1. Write a python function to list even and odd numbers in a list.

```
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
even, odd = list_even_and_odd_numbers(my_list)

print("Even numbers:", even)
print("Odd numbers:", odd)
```

```
Even numbers: [2, 4, 6, 8]
Odd numbers: [1, 3, 5, 7, 9]
```

2. Write and run a python program that asks the user to enter 8 integers (one at a time), and then prints out how many of those integers were even numbers. For example, if the user entered 19,6,9,20,13,7,6 and 1, then your program should print out 3 since 3 of those numbers were even.

```
even_count = 0

for i in range(8):
    try:
        number = int(input(f"Enter integer {i + 1}: "))
        if number % 2 == 0:
            even_count += 1
    except ValueError:
        print("Invalid input.")

print(f"Count of even numbers: {even_count}")
```

```
Enter integer 1: 19
Enter integer 2: 6
Enter integer 3: 9
Enter integer 4: 20
Enter integer 5: 13
Enter integer 6: 7
Enter integer 7: 6
```

```
Count of even numbers: 3
```

Double-click (or enter) to edit

3. Write a python program where you take any positive integer n, if n is even, divide it by 2 to get n/2. if n is odd, multiply it by 3 and add 1 to obtain 3n+1. Repeat the process until you reach 1.

```
def collatz(n):
    while n != 1:
        print(n, end=' -> ')
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
    print(1)

try:
    num = int(input("Enter a positive integer: "))
    if num <= 0:
        print("Please enter a positive integer.")
    else:
        collatz(num)
except ValueError:
    print("Invalid input.")
```

```
Enter a positive integer: 4
4 -> 2 -> 1
```

4. Write a python program to compute the sum of all the multiples of 3 or 5 below 500.

```
total_sum = 0

for num in range(1, 500):
```

```
for num in range(1, 500):
    if num % 3 == 0 or num % 5 == 0:
        total_sum += num

print("Sum of multiples of 3 or 5 below 500:", total_sum)
```

```
Sum of multiples of 3 or 5 below 500: 57918
```

5. To write a python program to find first 'n' prime numbers from a list of given numbers.

```
def is_prime(num):
    if num <= 1:
        return False
    if num <= 3:
        return True
    if num % 2 == 0 or num % 3 == 0:
        return False
    i = 5
    while i * i <= num:
        if num % i == 0 or num % (i + 2) == 0:
            return False
        i += 6
    return True

def find_first_n_primes(numbers, n):
    prime_numbers = []
    for num in numbers:
        if is_prime(num):
            prime_numbers.append(num)
            if len(prime_numbers) == n:
                break
    return prime_numbers

given_numbers = [2, 3, 5, 7, 10, 11, 13, 17, 19, 23]
n = 5

first_n_primes = find_first_n_primes(given_numbers, n)
print(f"The first {n} prime numbers from the given list are:", first_n_primes)
```

```
The first 5 prime numbers from the given list are: [2, 3, 5, 7, 11]
```

6. To write a python program to compute matrix

mutiplication.

```
A = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]

B = [[9, 8, 7],
     [6, 5, 4],
     [3, 2, 1]]

C = [[0, 0, 0],
     [0, 0, 0],
     [0, 0, 0]]

for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            C[i][j] += A[i][k] * B[k][j]
for row in C:
    print(row)
```

```
 [30, 24, 18]
 [84, 69, 54]
 [138, 114, 90]
```

7. Write a python function to count the number of vowels in a string.

```
def count_vowels(string):

    vowels = set("AEIOUaeiou")

    vowel_count = 0

    for char in string:
        if char in vowels:
            vowel_count += 1

    return vowel_count

input_string = "Hello, World!"
result = count_vowels(input_string)
print("Number of vowels:", result)
```

```
Number of vowels: 3
```

8. Write a python function for finding factorial for the given number using a recursive function.

```
def factorial(n):
  if n == 0:
    return 1
  else:
    return n * factorial(n - 1)
factorial(5)
```

```
120
```

9. Write a python function for generating the fibonacci series using the function.

```
def fibonacci_recursive(n):

    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]


    else:
        fib_series = fibonacci_recursive(n - 1)
        next_term = fib_series[-1] + fib_series[-2]
        fib_series.append(next_term)
        return fib_series

n = 10
result = fibonacci_recursive(n)
print(f"Fibonacci series of length {n}: {result}")
```

```
Fibonacci series of length 10: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

10. python program to display the given integer in reverse order using the function without an in-built function.

```
def reverse_integer(num):
    if num < 0:
        sign = "-"
        num = abs(num)
    else:
        sign = ""

    reversed_str = sign + str(num)[::-1]
    return int(reversed_str)


num = int(input("Enter an integer: "))

reversed_num = reverse_integer(num)

print("Reversed integer:", reversed_num)
```

```
 Enter an integer: 123
 Reversed integer: 321
```

11. Write a python function to display all integers within the range 200-300 whose sum of digits is an even number.

```
def is_even_sum_of_digits(num):

    digit_sum = sum(int(digit) for digit in str(num))


    return digit_sum % 2 == 0

def display_integers_with_even_digit_sum(start, end):
    if start > end:
        start, end = end, start

    for num in range(start, end + 1):
        if is_even_sum_of_digits(num):
            print(num)
```

```
start_range = 200
end_range = 300

print(f"Integers within the range {start_range}-{end_range} with even digit sums:")
display_integers_with_even_digit_sum(start_range, end_range)
```

```
Integers within the range 200-300 with even digit sums:
200
202
204
206
208
211
213
215
217
219
220
222
224
226
228
231
233
235
237
239
240
242
244
246
248
251
253
255
257
259
260
262
264
266
268
271
273
275
277
279
280
282
284
286
288
291
293
```

```
295
297
299
```

12. Write a python function to find the number of digits and sum of digits for a given integer.

```
def find_digits_and_sum(num):
    num_str = str(num)


    num_of_digits = len(num_str)


    digit_sum = sum(int(digit) for digit in num_str)

    return num_of_digits, digit_sum


num = int(input("Enter an integer: "))

num_of_digits, digit_sum = find_digits_and_sum(num)

print("Number of digits:", num_of_digits)
print("Sum of digits:", digit_sum)
```

```
Enter an integer: 1234
Number of digits: 4
Sum of digits: 10
```

13. Write functions called is_sorted that takes a list as a parameter and returns True if the list is sorted in ascending order and False otherwise and has_ duplicates that takes a list and returns True if there is any element that appears more than once. It should not modify the original list.

```
def is_sorted(arr):
```

```
    return all(arr[i] <= arr[i + 1] for i in range(len(arr) - 1))

def has_duplicates(arr):

    unique_elements = set()

    for item in arr:

        if item in unique_elements:
            return True
        unique_elements.add(item)


    return False

list1 = [1, 2, 3, 4, 5]
list2 = [1, 2, 2, 3, 4]
list3 = [5, 4, 3, 2, 1]


print("is_sorted(list1):", is_sorted(list1))
print("is_sorted(list2):", is_sorted(list2))
print("is_sorted(list3):", is_sorted(list3))


print("has_duplicates(list1):", has_duplicates(list1))
print("has_duplicates(list2):", has_duplicates(list2))
print("has_duplicates(list3):", has_duplicates(list3))
```

```
 is_sorted(list1): True
 is_sorted(list2): True
 is_sorted(list3): False
 has_duplicates(list1): False
 has_duplicates(list2): True
 has_duplicates(list3): False
```

14. Write functions called nested_sum that takes a list of integers and adds up the elements from all the nested lists and cumsum that takes a list of numbers and returns the cumulative sum; that is, a new list where the ith element is the sum of the first i + 1 elements from the original list.

```
def nested_sum(lst):
    total = 0
    for item in lst:
        if isinstance(item, list):
            total += nested_sum(item)
        else:
            total += item
    return total


nested_list = [1, 2, [3, 4], [5, [6, 7]]]
result = nested_sum(nested_list)
print("Nested Sum:", result)

def cumsum(numbers):
    cumulative_sum = []
    total = 0
    for num in numbers:
        total += num
        cumulative_sum.append(total)
    return cumulative_sum


input_list = [1, 2, 3, 4, 5]
result = cumsum(input_list)
print("Cumulative Sum:", result)
```

```
Nested Sum: 28
Cumulative Sum: [1, 3, 6, 10, 15]
```

Colab paid products - Cancel contracts here