =
=                  BSC COMPUTING               =
=         CS4051 FUNDAMENTALS OF COMPUTING     =
=                      LONDON                  =
=                    2023 - 2024               =
================================================


--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.):
```

9.2 Test 2: Enter Marks

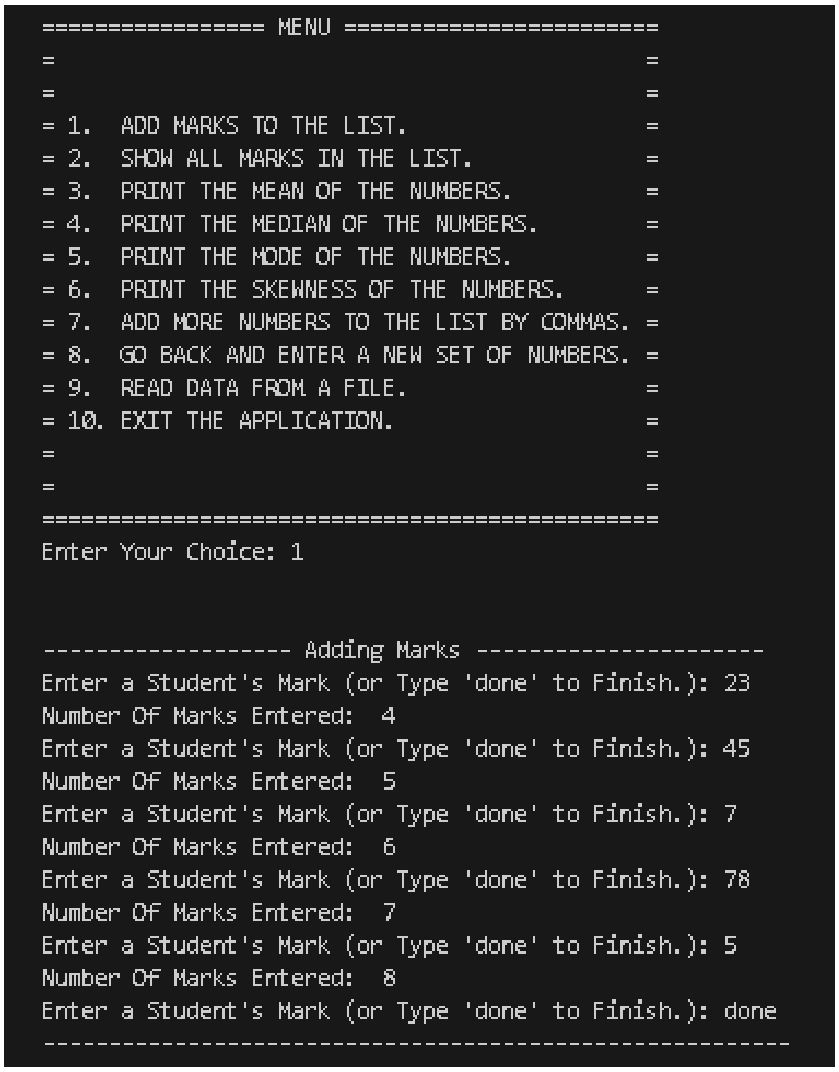
Adding at least two or more numbers before presenting to the Array List for presenting the Menu.



```
--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.): 55
Number Of Marks Entered:  1
Enter a Student's Mark (or Type 'done' to Finish.): 12
Number Of Marks Entered:  2
Enter a Student's Mark (or Type 'done' to Finish.): 5
Number Of Marks Entered:  3
Enter a Student's Mark (or Type 'done' to Finish.): done
--------------------------------------------------------------------

================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice:
```

9.3 Test 3: Adding marks to the list

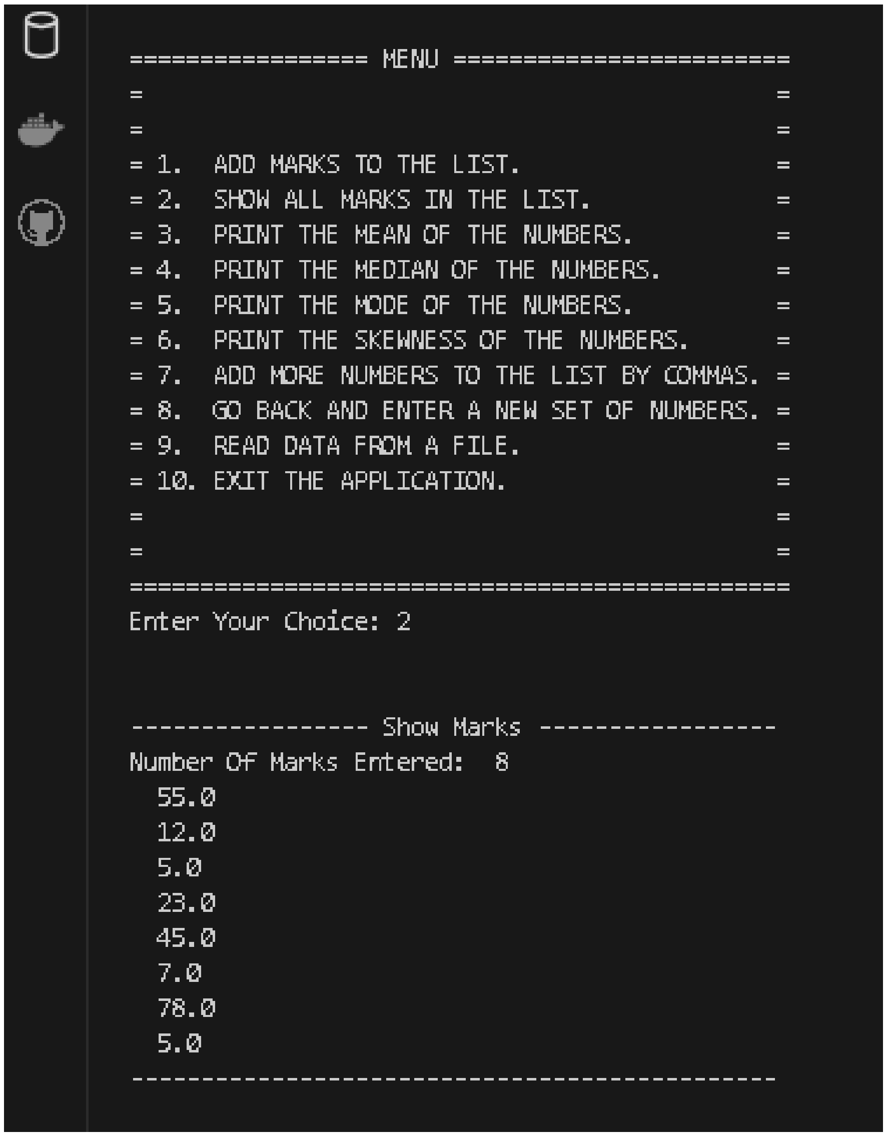
After entering at least two values in the console, you can now use the menu to select your program's functional directions. If you are successful, the program guides you to continue.



```
================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice: 1


------------------- Adding Marks ----------------------
Enter a Student's Mark (or Type 'done' to Finish.): 23
Number Of Marks Entered:  4
Enter a Student's Mark (or Type 'done' to Finish.): 45
Number Of Marks Entered:  5
Enter a Student's Mark (or Type 'done' to Finish.): 7
Number Of Marks Entered:  6
Enter a Student's Mark (or Type 'done' to Finish.): 78
Number Of Marks Entered:  7
Enter a Student's Mark (or Type 'done' to Finish.): 5
Number Of Marks Entered:  8
Enter a Student's Mark (or Type 'done' to Finish.): done
---------------------------------------------------------
```

9.4 Test 4: Displaying all marks the List

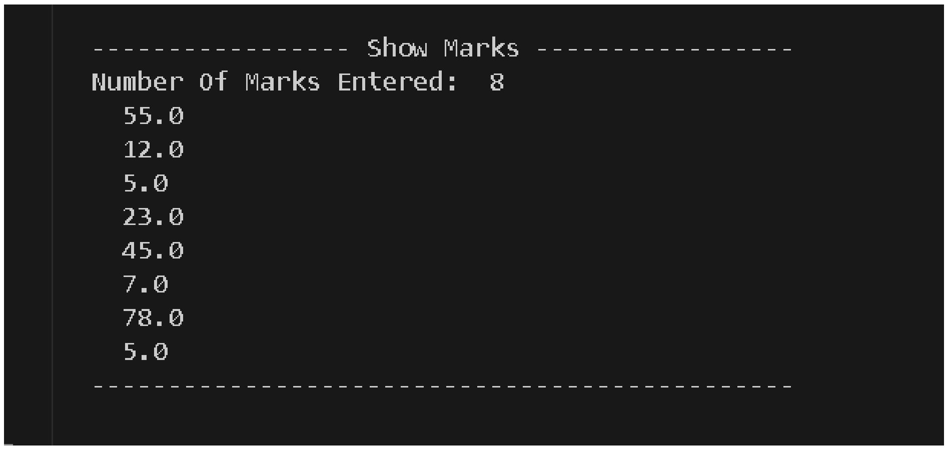
This the Display All action, iterating through marks to show their information.



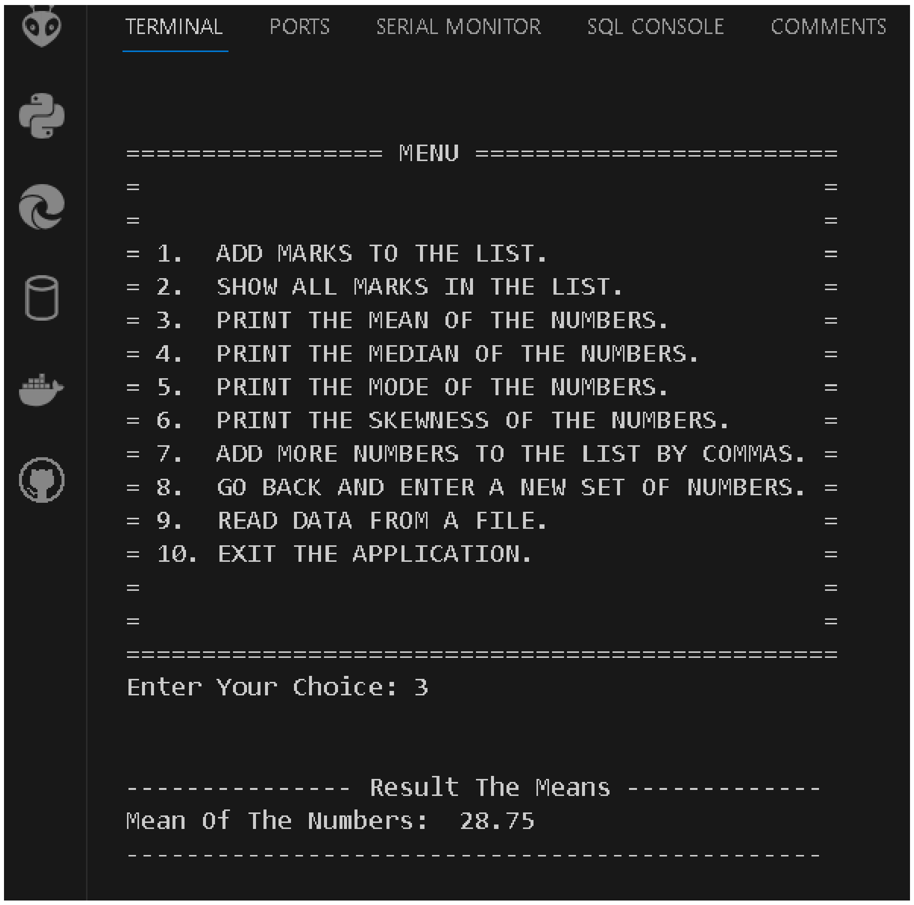
```
================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice: 2


----------------- Show Marks -----------------
Number Of Marks Entered:  8
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
----------------------------------------------
```

9.5 Test 5: The Means

The result the mean of numerical marks. It checks if the list is empty and returns 0 if so.



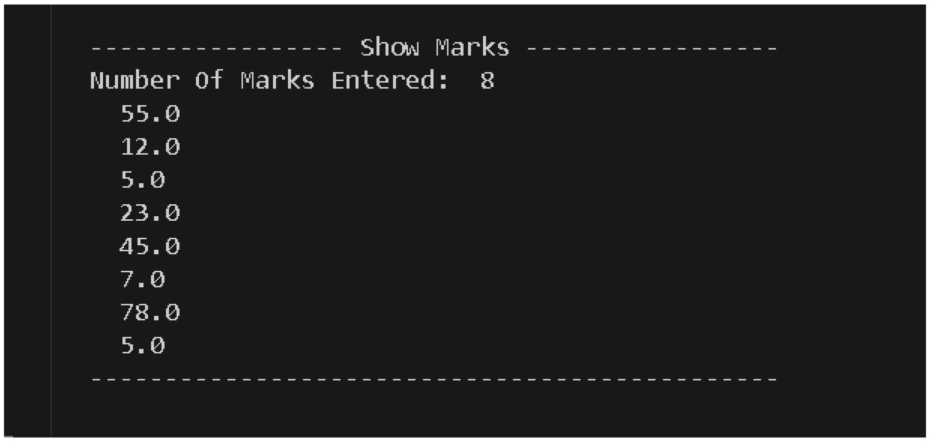
```
----------------- Show Marks -----------------
Number Of Marks Entered:  8
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
----------------------------------------------
```



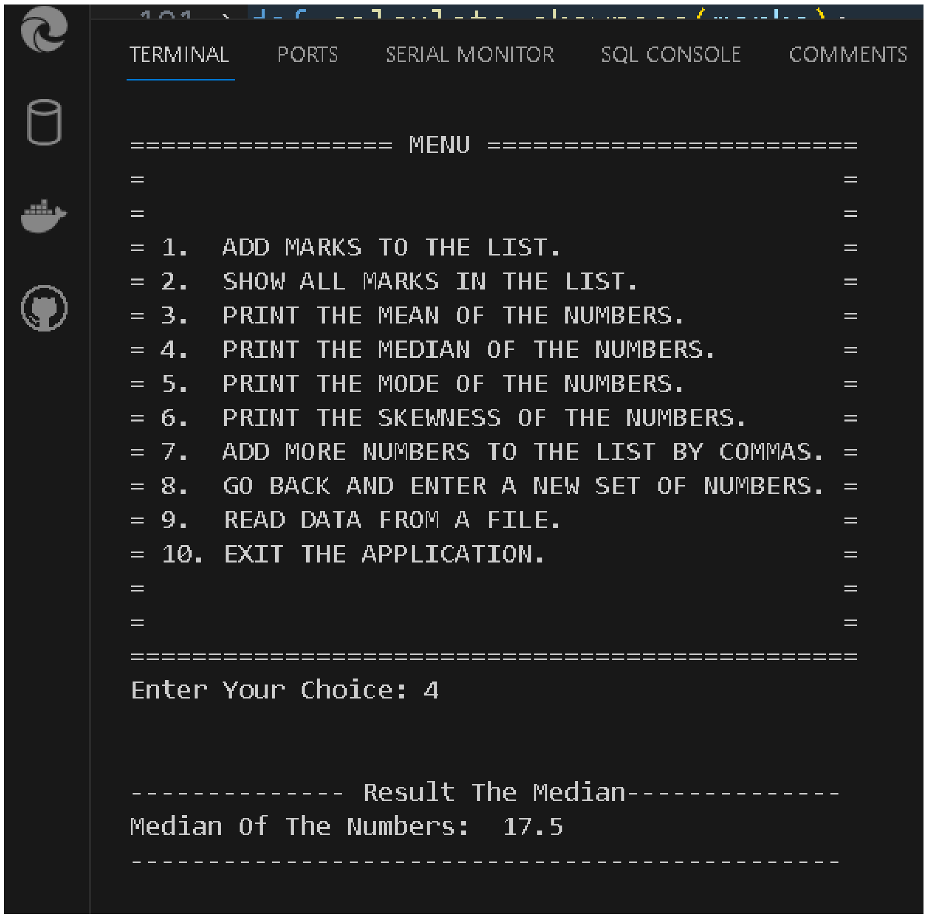
```
================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice: 3


--------------- Result The Means -------------
Mean Of The Numbers:  28.75
----------------------------------------------
```

9.6 Test 6: The Median

The result the median of a list of numerical marks. It handles both even and odd-length lists appropriately.



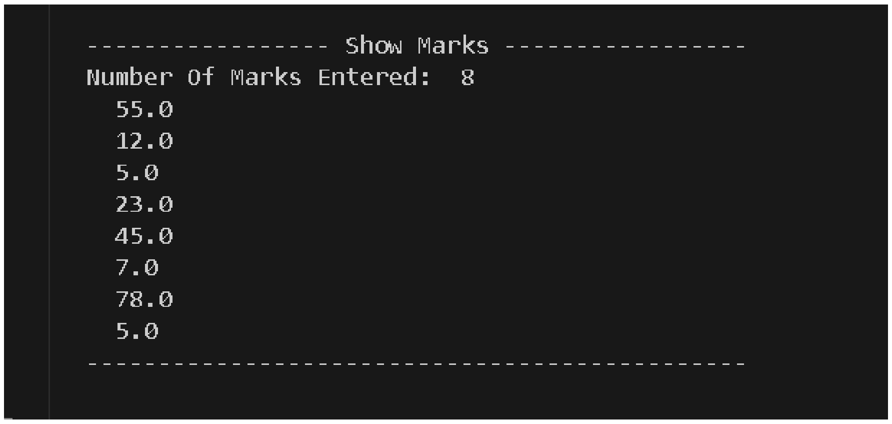
```
----------------- Show Marks -----------------
Number Of Marks Entered:  8
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
----------------------------------------------
```



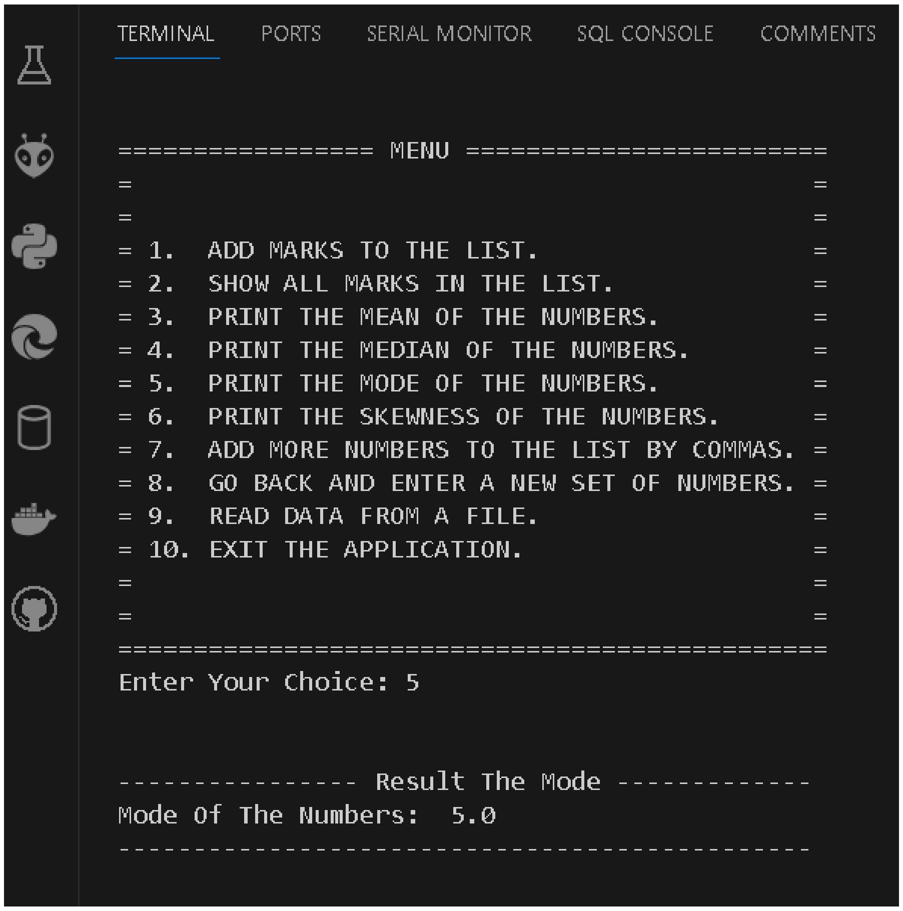
```
================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice: 4


-------------- Result The Median--------------
Median Of The Numbers:  17.5
----------------------------------------------
```

9.7 Test 7: The Mode

The menu option for printing the mode of numbers also involves calculating the mean of the marks in the list.



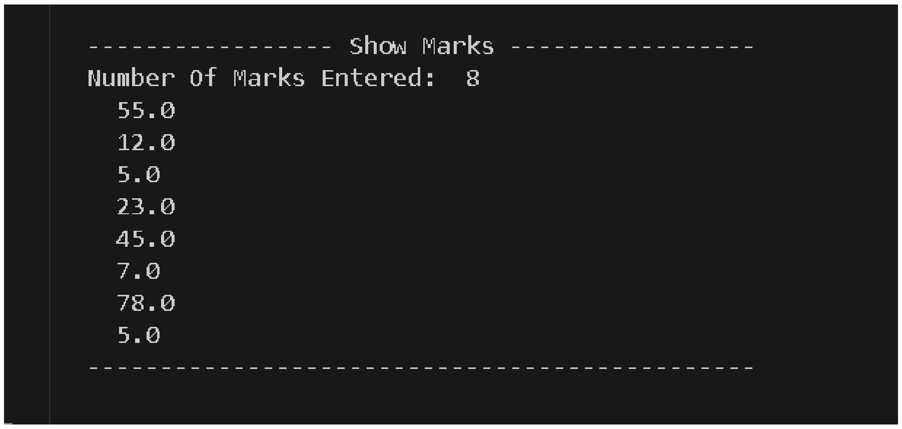
```
----------------- Show Marks -----------------
Number Of Marks Entered:  8
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
----------------------------------------------
```



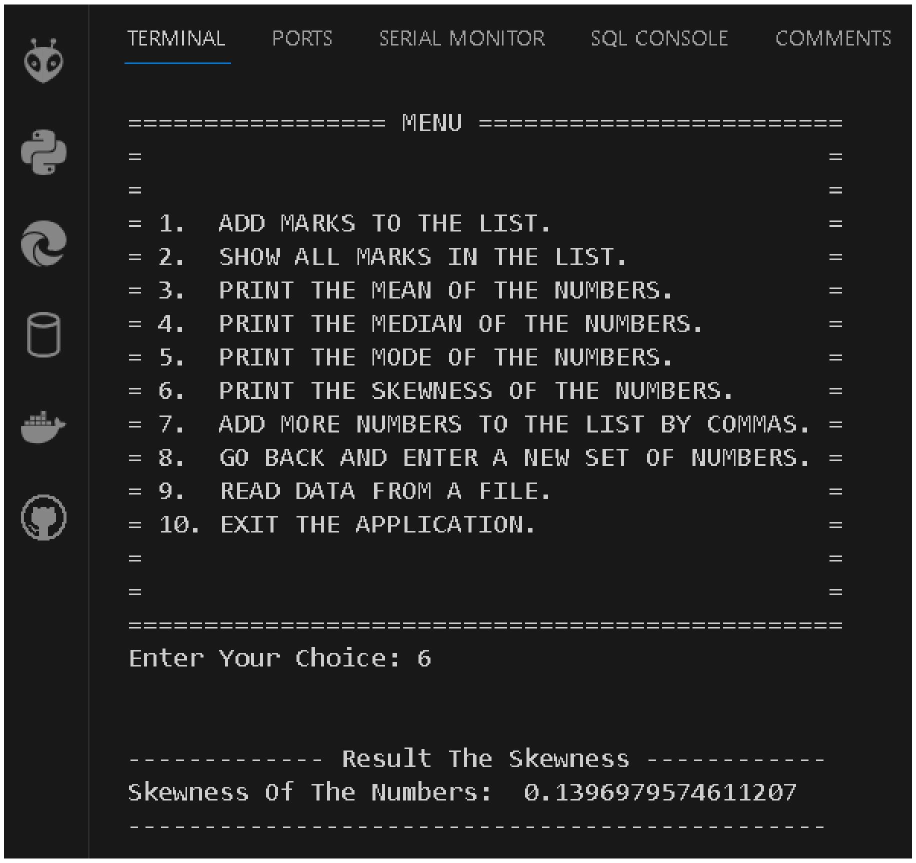
```
================= MENU ========================
=                                             =
=                                             =
= 1.   ADD MARKS TO THE LIST.                 =
= 2.   SHOW ALL MARKS IN THE LIST.            =
= 3.   PRINT THE MEAN OF THE NUMBERS.         =
= 4.   PRINT THE MEDIAN OF THE NUMBERS.       =
= 5.   PRINT THE MODE OF THE NUMBERS.         =
= 6.   PRINT THE SKEWNESS OF THE NUMBERS.     =
= 7.   ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.   GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.   READ DATA FROM A FILE.                 =
= 10.  EXIT THE APPLICATION.                  =
=                                             =
=                                             =
===============================================
Enter Your Choice: 5


---------------- Result The Mode -------------
Mode Of The Numbers:  5.0
----------------------------------------------
```

9.8 Test 8: The Skewness

Option 6 in the menu displays the skewness calculation result for the data present in the list when selected.



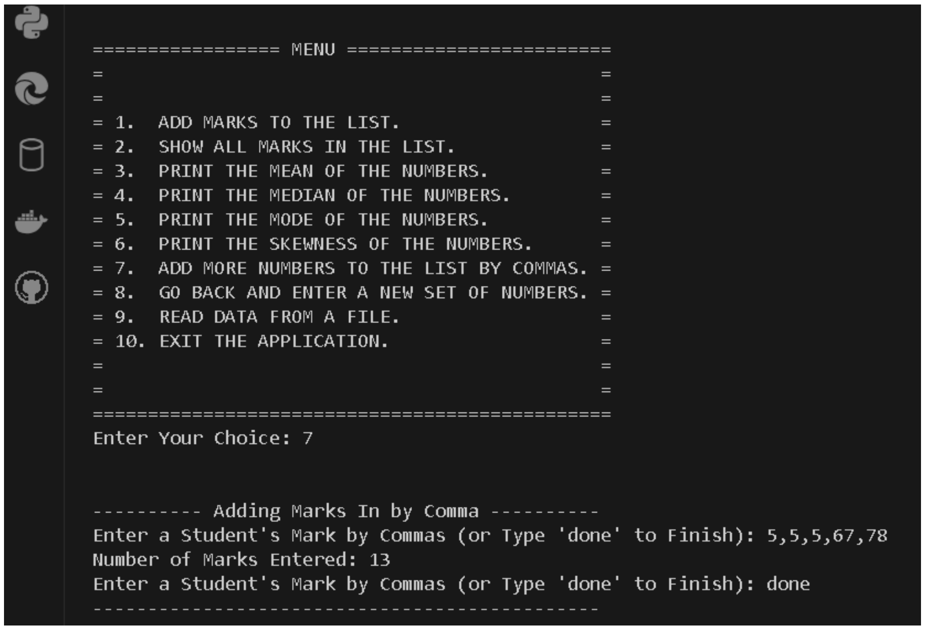
```
----------------- Show Marks -----------------
Number Of Marks Entered:  8
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
----------------------------------------------
```



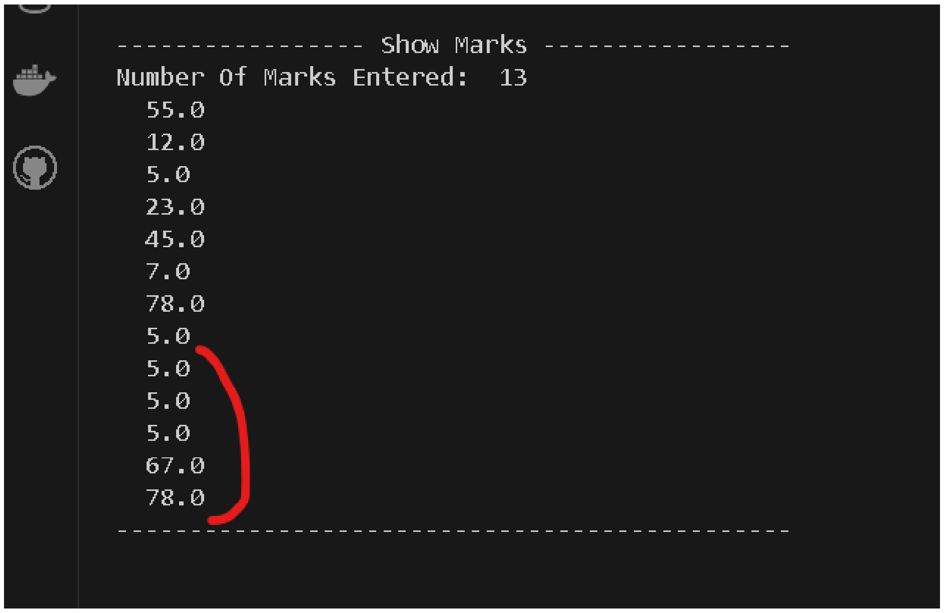
```
================= MENU ========================
=                                             =
=                                             =
= 1.   ADD MARKS TO THE LIST.                 =
= 2.   SHOW ALL MARKS IN THE LIST.            =
= 3.   PRINT THE MEAN OF THE NUMBERS.         =
= 4.   PRINT THE MEDIAN OF THE NUMBERS.       =
= 5.   PRINT THE MODE OF THE NUMBERS.         =
= 6.   PRINT THE SKEWNESS OF THE NUMBERS.     =
= 7.   ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.   GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.   READ DATA FROM A FILE.                 =
= 10.  EXIT THE APPLICATION.                  =
=                                             =
=                                             =
===============================================
Enter Your Choice: 6


------------- Result The Skewness ------------
Skewness Of The Numbers:  0.1396979574611207
----------------------------------------------
```

9.9 Test 9: Add marks in by comma.

Option 7 in the menu allows you to input marks separated by commas, which are then used for program measurements. You can view these marks by displaying all marks in the list.



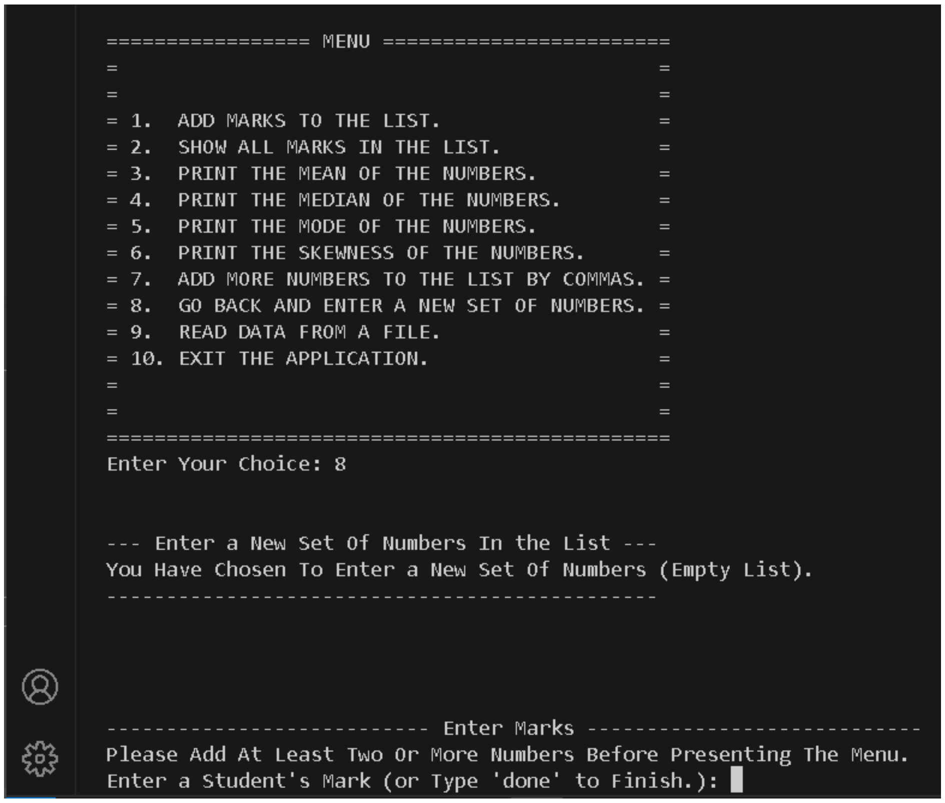
```
================= MENU =======================
=                                              =
=                                              =
= 1.  ADD MARKS TO THE LIST.                   =
= 2.  SHOW ALL MARKS IN THE LIST.              =
= 3.  PRINT THE MEAN OF THE NUMBERS.           =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.         =
= 5.  PRINT THE MODE OF THE NUMBERS.           =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.       =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS.  =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS.  =
= 9.  READ DATA FROM A FILE.                   =
= 10. EXIT THE APPLICATION.                    =
=                                              =
=                                              =
================================================
Enter Your Choice: 7


---------- Adding Marks In by Comma ----------
Enter a Student's Mark by Commas (or Type 'done' to Finish): 5,5,5,67,78
Number of Marks Entered: 13
Enter a Student's Mark by Commas (or Type 'done' to Finish): done
----------------------------------------------
```



```
----------------- Show Marks -----------------
Number Of Marks Entered:  13
  55.0
  12.0
  5.0
  23.0
  45.0
  7.0
  78.0
  5.0
  5.0
  5.0
  5.0
  67.0
  78.0
----------------------------------------------
```

9.10 Test 10: Add marks in by comma.

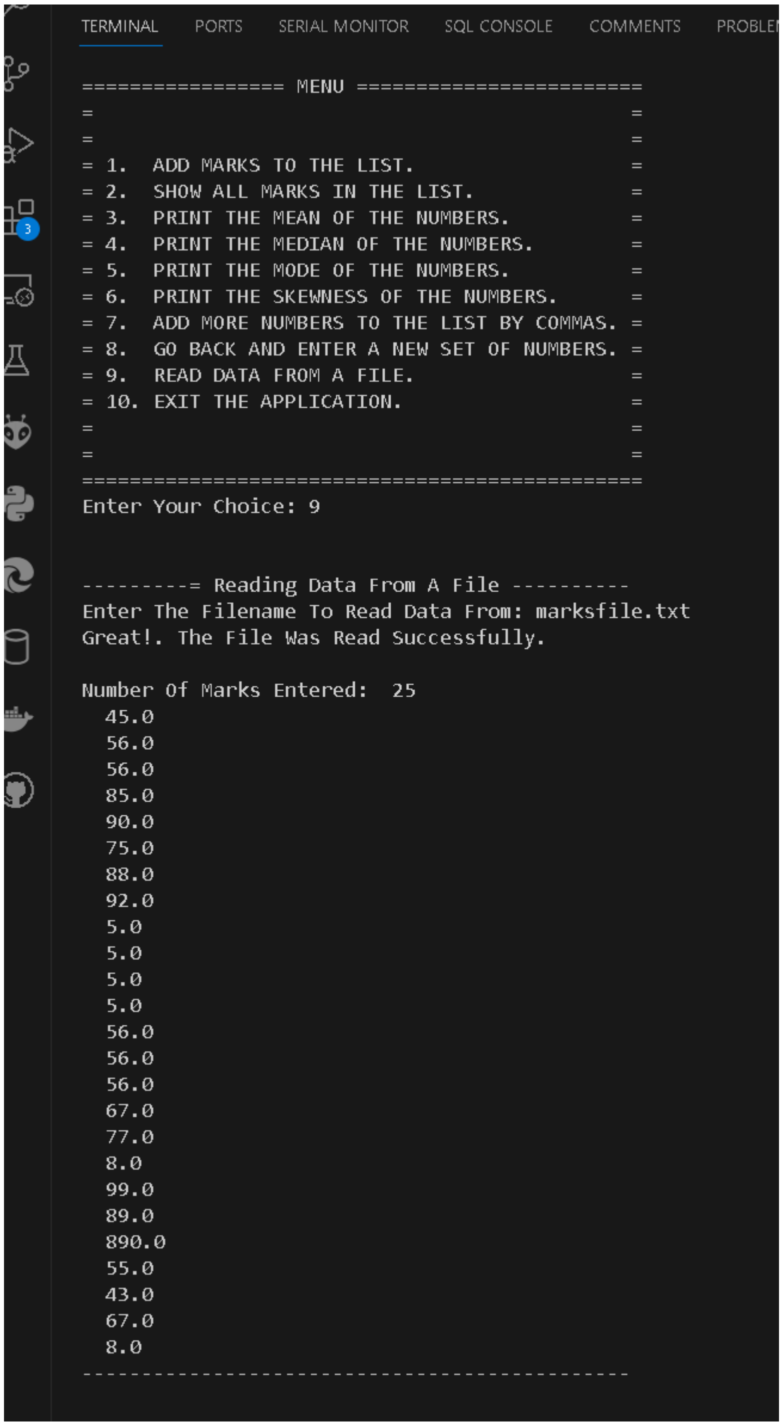
Choosing option 8 from the menu clears all marks from the list, prompting you to enter at least two marks to proceed with the program's features.



```
================= MENU ========================
=                                             =
=                                             =
= 1.  ADD MARKS TO THE LIST.                  =
= 2.  SHOW ALL MARKS IN THE LIST.             =
= 3.  PRINT THE MEAN OF THE NUMBERS.          =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =
= 5.  PRINT THE MODE OF THE NUMBERS.          =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                  =
= 10. EXIT THE APPLICATION.                   =
=                                             =
=                                             =
===============================================
Enter Your Choice: 8


--- Enter a New Set Of Numbers In the List ---
You Have Chosen To Enter a New Set Of Numbers (Empty List).
---------------------------------------------




--------------------------- Enter Marks -----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.):
```

9.11 Test 11: Read data from a file

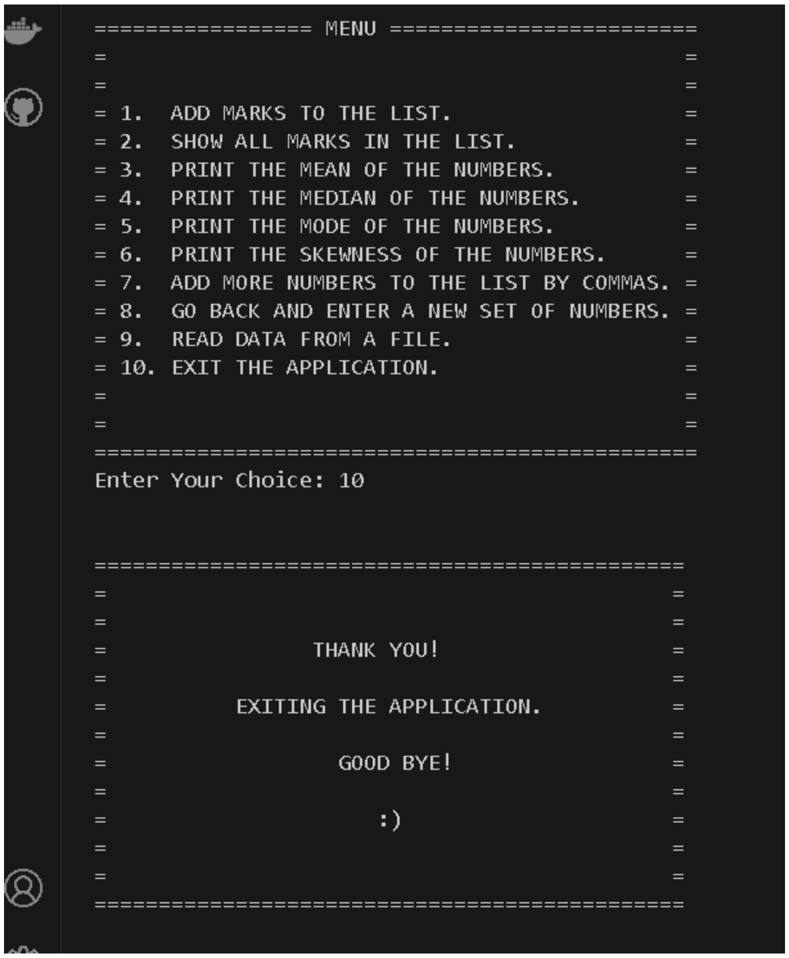
Selecting option 9 from the menu enables reading numerical data from files, facilitating calculations for the program's main functionalities displayed in the menu.



```
================ MENU =======================
=                                           =
=                                           =
= 1.  ADD MARKS TO THE LIST.                =
= 2.  SHOW ALL MARKS IN THE LIST.           =
= 3.  PRINT THE MEAN OF THE NUMBERS.        =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.      =
= 5.  PRINT THE MODE OF THE NUMBERS.        =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.    =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =
= 9.  READ DATA FROM A FILE.                =
= 10. EXIT THE APPLICATION.                 =
=                                           =
=                                           =
=============================================
Enter Your Choice: 9


---------= Reading Data From A File ----------
Enter The Filename To Read Data From: marksfile.txt
Great!. The File Was Read Successfully.

Number Of Marks Entered:  25
  45.0
  56.0
  56.0
  85.0
  90.0
  75.0
  88.0
  92.0
  5.0
  5.0
  5.0
  5.0
  56.0
  56.0
  56.0
  67.0
  77.0
  8.0
  99.0
  89.0
  890.0
  55.0
  43.0
  67.0
  8.0
----------------------------------------------
```

9.12 Test 12: Exit

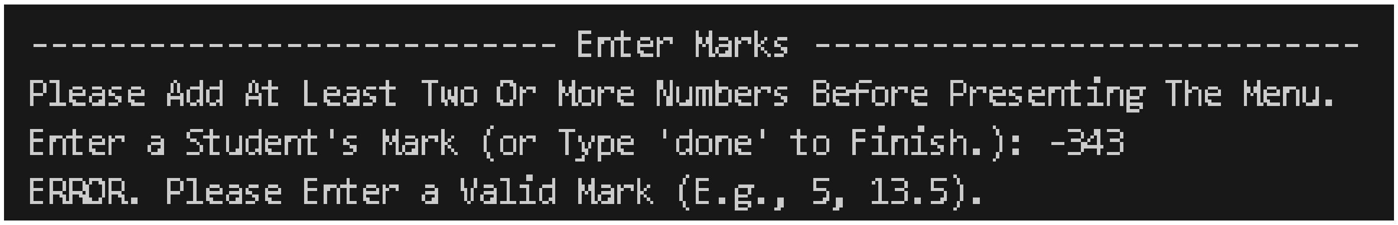
By choosing option 10 from the menu, you will gracefully exit the program and receive a warm farewell message.



```
================= MENU =========================
=                                              =
=                                              =
= 1.  ADD MARKS TO THE LIST.                   =
= 2.  SHOW ALL MARKS IN THE LIST.              =
= 3.  PRINT THE MEAN OF THE NUMBERS.           =
= 4.  PRINT THE MEDIAN OF THE NUMBERS.         =
= 5.  PRINT THE MODE OF THE NUMBERS.           =
= 6.  PRINT THE SKEWNESS OF THE NUMBERS.       =
= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS.  =
= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS.  =
= 9.  READ DATA FROM A FILE.                   =
= 10. EXIT THE APPLICATION.                    =
=                                              =
=                                              =
================================================
Enter Your Choice: 10


==============================================
=                                            =
=                                            =
=                THANK YOU!                  =
=                                            =
=          EXITING THE APPLICATION.          =
=                                            =
=                 GOOD BYE!                  =
=                                            =
=                    :)                      =
=                                            =
=                                            =
==============================================
```

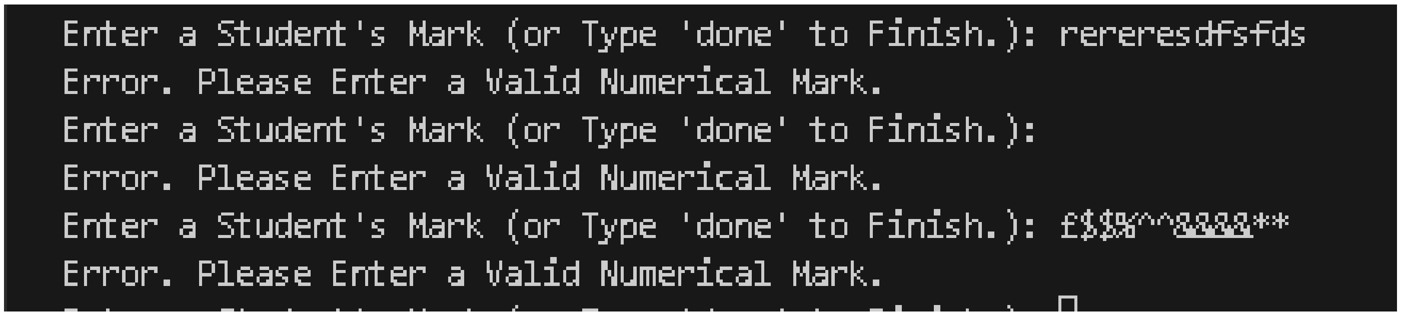
9.13 Test 13: Error Handling

- If the user inputs negative numbers, the program displays an error message and prompts for valid non-negative marks.



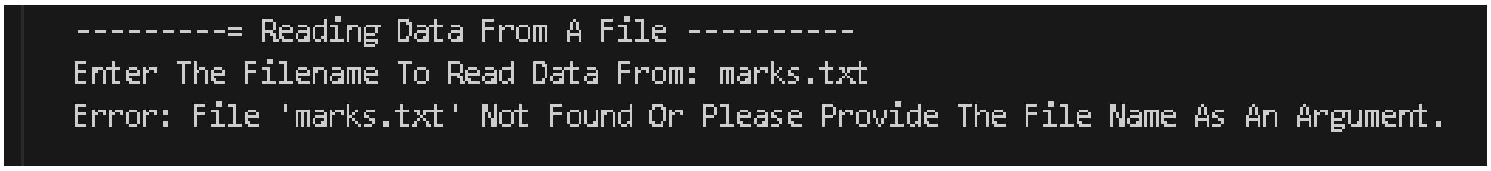
```
-------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.): -343
ERROR. Please Enter a Valid Mark (E.g., 5, 13.5).
```

- Special characters or letters instead of numerical marks prompt an error message, guiding the user to enter valid numerical input.

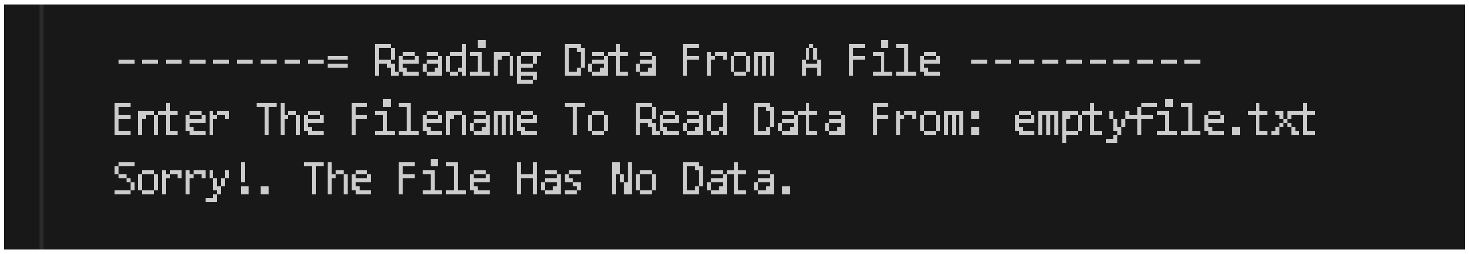


```
  Enter a Student's Mark (or Type 'done' to Finish.): rereresdfsfds
  Error. Please Enter a Valid Numerical Mark.
  Enter a Student's Mark (or Type 'done' to Finish.):
  Error. Please Enter a Valid Numerical Mark.
  Enter a Student's Mark (or Type 'done' to Finish.): £$$%^^&&&&**
  Error. Please Enter a Valid Numerical Mark.
```

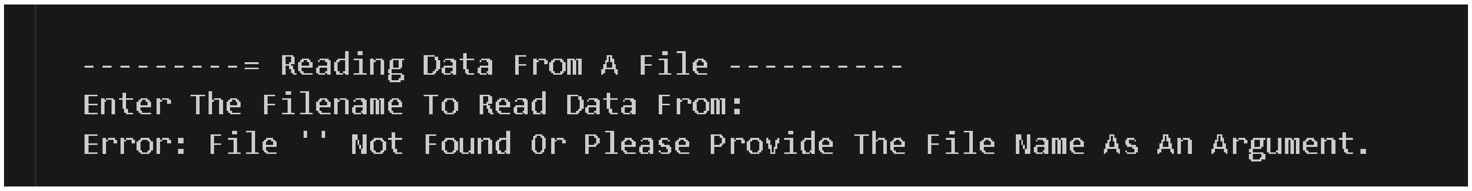
- Exception handling is used to address errors such as file not found or insufficient data points for calculations and if it is null.



```
  ---------= Reading Data From A File ----------
  Enter The Filename To Read Data From: marks.txt
  Error: File 'marks.txt' Not Found Or Please Provide The File Name As An Argument.
```

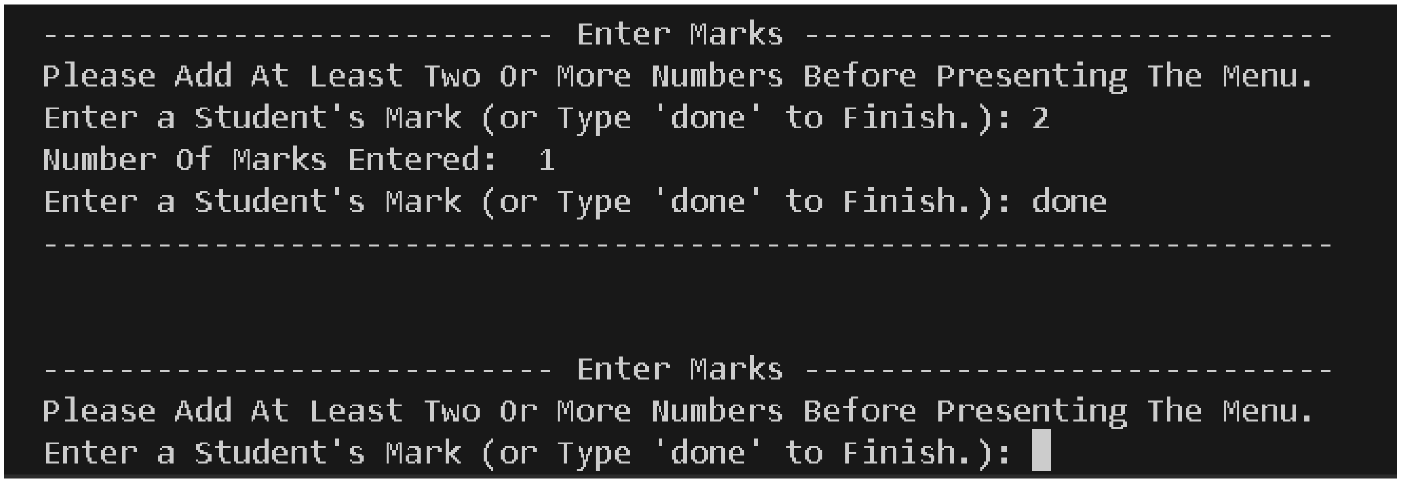


```
---------= Reading Data From A File ----------
Enter The Filename To Read Data From: emptyfile.txt
Sorry!. The File Has No Data.
```

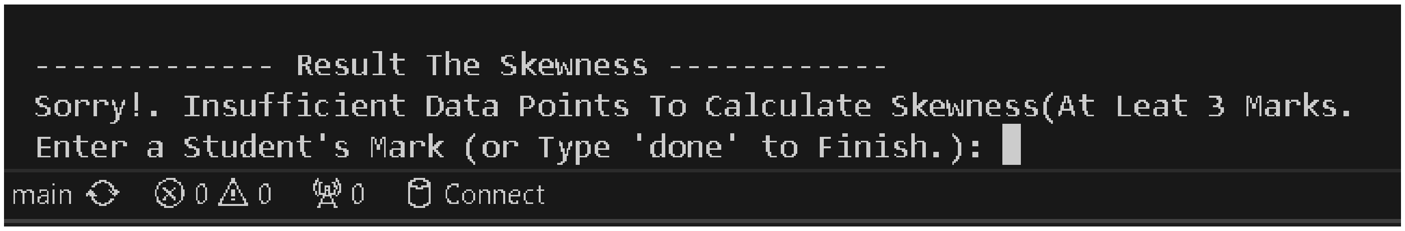


```
---------= Reading Data From A File ----------
Enter The Filename To Read Data From:
Error: File '' Not Found Or Please Provide The File Name As An Argument.
```

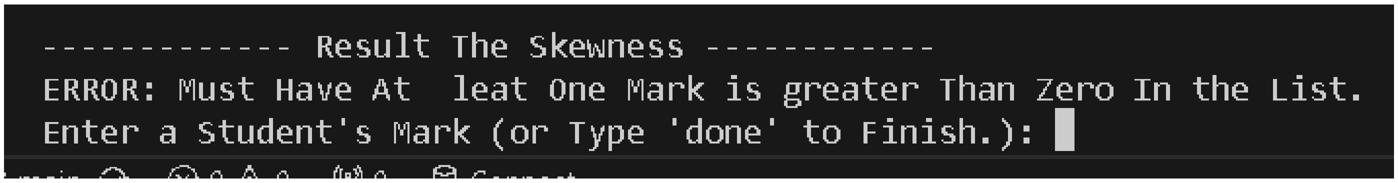
- Before performing certain operations (e.g., mean, median, mode, skewness), the program ensures that there are at least two marks entered to avoid errors related to insufficient data.



```
--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.): 2
Number Of Marks Entered:  1
Enter a Student's Mark (or Type 'done' to Finish.): done
--------------------------------------------------------------------



--------------------------- Enter Marks ----------------------------
Please Add At Least Two Or More Numbers Before Presenting The Menu.
Enter a Student's Mark (or Type 'done' to Finish.):
```



```
------------- Result The Skewness ------------
Sorry!. Insufficient Data Points To Calculate Skewness(At Leat 3 Marks.
Enter a Student's Mark (or Type 'done' to Finish.):
```



```
------------- Result The Skewness ------------
ERROR: Must Have At  leat One Mark is greater Than Zero In the List.
Enter a Student's Mark (or Type 'done' to Finish.):
```

Overall, the program provides robust error handling and a comprehensive set of statistical functionalities for analyzing student marks, enhancing usability and reliability.

10. CONCLUSION

In conclusion, the student marks calculation application, which uses the MVC pattern, offers a streamlined solution for managing and analyzing academic data. Through a modular design and clear separation of concerns, it ensures efficient data processing, user interaction and presentation. By providing statistical functionalities such as calculation of mean, median, mode and skewness, along with easy-to-use features such as data entry validation and file reading capabilities, the program improves usability and productivity. With its structured architecture and comprehensive functionality, the application is a testament to effective software design principles, empowering users and system administrators in their data management and analysis efforts.

11. Appendix

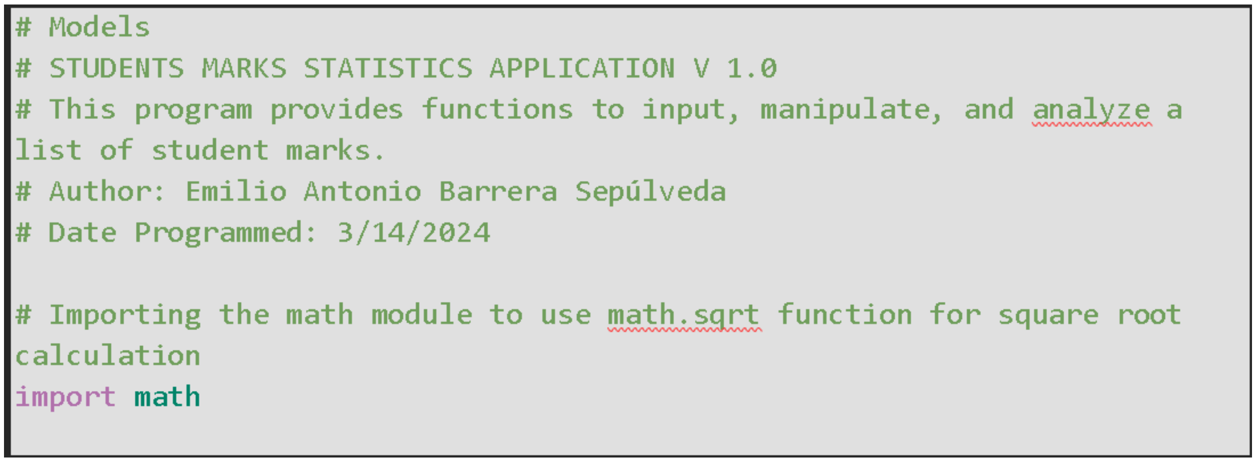
This coursework presents the complete source code for a student’s marks statistics application. The code is well-structured and includes comments to explain its functionality. This application can be used for various purposes, such as:

- Calculating student grades based on marks: The application allows you to enter student marks (individually or as a comma-separated list) and then calculates various statistics like mean, median, mode, and skewness.

- Learning Python programming concepts: This code demonstrates practical use of Python concepts like control flow statements, data structures (lists), functions, and working with user input.

- Adapting for other calculations: The core functionality of collecting and processing numerical data can be adapted to calculate grades using different weighting schemes or for other use cases beyond student grades.

By including these details, you provide a clearer understanding of the code's significance and potential applications for the reader.



```python
# Models
# STUDENTS MARKS STATISTICS APPLICATION V 1.0
# This program provides functions to input, manipulate, and analyze a list of student marks.
# Author: Emilio Antonio Barrera Sepúlveda
# Date Programmed: 3/14/2024

# Importing the math module to use math.sqrt function for square root calculation
import math
```

```
"""
 Primary statistical functions implemented within the program.
"""

# Function enter marks to the list
def enter_marks_to_list(marks):
    """
    This function allows the user to input marks until they choose to
stop.
    It validates each input to ensure it is a valid numerical mark.

    Args:
    marks (list): A list to which the entered marks will be appended.

    Returns:
    None
    """
    while True:  # Start an infinite loop
        # Prompt the user to enter a mark
        mark_input = input("Enter a Student's Mark (or Type 'done' to
Finish.): ")
        # Check if user wants to finish entering marks
        if mark_input.lower() == "done":
            # Exit the loop if user inputs 'done'
            break
        try:
            # Convert input to float
            mark = float(mark_input)
             # Check if mark is valid (assuming negative marks are not
allowed)
            if mark > -1:
                #Append mark to the list
                marks.append(mark)
                # Print number of marks entered
                print("Number Of Marks Entered: ", len(marks))
            else:
                # Print error message for invalid mark
                print("ERROR. Please Enter a Valid Mark (E.g., 5, 
13.5).")
        except ValueError:
            # Print error message for non-numerical input
            print("Error. Please Enter a Valid Numerical Mark.")

# Function enter marks to list by commas
def enter_marks_to_list_by_comma(marks):
    """
    This function allows the user to input marks until they choose to 
stop.
    It validates each input to ensure it is a valid numerical mark.

    Args:
    marks (list): A list to which the entered marks will be appended.

    Returns:
    None
    """
    while True:  # Start an infinite loop
        # Prompt the user to enter a mark
        mark_input = input("Enter a Student's Mark by Commas (or Type 
'done' to Finish): ")
        # Check if user wants to finish entering marks
        if mark_input.lower() == "done":
            # Exit the loop if user inputs 'done'
            break
        elif ',' in mark_input:
            try:
                # Split the input by commas, convert each substring to 
float, and add to the marks list
                marks.extend(map(float, mark_input.split(',')))
                # Print the number of marks entered
                print("Number of Marks Entered:", len(marks))
            except ValueError:
                # Print error message for non-numerical
input
                print("ERROR. Please Enter a Valid Numerical Mark.")

# Define a function named ShowAllMarks that takes a list named marks as
input
def Show_All_Marks(marks):
    # Define a function named ShowAllMarks that takes a list named marks
as input
    print("Number Of Marks Entered: ", len(marks) ,"\t")
    for mark in marks:
        # Iterate over each element in the marks list
        print(" ", mark)
        # Convert each element in the marks list to a string using map()
and str() functions
        # Then join all the elements together with a space in between
using join()
        # Finally, print the result

# Function Calculate the mean of the marks
def calculate_mean(marks):
    """
    This function calculates the mean of the entered marks.

    Args:
    marks (list): A list of numerical marks.

    Returns:
    float: The mean of the marks. If the list is empty, returns 0.
    """
    # Check if marks list is empty
    if not marks:
        # Return 0 if list is empty
        return 0
    # Calculate mean of marks and return it
    return sum(marks) / len(marks)
```

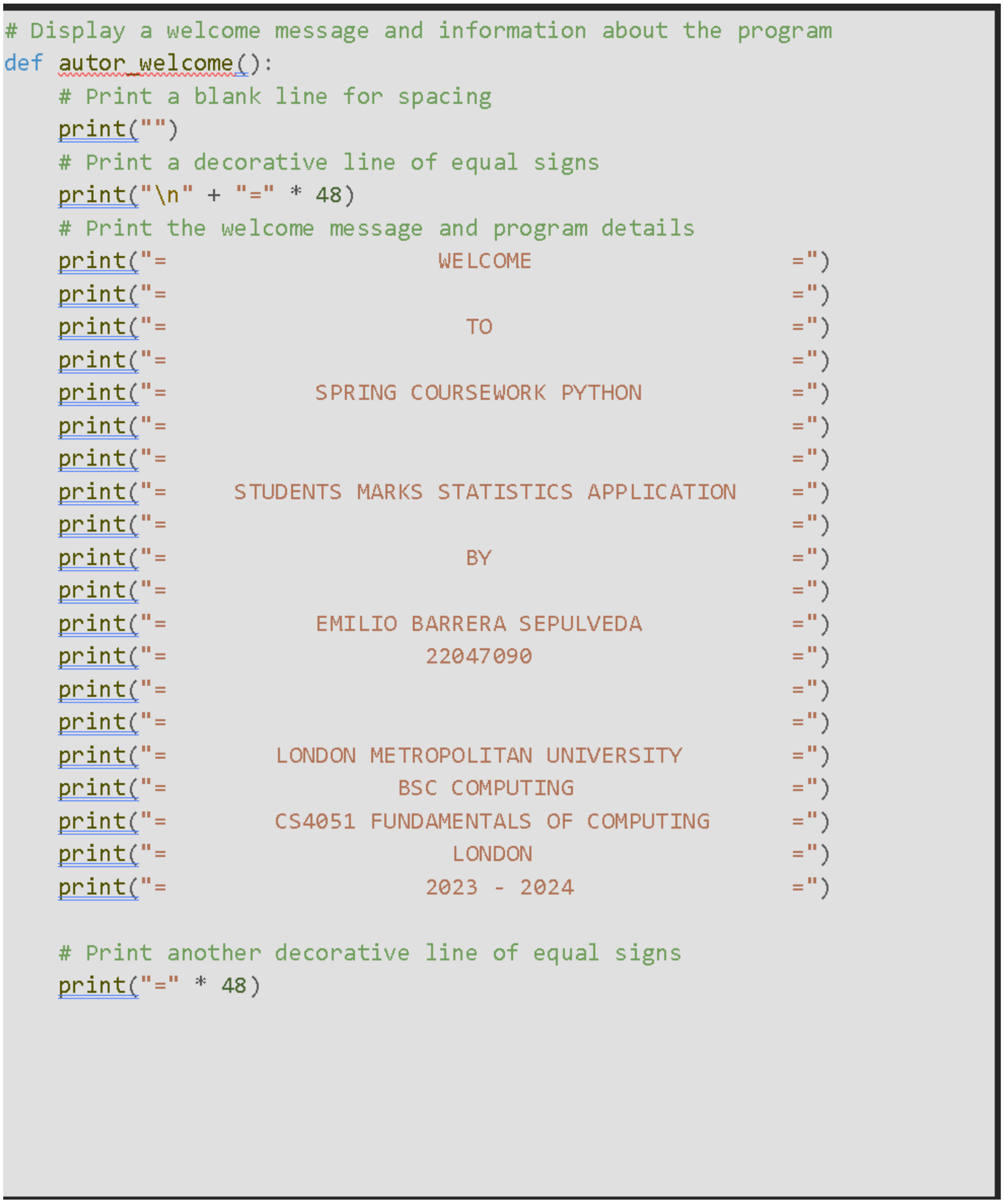
```python
# Function Calculate the median of the marks
def calculate_median(marks):
    """
    Calculate the median of a list of marks.

    Args:
    marks (list): A list of numerical marks.

    Returns:
    float: The median of the marks.
    """

    if not marks:  # Check if marks list is empty
        return 0  # Return 0 if list is empty

    # Sort the list of marks in ascending order
    sorted_marks = sorted(marks)
    # Get the number of marks in the list
    n = len(sorted_marks)

    # Check if the number of marks is even
    if n % 2 == 0:
        # Calculate the median for even-length lists
        # by taking the average of the two middle values
        return (sorted_marks[n // 2 - 1] + sorted_marks[n // 2]) / 2
    else:
        # Calculate the median for odd-length lists
        # by returning the middle value
        return sorted_marks[n // 2]
```

```
# Function Calculate the mode of the marks
def calculate_mode(marks):
    # Check if the marks list is empty
    if not marks:
        return None  # Return None if the list is empty, as there is no
mode

    # Create an empty dictionary to store the count of each mark
    marks_count = {}

    # Count the occurrences of each mark in the list
    for mark in marks:
        # Increase the count of the current mark by 1
        # Using marks_count.get() to retrieve the count of the current
mark, or 0 if not present
        marks_count[mark] = marks_count.get(mark, 0) + 1

    # Find the maximum count of occurrences
    max_count = max(marks_count.values())

    # Find all marks that have the maximum count (the mode(s))
    mode = [key for key, value in marks_count.items() if value ==
max_count]

    # Return the first mode if it exists, otherwise return None
    return mode[0] if mode else None

# Function Calculate the standard deviation of the marks
def calculate_standard_deviation(marks):
    """
    Calculate the standard deviation of a list of marks.

    Args:
    marks (list): A list of numerical marks.

    Returns:
```

```
    float: The standard deviation of the marks.
    """
    # Check if the marks list is empty
    if not marks:
        # If the marks list is empty, return 0 as there are no marks to
compute the standard deviation
        return 0

    # Calculate the mean of the marks using a separate function
calculate_mean
    # Above line uses a separate function named calculate_mean to
compute the mean of the marks.
    # It's done this way to break down the computation into smaller,
more manageable functions.
    mean = calculate_mean(marks)

    # Calculate the variance of the marks
    # Variance is the average of the squared differences from the mean
    # The above line uses a generator expression to iterate over each
mark in the marks list.
    # For each mark, it calculates the squared difference from the mean
and sums them up.
    # Then it divides the sum by the total number of marks to get the
average squared difference.
    variance = sum((mark - mean) ** 2 for mark in marks) / len(marks)

    # Return the square root of the variance as the standard deviation
    # Standard deviation is the square root of variance
    # The square root of the variance is returned as the standard
deviation.
    # This is because the standard deviation is the measure of how
spread out the values in a dataset are.
    # By returning the square root of the variance, we're providing a
measure of the spread that is in the same units as the original data.
    return math.sqrt(variance)
```

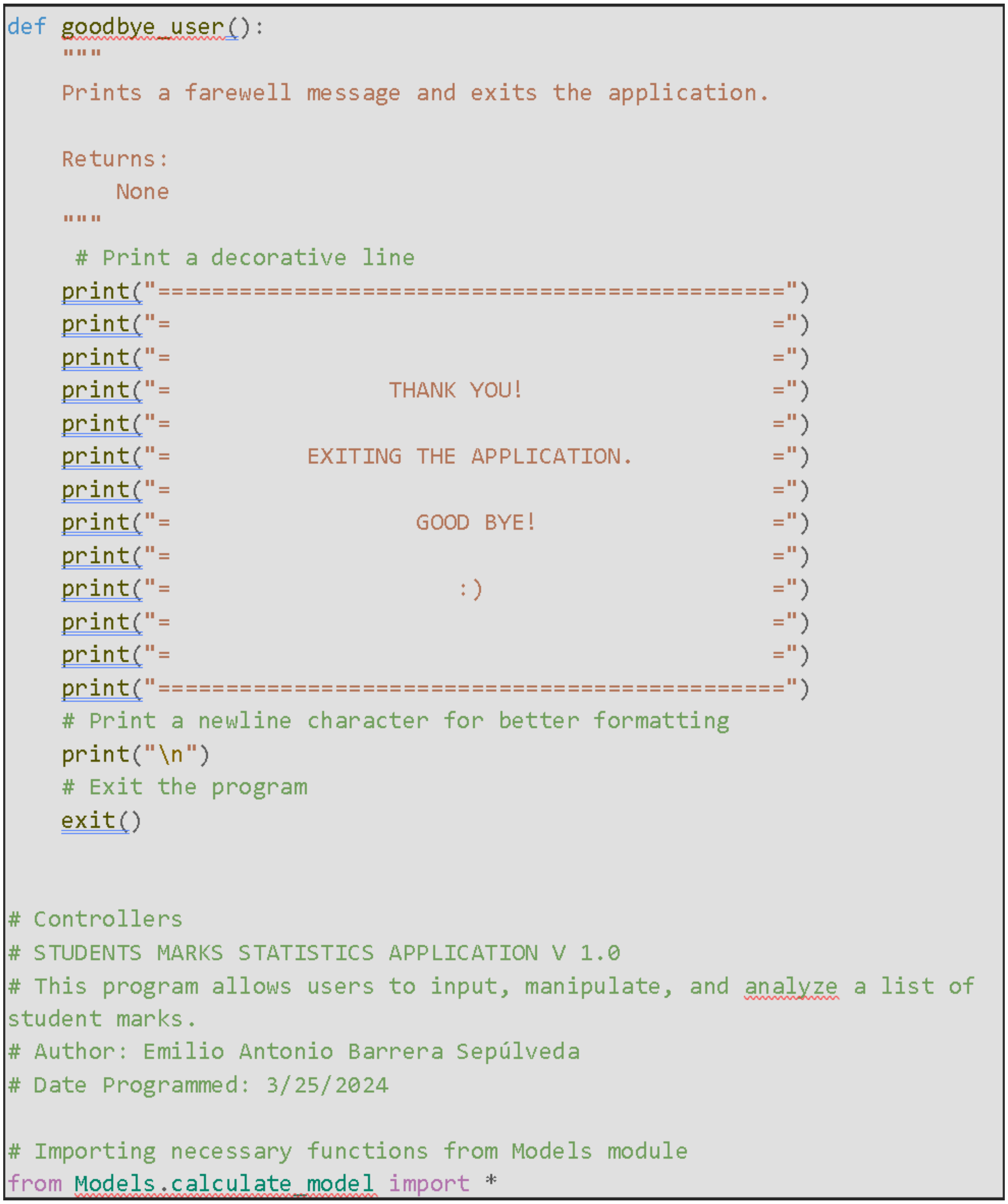
```
# Function Calculate the skewness of the marks
def calculate_skewness(marks):
    """
    Calculate the skewness of a list of marks.

    Args:
    marks (list): A list of numerical marks.

    Returns:
    float: The skewness of the marks.
    """
    # Check for empty list or insufficient data points
    if len(marks) < 3:
        raise ValueError("Sorry!. Insufficient Data Points To Calculate
Skewness(At Leat 3 Marks.")
    #Inssufficient Data Points To Calculate Skewness.

    # Calculate the mean of the marks
    # Above line uses a separate function named calculate_mean to
compute the mean of the marks.
    # It's done this way to break down the computation into smaller,
more manageable functions.
    mean = calculate_mean(marks)

    n = len(marks)  # Get the number of marks

    # Calculate the numerator of the skewness formula
    numerator = sum((mark - mean) ** 3 for mark in marks)
    # The above line uses a generator expression to iterate over each
mark in the marks list.
    # For each mark, it calculates the cubed difference from the mean
and sums them up.

    # Calculate the denominator of the skewness formula
    denominator = (n - 1) * (n - 2) *
(calculate_standard_deviation(marks) ** 3)
```

```
    # The denominator of the skewness formula involves the cube of the 
standard deviation,
    # which is calculated using the calculate_standard_deviation 
function.
    # The formula also depends on the number of elements in the list.

    # Check for division by zero
    if denominator == 0:
        raise ValueError("ERROR: Must Have At  leat One Mark is greater 
Than Zero In the List.")

    # Return the skewness value
    return numerator / denominator
    # The skewness value is calculated by dividing the numerator by the 
denominator,
    # following the skewness formula.

# Read data from a file and populate the list of marks.
def read_data_from_file(filename):
    """
    Read data from a file and populate the list of marks.

    Args:
    filename (str): The name of the file to read data from.

    Returns:
    list: A list containing the numerical marks read from the file.
    """
    try:
        # Attempt to open the file for reading
        with open(filename, "r") as file:
            # Read the contents of the file
            data = file.read()
            if data != '':
                print("Great!. The File Was Read Successfully.")
                # Split the data by commas and convert each element to a
float, then store in a list
                marks = list(map(float, data.split(',')))
                # Return the list of marks
                return marks
            else:
                print("Sorry!. The File Has No Data.")
                return ""
    except FileNotFoundError:
        # Handle the case where the file is not found
        print(f"Error: File '{filename}' Not Found Or Please Provide The
File Name As An Argument.")
        # Return an empty list
        return []
    except ValueError:
        # Handle the case where the file contains invalid numerical data
        print("Error: File contains invalid numerical data.")
        # Return an empty list
        return []



# Views
# STUDENTS MARKS STATISTICS APPLICATION V 1.0
# This program provides functionalities for managing and analyzing
student marks.
# Author: Emilio Antonio Barrera Sepúlveda
# Date Programmed: 3/20/2024
```

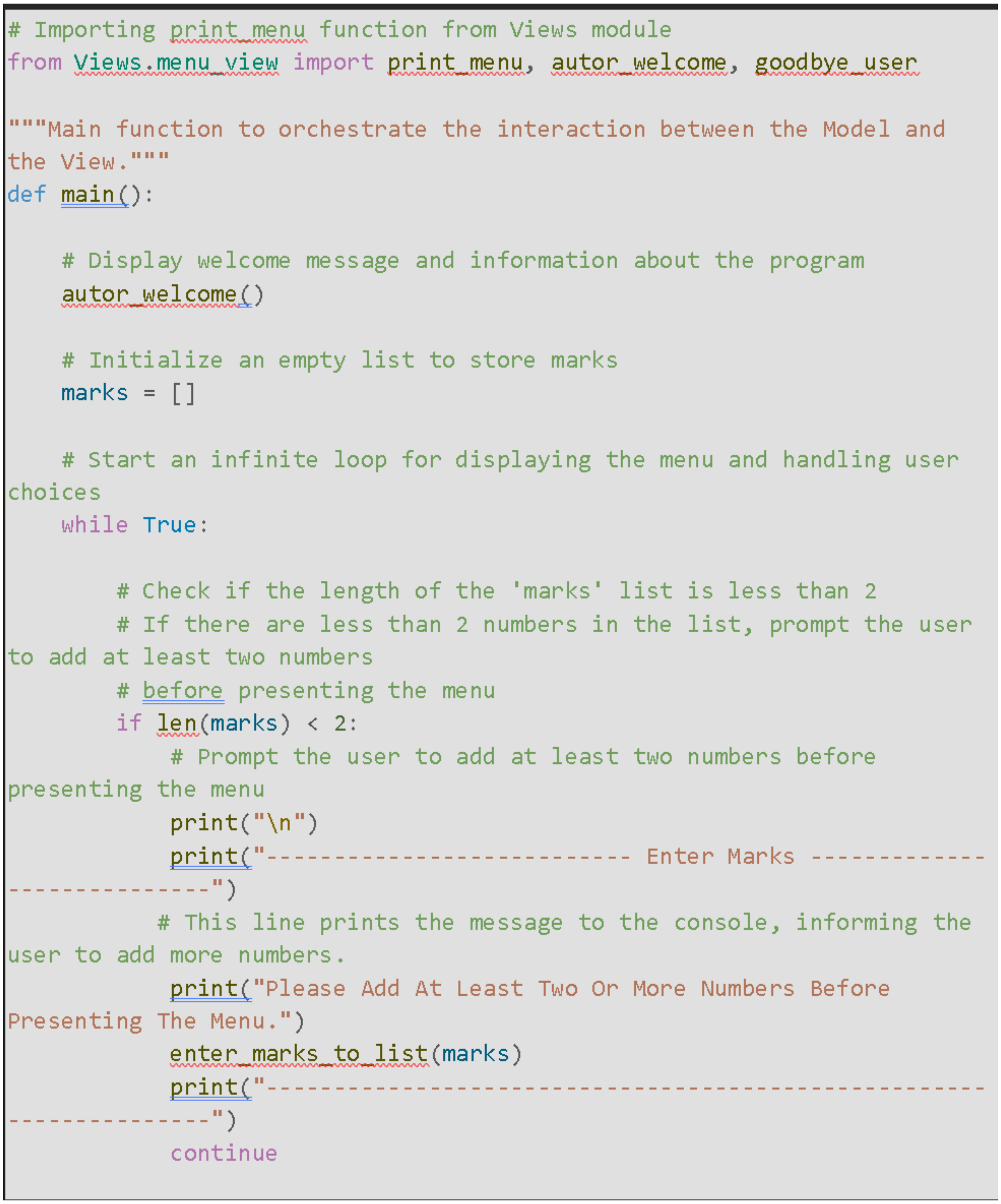


```python
# Display a welcome message and information about the program
def autor_welcome():
    # Print a blank line for spacing
    print("")
    # Print a decorative line of equal signs
    print("\n" + "=" * 48)
    # Print the welcome message and program details
    print("=                    WELCOME                    =")
    print("=                                              =")
    print("=                      TO                      =")
    print("=                                              =")
    print("=           SPRING COURSEWORK PYTHON           =")
    print("=                                              =")
    print("=                                              =")
    print("=     STUDENTS MARKS STATISTICS APPLICATION    =")
    print("=                                              =")
    print("=                      BY                      =")
    print("=                                              =")
    print("=           EMILIO BARRERA SEPULVEDA           =")
    print("=                   22047090                   =")
    print("=                                              =")
    print("=                                              =")
    print("=        LONDON METROPOLITAN UNIVERSITY        =")
    print("=                 BSC COMPUTING                =")
    print("=        CS4051 FUNDAMENTALS OF COMPUTING      =")
    print("=                     LONDON                   =")
    print("=                   2023 - 2024                =")

    # Print another decorative line of equal signs
    print("=" * 48)
```

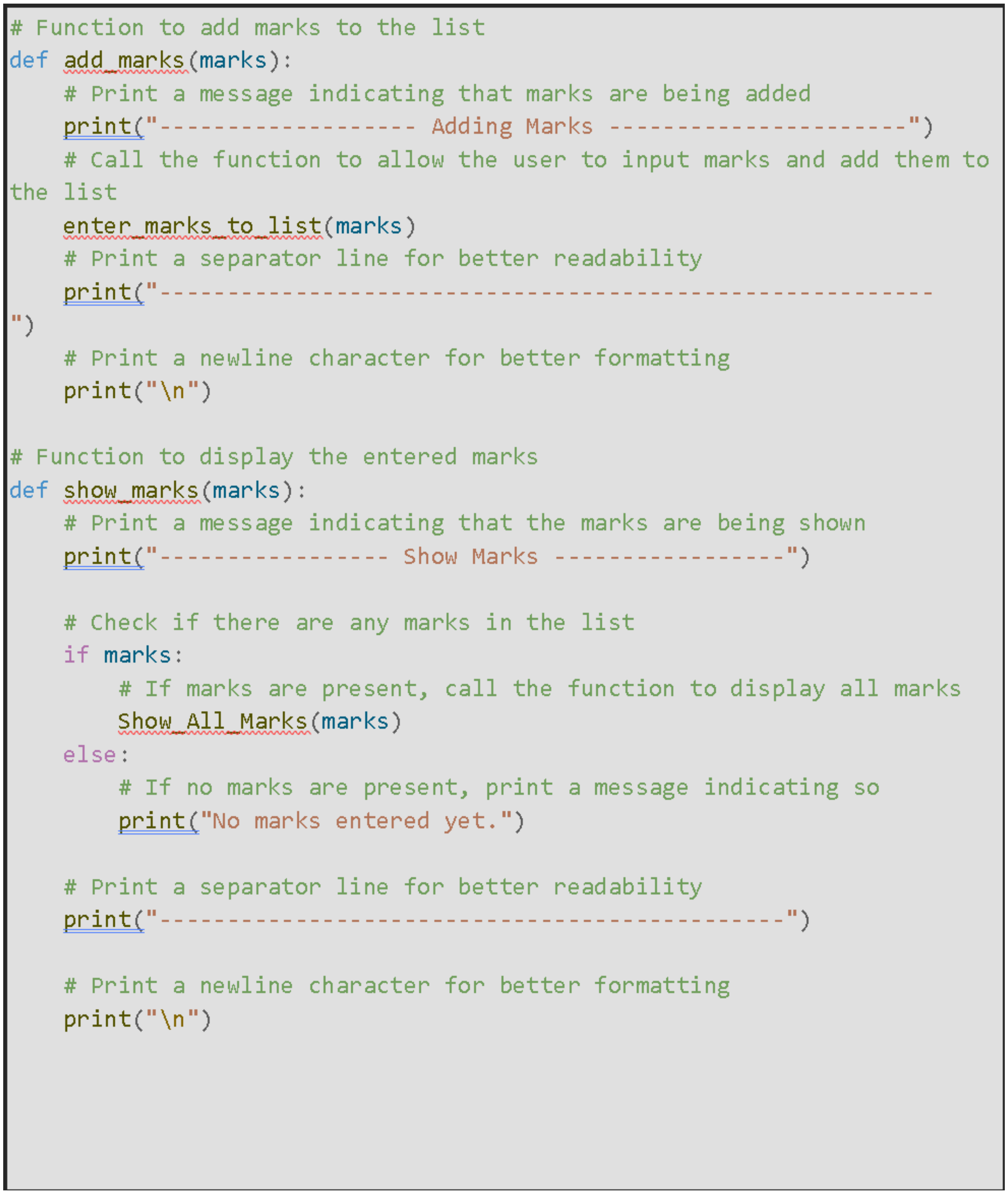
```
# This function prints the menu options for the application.
def print_menu():
    print("\n================= MENU ========================")  # Print
menu header
    print("=                                             =")  # Print
menu head
    print("=                                             =")  # Print
menu head
    print("= 1.  ADD MARKS TO THE LIST.                  =")  # Print
option 1
    print("= 2.  SHOW ALL MARKS IN THE LIST.             =")  # Print
option 2
    print("= 3.  PRINT THE MEAN OF THE NUMBERS.          =")  # Print
option 3
    print("= 4.  PRINT THE MEDIAN OF THE NUMBERS.        =")  # Print
option 4
    print("= 5.  PRINT THE MODE OF THE NUMBERS.          =")  # Print
option 5
    print("= 6.  PRINT THE SKEWNESS OF THE NUMBERS.      =")  # Print
option 6
    print("= 7.  ADD MORE NUMBERS TO THE LIST BY COMMAS. =")  # Print
option 7
    print("= 8.  GO BACK AND ENTER A NEW SET OF NUMBERS. =")  # Print
option 8
    print("= 9.  READ DATA FROM A FILE.                  =")  # Print
option 9
    print("= 10. EXIT THE APPLICATION.                   =")  # Print
option 10
    print("=                                             =")  # Print
menu footer
    print("=                                             =")  # Print
menu footer
    print("===============================================")  # Print
menu footer

# Exit the function and the program
```



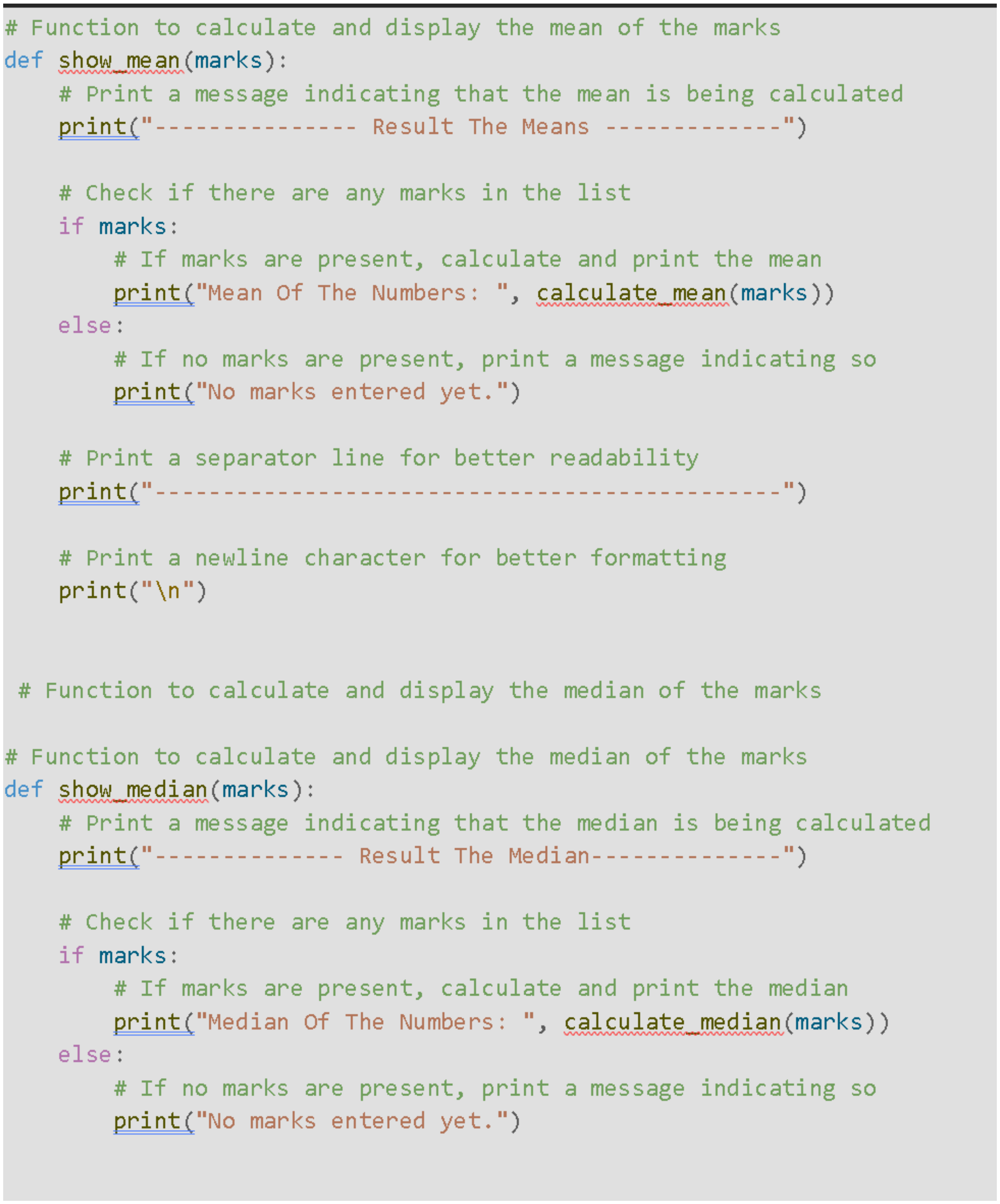
```python
def goodbye_user():
    """
    Prints a farewell message and exits the application.

    Returns:
        None
    """
     # Print a decorative line
    print("==============================================")
    print("=                                            =")
    print("=                                            =")
    print("=                THANK YOU!                  =")
    print("=                                            =")
    print("=          EXITING THE APPLICATION.          =")
    print("=                                            =")
    print("=                  GOOD BYE!                 =")
    print("=                                            =")
    print("=                     :)                     =")
    print("=                                            =")
    print("=                                            =")
    print("==============================================")
    # Print a newline character for better formatting
    print("\n")
    # Exit the program
    exit()


# Controllers
# STUDENTS MARKS STATISTICS APPLICATION V 1.0
# This program allows users to input, manipulate, and analyze a list of
student marks.
# Author: Emilio Antonio Barrera Sepúlveda
# Date Programmed: 3/25/2024

# Importing necessary functions from Models module
from Models.calculate_model import *
```



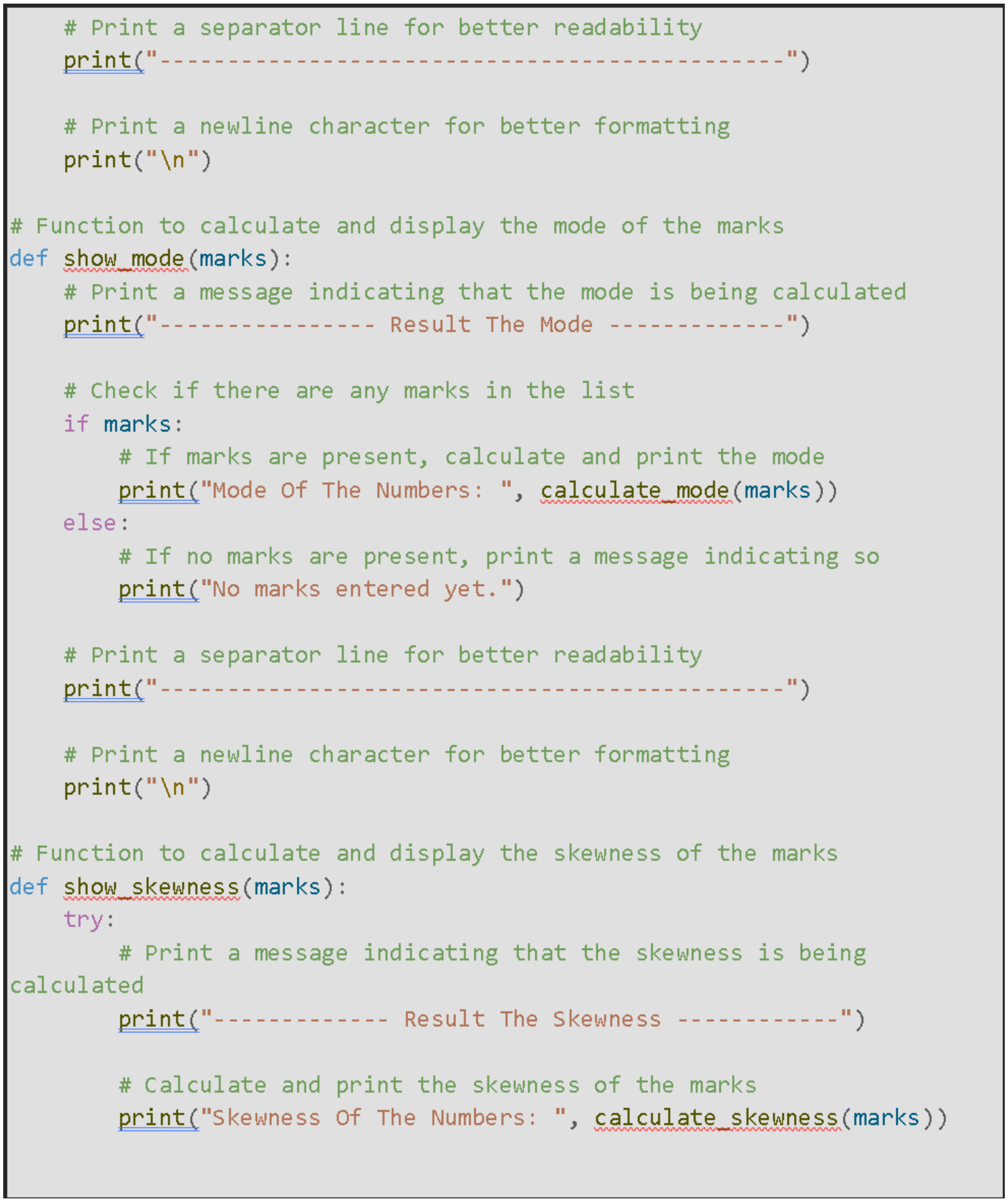
```python
# Importing print_menu function from Views module
from Views.menu_view import print_menu, autor_welcome, goodbye_user

"""Main function to orchestrate the interaction between the Model and
the View."""
def main():

    # Display welcome message and information about the program
    autor_welcome()

    # Initialize an empty list to store marks
    marks = []

    # Start an infinite loop for displaying the menu and handling user
choices
    while True:

        # Check if the length of the 'marks' list is less than 2
        # If there are less than 2 numbers in the list, prompt the user
to add at least two numbers
        # before presenting the menu
        if len(marks) < 2:
            # Prompt the user to add at least two numbers before
presenting the menu
            print("\n")
            print("--------------------------- Enter Marks -------------
---------------")
           # This line prints the message to the console, informing the
user to add more numbers.
            print("Please Add At Least Two Or More Numbers Before
Presenting The Menu.")
            enter_marks_to_list(marks)
            print("--------------------------------------------------
---------------")
            continue
        # Display the menu options
        print_menu()

        # Prompt the user to enter their choice
        choice = input("Enter Your Choice: ")
        print("\n")

        # Dictionary to map user choices to corresponding functions
        switch = {
            '1': add_marks,
            '2': show_marks,
            '3': show_mean,
            '4': show_median,
            '5': show_mode,
            '6': show_skewness,
            '7': add_marks_by_comma,
            '8': enter_new_set,
            '9': read_from_file,
            '10': exit_program,
        }

       # Execute the chosen function based on the user's input
        if choice in switch:
         # If the user's choice is a valid option in the menu, the
corresponding function is called
         # with 'marks' list as argument
         switch[choice](marks)
        else:
        # If the user's choice is not valid, an error message is
displayed
         print("Invalid Choice. Please Enter A Number From 1 To
10.")
```



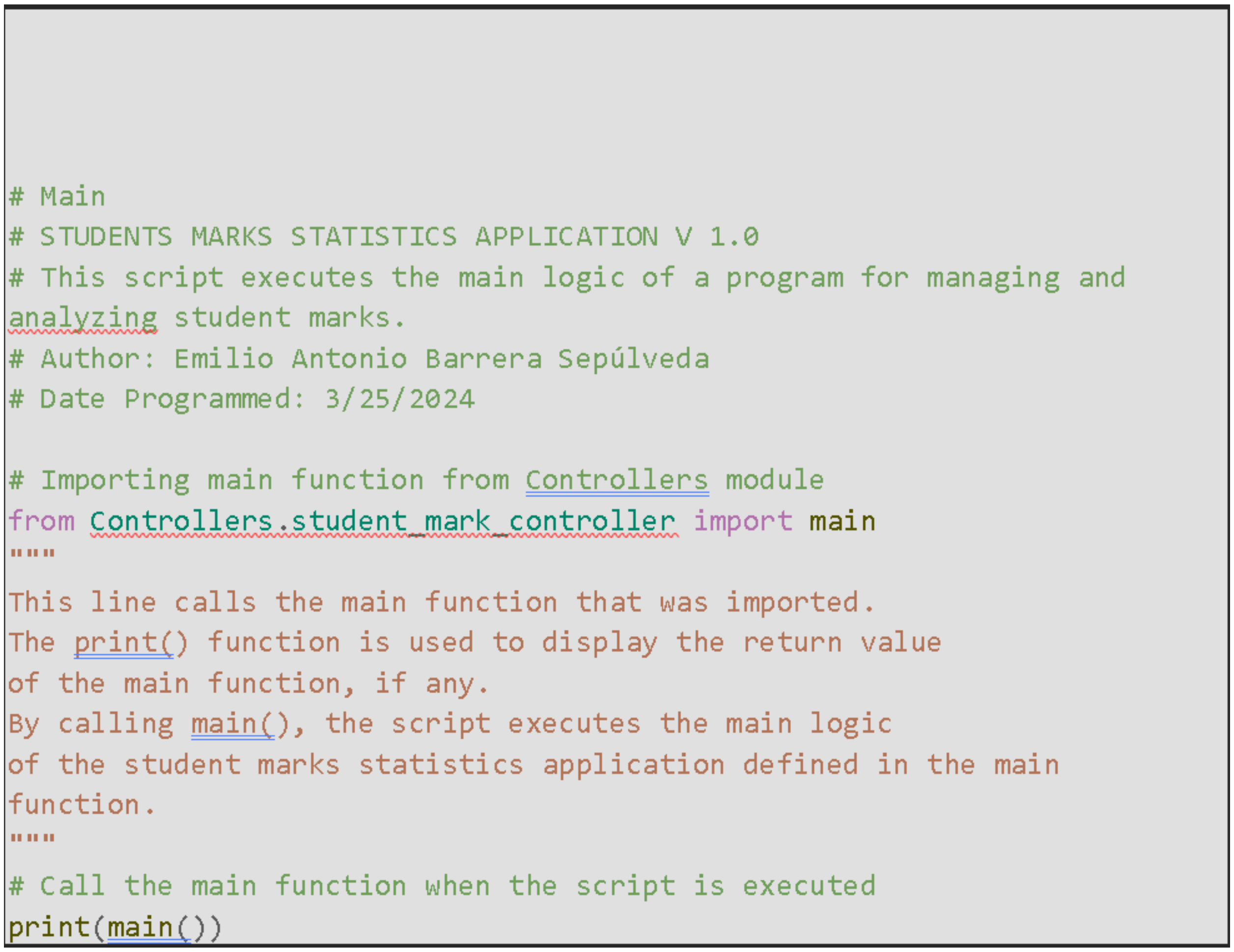
```
# Function to add marks to the list
def add_marks(marks):
    # Print a message indicating that marks are being added
    print("------------------- Adding Marks ----------------------")
    # Call the function to allow the user to input marks and add them to
the list
    enter_marks_to_list(marks)
    # Print a separator line for better readability
    print("---------------------------------------------------------
")
    # Print a newline character for better formatting
    print("\n")

# Function to display the entered marks
def show_marks(marks):
    # Print a message indicating that the marks are being shown
    print("----------------- Show Marks -----------------")

    # Check if there are any marks in the list
    if marks:
        # If marks are present, call the function to display all marks
        Show_All_Marks(marks)
    else:
        # If no marks are present, print a message indicating so
        print("No marks entered yet.")

    # Print a separator line for better readability
    print("----------------------------------------------")

    # Print a newline character for better formatting
    print("\n")
```



```python
# Function to calculate and display the mean of the marks
def show_mean(marks):
    # Print a message indicating that the mean is being calculated
    print("--------------- Result The Means -------------")

    # Check if there are any marks in the list
    if marks:
        # If marks are present, calculate and print the mean
        print("Mean Of The Numbers: ", calculate_mean(marks))
    else:
        # If no marks are present, print a message indicating so
        print("No marks entered yet.")

    # Print a separator line for better readability
    print("----------------------------------------------")

    # Print a newline character for better formatting
    print("\n")


 # Function to calculate and display the median of the marks

# Function to calculate and display the median of the marks
def show_median(marks):
    # Print a message indicating that the median is being calculated
    print("-------------- Result The Median--------------")

    # Check if there are any marks in the list
    if marks:
        # If marks are present, calculate and print the median
        print("Median Of The Numbers: ", calculate_median(marks))
    else:
        # If no marks are present, print a message indicating so
        print("No marks entered yet.")
```



```
    # Print a separator line for better readability
    print("----------------------------------------------")

    # Print a newline character for better formatting
    print("\n")

# Function to calculate and display the mode of the marks
def show_mode(marks):
    # Print a message indicating that the mode is being calculated
    print("---------------- Result The Mode -------------")

    # Check if there are any marks in the list
    if marks:
        # If marks are present, calculate and print the mode
        print("Mode Of The Numbers: ", calculate_mode(marks))
    else:
        # If no marks are present, print a message indicating so
        print("No marks entered yet.")

    # Print a separator line for better readability
    print("----------------------------------------------")

    # Print a newline character for better formatting
    print("\n")

# Function to calculate and display the skewness of the marks
def show_skewness(marks):
    try:
        # Print a message indicating that the skewness is being
calculated
        print("------------- Result The Skewness ------------")

        # Calculate and print the skewness of the marks
        print("Skewness Of The Numbers: ", calculate_skewness(marks))
        # Print a separator line for better readability
        print("----------------------------------------------")
    except ValueError as ve:
        # If there's a ValueError (likely due to insufficient data),
print the error message
        print(ve)
        # Prompt the user to enter more marks
        enter_marks_to_list(marks)

# Function to add marks to the list using comma-separated input
def add_marks_by_comma(marks):
    try:
        # Print a message indicating that marks are being added using
comma-separated input
        print("---------- Adding Marks In by Comma ----------")

        # Call the function to allow the user to input marks using
comma-separated input and add them to the list
        enter_marks_to_list_by_comma(marks)

        # Print a separator line for better readability
        print("----------------------------------------------")
    except ValueError as ve:
        # If there's a ValueError (likely due to incorrect input
format), print the error message
        print(ve)
        # Print another separator line for better readability
        print("----------------------------------------------")
        # Print a newline character for better formatting
        print("\n")

# Function to clear the list and enter a new set of marks
def enter_new_set(marks):
    # Print a message indicating that a new set of marks is being
entered
    print("--- Enter a New Set Of Numbers In the List ---")
    # Clear the existing marks list
    marks.clear()

    # Print a message indicating that the marks list is now empty
    print("You Have Chosen To Enter a New Set Of Numbers (Empty List).")

    # Print a separator line for better readability
    print("----------------------------------------------")

    # Print a newline character for better formatting
    print("\n")

    # Return the empty marks list
    return marks

# Function to read marks from a file and add them to the list
def read_from_file(marks):
    try:
        # Print a message indicating that data is being read from a file
        print("---------= Reading Data From A File ----------")

        # Prompt the user to enter the filename from which data will be
read
        filename = input("Enter The Filename To Read Data From: ")

        # Read data from the file and add it to the marks list
        marks += read_data_from_file(filename)

        # Print a newline character for better formatting
        print("")
        # Show all the marks in the list
        Show_All_Marks(marks)


        # Print a separator line for better readability
        print("----------------------------------------------")

        # Print a newline character for better formatting
        print("\n")

        # Return the updated marks list
        return marks
    except ValueError as ve:
        # If there's a ValueError (likely due to incorrect input), print the error message
        print(ve)
        # Print a separator line for better readability
        print("----------------------------------------------")

# Function to exit the program
def exit_program(marks):
    # Clear the marks list
    marks = []

    # Call the function to display a goodbye message to the user
    goodbye_user()

    # Print a newline character for better formatting
    print("\n")
```



```python
# Main
# STUDENTS MARKS STATISTICS APPLICATION V 1.0
# This script executes the main logic of a program for managing and
analyzing student marks.
# Author: Emilio Antonio Barrera Sepúlveda
# Date Programmed: 3/25/2024

# Importing main function from Controllers module
from Controllers.student_mark_controller import main
"""
This line calls the main function that was imported.
The print() function is used to display the return value
of the main function, if any.
By calling main(), the script executes the main logic
of the student marks statistics application defined in the main
function.
"""
# Call the main function when the script is executed
print(main())
```