**AI ASSISTED CODING**

**ASSIGNMENT-2**

**NAME :** YUVARAJ REGULA

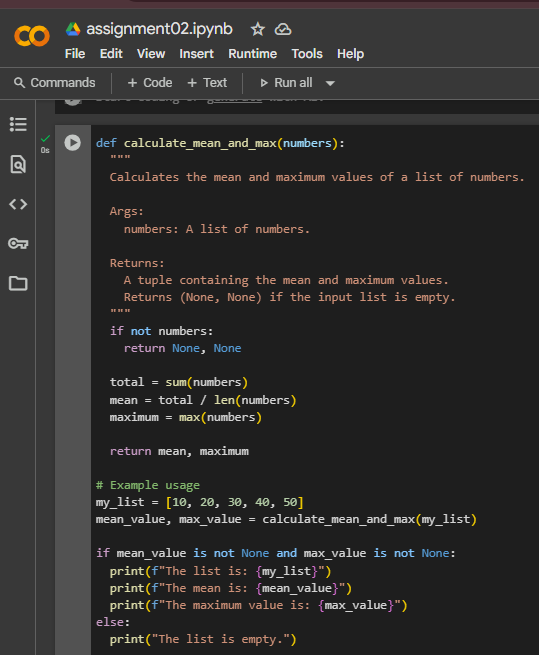
**HALL.NO :** 2403A52386

**BATCH.NO :** AIML-14

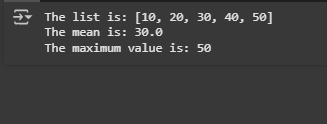
**PROMPT 01 :**

I need a code fuction that reads a list of numbers and calculates the mean and maximum values

**CODE :**

****

**OUTPUT:**

****

**CODE EXPLANATION:**

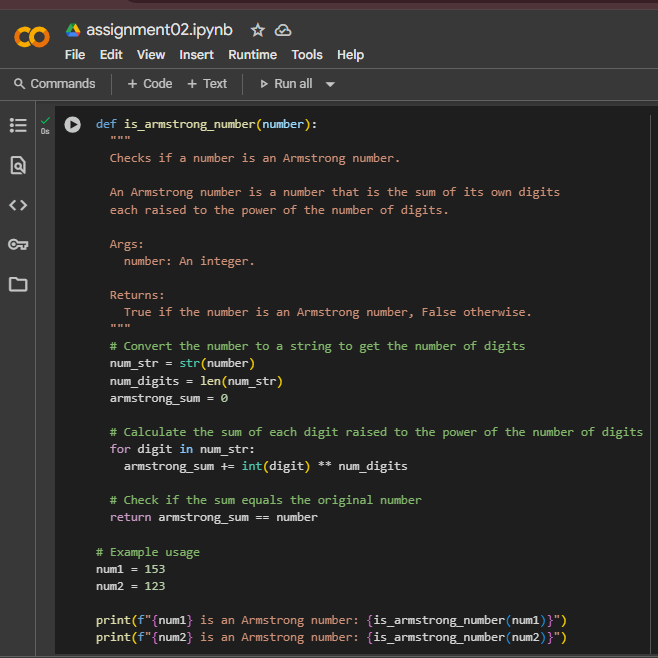
Function: calculate\_mean\_and\_max(numbers)

1. Purpose: Calculates the *mean (average)* and *maximum* values from a list of numbers.
2. Empty list check: If the list is empty, returns (None, None).
3. Total calculation: Uses sum(numbers) to get the total of all elements.
4. Mean calculation: Divides the total by the number of elements (len(numbers)).
5. Max calculation: Finds the largest number in the list using max(numbers).
6. Return values: Returns both the mean and maximum as a tuple (mean, maximum).
7. Creates a sample list: my\_list = [10,. Calls the function: mean\_value, max\_value = calculate\_mean\_and\_max(my\_list)`.
8. Checks if results are not None (list is not empty).
9. If values exist, prints:
   * The original list.
   * The mean value.
   * The maximum value.
10. If the list is empty, prints "The list is empty.".

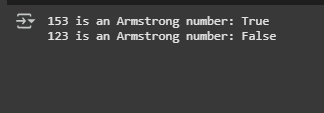
**PROMPT 02 :**

I need python code ,function that checks wether a number is an amstrong number

**CODE :**

****

**OUTPUT :**

****

**CODE EXPLANATION :**

Function: is\_armstrong\_number(number)

1. Purpose: Checks if a number is an *Armstrong number*.
2. Convert to string: num\_str = str(number) to handle digits easily.
3. Get digit count: num\_digits = len(num\_str) stores total digits.
4. Initialize sum: armstrong\_sum = 0.
5. Loop through digits:
   * Convert each to int.
   * Raise to num\_digits power.
   * Add to armstrong\_sum.
6. Compare with original number:  
   return armstrong\_sum == number → True if Armstrong, else False.

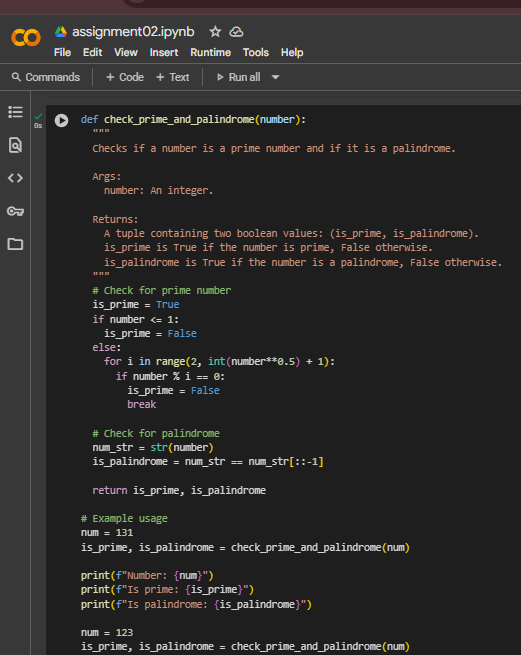
Example usage

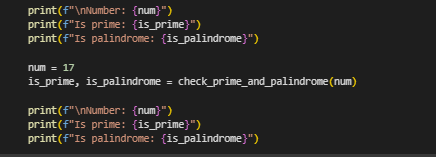
1. Checks if 153 and 123 are Armstrong numbers using the function.
2. Prints results in format:  
   "XXX is an Armstrong number: True/False".

**PROMPT 03 :**

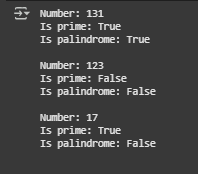
I need a python code , function check the number is prime number or a palindrome

**CODE :**

****

****

**OUTPUT :**

****

**CODE EXPLANATION :**

Function: check\_prime\_and\_palindrome(number)

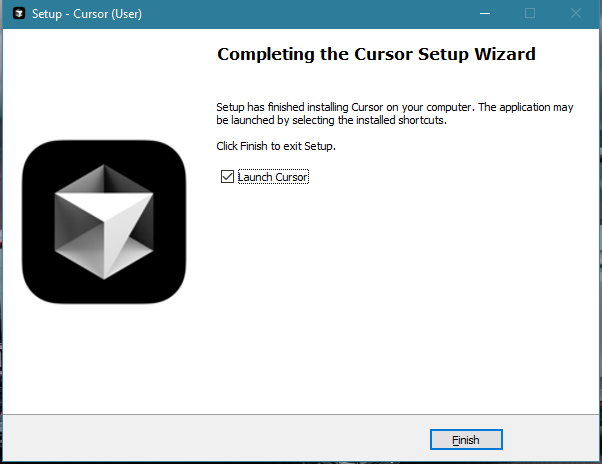
1. Purpose: Checks if a number is prime and if it is a palindrome.
2. Prime check:
   * If number ≤ 1 → Not prime.
   * Else → Check divisibility from 2 to √number.
   * If divisible → Not prime, else prime.
3. Palindrome check:
   * Convert number to string: num\_str = str(number).
   * Compare with reversed string: num\_str[::-1].
   * If equal → palindrome.
4. Return values:
   * (is\_prime, is\_palindrome) as a tuple of booleans.

Example usage

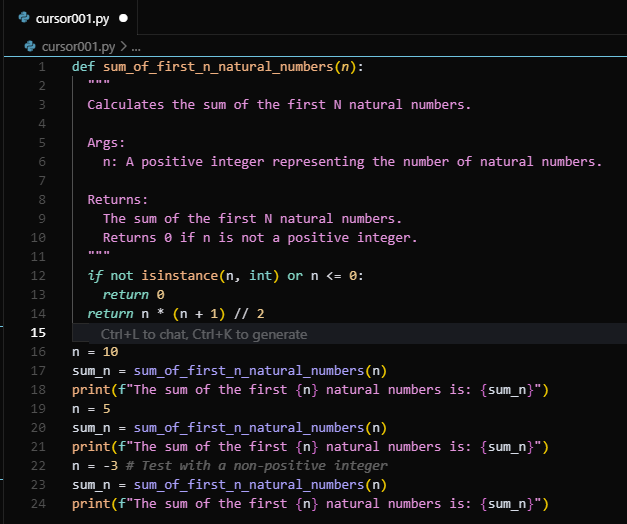
1. Test with numbers 131, 123, and 17.
2. Print:
   * The number.
   * Whether it’s prime.
   * Whether it’s a palindrome.

**PROMPT 04 :**

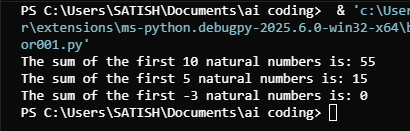
i need a pyhton code , function sum of the first N natural numbers and test the output



**CODE :**

****

**OUTPUT :**

****

**CODE EXPLANATION :**

**Function Purpose**:  
Calculates the sum of the first n natural numbers using the mathematical formula:

Sum=n(n+1)2\text{Sum} = \frac{n(n+1)}{2}Sum=2n(n+1)​

**Input Validation**:  
Checks if the input n is a positive integer.

* If not, returns 0.

**Sum Calculation**:  
Uses integer division (//) to compute the sum efficiently without loops.

**Function Calls**:

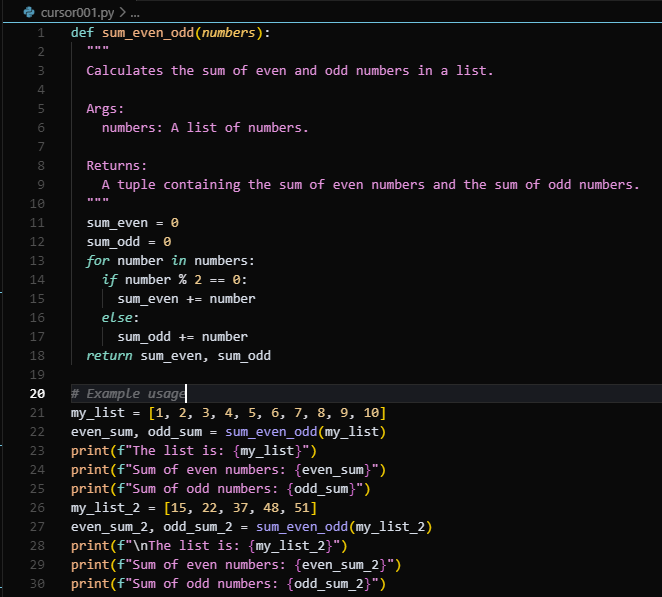
* n = 10 → Prints the sum of first 10 natural numbers.
* n = 5 → Prints the sum of first 5 natural numbers.
* n = -3 → Invalid input, so it prints 0.

**Output**:  
Demonstrates how the function behaves for valid and invalid inputs.

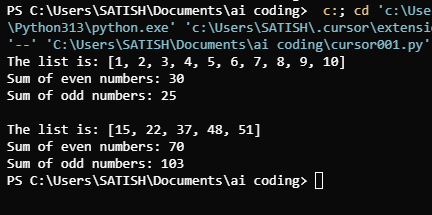
**PROMPT 05 :**

i need a pyhton code , sum of even numbers and odd numbes

**CODE :**

****

**OUTPUT :**

****

**CODE EXPLANATION :**

**Function Purpose**:  
Calculates and returns the sum of even and odd numbers from a given list.

**Parameters:**

* numbers: A list of integers.

**Initialization:**

* sum\_even and sum\_odd are both initialized to 0.

**Iteration and Condition:**

* For each number in the list:
  + If the number is even (number % 2 == 0), it's added to sum\_even.
  + If the number is odd, it's added to sum\_odd.

**Return Value:**  
Returns a tuple: (sum\_even, sum\_odd).

**Example Usage:**

* For my\_list = [1, 2, 3, ..., 10]:
  + Even sum = 2 + 4 + 6 + 8 + 10 = 30
  + Odd sum = 1 + 3 + 5 + 7 + 9 = 25
* For my\_list\_2 = [15, 22, 37, 48, 51]:
  + Even sum = 22 + 48 = 70
  + Odd sum = 15 + 37 + 51 = 103

**Output:**  
Displays the original list and the respective even and odd sums.