

# TASK:1

PROMPT GIVEN: write a Python function that reads a list of numbers and calculates the mean, minimum, and maximum values..

```
def calculate_stats(numbers):  
    """  
    Calculates the mean, minimum, and maximum values of a list of numbers.  
  
    Args:  
        numbers: A list of numbers.  
  
    Returns:  
        A tuple containing the mean, minimum, and maximum values.  
    """  
    if not numbers:  
        return None, None, None # Return None for empty list  
  
    mean = sum(numbers) / len(numbers)  
    minimum = min(numbers)  
    maximum = max(numbers)  
  
    return mean, minimum, maximum  
  
# Example usage:  
my_list = [10, 20, 30, 40, 50]  
mean_value, min_value, max_value = calculate_stats(my_list)  
  
print(f"List: {my_list}")  
print(f"Mean: {mean_value}")  
print(f"Minimum: {min_value}")  
print(f"Maximum: {max_value}")
```

```
List: [10, 20, 30, 40, 50]  
Mean: 30.0  
Minimum: 10  
Maximum: 50
```

# TASK 2

Prompt given :write a python function that check whether the number is a amstrong number...

```
def is_armstrong(number):
```

```
    """Checks if a number is an Armstrong number."""
```

```
    num_str = str(number)
```

```
    n = len(num_str)
```

```
    armstrong_sum = 0
```

```
    for digit in num_str:
```

```
    armstrong_sum += int(digit) ** n
return armstrong_sum == number
```

```
# Example usage
```

```
num = 153
```

```
if is_armstrong(num):
```

```
    print(f"{num} is an Armstrong number.")
```

```
else:
```

```
    print(f"{num} is not an Armstrong number.")
```

```
num = 123
```

```
if is_armstrong(num):
```

```
    print(f"{num} is an Armstrong number.")
```

```
else:
```

```
    print(f"{num} is not an Armstrong number.")
```

OUTPUT: 153 is an Armstrong number.

123 is not an Armstrong number.

## **COPILOT CODE:**

```
def is_armstrong(number):
```

```
    """Check if a number is an Armstrong number."""
```

```
    digits = [int(d) for d in str(number)]
```

```
    power = len(digits)
```

```
    total = sum(d ** power for d in digits)
```

```
    return total == number
```

```
# Example usage
```

```
example_numbers = [153, 370, 371, 9474, 123, 100]
```

```
for num in example_numbers:  
    print(f"{num} is Armstrong: {is_armstrong(num)}")
```

OUTPUT: 153 is Armstrong: True

370 is Armstrong: True

371 is Armstrong: True

9474 is Armstrong: True

123 is Armstrong: False

100 is Armstrong: False

PS C:\Users\ramsa>..

PROMPT GIVEN: write a python function that check whether the number is a amstrong number...

Here's a Python function to check whether a number is an Armstrong number:

Python

Copy

```
def is_armstrong(number):
    # Convert the number to a string to easily iterate over digits
    digits = str(number)
    num_digits = len(digits)

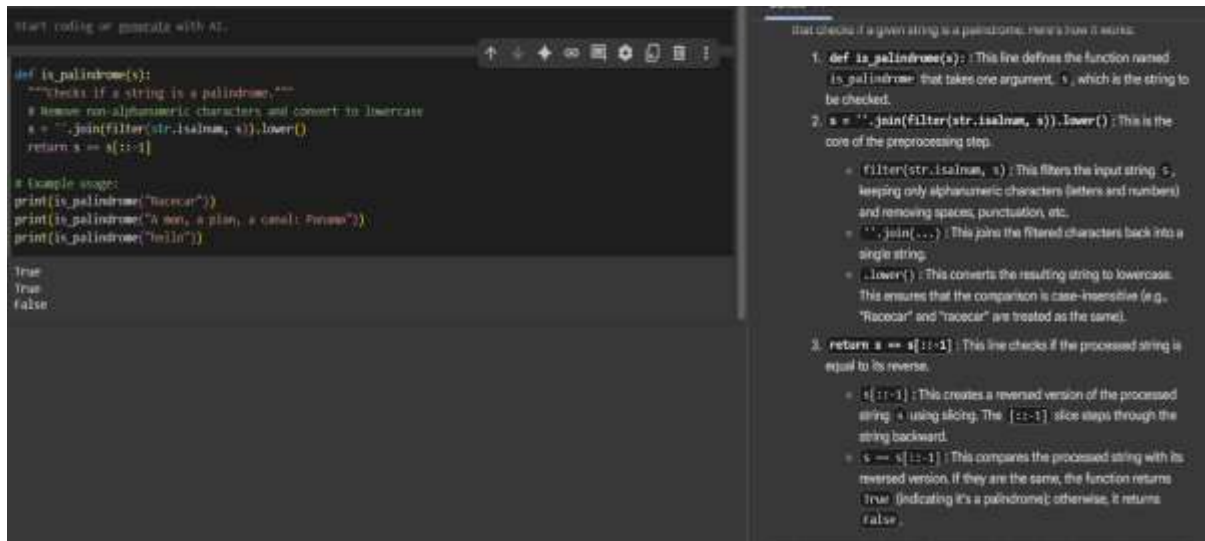
    # Calculate the sum of digits each raised to the power of num_digits
    total = sum(int(digit) ** num_digits for digit in digits)

    # Check if the total equals the original number
    return total == number

# Example usage
print(is_armstrong(153)) # True
print(is_armstrong(9474)) # True
print(is_armstrong(123)) # False
```

PROMPT :write a python function to check whether a number is an amstrong number..

# TASK 3



The screenshot shows a code editor with a Python function `is_palindrome` on the left and its explanation on the right. The function takes a string `s` and returns `True` if it is a palindrome, otherwise `False`. The explanation breaks down the function into three main steps: defining the function, preprocessing the string, and checking for a palindrome.

```
def is_palindrome(s):
    """Checks if a string is a palindrome."""
    # Remove non-alphanumeric characters and convert to lowercase
    s = ''.join(filter(str.isalnum, s)).lower()
    return s == s[::-1]

# Example usage:
print(is_palindrome("Racecar"))
print(is_palindrome("A man, a plan, a canal: Panama"))
print(is_palindrome("hello"))
```

True  
True  
False

that checks if a given string is a palindrome. Here's how it works:

1. **def is\_palindrome(s):** : This line defines the function named `is_palindrome` that takes one argument, `s`, which is the string to be checked.
2. **s = ''.join(filter(str.isalnum, s)).lower()** : This is the core of the preprocessing step.
  - `filter(str.isalnum, s)` : This filters the input string `s`, keeping only alphanumeric characters (letters and numbers) and removing spaces, punctuation, etc.
  - `''.join(...)` : This joins the filtered characters back into a single string.
  - `.lower()` : This converts the resulting string to lowercase. This ensures that the comparison is case-insensitive (e.g., "Racecar" and "racecar" are treated as the same).
3. **return s == s[::-1]** : This line checks if the processed string is equal to its reverse.
  - `s[::-1]` : This creates a reversed version of the processed string `s` using slicing. The `[::-1]` slice steps through the string backward.
  - `s == s[::-1]` : This compares the processed string with its reversed version. If they are the same, the function returns `True` (indicating it's a palindrome); otherwise, it returns `False`.

**EXPLANATION:** This code snippet defines a function called `is_palindrome` that checks if a given string is a palindrome. Here's how it works:

1. **def is\_palindrome(s):**: This line defines the function named `is_palindrome` that takes one argument, `s`, which is the string to be checked.
2. **s = ''.join(filter(str.isalnum, s)).lower()**: This is the core of the preprocessing step.
  - `filter(str.isalnum, s)`: This filters the input string `s`, keeping only alphanumeric characters (letters and numbers) and removing spaces, punctuation, etc.
  - `''.join(...)`: This joins the filtered characters back into a single string.
  - `.lower()`: This converts the resulting string to lowercase. This ensures that the comparison is case-insensitive (e.g., "Racecar" and "racecar" are treated as the same).
3. **return s == s[::-1]**: This line checks if the processed string is equal to its reverse.
  - `s[::-1]`: This creates a reversed version of the processed string `s` using slicing. The `[::-1]` slice steps through the string backward.
  - `s == s[::-1]`: This compares the processed string with its reversed version. If they are the same, the function returns `True` (indicating it's a palindrome); otherwise, it returns `False`.

In summary, the function first cleans the input string by removing non-alphanumeric characters and converting it to lowercase, and then it checks if the cleaned string reads the same forwards and backward.

**PROMPT GIVEN: 1.** Write a python function to check the palindrome...

**2.** Explain the function that had given by the gemini above...

# TASK 4

## CURSUR AI CODE:

```
def sum_first_n(n: int) -> int:

    Uses the formula  $n * (n + 1) // 2$ .

    Examples:
    >>> sum_first_n(0)
    0
    >>> sum_first_n(1)
    1
    >>> sum_first_n(5)
    15
    >>> sum_first_n(10)
    55
    """
    if n < 0:
        raise ValueError("n must be non-negative")
    return n * (n + 1) // 2

if __name__ == "__main__":
    example_numbers = [0, 1, 5, 10, 100]
    for n in example_numbers:
        print(f"sum_first_n({n}) = {sum_first_n(n)}")]
```

**PROMPT GIVEN:** Generate a Python function for sum of the first N natural numbers.

OUTPUT:  $\text{sum\_first\_n}(0) = 0$

$\text{sum\_first\_n}(1) = 1$

$\text{sum\_first\_n}(5) = 15$

$\text{sum\_first\_n}(10) = 55$

$\text{sum\_first\_n}(100) = 5050$

## GEMINI AI CODE:

```
def sum_of_natural_numbers(n):
    """Calculates the sum of the first N natural numbers."""
    if n < 1:
        return "Input must be a positive integer"
    else:
        return n * (n + 1) // 2

# Example usage:
n = 10
sum_n = sum_of_natural_numbers(n)
print(f"The sum of the first {n} natural numbers is: {sum_n}")

The sum of the first 10 natural numbers is: 55
```

Prompt Given: Generate a Python function for sum of the first N natural numbers.

Output: The sum of the first 10 natural numbers is: 55.

# TASK 5

- Python program to calculate the sum of odd numbers and even numbers in a given tuple.

```
# Given tuple
```

```
numbers = (12, 3, 5, 8, 15, 20, 7, 10)
```

```
# Initialize sums
```

```
sum_even = 0
```

```
sum_odd = 0
```

```
# Loop through the tuple
```

```
for num in numbers:
```

```
    if num % 2 == 0:
```

```
        sum_even += num
```

```
    else:
```

```
        sum_odd += num
```

```
# Display results
```

```
print("Sum of even numbers:", sum_even)
```

```
print("Sum of odd numbers:", sum_odd)
```

## **OUTPUT:**

Sum of even numbers: 50

Sum of odd numbers: 30

## SCREENSHOT:

```
C:\Users\ramsa\Downloads\WEB DESIGNING> python3 arassisted.pyt > ...

1  # Given tuple
2  numbers = (12, 3, 5, 8, 15, 20, 7, 10)
3
4  # Initialize sums
5  sum_even = 0
6  sum_odd = 0
7
8  # Loop through the tuple
9  for num in numbers:
10     if num % 2 == 0:
11         sum_even += num
12     else:
13         sum_odd += num
14 # Display results
15 print("Sum of even numbers:", sum_even)
16 print("Sum of odd numbers:", sum_odd)
17
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\ramsa> & C:/Users/ramsa/AppData/Local/Microsoft/Windows/PowerShell/PowerShell.exe -Command python3 arassisted.pyt

Sum of even numbers: 50
Sum of odd numbers: 30
PS C:\Users\ramsa> & C:/Users/ramsa/AppData/Local/Microsoft/Windows/PowerShell/PowerShell.exe -Command python3 arassisted.pyt
```

## EXPLANATION:

We have a set of numbers stored in a tuple.

We want to find two things:

1. The total (sum) of all the **even** numbers.
2. The total (sum) of all the **odd** numbers.

To do this:

- First, we create two counters: one for even numbers and one for odd numbers. Both start at zero.
- Then, we look at each number one by one.



- If the number can be divided by 2 without leaving anything left over, it's an **even** number, so we add it to the “even” total.
- Otherwise, it's an **odd** number, so we add it to the “odd” total.
- After checking all the numbers, we show both totals.

For example, with these numbers:

12, 3, 5, 8, 15, 20, 7, 10

- Even numbers are: 12, 8, 20, 10 → sum = **50**
- Odd numbers are: 3, 5, 15, 7 → sum = **30**

So the program will display:

Sum of even numbers: 50

Sum of odd numbers: 30....