COURSE WORK - CS4051

WORK OF SANDRO ZAKAIDZE

ID:23028884

CS4051

FUNDAMENTALS OF COMPUTING

2024 – SPRING

## Table of Contents

Abstract............................................................................................................ 2

Introduction.......................................................................................................3

Methodology......................................................................................................4

Implementation .................................................................................................5

Testing...............................................................................................................6

Conclusion........................................................................................................7

References ........................................................................................................8

Appendix ...........................................................................................................9

GitHub Repository .............................................................................................10

## Abstract

*This project presents a tool designed to provide comprehensive insight into students' performance data. Implemented using Python programming language, this tool enables users to input a series of numerical grades and analyze them through statistical measures, such as a mean, median and mode. The application also includes a user-friendly menu system, which guides users through different functionalities, including recalculating new data sets and exiting the tool. Using an intuitive command-line interface, this application ensures accuracy and accessibility in calculating essential statistics. The clear and visually distinct menu structure enhances usability and efficiency.*

## Introduction

The CS4051 course work for the academic year 2023-2024 presents a practical application of Python programming. The application begins by prompting the user to enter a list of numerical marks one at a time. The user is guided through entering grades until they indicate completion by typing “done”. The application verifies that at least two grades have been entered before proceeding. Once the data entry phase is complete, the user is presented with a menu offering the following options: calculating the mean, median and mode. Additionally, the user can choose to re-enter a new set of marks or exit the program entirely. The menu is implemented using a loop to control the display until a valid choice is selected. In the second part of the application, only numerical values are accepted, prompting the users to correct any invalid input, and checks that the minimum numbers of marks are entered. Furthermore, this part introduces the calculation of skewness, which offers additional insight into the dataset’s distribution. For the last part of application, we further extended the flexibility of the application in several ways. First, it allows users to enter multiple marks at once as a comma-separated string. Simplifying data entry. The application also provides an option to append new marks to the existing dataset and read data directly from a file on the computer's hard drive, enhancing data management capabilities.

## Methodology

An Agile methodology was used to guide the development process, emphasizing iterative progress through three primary stages that built upon one another:

1. Part 1: The basic framework for data entry and initial statistical analysis.
2. Part 2: Enhanced robustness by implementing input validation, ensuring meaningful data entry. And introducing more sophisticated calculations.
3. Part 3: Improved data input options and expanded the application’s flexibility through file reading capabilities.

The Agile approach was chosen for its flexibility and ability to incorporate continuous feedback. This allowed iterative refinement and incremental additions to the application, ensuring that each part met specific requirements before moving on.

* **Application Structure**

The Application has a **hierarchical structure** that consists of well-defined functions encapsulated in a modular manner. This structure separates data entry, statistical calculations, and menu interactions into distinct functions, ensuring clarity and maintainability. The main function coordinates the execution flow, invoking specific functions based on user input.

## Implementation

**1. Global Variables:**

**“****student\_grades”:** An empty dictionary used to store student names and their respective grades**.**

**“subjects”:** A predefined list of subjects for which grades are entered.

**2. Adding\_grades\_of\_students() Function:**

* **Purpose:** To collect and store individual student’s grades for each predefined subject.
* **Details**:

1. Prompts the user for a student’s name
2. Initializes a dictionary to store grades for the current student.
3. Iterates over the list of subjects, collecting and storing each grade using user input.
4. Updates the **“students\_grades”** dictionary with this student’s grades.

* **Interaction:** This function uses the **“Subjects”** list and updates the global **“students\_grades”** dictionary.

**3. Loop to collect Student Grades:**

* **Purpose**:
  + - Repeatedly calls “adding\_grades\_of\_students()” to collect data for multiple students.
    - Details:
      * + Prompts the user for the number of students.
        + Iterates to call  **“adding\_grades\_of\_students()”** for each student.

4**. “print” statement to display grades:**

* **Purpose:** Displays all student names and their grades**.**
* **Details:**
  + - Loops through **“students\_grades”** to print each student’s name and their grades.

5. **Statistical Functions:**

* **“calculate\_mean(grades)”:**

Calculates and returns the mean(average) of a list of grades.

* **“calculate\_median(grades)”:**

Returns the median value of a sorted list of grades.

* **“calculate\_mode(grades)”:**

Calculates and returns the mode(s) (most frequently occurring values of a list of grades.

* **“calculate\_skeewness(grades)”:**

Returns the skewness (asymmetry of the distribution) of a list of grades

**6.“display\_menu\_and\_get\_****choice()” Function**:

* + - **Purpose**: Displays a menu to the user and gets their choice.

**Details**:

* + - * Prints five different options to the user.
      * Prompts the user to input a choice and returns it.

**7.** **Main() Function:**

* **Purpose: Starts the entire application workflow.**
* **Details:**

Based on user choice:

Option 1: calculates and prints the mean using **calculate\_mean()**.

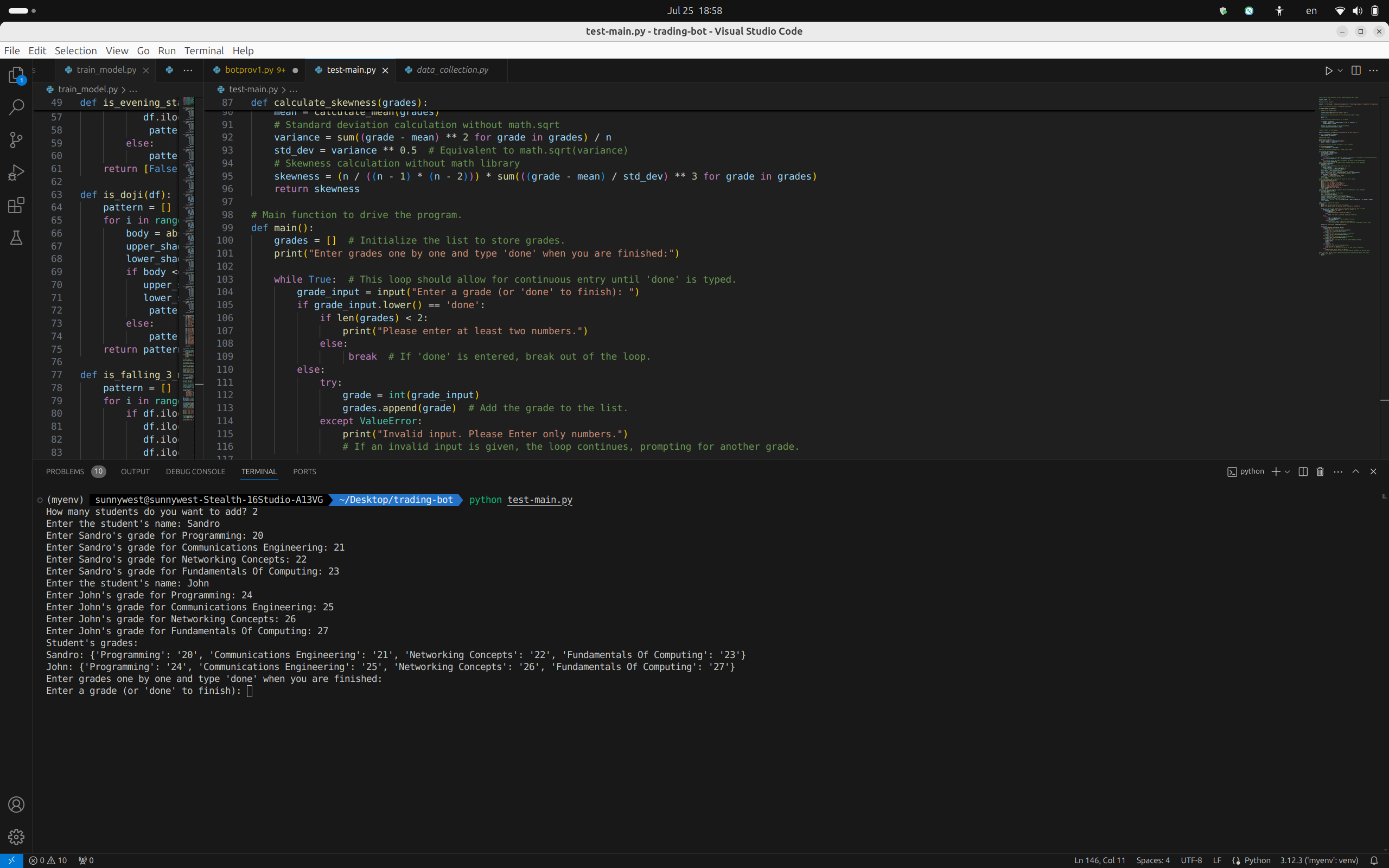
Option 2: calculates and prints the median using **calculate\_median()**.

Option 3: calculates and prints the mode using **calculate\_mode()**.

Option 4: clears the list and re-invokes **main()** to restart grade entry.

Option 5: exit the application.

# Testing



# Conclusion

The development of the Student Grades Management and Statistical Analysis tool has been a successful endeavor in applying fundamental computing concepts and Python programming skills. This project demonstrates the effective use of Python for data entry, validation, and statistical analysis, providing a robust and user-friendly application for managing student grades.

The tool's ability to handle multiple students and subjects, coupled with its statistical analysis capabilities, makes it a versatile and valuable application. By implementing a clear and intuitive menu system, users can easily navigate through various functionalities, ensuring an accessible and efficient user experience.

Through rigorous testing, the application has proven to be reliable, accurately calculating mean, median, mode, and skewness for the entered grades. The implementation of input validation further enhances the robustness of the tool, preventing errors and ensuring meaningful data entry.

The project has also demonstrated the importance of an iterative development process, using an Agile methodology to refine and expand the application's features. This approach allowed for continuous improvement and incorporation of feedback, resulting in a well-rounded and comprehensive final product.

In conclusion, this project not only fulfills the academic requirements of the CS4051 coursework but also provides practical experience in software development and statistical analysis. The successful completion of this project underscores the effectiveness of Python as a programming language for developing functional and user-friendly applications. This tool can be further extended and enhanced, potentially incorporating additional statistical measures and more advanced data management capabilities, providing even greater value to its users.

The skills and knowledge gained through this project will be invaluable in future software development endeavors, and the completed application serves as a testament to the potential of combining sound programming practices with effective project management techniques.

**References**

# Objects First with Java: A Practical Introduction Using BlueJ, Global Edition Paperback – International Edition, 24 Jun. 2016

1. Python Programming: An introduction to computer science by John Zelle
2. Python Programming Third Edition by Michael Dawson.

# Appendix

# Initialize an empty dictionary to store student names and their grades.

students\_grades = {}

#Define a list of subjects.

subjects = ["Programming", "Communications Engineering", "Networking Concepts", "Fundamentals Of Computing"]

# This will define the function to store grades and students

def adding\_grades\_of\_students():

# Ask for the student's name

student\_name = input("Enter the student's name: ")

# Initialize an empty dictionary to store the current student's grades.

grades = {}

# Loop through each subject and ask for the grade.

for subject in subjects:

grade = input(f"Enter {student\_name}'s grade for {subject}: ")

grades[subject] = grade

# Store the student's grades in the main dictionary.

students\_grades[student\_name] = grades

# adding students and their grades.

number\_of\_students = int(input("How many students do you want to add? "))

for i in range(number\_of\_students):

adding\_grades\_of\_students()

# Display the grades

print("Student's grades: ")

for student, grades in students\_grades.items():

print(f"{student}: {grades}")

# Function to calculate the mean (average) of a list of grades.

def calculate\_mean(grades):

return sum(grades) / len(grades)

# Function to calculate the median of a sorted list of grades.

def calculate\_median(grades):

sorted\_grades = sorted(grades)

n = len(sorted\_grades)

mid = n // 2

if n % 2 == 0:

# If the list has an even number of elements, the median is the average of the two middle elements.

return(sorted\_grades[mid - 1] + sorted\_grades[mid]) / 2

else:

# If the list has an odd number of elements, the median is the middle element.

return sorted\_grades[mid]

# This function will calculate the mode(s) (most frequent values in a list of grades.

def calculate\_mode(grades):

frequency = {}

# This counts the frequency of each grade in the list.

for grade in grades:

frequency[grade] = frequency.get(grade, 0) + 1

# This will determine the maximum frequency.

max\_frequency = max(frequency.values())

# This identifies all grades that have the maximum frequency.

modes = [key for key, val in frequency.items() if val == max\_frequency]

# This is will output message if all grades are equally common.

if len(modes) == len(grades):

return 'All numbers are equally common'

# Otherwise, return the list of mode.

return modes

# This will display a menu of options and return the user's choice.

def display\_menu\_and\_get\_choice():

print("\nMenu of choices: ")

print("1. Print the mean of the numbers")

print("2. Print the median of the numbers")

print("3. print the mode of the numbers")

print("4. Go back and enter a new set of numbers")

print("5. Exit the application")

choice = input("Enter your choice: ")

return choice

# Function to calculate skewness (asymmetry of the distribution) of a list of grades.

def calculate\_skewness(grades):

n = len(grades)

# This calculates the mean of the grades.

mean = calculate\_mean(grades)

# Standard deviation calculation without math.sqrt

variance = sum((grade - mean) \*\* 2 for grade in grades) / n

std\_dev = variance \*\* 0.5 # Equivalent to math.sqrt(variance)

# Skewness calculation without math library

skewness = (n / ((n - 1) \* (n - 2))) \* sum(((grade - mean) / std\_dev) \*\* 3 for grade in grades)

return skewness

# Main function to drive the program.

def main():

grades = [] # Initialize the list to store grades.

print("Enter grades one by one and type 'done' when you are finished:")

while True: # This loop should allow for continuous entry until 'done' is typed.

grade\_input = input("Enter a grade (or 'done' to finish): ")

if grade\_input.lower() == 'done':

if len(grades) < 2:

print("Please enter at least two numbers.")

else:

break # If 'done' is entered, break out of the loop.

else:

try:

grade = int(grade\_input)

grades.append(grade) # Add the grade to the list.

except ValueError:

print("Invalid input. Please Enter only numbers.")

# If an invalid input is given, the loop continues, prompting for another grade.

print(f"You have entered {len(grades)} grades.")

while True:

choice = display\_menu\_and\_get\_choice()

if choice == '1':

# Option 1: This prints the mean of the grades.

print(f"Mean: {calculate\_mean(grades)}")

elif choice == '2':

# Option 2: This will print the median of the grades.

print(f"Median: {calculate\_median(grades)}")

elif choice == '3':

# Option 3: This will print the mode of the grades.

print(f"Mode: {calculate\_mode(grades)}")

elif choice == '4':

# Option 4: This will clear the list and restart the main function.

grades.clear()

main()

continue

elif choice == '5':

# Option 5: This will exit the application.

print("Exiting the application.")

break # If '5' is entered, break out of the loop to exit the program.

else:

print("Invalid choice. Please try again.")

# If an invalid choice is given, the loop continues, prompting for the menu again.

# This will check if the script is being run directly (not imported) and then it runs main();

if \_\_name\_\_ == "\_\_main\_\_":

main()

# GitHub Repository:

https://github.com/Rememberwhy/python-uni-project