**AI ASSISTED CODING ASSIGNMENT -02**

**HALLTICKET NUMBER : 2403A52373**

**NAME :KODURU ESWAR REDDY**

**Prompt 01:** Write a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values

**Code:**A screenshot of a computer program

AI-generated content may be incorrect.

**Code explanation:**

* def analyze\_numbers(numbers):
* Defines a function taking a list of numbers.
* if not numbers:
* return None, None, None
* If the list is empty, returns None for all.
* mean\_value = sum(numbers) / len(numbers)
* Calculates the average (mean).
* min\_value = min(numbers)
* Finds the smallest number.
* max\_value = max(numbers)
* Finds the largest number.
* return mean\_value, min\_value, max\_value
* Returns the mean, minimum, and maximum.
* my\_list = [10, 20, 30, 40, 50]
* Creates a sample list.
* mean, minimum, maximum = analyze\_numbers(my\_list)
* Calls the function and stores results.
* if mean is not None:
* Checks if the list was not empty.
* print(f"List: {my\_list}")
* print(f"Mean: {mean}")
* print(f"Minimum: {minimum}")
* print(f"Maximum: {maximum}")
* Prints the list and results if not empty.
* else:
* print("The list is empty.")
* Prints a message if the list is empty..

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Prompt 02: Write a Python code that checks whether a number is an Armstrong number or not**

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

def is\_armstrong\_number(number):

This line defines a Python function called is\_armstrong\_number that takes one argument, number, which is the integer you want to check.

  # Convert the number to a string to easily access its digits  
  num\_str = str(number)

This converts the input number into a string. This is done so we can easily iterate through each digit of the number.

  num\_digits = len(num\_str)

This line calculates the number of digits in the original number by getting the length of the string representation.

  armstrong\_sum = 0

This initializes a variable armstrong\_sum to 0. This variable will store the sum of the digits raised to the power of the number of digits.

  # Iterate through each digit  
  For digit in num\_str:

This starts a for loop that iterates through each character (which is a digit as a string) in the num\_str.

    armstrong\_sum += int(digit) \*\* num\_digits

Inside the loop, for each digit:

* int(digit) converts the digit character back into an integer.
* \*\* num\_digits raises that integer digit to the power of the total number of digits in the original number.
* armstrong\_sum += ... adds this result to the armstrong\_sum.

  # Check if the sum is equal to the original number  
  return armstrong\_sum == number

After the loop finishes, this line checks if the calculated Armstrong sum is equal to the original number. It returns True if they are equal (meaning it's an Armstrong number) and False otherwise.

# Example usage:  
num\_to\_check = 153

This is a comment indicating the start of example usage. This line sets a variable num\_to\_check to the value 153 for testing.

if is\_armstrong\_number(num\_to\_check):  
  print(f"{num\_to\_check} is an Armstrong number.")  
Else:  
  print(f"{num\_to\_check} is not an Armstrong number.")

This calls the is\_armstrong\_number function with 153. If the function returns True, it prints that 153 is an Armstrong number. Otherwise, it prints that it is not.

num\_to\_check = 123

This sets num\_to\_check to a different value, 123, for another test.

if is\_armstrong\_number(num\_to\_check):  
  print(f"{num\_to\_check} is an Armstrong number.")  
Else:  
  print(f"{num\_to\_check} is not an Armstrong number.")

This is the same check and print block as before, but for the number 123.

**Output:**

**A grey rectangular sign with white text

AI-generated content may be incorrect.**

**Prompt 03:** Write a Python code to check if a number is prime or a palindrome using a function

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

import math

Imports the math module (not strictly needed here, but common for number tasks).

def is\_prime(number):

Defines a function is\_prime to check if a number is prime.

  if number <= 1: return False

Numbers 1 or less are not prime.

  if number <= 3: return True

2 and 3 are prime.

  if number % 2 == 0 or number % 3 == 0: return False

If divisible by 2 or 3, it's not prime.

  i = 5  
  while i \* i <= number:  
    if number % i == 0 or number % (i + 2) == 0:  
      return False  
    i += 6

Efficiently checks for divisors starting from 5.

  return True

If no divisors found, it's prime.

def is\_palindrome(number):

Defines a function is\_palindrome to check if a number is a palindrome.

  num\_str = str(number)

Converts the number to a string.

  return num\_str == num\_str[::-1]

Checks if the string is the same forwards and backward.

# Example usage:

Shows how to use the functions.

num\_to\_check\_prime = 17  
if is\_prime(num\_to\_check\_prime):  
  print(f"{num\_to\_check\_prime} is a prime number.")  
Else:  
  print(f"{num\_to\_check\_prime} is not a prime number.")

Checks and prints if 17 is prime.

num\_to\_check\_prime = 15  
if is\_prime(num\_to\_check\_prime):  
  print(f"{num\_to\_check\_prime} is a prime number.")  
Else:  
  print(f"{num\_to\_check\_prime} is not a prime number.")

Checks and prints if 15 is prime.

num\_to\_check\_palindrome = 121  
if is\_palindrome(num\_to\_check\_palindrome):  
  print(f"{num\_to\_check\_palindrome} is a palindrome number.")  
Else:  
  print(f"{num\_to\_check\_palindrome} is not a palindrome number.")

Checks and prints if 121 is a palindrome.

num\_to\_check\_palindrome = 123  
if is\_palindrome(num\_to\_check\_palindrome):  
  print(f"{num\_to\_check\_palindrome} is a palindrome number.")  
Else:  
  print(f"{num\_to\_check\_palindrome} is not a palindrome number.")

Checks and prints if 123 is a palindrome.

**Output:**

**A grey screen with white text

AI-generated content may be incorrect.**

**Prompt 04:** 

**Code:**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

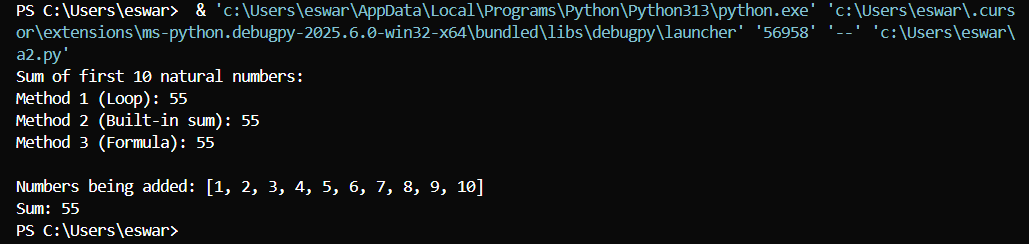
1. Loop method: Uses a for loop to iterate through numbers 1 to 10 and add them
2. Built-in sum method: Uses Python's built-in sum() function with range(1, 11)
3. Mathematical formula: Uses the formula n\*(n+1)/2, where n=10

The program will output:

* The sum calculated using each method (all should give the same result: 55)
* The list of numbers being added (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
* The final sum

You can run this program, and it will show that the sum of the first 10 natural numbers is 55.

**Output:**

****

**Prompt 05:** Write a Python code that gives the sum of even numbers and odd numbers

**Code:**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**Code explanation:**

Line 1: def sum\_even\_and\_odd\_upto(n: int):

* Defines function that takes integer n

Line 2: even\_sum = sum(number for number in range(1, n + 1) if number % 2 == 0)

* Sums all even numbers from 1 to n (numbers divisible by 2)

Line 3: odd\_sum = sum(number for number in range(1, n + 1) if number % 2 != 0)

* Sums all odd numbers from 1 to n (numbers not divisible by 2)

Line 4: return even\_sum, odd\_sum

* Returns both sums as a tuple

Line 6: if \_\_name\_\_ == "\_\_main\_\_":

* Only runs code below when script is executed directly

Line 7: n = 10

* Sets upper limit to 10

Line 8: even\_sum, odd\_sum = sum\_even\_and\_odd\_upto(n)

* Calls function and stores results

Lines 9-11: Print statements

* Display the range and both sums

**Output:**

**A computer screen with white text

AI-generated content may be incorrect.**