**ASSIGNMENT – 2.1**

**TASK-1:**

Write a python function that reads a list of numbers and calculate the mean ,medium, and maximum values

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screen shot of a computer

AI-generated content may be incorrect.

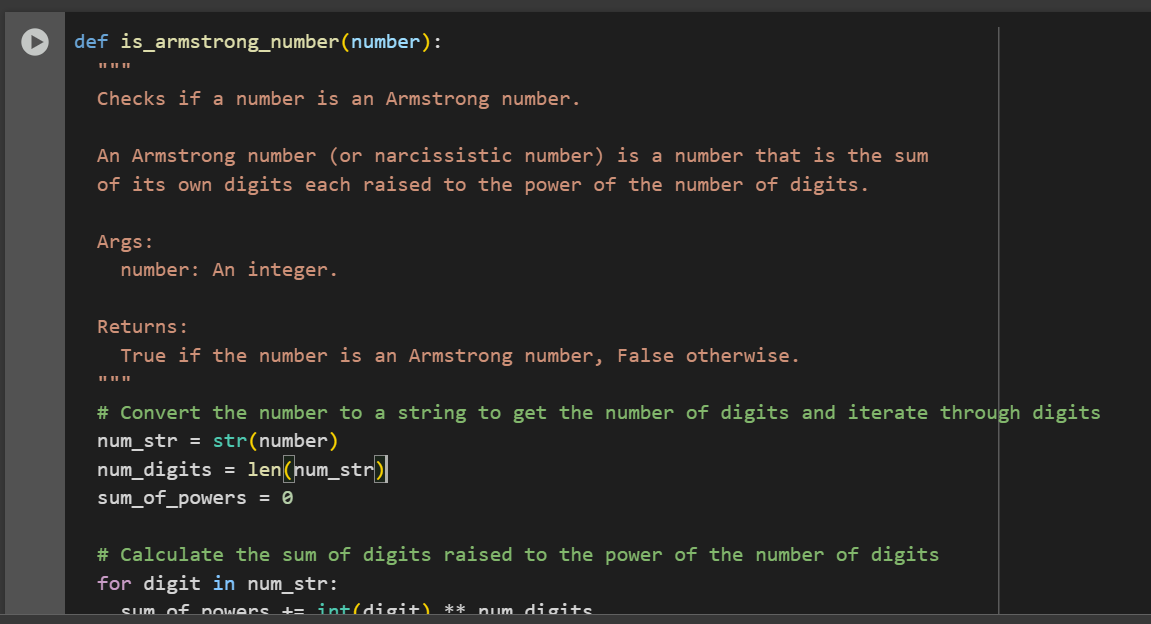
**EXPLANATION:**

1. **def calculate\_stats(numbers):**: This line defines a Python function named calculate\_stats that takes one argument, numbers. This numbers argument is expected to be a list of numbers.
2. **Docstring ("""Docstring goes here""")**: The triple-quoted text inside the function is called a docstring. It explains what the function does, its arguments (Args), and what it returns (Returns). This is good practice for making your code understandable.
3. **if not numbers:**: This is a conditional statement that checks if the input list numbers is empty. If the list is empty, it means there are no numbers to calculate statistics for.
4. **return None, None, None # Return None for empty list**: If the list is empty (the condition if not numbers: is true), the function returns three None values. This indicates that the statistics could not be calculated because there was no data.
5. **mean = sum(numbers) / len(numbers)**: This line calculates the mean (average) of the numbers in the list.
   * sum(numbers): This built-in Python function calculates the sum of all the numbers in the numbers list.
   * len(numbers): This built-in Python function calculates the number of elements (the length) of the numbers list.
   * The sum is then divided by the length to get the average.
6. **minimum = min(numbers)**: This line uses the built-in Python function min() to find the smallest number in the numbers list and assigns it to the variable minimum.
7. **maximum = max(numbers)**: This line uses the built-in Python function max() to find the largest number in the numbers list and assigns it to the variable maximum.
8. **return mean, minimum, maximum**: This line returns the calculated mean, minimum, and maximum values as a tuple.
9. **my\_list = [10, 20, 30, 40, 50]**: This line creates a list of numbers called my\_list to demonstrate how to use the function.
10. **mean\_value, min\_value, max\_value = calculate\_stats(my\_list)**: This line calls the calculate\_stats function with my\_list as the input. The function returns a tuple of three values, which are then unpacked and assigned to the variables mean\_value, min\_value, and max\_value respectively.
11. **print(f"List: {my\_list}")**: This line prints the original list. The f"" syntax is a f-string, which allows you to embed the value of variables directly within the string.
12. **print(f"Mean: {mean\_value}")**: This line prints the calculated mean value.
13. **print(f"Minimum: {min\_value}")**: This line prints the calculated minimum value.
14. **print(f"Maximum: {max\_value}")**:

**TASK – 2:**

write a python function to check whether a number is amstrong number

compare it with gemini and copilot



A computer screen shot of a number

AI-generated content may be incorrect.

**OUTPUT:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **Function Definition**: def is\_armstrong\_number(number): defines the function that takes one argument, number.
2. **Docstring**: The text within triple quotes explains what the function does, its arguments, and what it returns.
3. **Convert to String**: num\_str = str(number) converts the input integer number into a string (num\_str). This is done to easily access each digit and determine the number of digits.
4. **Get Number of Digits**: num\_digits = len(num\_str) calculates the number of digits in the number by getting the length of the string.
5. **Initialize Sum**: sum\_of\_powers = 0 initializes a variable to store the sum of the digits raised to the power of the number of digits.
6. **Iterate Through Digits**: The for digit in num\_str: loop iterates through each character (digit) in the num\_str string.
7. **Calculate Sum of Powers**: Inside the loop, sum\_of\_powers += int(digit) \*\* num\_digits performs the following steps:
   * int(digit): Converts the character digit back into an integer.
   * \*\* num\_digits: Raises the integer digit to the power of num\_digits (the total number of digits in the original number).
   * sum\_of\_powers += ...: Adds the result to the sum\_of\_powers variable.
8. **Check for Armstrong Number**: return sum\_of\_powers == number compares the calculated sum\_of\_powers with the original number. If they are equal, the function returns True, indicating that the number is an Armstrong number. Otherwise, it returns False.
9. **Example Usage**: The lines after the function definition demonstrate how to use the function with the numbers 153 (an Armstrong number) and 123 (not an Armstrong number) and print the results.

**TASK-3:**

Write a python program to check whether the number is prime or palindrome

A computer screen with colorful text

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**

A black screen with white text

AI-generated content may be incorrect.

**EXPLANATION :**

**This code defines two functions:**

* **is\_prime(num): This function checks if a given number num is a prime number.**
  + **It first handles the base cases: numbers less than or equal to 1 are not prime.**
  + **It then iterates from 2 up to the square root of the number. If any number in this range divides num evenly, num is not prime and the function returns False.**
  + **If the loop finishes without finding any divisors, the number is prime and the function returns True.**
* **is\_palindrome(num): This function checks if a given number num is a palindrome.**
  + **It converts the number to a string.**
  + **It then compares the string with its reverse. If they are the same, the number is a palindrome and the function returns True; otherwise, it returns False**

**TASK-4 :**

Write a python function to generate the sum of first n natural numbers

A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**

**A black background with white text

AI-generated content may be incorrect.**

**EXPLANATION:**

This code defines a function called sum\_of\_natural\_numbers(n).

* This function takes one argument, n, which represents the number of natural numbers you want to sum.
* It first checks if n is less than 1. If it is, it returns a message indicating that the input should be a positive integer.
* If n is 1 or greater, it uses the formula n \* (n + 1) // 2 to calculate the sum of the first n natural numbers. This is a well-known formula for this calculation, and // 2 ensures integer division.
* Finally, the code shows example usage of the function with n=10 and n=5, printing the results.

**TASK-5:**

Write a python program to calculate the sum of odd and even numbers in a given tuple

A screenshot of a computer program

AI-generated content may be incorrect.

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**EXPLANATION :**

This code defines a function called sum\_odd\_even(numbers).

* This function takes one argument, numbers, which is expected to be a tuple (or any iterable) of numbers.
* Inside the function, two variables sum\_even and sum\_odd are initialized to 0. These will store the running sums of even and odd numbers.
* The code then iterates through each number in the input numbers.
* For each number, it checks if the number is even using the modulo operator (%). If number % 2 == 0 is true, the number is even and is added to sum\_even. Otherwise, it's odd and is added to sum\_odd.
* Finally, the function returns both sum\_even and sum\_odd as a tuple.