STUDENTS MARKS STATISTICS APPLICATION V 1.0

CS4051 23-24 Spring Coursework in Python

By

Emilio Antonio Barrera Sepúlveda
22047090

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

Table of Content

1. Introduction ---------- 5

2. Aim ---------- 6

3. Literature Review---------- 6

3.1 Mean ---------- 6

3.1 Median ---------- 6

3.3 Mode ---------- 6

3.4 Skewness ---------- 6

4. The GitHub Link---------- 7

5. Model ---------- 7

5.1 Models ---------- 7

5.2 Views ---------- 7

5.3 Controllers---------- 7

5.4 Main Program---------- 8

6. Pseudocode ---------- 9

7. Data (Input-Output Data Structures)----------12

7.1 Input data structure ----------12

7.2 Output data structure----------12

7.3 Example ----------12

7.4 Data validation ----------13

8. Description ----------14

8.1 Provides an overview of the program----------14

8.1.1 Main entry point of the application ----------14

8.1.2 Views----------15

8.1.2.1 menu_view.py----------15

8.1.3 Models----------16

8.1.3.1 calculate_model.py ----------16

8.1.4 Controllers ----------18

8.1.4.1 student_mark_controller.py ---------- 18

8.2 Covers input and output operations ----------20

8.2.1 Input Operations ---------- 20

8.2.2 Output Operations----------23

9. Testing ----------30

9.1 Test 1: Running my program from Terminal ----------30

9.1.1 My first step was changing the directory to the location of my project --31

9.1.2 My next step was compiling my main Python file ----------31

9.1.3 My final step was running the file ----------31

9.2 Test 2: Enter Marks----------32

9.3 Test 3: Adding marks to the list ---------- 33

9.4 Test 4: Displaying all marks the Lis ----------34

9.5 Test 5: The Means ----------35

9.6 Test 6: The Median ----------35

9.7 Test 7: The Mode----------37

9.8 Test 8: The Skewness ----------38

9.9 Test 9: Add marks in by comma ----------39

9.10 Test 10: Add marks in by comma----------40

9.11 Test 11: Read data from a file----------41

9.12 Test 12: Exit----------42

9.13 Test 13: Error Handling----------43

10. Conclusion----------46

11 References----------47

12. Appendix----------48

1. INTRODUCTION

This Python app uses the MVC pattern, managing data, interfaces, and logic. The Model processes data and stats, while the View offers a user-friendly interface. The Controller mediates between them, ensuring smooth interaction. With MVC, it's organized, scalable, and efficient for student assessment management and analysis.

2. Aim

The Python [1] program employs MVC, enabling users to input marks for statistical calculations like mean, median, mode, and skewness.

3. Literature Review

Understanding statistical [3] measures like mean, median, mode, and skewness is crucial for interpreting data distributions, offering unique perspectives for analysis.

3.1 Mean

The mean is the sum of all values divided by the total count, indicating central tendency, but sensitive to outliers.

3.2 Median

The median is the middle value in a sorted dataset, robust against outliers, unlike the mean, providing stability for skewed datasets.

3.3 Mode

The mode is the most frequent value in a dataset, determined by its highest frequency, applicable to both numerical and categorical data.

3.4 Skewness

Skewness measures dataset asymmetry using mean, median, and standard deviation. Positive values indicate right skewness, negative values imply left skewness, and 0 suggests symmetry.

4. The GitHub links Code

https://github.com/emiliobs/CS405StudentMarksStatisticsCoursework

5. Model

The Model-View-Controller (MVC) [2] architectural pattern divides applications into three components:

5.1 Model:

- Functions calculate statistical measures like mean, median, mode, and skewness.
- A function extracts marks from a file into a list.

5.2 View

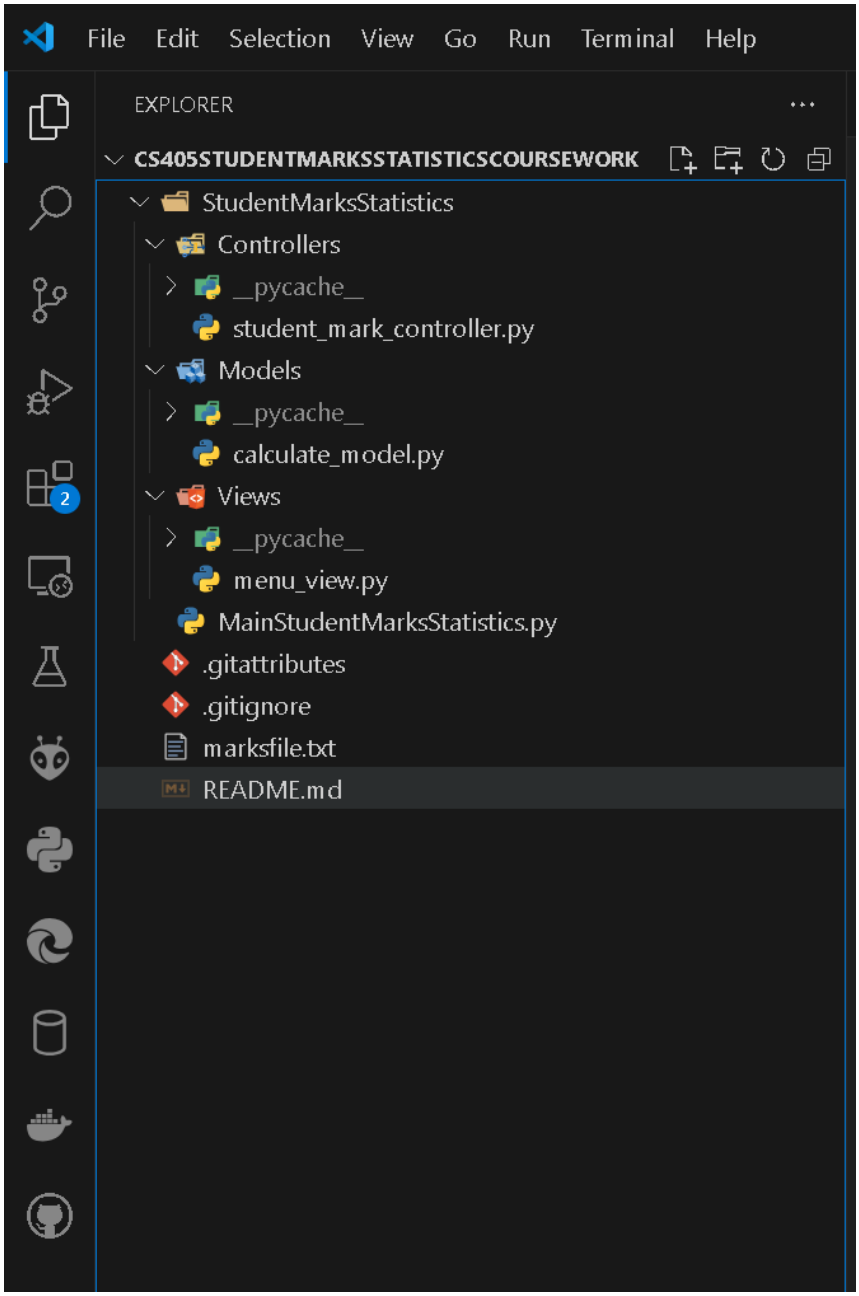
Functions display a user interface, prompt mark input, and show statistics or errors.

5.3 Controller

- initializes a list to store marks.
- Implements a loop for menu options and user input handling.

5.4 Main Program

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



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)

Explaining input-output data structures involves clarifying data format for system input and output presentation for user comprehension.

7.1 Input Data Structure

The summary outlines the input data structure for the student marks calculation application. Users input individual marks sequentially via the keyboard, with each mark expected to be a numerical value. Entry ends with a specified keyword like "done", facilitating user interaction and data input.

7.2 Output Data Structure

The summary outlines the presentation format for output data, including the mean of entered marks. Typically, mean values are shown as floating-point numbers with additional context like total marks. This output is displayed as formatted text on the console, ensuring clarity for users.

7.3 Example

- Input Data

  Users input marks sequentially (e.g., 55, 5, 45, 45) individually, concluding with "done" to indicate entry completion.

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS    PROBLEMS    OUTPUT

=                    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:

  The mean, calculated as 37.5, is displayed alongside the total number of marks entered: "You have entered 3 mark(s). The mean is: 37.5".

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS    PROBLEMS    OUTPUT

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

7.4 Data Validation

Input data validation ensures criteria compliance (e.g., numeric format, range). Error handling addresses invalid inputs, providing user feedback.

```
--------------------------- 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

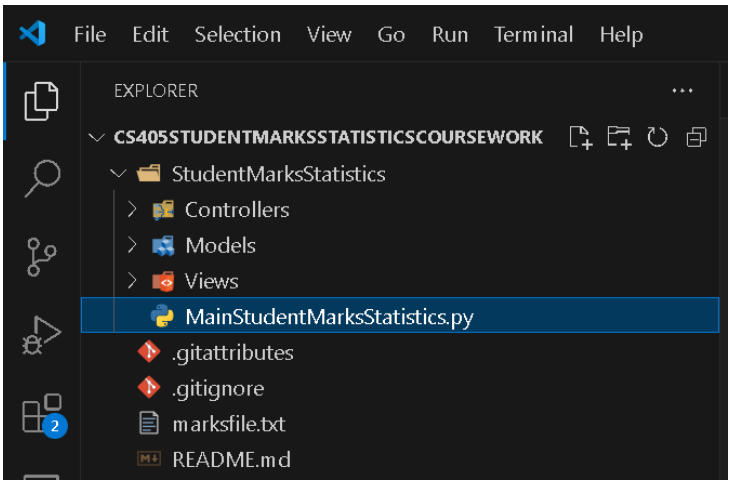
The application follows MVC, ensuring modularity and efficiency in managing student marks data with diverse input and output options.

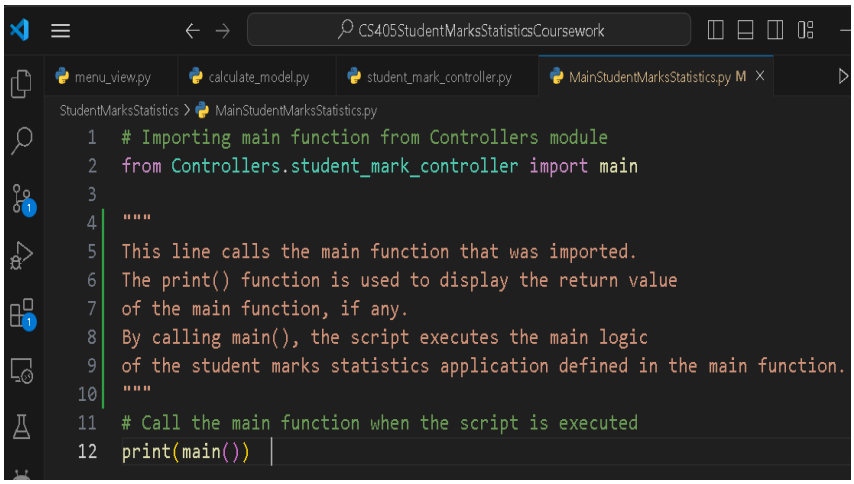
8.1 Provides an overview of the program

The program prompts for input, validating for numeric values, allowing analysis and manipulation with a menu interface for seamless interaction.

8.1.1 Main entry point of the application

The code imports the main function from Controllers, serving as the entry point, managing initialization, input processing, calculations, and results.



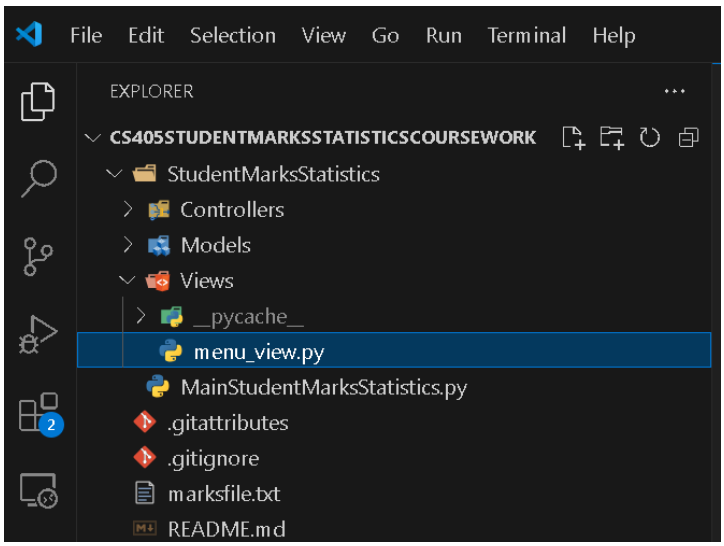


8.1.2 Views

- 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

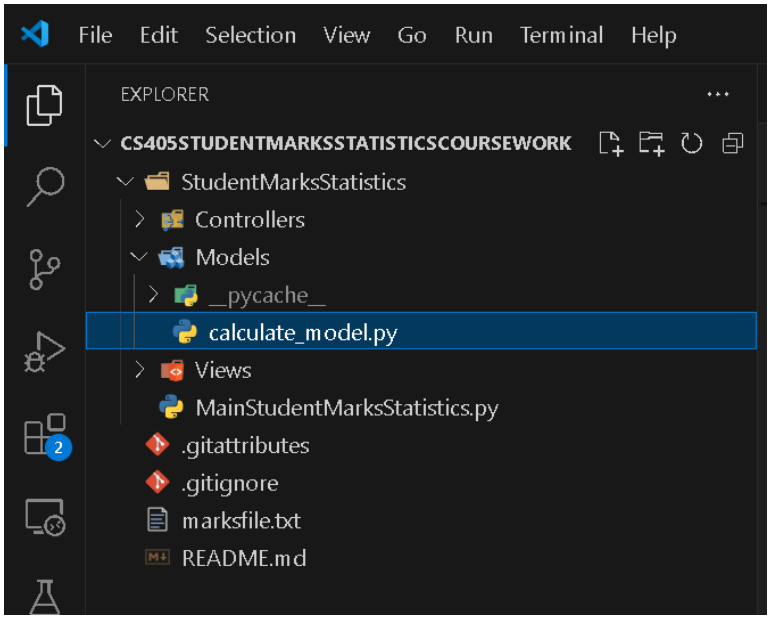
The provided code defines functions for a student marks statistics application. autor_welcome() displays program information and a welcome message. print_menu() presents menu options for the application. goodbye_user() prints a farewell message and exits the program gracefully, enhancing user interaction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

```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
```

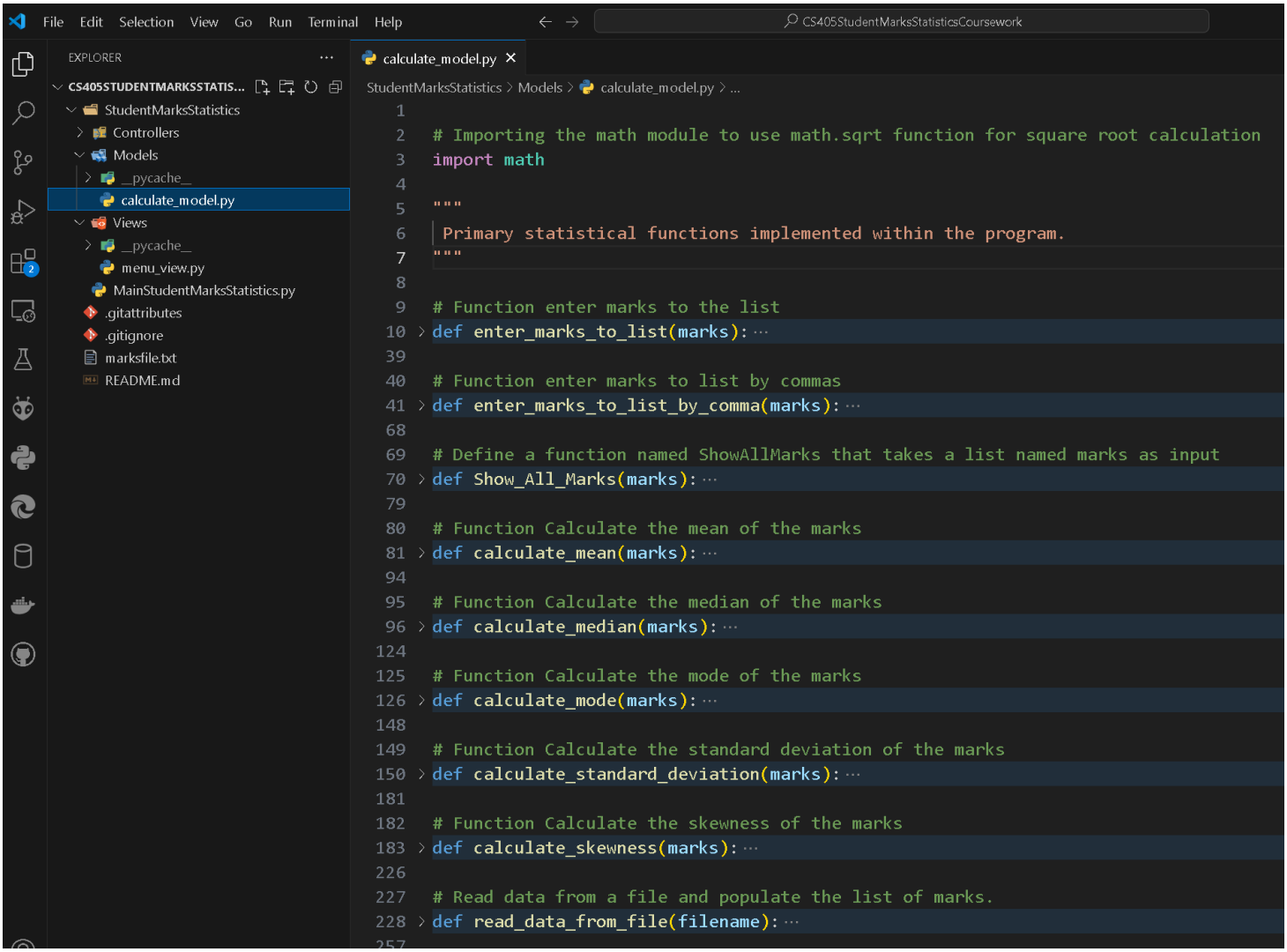
8.1.3 Models

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



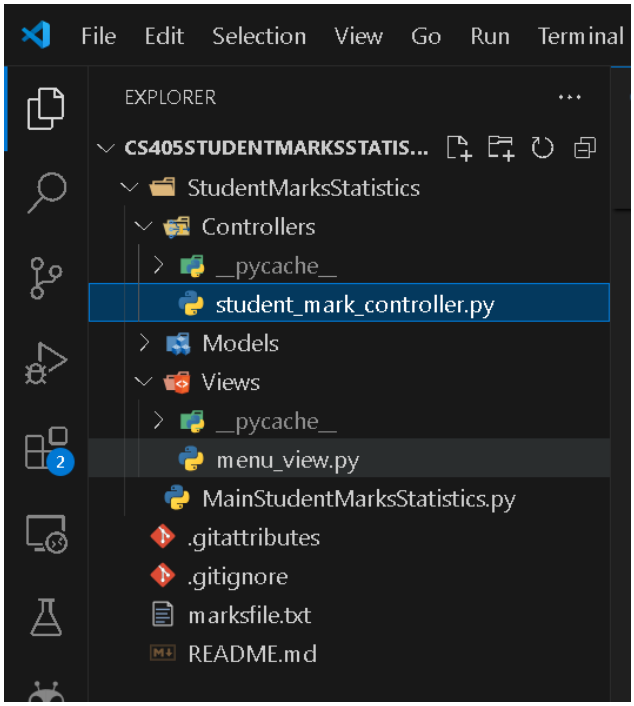
8.1.3.1 calculate_model.py

The code facilitates analysis of student marks, offering functionalities for entering marks individually or as comma-separated lists, displaying marks, and calculating various statistics including mean, median, mode, standard deviation, and skewness. It also includes file reading capabilities for data retrieval from CSV files.


File Edit Selection View Go Run Terminal Help
CS405StudentMarksStatisticsCoursework
EXPLORER
CS405STUDENTMARKSSTATIS...
StudentMarksStatistics
Controllers
Models
__pycache__
calculate_model.py
Views
__pycache__
menu_view.py
MainStudentMarksStatistics.py
.gitattributes
.gitignore
marksfile.txt
README.md
calculate_model.py
StudentMarksStatistics > Models > calculate_model.py > ...
# 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

The Python code serves as a comprehensive statistical tool, utilizing a menu-driven interface and modular design principles. Importing functions from calculate_model and menu_view modules, it handles user interactions, statistical calculations, input validation, file handling, and provides an exit option. This user-friendly tool allows users to input, manipulate, and analyze numerical data effectively.

```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): ...
```

```python
# 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): ...
```

8.2 Covers input and output operations

The code facilitates user interaction through console input and provides feedback and results via print statements. Users input marks, select menu options, and potentially provide file names. The program processes this input, performs calculations or file operations, and presents results or feedback accordingly.

8.2.1 Input Operations

The code snippet ensures adequate data for statistical analysis by prompting users to input numerical marks and validating each entry. It facilitates user interaction through mark input and menu selection, enhancing user experience and program functionality for managing student marks.

- At least two marks are entered before code

```
    # 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.

```python
        # 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)**

```python
# 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 presenting information through console printing, file writing, or graphical interfaces. In the provided code snippet, console printing includes messages, prompts, and statistical results, aiding user interaction. Feedback includes confirming mark additions, displaying errors, and guiding menu interactions. Overall, output operations facilitate user interaction and information dissemination.

- 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 [4] 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 computes the median of a list [4] of marks. It checks for an empty list, sorts it, then calculates the median based on even or odd mark counts.

```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 finds the mode of a list of marks by counting occurrences. It creates a dictionary to tally mark frequencies, then retrieves the mark(s) with the highest count as mode(s). If multiple modes, it returns the first; otherwise, None.

```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 skewness, assessing distribution asymmetry. It validates data points, computes mean, and initializes variables for skewness formula. It then calculates skewness value based on differences between marks and mean.

```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 retrieves numerical data from a designated file, returning a list of marks. It handles file opening, FileNotFoundError, and data parsing. Empty files prompt a message, while invalid data triggers an error message.

```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.

  Error handling in the code ensures users are notified of input issues like non-numerical values, invalid file names, or file reading errors. Descriptive error messages improve user experience and troubleshooting.

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

```python
            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.")
```

```python
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.")
```

```python
            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 code includes a function, goodbye_user(), which prints a farewell message, a decorative line, and a smiley face before exiting the application using exit(). This enhances user experience and expresses gratitude 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 Student Marks Statistics Application covers unit, integration, UI, boundary, error handling, file I/O, usability, and documentation testing.

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                    python

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
---------------------------------------------
```

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS

================= 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
--------------------------------------------
```

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS

================= 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
---------------------------------------------
```

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS

================= 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
---------------------------------------------
```

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS

================= 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.

```
TERMINAL    PORTS    SERIAL MONITOR    SQL CONSOLE    COMMENTS    PROBLEM

================= 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.):
main  0 0  0  Connect
```

```
------------- 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

The Python application, employing MVC, offers streamlined academic data management and analysis. Modular design and separation of concerns ensure efficient processing, interaction, and presentation. Statistical functionalities like mean, median, mode, and skewness, alongside user-friendly features, enhance usability and productivity, embodying effective software design principles for empowered users and administrators.

11. References

[1] Learn Python - Free Interactive Python tutorial (2024) Learn Python - Free Interactive Python Tutorial. Available at: https://www.learnpython.org/.

[2] Real Python (2024) *Model-view-controller (MVC) in Python web apps: Explained with Lego*, *Real Python*. Available at: https://realpython.com/lego-model-view-controller-python/).

[3] CalculatorSoup, L. (2023) Mean, median, mode calculator, CalculatorSoup. Available at: https://www.calculatorsoup.com/calculators/statistics/mean-median-mode.php.

[4] GfG (2023) Python lists, GeeksforGeeks. Available at: https://www.geeksforgeeks.org/python-lists/.

12. Appendix

This coursework provides Python source code for a student marks statistics application. It's well-structured with explanatory comments. The application calculates grades based on marks, demonstrates Python concepts, and can be adapted for various calculations, broadening its utility and educational value for readers.

```
# 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 *
```

```
# 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())
```