GRADES CALCULATOR

Contents

Abstract

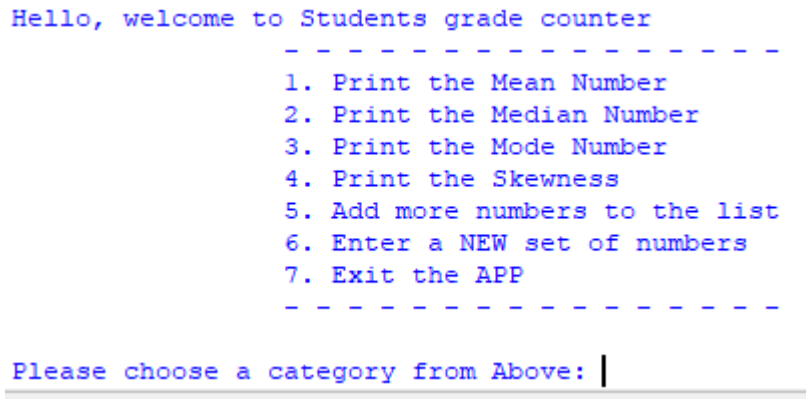
"In the course Fundamentals of Computing, we embarked on a journey from the inception of the first computer to the advancements in machine learning and artificial intelligence. Initially, we delved into understanding how hardware collaborates with software and the essential functions of a computer.

We explored the meaning of algorithms and their significance, along with different methods of interpreting our code, which led us to pseudocode. Additionally, we familiarized ourselves with computer architecture, machine language, and various programming languages. We particularly focused on Python programming language, because of the coursework which we must accomplish. Also, Python plays a pivotal role in the development of machine learning and artificial intelligence.

The course not only provided insights into the history and future of computing but also engaged us with practical exercises in Python. Furthermore, we had the unique opportunity to immerse ourselves in computing history by visiting the science museum where we encountered various challenges."

Introduction

The coursework provides an opportunity to become acquainted with Python and grasp the logical connections between human cognition and machine processes. In this assignment, we are tasked with devising a logical algorithm to address a specific problem. This algorithm will entail a script that stores a list of numbers inputted by the user. By selecting options from a menu provided by the program's author, users can manipulate this list and receive the resultant output.



As depicted, there exists a menu offering various options for user interaction, enabling manipulation of the list according to user preferences. Each option corresponds to a specific function, triggered upon selection by the user from the main menu. Throughout the project, we will thoroughly examine each function, accompanied by visuals and comprehensive discussions elucidating their purpose. Testing the code will further solidify our understanding and ensure its efficacy in fulfilling its intended objectives.

Here you can find the code as well as in appendix.

https://github.com/sergiumi/Grades-Counter

Literature Review

There are multiple methods for calculating the mean of a set of numbers, such as the arithmetic mean, which adds up the numbers and divides by the count, and the geometric mean, which involves taking the nth root of the product of the numbers. Despite these different approaches, they typically yield similar results for most datasets. (Almond, 2023)

In statistics, the median is a measure that represents the middle value in a sorted list of numbers, either ascending or descending. It's considered a robust measure of central tendency as it isn't influenced by extreme values like the mean. Essentially, it's the point where half of the data falls above and half below, marking the midpoint of the dataset. (Ganti, 2024)

The mode signifies the value that occurs with the highest frequency within a dataset, which can be singular, multiple, or non-existent. Additionally, common metrics for central tendency encompass the mean, representing the average of a dataset, and the median, which denotes the middle value within the dataset. (Hayes, 2024)

Standard deviation, a statistical measure of a dataset's dispersion in relation to its mean, is computed as the square root of the variance. This entails assessing each data point's deviation from the mean and then taking the square root of the variance. (Hargrave, 2023)

Skewness refers to the extent of asymmetry observed in a probability distribution. When data points in a bell curve are not evenly spread on both sides of the median, the curve is said to be skewed. Skewness can take the form of positive skewness, where the distribution leans towards the right, or negative skewness, where it leans towards the left. A perfectly symmetrical distribution, like a normal distribution, exhibits zero skewness. (Chen, 2024)

Methodology

The project's objective is to ensure timely delivery of the final product, functioning seamlessly without bugs or program crashes, and maintaining user-friendliness, even in the event of errors. Opting for the Waterfall Methodology aligns with this goal, particularly since the project is solely managed by the author. This approach allows for meticulous attention to each phase, ensuring that each part operates perfect before progressing to the next stage. Although it may require additional time, this approach prioritizes precision and reliability over flexibility throughout the project phases.

Between the flat and hierarchical structures, I opt for the hierarchical structure because it facilitates a more systematic approach to coding, characterized by a sequential, top-down methodology. This structure accommodates distinct phases and dependencies, established by various functions containing conditional statements, thus forming a cohesive hierarchy.

Implementation

1) Main Menu Function

The mainMenu() function serves to interpret user input from a menu, effecting changes to a list accordingly. It employs an indefinite while loop to continually prompt the user until their response corresponds to one of the menu options. Upon exiting the loop, it prints the modified list using the print function. Next, the input function captures and stores the user's input number in the chosenCategory variable. To determine the selected category, an if, elif, else statement is utilized, functioning similarly to a switch case. If the input matches the first number, it invokes the meanNumber() function; otherwise, it proceeds to the next elif statement, evaluating each condition sequentially. If none of the conditions are met, the program displays an "invalid choice" message. And the “continue” method send the user to loop again.

```
# author (Sergiu Mita 22030807)
# version (version:3  date:06/05/24)
def mainMenu():
    while True: # I used while loop in this function to return to the main menu every time when an actiti
        print("""Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -""") # this is a print statement that will print this str
        print() # it's an empty print statement to print an empty line after the mainMenu
        chosenCategory = input("Please choose a category from Above: ") # chosenCcategory is a local vari
        print() # it's an empty print statement to print an empty line after the choosing category

        if chosenCategory == "1": # an if statement is and function which allow us to make a decision,
                    # I used an if statement to allow the user to choose a category from the list which w
            numbers = multiNumbers()  # we are saying multi number function to the "numbers" which is a l
                    # to manipulate with the "numberList" from the "multiNumber" function, because the "m
            print(f'The MEAN number of your list {numbers} is {meanNumber(numbers)}') #in the print state
                    # and printing the updated list through "numbers" variable and calling a function whi
                    # this parameter allows to call the same function but with a different input if we ha
            print() # it's an empty print statement to print an empty line after to be easier to delimita

        elif chosenCategory == "2": # elif strands from else and if command which is used in this case as
                   # to check the input from the user and to give the right answer when teh function was
            numbers = multiNumbers()
            print(f'The MEDIAN number of your sorted list {sorted(numbers)} is {medianNumber(numbers)}')
                    #and printing the updateted  sorted list through "numbers" variable and calling a fun
            print()
        elif chosenCategory == "3": # if the user will type 3 the next print will be executed
            numbers = multiNumbers()
            print(f'The MODE number of your list {numbers} is {modeNumber(numbers)}') # print will show o
            print()
        elif chosenCategory == "4": # if the user will type 4 the next print will be executed
            numbers = multiNumbers()
            print(f'The STANDARD DEVIATION of your list {numbers} is {standard_dev(numbers)}') # print wi
            print(f'The SKEWNESS of your list {numbers} is {Skewness(numbers)}') # print will show on the
            print()
        elif chosenCategory == "5": # if the user will type 5 the program will call the multiNumbers func
            multiNumbers() #calling the multiNumbers function to add new numbers to the existing list
            print()
        elif chosenCategory == "6": # if the user will type 6 the program will clear the list
                    # and call the "multiNumbers" function which will allow user to insert a new roll of
            numberList.clear()  # this inbuilt list method allow us to clear the list
            print("You can add a new set of numbers...") # printing a message to be more user friendly
            multiNumbers()  # adding numbers to the cleared list so it looks like a new list
            print()
        elif chosenCategory == 7: # if the user will type 7 the program will print some messages and afte
            print("See you later")
            print("..............")
            print("Exiting the app...")
            exit() # the exit inbuilt method allows us to close the program from the terminal

        else:
            print("Invalid choice, please choose a valid option.") # if the user will typewill type a dif
            continue # the continue keyword will return bak to the main menu each time an non menu nuber
```

```
FUNCTION mainMenu():
    WHILE True:  # Continuously display the main menu until the user chooses to exit
        PRINT "Hello, welcome to Students grade counter"
        PRINT "- - - - - - - - - - - - - - - - - -"
        PRINT "1. Print the Mean Number"
        PRINT "2. Print the Median Number"
        PRINT "3. Print the Mode Number"
        PRINT "4. Print the Skewness"
        PRINT "5. Add more numbers to the list"
        PRINT "6. Enter a NEW set of numbers"
        PRINT "7. Exit the APP"
        PRINT "- - - - - - - - - - - - - - - - -"
        PRINT  # Print an empty line

        chosenCategory = GET_USER_INPUT("Please choose a category from Above: ")  # Get user's choice
        PRINT  # Print an empty line

        IF chosenCategory == "1":
            numbers = CALL multiNumbers()  # Add more numbers to the list
            PRINT "The MEAN number of your list {numbers} is {meanNumber(numbers)}"

        ELIF chosenCategory == "2":
            numbers = CALL multiNumbers()
            PRINT "The MEDIAN number of your sorted list {sorted(numbers)} is {medianNumber(numbers)}"

        ELIF chosenCategory == "3":
            numbers = CALL multiNumbers()
            PRINT "The MODE number of your list {numbers} is {modeNumber(numbers)}"

        ELIF chosenCategory == "4":
            numbers = CALL multiNumbers()
            PRINT "The STANDARD DEVIATION of your list {numbers} is {standard_dev(numbers)}"
            PRINT "The SKEWNESS of your list {numbers} is {Skewness(numbers)}"

        ELIF chosenCategory == "5":
            CALL multiNumbers()  # Add more numbers to the list
            PRINT  # Print an empty line

        ELIF chosenCategory == "6":
            CALL clearList()  # Clear the list and add a new set of numbers
            PRINT "You can add a new set of numbers..."
            CALL multiNumbers()  # Add numbers to the cleared list
            PRINT  # Print an empty line

        ELIF chosenCategory == "7":
            PRINT "See you later"
            PRINT "............."
            PRINT "Exiting the app..."
            EXIT_PROGRAM()  # Exit the application

        ELSE:
            PRINT "Invalid choice, please choose a valid option."
            CONTINUE  # Return to the main menu if an invalid option is chosen
```

2) Multi Number Function

The main purpose of the multiNumbers() function is to ensure adherence to the rules while populating the list. I created this custom function with a while loop to iteratively execute the same steps until the user inputs the correct character. Initially, I implemented a try-except block to capture errors without causing program crashes. Subsequently, the function prompts the user to input a number, and if the user enters 'n', the program checks if there are at least two elements in the list. If there are, it breaks the loop; otherwise, it prompts the user to add one more element to the list. If the user doesn't input 'n', the program prompts them to add another grade. If the input is less than 0, a message is printed to enter a larger value. Conversely, if the input is an integer or float and greater than 0, it is added to the list. Finally, the function prints the list and the number of grades within it and returns the updated list."

```
def multiNumbers():
    while True:  # The  indefinite while loop was created to repeat the same action until the usel w
        try: # The try function was implemented to catch the error witout crashing the program and
            number = input("Insert a grade and press Enter to continue or 'n' to stop: ") # the "num
              #and colleteded by the program with the command input
            if number == 'n':  # if user inputs 'n', the program will exit the loop and proceed to t
                if len(numberList) < 2: # if the length of the list is less than 2 the user will be
                    print("Please add at least two grades to the list.") # and every time the progra
                    continue # this keyword allows us to go back and ask the user to insert more gra
                else: # if the condition is satisfied the program will go to the next keyword
                    break  # this give us power to stop the indefinite loop
            else: # if the user started to inser grades the next scenario will apply
                number = float(number) # the numbers will be converted to float
                if number <= 0: # if the input number is less than 0 the user will be asked to add a
                    print("Please enter a grade bigger than '0'")
                else: #if the number is bigger then 0 it will be added to the created list called nu
                    numberList.append(number)  # an element can be added using different functions,
        except ValueError: #the except keyword is a part of the try method which will catch the err
              # and will display a help message for the user without crashing the program
            print("Invalid input. Please type a number or 'n' to stop.") # this message will be disp
              #if the user will type a character which is not of type int or float
    print()
    print("You have entered " + str(len(numberList)) + " numbers. Your list is: " + str(numberList))
              # and how many elements it contains and to be able to print it I converted the results
    return numberList # the return keyword allows us to return the output of the changes which was m
```

```
FUNCTION multiNumbers():
    numberList = empty list  # Initialize an empty list to store numbers
    WHILE True:
        TRY:
            number = INPUT("Insert a grade and press Enter to continue or 'n' to stop: ")  # Prompt user for input
            IF number == 'n':
                IF LENGTH(numberList) < 2:
                    PRINT("Please add at least two grades to the list.")
                    CONTINUE  # Continue to the next iteration of the loop
                ELSE:
                    BREAK  # Exit the loop
            ELSE:
                number = CONVERT_TO_FLOAT(number)  # Convert input to float
                IF number <= 0:
                    PRINT("Please enter a grade bigger than '0'")
                ELSE:
                    ADD number TO numberList  # Add number to the list
        EXCEPT ValueError:
            PRINT("Invalid input. Please type a number or 'n' to stop.")
    PRINT("You have entered " + CONVERT_TO_STRING(LENGTH(numberList)) + " numbers. Your list is: " + CONVERT_TO_STRING(numberList))
    RETURN numberList
```

3) Mean Number Function

The meanNumber(aList) function calculates the average of a list. To create this custom function, begin with the def keyword followed by the function name and a parameter for the list, allowing easy access. Inside the function, initialize a local variable meanNumber to zero. Then, iterate through the list using a for loop, adding each item to the local variable. Next, create a new variable representing the result of dividing the sum of the items by the number of items in the list, determined using the len() command. Finally, return the result.

```
# The meanNumber function is a method which calculates the average number fron the inseted list
# author (Sergiu Mita 22030807)
# version (version:1  date:06/05/24)
def meanNumber(aList): # The "meanNumber()" function accepts a list as a parameter, which needs
    # In our case, the list is populated by the function "multiNumbers", so our parameter will

    meanNumber = 0 # to calculate the average number first create a variable for it and assign
    for number in aList: # looping through the list
        meanNumber += number # and adding each element of the list to that variable

    mean = meanNumber / len(aList) # as result the mean number is the sum of all the elements i
    return mean # at the end we have to return the result
```

```
FUNCTION meanNumber(aList):
    meanNumber = 0  # Initialize a variable to store the sum of numbers in the list
    FOR EACH number IN aList:  # Iterate through each element in the list
        meanNumber += number  # Add the current number to the sum

    mean = meanNumber / LENGTH_OF(aList)  # Calculate the mean by
                        # dividing the sum by the number of elements in the list
    RETURN mean  # Return the calculated mean
```

4) Median number method

This medianNumber(aList) function calculates the median of a list of numbers. It arranges the input list in ascending order using the sorted() function and assigns the result to the variable sorted_numbers. By sorting the list ensures that the middle elements can be easily identified. After the program calculates the length of the sorted list using the len() command and assigns it to the variable n. This will help determine whether the number of elements is even or odd. If the number of elements (n) is even the function calculates the median as the average of the two middle elements. It retrieves the two middle elements using list indexing adds them together, and divides by 2 to get the average. If the number of elements is odd, the function simply returns the middle element of the sorted list. If there is an odd number of elements, the median is the middle value. The function returns the calculated median value.

```
# The medianNumber() function verify the middle number of a arranged list from min to max
# it identifies the middle number in the sorted list if the list length is odd
# or the average of the two middle numbers if the list length is even
# author (Sergiu Mita 22030807)
# version (version:1  date:06/05/24)
def medianNumber(aList): # The "meanNumber()" function accepts a list as a parameter, which needs to be specified wh
    sorted_numbers = sorted(aList) # first we create the list to sort from min to max valuewith the list method sort
    n = len(sorted_numbers) # after calculating the length of the list with the len functoin
    if n % 2 == 0: # if the number of elements is even, median is the average of the middle two elements
        return (sorted_numbers[n // 2 - 1] + sorted_numbers[n // 2]) / 2 # returning the output of the median numbe
                #by calculating the average of the two middle numbers that we determined by their index numbers

    else:# if the number of elements is odd, median is the middle element
        return sorted_numbers[n // 2] # returning the output of the median number
                # by using floor division which will return the amount containing a number without the remainder
```

```
FUNCTION medianNumber(aList):
    sorted_numbers = SORT(aList)  # Sort the list in ascending order
    n = LENGTH_OF(sorted_numbers)  # Get the length of the sorted list

    IF n MOD 2 == 0 THEN  # If the number of elements is even
        middle_index1 = n // 2 - 1  # Calculate the index of the first middle element
        middle_index2 = n // 2  # Calculate the index of the second middle element
        median = (sorted_numbers[middle_index1] + sorted_numbers[middle_index2]) / 2  #
Calculate the average of the two middle elements
        RETURN median  # Return the median value

    ELSE:  # If the number of elements is odd
        middle_index = n // 2  # Calculate the index of the middle element
        median = sorted_numbers[middle_index]  # Get the middle element as the median
        RETURN median  # Return the median value
```

5) Mode number function

This modeNumber(aList) function calculates the mode(s) of a list of numbers. First, create a dictionary to store the counts by initializing an empty dictionary named countNr to store the counts of each unique number in the input list. The keys of this dictionary represent the unique numbers in the list, and the values represent the counts of each number. Using the for loop iterates through each element num in the input list. For each num, we check if it already exists as a key in the countNr dictionary. If it does, it increments the count associated with that number by 1. If not, it initializes the count of that number to 1. After counting the occurrences of each number, we determine the maximum count among all the numbers in the countNr dictionary using the max() command applied to the values of the dictionary. Creating an empty list named modes to store the mode(s) (numbers with the highest count). Then, iterates over each key-value pair (num, count) in the countNr dictionary. For each pair, if the count of the number equals the maximum count found earlier, it appends the number (num) to the modes list. as a result the function returns the list of mode(s) found in the input list.

```
# Mode Number function is checking which number appears the most in the list
# author (Sergiu Mita 22030807)
# version (version:3  date:06/05/24)
def modeNumber(aList):
    countNr = {} # create a dictionary to store the pair
    # of how many times the number belongs to the list and the number itself

    for num in aList: # check how many times the same number can be found in the list
        if num in countNr: # if the number belongs to the list
            countNr[num] += 1 #the counter in the dictionar is growing up
        else: # if not it is assign to 1
            countNr[num] = 1

    max_count = max(countNr.values()) # find the number(s) with the highest count
    modes = []  # initialize an empty list to store the modes

    for num, count in countNr.items():  # iterate over key-value pairs in count_dict
        if count == max_count:  # check if the count equals the maximum count
            modes.append(num)  # if yes, add the number to the modes list
    return modes # returning the final result
```

```
FUNCTION modeNumber(aList):
    countNr = {}  # Create an empty dictionary to store the count of each number

    # Loop through each number in the list
    FOR EACH num IN aList:
        # Check if the number is already in the dictionary
        IF num EXISTS IN countNr:
            # Increment the count of the number by 1
            INCREMENT countNr[num]
        ELSE:
            # Initialize the count of the number to 1
            countNr[num] = 1

    max_count = MAXIMUM_VALUE(countNr.values())  # Find the maximum count of any number

    modes = []  # Initialize an empty list to store the modes

    # Loop through each key-value pair in the dictionary
    FOR EACH num, count IN countNr.items():
        # Check if the count of the current number is equal to the maximum count
        IF count EQUALS max_count:
            # Add the number to the list of modes
            APPEND num TO modes

    # Return the list of modes
    RETURN modes
```

6) Skewness function

The Skewness(aList) function calculates the skewness of a list of numbers using a formula that involves the mean, median, and standard deviation. In this formula, the standard deviation is calculated using the standard_dev(aList) function. Therefore, the Skewness(aList) function relies on the standard_dev(aList) function to compute the standard deviation necessary for skewness calculation. This demonstrates the correlation between the two functions. The standard_dev(aList) function calculates the standard deviation of a list of numbers using the formula for population standard deviation. To calculate standard deviation, we are starting by initializing a variable sumSquaredDif to store the sum of squared differences. This variable will be used to accumulate the squared differences between each element of the list and the mean. By using a definite loop iterates through each element "i" in the input list. For each element, the function calculates the squared difference between the element and the mean of the list (calculated using the meanNumber(aList) function). The squared difference is then added to the sumSquaredDif variable. After iterating through all elements, it calculates the variation by dividing the sum of squared differences (sumSquaredDif) by the length of the list. This gives the variance of the population. By taking the square root of the variation we can calculate Standard Deviation. This gives the standard deviation of the population. And as always we have to return the result of calculated standard deviation. To use it in the Skewness function.

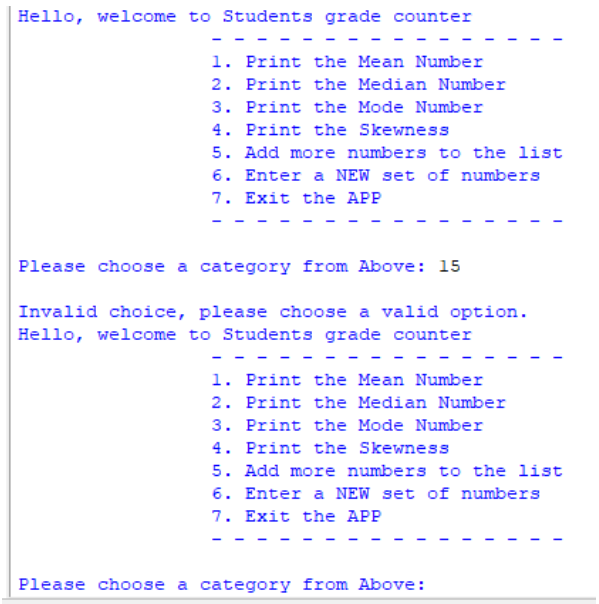
```
# Standard_dev method calculates the standard deviation by the formula by population
# from research which will be used in the skewness calculation
# author (Sergiu Mita 22030807)
# version (version:3  date:07/05/24)
def standard_dev(aList):

    sumSquaredDif = 0 # Initialize a variable to store the sum of squared differences

    for i in aList: # Iterate through each element i in the list
        squaredDif = (i - meanNumber(aList)) ** 2 # calculate the squared difference b
        sumSquaredDif += squaredDif # add each of the squared difference to the sum of

    variation = sumSquaredDif / len(aList) # calculate the variation by dividing the s
    st_dev = variation ** 0.5 # calculate the standard deviation by taking the square
    return st_dev # return the calculated standard deviation


# Skewness function calculates the asymmetry of the grades in the list by using a form
# author (Sergiu Mita 22030807)
# version (version:1  date:07/05/24)
def Skewness(aList): # a new procedure with a parameter
    Skew = 3*(meanNumber(aList)- medianNumber(aList))/ standard_dev(aList) # creating
        # skew and assigning a formula that calculates skewness
    return  Skew # returning the result of the asymmetry in the list
```

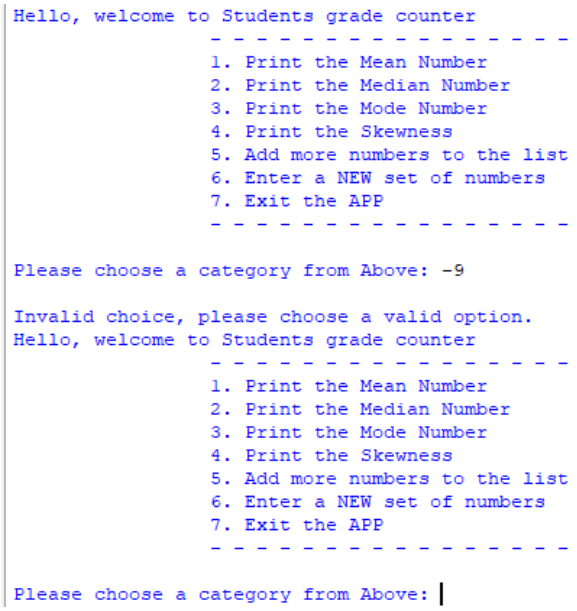
```
FUNCTION standard_dev(aList):
    sumSquaredDif = 0  # Initialize a variable to store the sum of squared differences

    # Iterate through each element i in the list
    FOR EACH i IN aList:
        # Calculate the squared difference between the current element and the mean of the list
        squaredDif = (i - meanNumber(aList)) ** 2

        # Add the squared difference to the sum of squared differences
        sumSquaredDif += squaredDif

    # Calculate the variation by dividing the sum of squared differences by the length of the list
    variation = sumSquaredDif / LENGTH_OF(aList)

    # Calculate the standard deviation by taking the square root of the variation
    st_dev = SQUARE_ROOT(variation)

    # Return the calculated standard deviation
    RETURN st_dev

FUNCTION Skewness(aList):
    # Calculate skewness using the formula: 3 * (mean - median) / standard deviation
    skew = 3 * (meanNumber(aList) - medianNumber(aList)) / standard_dev(aList)

    # Return the calculated skewness
    RETURN skew
```

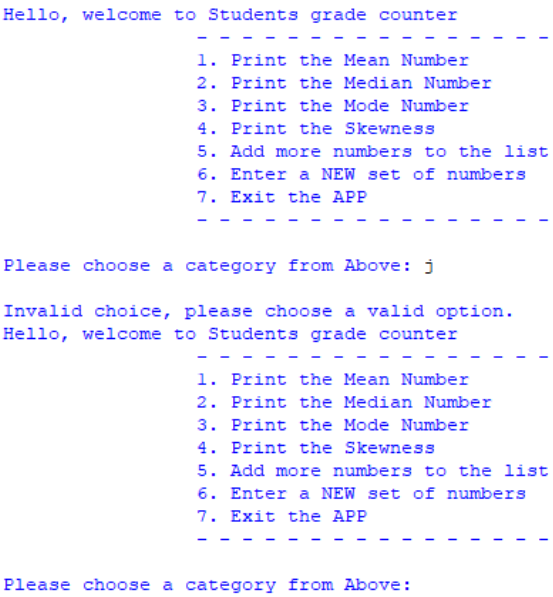
Testing

Test 1 Main menu incorrect input:

I've tested the program three times, deliberately inputting incorrect data each time. On each occasion, the message "Invalid choice. Please select a valid option" appeared.







Test 2 Main menu correct option:

Upon entering the correct option, the program functions as expected. In the following images, you can observe some of the testing with successful attempts. You can be assured of this because, following the execution of the mean number function, the main menu promptly reappears.

a) The meanNumber(aList) function.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 1

The MEAN number will be printed
Insert a grade and press Enter to continue or 'n' to stop: 4
Insert a grade and press Enter to continue or 'n' to stop: 3.5
Insert a grade and press Enter to continue or 'n' to stop: 12
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 3 numbers. Your list is: [4.0, 3.5, 12.0]
The MEAN number of your list [4.0, 3.5, 12.0] is 6.5

Hello, welcome to Students grade counter
```

b) The medianNumber(aList) function when the length list is odd.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 2

The MEDIAN number will be printed
Insert a grade and press Enter to continue or 'n' to stop: 4
Insert a grade and press Enter to continue or 'n' to stop: 8
Insert a grade and press Enter to continue or 'n' to stop: 6
Insert a grade and press Enter to continue or 'n' to stop: 7
Insert a grade and press Enter to continue or 'n' to stop: 3
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 5 numbers. Your list is: [4.0, 8.0, 6.0, 7.0, 3.0]
The MEDIAN number of your sorted list [3.0, 4.0, 6.0, 7.0, 8.0] is 6.0

Hello, welcome to Students grade counter
```

c) The medianNumber(aList) function when the length list is even.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 2

The MEDIAN number will be printed
Insert a grade and press Enter to continue or 'n' to stop: 1
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 6 numbers. Your list is: [4.0, 8.0, 6.0, 7.0, 3.0, 1.0]
The MEDIAN number of your sorted list [1.0, 3.0, 4.0, 6.0, 7.0, 8.0] is 5.0

Hello, welcome to Students grade counter
```

Here you can observe that the middle numbers were added (4 and 6) and divided by 2 which is 5.

d) The modeNumber() function.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 3

The MODE number will be printed
Insert a grade and press Enter to continue or 'n' to stop: 1
Insert a grade and press Enter to continue or 'n' to stop: 6
Insert a grade and press Enter to continue or 'n' to stop: 5
Insert a grade and press Enter to continue or 'n' to stop: 6
Insert a grade and press Enter to continue or 'n' to stop: 4
Insert a grade and press Enter to continue or 'n' to stop: 8
Insert a grade and press Enter to continue or 'n' to stop: 4
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 7 numbers. Your list is: [1.0, 6.0, 5.0, 6.0, 4.0, 8.0, 4.0]
The MODE number of your list [1.0, 6.0, 5.0, 6.0, 4.0, 8.0, 4.0] is [6.0, 4.0]

Hello, welcome to Students grade counter
```

In this scenario, we have two repetitive numbers in the list: 6 and 4.

e) The Skewness() method.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 4

The SKEWNESS will be printed
Insert a grade and press Enter to continue or 'n' to stop: 12
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 8 numbers. Your list is: [1.0, 6.0, 5.0, 6.0, 4.0, 8.0, 4.0, 12.0]
The STANDARD DEVIATION of your list [1.0, 6.0, 5.0, 6.0, 4.0, 8.0, 4.0, 12.0] is 3.031088913245535
The SKEWNESS of your list [1.0, 6.0, 5.0, 6.0, 4.0, 8.0, 4.0, 12.0] is 0.24743582965269675

Hello, welcome to Students grade counter
```

In this case I decided to print both Skewness and Standard deviation numbers.

f) Add more numbers to the list will execute multiNumbers() function one more time.

```
You have entered 2 numbers. Your list is: [1.0, 2.0]

Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 5

You can add more numbers to the list
Insert a grade and press Enter to continue or 'n' to stop: 5
Insert a grade and press Enter to continue or 'n' to stop: 8
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 4 numbers. Your list is: [1.0, 2.0, 5.0, 8.0]

Hello, welcome to Students grade counter
```

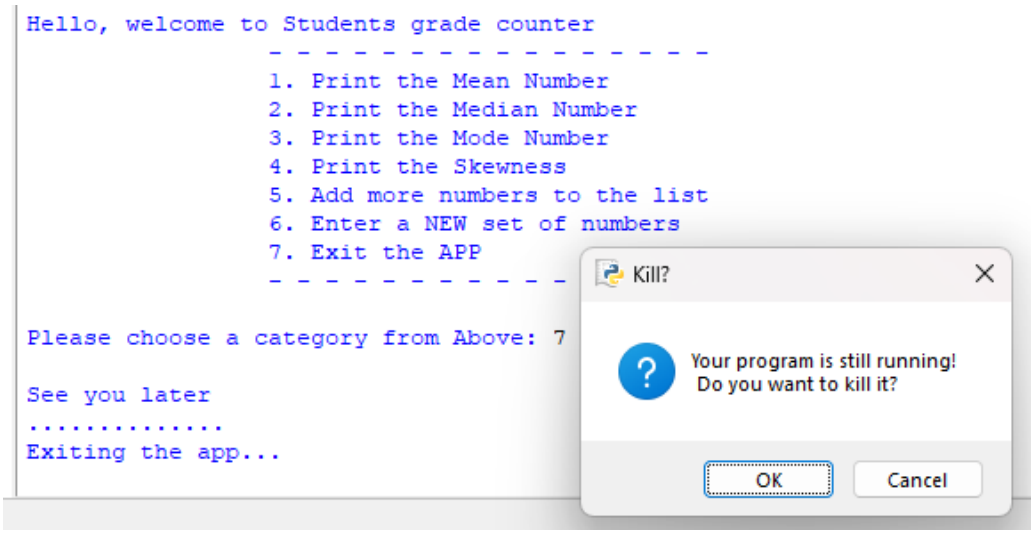
It's evident that we initially had two numbers in the list (1, 2). After invoking the method to add more numbers, the list now contains four numbers: 1, 2, 5, and 8.

g) Add a new list of numbers.

```
You have entered 4 numbers. Your list is: [1.0, 2.0, 5.0, 8.0]

Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - -

Please choose a category from Above: 6

You can add a new set of numbers...
Insert a grade and press Enter to continue or 'n' to stop: 6
Insert a grade and press Enter to continue or 'n' to stop: 7.2
Insert a grade and press Enter to continue or 'n' to stop: 3
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 3 numbers. Your list is: [6.0, 7.2, 3.0]

Hello, welcome to Students grade counter
```

In the image, it's apparent that there is already a list of 4 numbers from the previous test. Now, after choosing to enter a new list, we can see a new list with only 3 numbers. These numbers are different from the previous list because we first used the command clear() to empty the list and then executed the multiNumber() function again

h) The exit() function



This action was executed by invoking the built-in method exit().

Test 3 The multiNumber() function by typing the wrong input.

```
Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
                1. Print the Mean Number
                2. Print the Median Number
                3. Print the Mode Number
                4. Print the Skewness
                5. Add more numbers to the list
                6. Enter a NEW set of numbers
                7. Exit the APP
                - - - - - - - - - - - - - - - - -

Please choose a category from Above: 1

The MEAN number will be printed
Insert a grade and press Enter to continue or 'n' to stop: n
Please add at least two grades to the list.
Insert a grade and press Enter to continue or 'n' to stop: 1
Insert a grade and press Enter to continue or 'n' to stop: n
Please add at least two grades to the list.
Insert a grade and press Enter to continue or 'n' to stop: 8.2
Insert a grade and press Enter to continue or 'n' to stop: -15
Please enter a grade bigger than '0'
Insert a grade and press Enter to continue or 'n' to stop: h
Invalid input. Please type a number or 'n' to stop.
Insert a grade and press Enter to continue or 'n' to stop: 141
Invalid input. Please type a number or 'n' to stop.
Insert a grade and press Enter to continue or 'n' to stop: .
Invalid input. Please type a number or 'n' to stop.
Insert a grade and press Enter to continue or 'n' to stop: n

You have entered 2 numbers. Your list is: [1.0, 8.2]
The MEAN number of your list [1.0, 8.2] is 4.6

Hello, welcome to Students grade counter
                - - - - - - - - - - - - - - - - -
```

Each time we attempted to insert a character that wasn't of type float or integer, or when the list contained fewer than two numbers, the program provided us with a prompt.

Conclusion

Working on this project was a valuable experience where I applied the knowledge gained throughout the course and got acquainted with crafting logical algorithms using the Python programming language. The project presented numerous challenges as it required integrating various methods and functions to develop a functional program free of bugs and crashes. I utilized the try method to handle errors, implemented a while loop for indefinite iteration, and leveraged for loops to navigate through lists. Additionally, I employed if-else statements in various contexts. However, the most demanding aspect was writing thorough comments, which proved time-consuming yet essential for clarity and understanding.

Reference

Almond, N., 2023. *What Is Mean In Maths.* [Online]
Available at: https://thirdspacelearning.com/blog/what-is-mean-average/
[Accessed 09 05 2024].

Chen, J., 2024. *Right Skewed vs. Left Skewed Distribution.* [Online]
Available at: https://www.investopedia.com/terms/s/skewness.asp
[Accessed 09 05 2024].

Foundation, P. S., 2001. *https://www.python.org/.* [Online]
Available at: https://www.python.org/
[Accessed 08 04 2024].

Ganti, A., 2024. *Median: What It Is and How To Calculate It.* [Online]
Available at: https://www.investopedia.com/terms/m/median.asp
[Accessed 09 05 2024].

Hargrave, M., 2023. *Standard Deviation Formula and Uses vs. Variance.* [Online]
Available at: https://www.investopedia.com/terms/s/standarddeviation.asp
[Accessed 09 05 2024].

Hayes, A., 2024. *Mode: What It Is in Statistics.* [Online]
Available at: https://www.investopedia.com/terms/m/mode.asp
[Accessed 09 05 2024].

www.w3schools.com, 1999. *Learn Python.* [Online]
Available at: https://my-learning.w3schools.com/
[Accessed 05 04 2024].

Zelle, J. M., 2016. *Python programming: an introduction to computer science.* 3rd ed. Portland, Oregon: Franklin, Beedle & Associates Inc .

Appendix

#The program manipulates a user-added list of numbers and generates output based on user requirements.

#It can print the mean, median, mode, and skewness of the list.

numberList = [] #creating an empty list to store the grades of the students inside to be able to manipulate with them in our program

This is a function which is called mainMenu here is going to be dysplayed the menu where the user can chose to do a chenge to his own list

author (Sergiu Mita 22030807)

version (version:3 date:06/05/24)

def mainMenu():

    while True: # I used while loop in this function to return to the main menu every time when an actitin from the menu list was succeseful done

        print("""Hello, welcome to Students grade counter

            - - - - - - - - - - - - - - - -

            1. Print the Mean Number

            2. Print the Median Number

            3. Print the Mode Number

            4. Print the Skewness

            5. Add more numbers to the list

            6. Enter a NEW set of numbers

            7. Exit the APP

            - - - - - - - - - - - - - - - -""") # this is a print statement that will print this string, the 3 quotation marks were used to print more than one line of this string

        print() # it's an empty print statement to print an empty line after the mainMenu

```
    chosenCategory = input("Please choose a category from Above: ") # chosenCcategory is a local variable that will store the input from the user

    print() # it's an empty print statement to print an empty line after the choosing category


    if chosenCategory == "1": # an if statement is and function which allow us to make a decision,

            # I used an if statement to allow the user to choose a category from the list which was printed above if the user  will type 1 the next print will be executed

        numbers = multiNumbers()  # we are saying multi number function to the "numbers" which is a local variable to be able

            # to manipulate with the "numberList" from the "multiNumber" function, because the "multiNumber" returning the updated list

        print(f'The MEAN number of your list {numbers} is {meanNumber(numbers)}') #in the print statement, we are printing a message for the user

            # and printing the updated list through "numbers" variable and calling a function which is the "meanNumber" of parameter numbers,

            # this parameter allows to call the same function but with a different input if we have multiple inputs"'

        print() # it's an empty print statement to print an empty line after to be easier to delimitate what happened


    elif chosenCategory == "2": # elif strands from else and if command which is used in this case as an swich mode

            # to check the input from the user and to give the right answer when teh function was called  if the user will type 2 the next print will be executed

        numbers = multiNumbers()

        print(f'The MEDIAN number of your sorted list {sorted(numbers)} is {medianNumber(numbers)}') #in the print statement, we are printing a message for user

            #and printing the updateted  sorted list through "numbers" variable and calling a function which is the "medianNumber" of parameter numbers

        print()


    elif chosenCategory == "3": # if the user will type 3 the next print will be executed

        numbers = multiNumbers()
```

```
        print(f'The MODE number of your list {numbers} is {modeNumber(numbers)}') # print will show on the screen the user list and  modeNumber function answer

        print()


    elif chosenCategory == "4": # if the user will type 4 the next print will be executed

        numbers = multiNumbers()

        print(f'The STANDARD DEVIATION of your list {numbers} is {standard_dev(numbers)}') # print will show on the screen the user list and standard_dev function answer

        print(f'The SKEWNESS of your list {numbers} is {Skewness(numbers)}') # print will show on the screen the user list and Skewness function answer

        print()


    elif chosenCategory == "5": # if the user will type 5 the program will call the multiNumbers function which will allow user to add more grades to the list

        multiNumbers() #calling the multiNumbers function to add new numbers to the existing list

        print()


    elif chosenCategory == "6": # if the user will type 6 the program will clear the list

            # and call the "multiNumbers" function which will allow user to insert a new roll of grades to the list

        numberList.clear()  # this inbuilt list method allow us to clear the list

        print("You can add a new set of numbers...") # printing a message to be more user friendly

        multiNumbers()  # adding numbers to the cleared list so it looks like a new list

        print()


    elif chosenCategory == "7": # if the user will type 7 the program will print some messages and after will ask if you are agree to close the display window

        print("See you later")

        print(".............")
```

```
            print("Exiting the app...")
            exit() # the exit inbuilt method allows us to close the program from the terminal

        else:
            print("Invalid choice, please choose a valid option.") # if the user will typewill type a differnt number betwen 1 and 7 this message will be printed
            continue # the continue keyword will return bak to the main menu each time an non menu nuber was entered


#the custom created "multiNumbers()" function allows us to insert every symbol from the keyboard and create logical instructions
#will give hints to the user to help them insert the correct symbols which are numbers in primitive types like integer and float
# author (Sergiu Mita 22030807)
# version (version:6  date:05/05/24)
def multiNumbers():
    while True:  # The  indefinite while loop was created to repeat the same action until the usel will insert a true value

        try: # The try function was implemented to catch the error witout crashing the program and to be user fridly

            number = input("Insert a grade and press Enter to continue or 'n' to stop: ") # the "number" local variable will store the value typed by the user
             #and colleteded by the program with the command input

            if number == 'n':  # if user inputs 'n', the program will exit the loop and proceed to the next statement but only if the list contains more than 2 numbers

                if len(numberList) < 2: # if the length of the list is less than 2 the user will be asked to insert more grades
```

```
                print("Please add at least two grades to the list.") # and every time the program will check its correctness because of the while loop
                continue # this keyword allows us to go back and ask the user to insert more grades because the condition was not satisfied

            else: # if the condition is satisfied the program will go to the next keyword
                break  # this give us power to stop the indefinite loop

        else: # if the user started to inser grades the next scenario will apply
            number = float(number) # the numbers will be converted to float

            if number <= 0: # if the input number is less than 0 the user will be asked to add a positive number
                print("Please enter a grade bigger than '0'")

            else: #if the number is bigger then 0 it will be added to the created list called numberList
                numberList.append(number)  # an element can be added using different functions, but I chose to add, which will add it to the end of the lis

    except ValueError: #the except keyword is a part of the try method which will catch the error
        # and will display a help message for the user without crashing the program
        print("Invalid input. Please type a number or 'n' to stop.") # this message will be displayed
        #if the user will type a character which is not of type int or float

print()
print("You have entered " + str(len(numberList)) + " numbers. Your list is: " + str(numberList)) # tThis print statement will display the list
# and how many elements it contains and to be able to print it I converted the results to string type
```

```
    return numberList # the return keyword allows us to return the output of the changes which was made to the initial list



# The meanNumber function is a method which calculates the average number fron the inseted list

# author (Sergiu Mita 22030807)

# version (version:1  date:06/05/24)

def meanNumber(aList): # The "meanNumber()" function accepts a list as a parameter, which needs to be specified when the function is called

    # In our case, the list is populated by the function "multiNumbers", so our parameter will be the result of calling the "multiNumbers" function


    meanNumber = 0 # to calculate the average number first create a variable for it and assign it to 0

    for number in aList: # looping through the list

        meanNumber += number # and adding each element of the list to that variable


    mean = meanNumber / len(aList) # as result the mean number is the sum of all the elements in the list divided by the length of the list

    return mean # at the end we have to return the result



# The medianNumber() function verify the middle number of a arranged list from min to max

# it identifies the middle number in the sorted list if the list length is odd

# or the average of the two middle numbers if the list length is even

# author (Sergiu Mita 22030807)

# version (version:1  date:06/05/24)

def medianNumber(aList): # The "meanNumber()" function accepts a list as a parameter, which needs to be specified when the function is called
```

```
    sorted_numbers = sorted(aList) # first we create the list to sort from min to max valuewith the list method sorted

    n = len(sorted_numbers) # after calculating the length of the list with the len functoin

    if n % 2 == 0: # if the number of elements is even, median is the average of the middle two elements

        return (sorted_numbers[n // 2 - 1] + sorted_numbers[n // 2]) / 2 # returning the output of the median number

                #by calculating the average of the two middle numbers that we determined by their index numbers


    else:# if the number of elements is odd, median is the middle element

        return sorted_numbers[n // 2] # returning the output of the median number

                # by using floor division which will return the amount containing a number without the remainder



# Mode Number function is checking which number appears the most in the list

# author (Sergiu Mita 22030807)

# version (version:3  date:06/05/24)

def modeNumber(aList):

    countNr = {} # create a dictionary to store the pair

    # of how many times the number belongs to the list and the number itself


    for num in aList: # check how many times the same number can be found in the list by looping the list

        if num in countNr: # if the number belongs to the list

            countNr[num] += 1 #the counter in the dictionar is growing up

        else: # if not it is assign to 1

            countNr[num] = 1


    max_count = max(countNr.values()) # find the number(s) with the highest count
```

```
    modes = []  # initialize an empty list to store the modes

    for num, count in countNr.items():  # iterate over key-value pairs in count_dict
        if count == max_count:  # check if the count equals the maximum count
            modes.append(num)  # if yes, add the number to the modes list
    return modes # returning the final result



# Standard_dev method calculates the standard deviation by the formula by population
# from research which will be used in the skewness calculation
# author (Sergiu Mita 22030807)
# version (version:3  date:07/05/24)
def standard_dev(aList):

    sumSquaredDif = 0 # Initialize a variable to store the sum of squared differences

    for i in aList: # Iterate through each element i in the list
        squaredDif = (i - meanNumber(aList)) ** 2 # calculate the squared difference between the current element and the mean
        sumSquaredDif += squaredDif # add each of the squared difference to the sum of squared differences

    variation = sumSquaredDif / len(aList) # calculate the variation by dividing the sum of squared differences by the length of the list
    st_dev = variation ** 0.5 # calculate the standard deviation by taking the square root of the variation which is the same as 1**0.05
    return st_dev # return the calculated standard deviation
```

```
# Skewness function calculates the asymmetry of the grades in the list by using a formula
# author (Sergiu Mita 22030807)
# version (version:1  date:07/05/24)
def Skewness(aList): # a new procedure with a parameter
    Skew = 3*(meanNumber(aList)- medianNumber(aList))/ standard_dev(aList) # creating a local variable
        # skew and assigning a formula that calculates skewness
    return  Skew # returning the result of the asymmetry in the list


mainMenu() #calling the main method to start the program
```