1.fibonacci series using recursion

def fibonacci(n):

if n<0:

print('incorrect input')

return

elif n==0:

return 0

elif n==1:

return 1

else:

return fibonacci(n-1)+fibonacci(n-2)

for i in range(10):

print(fibonacci(i),end=' ')

2.armstrong number using recursion

def is\_armstrong(num):

original\_num=num

order=len(str(num))

sum\_of\_powers=0

while num>0:

digit=num%10

sum\_of\_powers+=digit\*\*order

num=num//10

return sum\_of\_powers==original\_num

number=int(input('enter the number: '))

if is\_armstrong(number):

print(number,'is an armstrong number')

else:

print(number,'is not an armstrong number')

3.FIND THE GCD OF TWO NUMBERS USING RECURSION

def gcd(a,b):

while b!=0:

a,b=b,a%b

return a

n1=12

n2=15

result=gcd(n1,n2)

print(f'gcd of {n1} and {n2} is {result}')

4.FIND THE LARGEST ELEMENT IN ARRAY USING RECURSION

def largest\_element(arr, i):

if i == len(arr) - 1:

return arr[i]

else:

largest = largest\_element(arr, i + 1)

return largest if largest > arr[i] else arr[i]

arr = [10, 25, 12, 3, 70]

largest = largest\_element(arr, 0)

print(f"Largest element in the array is {largest}")

5.FIND THE FACTORIAL OF NUMBER USING RECURSION

def factorial(n):

if n==0:

return 1

else:

return n\*factorial(n-1)

num=5

fact=factorial(num)

print('factorial of',num,'is',fact)

6.WRITE A PROGRAM FOR TO COPY ONE STRING TO ANOTHER USING RECURSION

def copy\_string(source, dest, i):

if source[i] == '\0':

return

else:

dest[i] = source[i]

copy\_string(source, dest, i + 1)

source\_str = "Hello"

dest\_str = [None] \* len(source\_str) + ["\0"] # Create destination with null terminator

copy\_string(source\_str, dest\_str, 0)

print(f"Copied string: {''.join(dest\_str[:-1])}") # Remove null terminator from output

7. REVERSE A STRING USING RECURSION

def reverse\_string(string, i):

if i == len(string) // 2:

return

else:

temp = string[i]

string[i] = string[len(string) - i - 1]

string[len(string) - i - 1] = temp

reverse\_string(string, i + 1)

text = "World"

reverse\_string(text, 0)

print(f"Reversed string: {text}")

8.PRIME NUMBERS USING RECURSION

def sieve\_of\_eratosthenes(n):

primes = [True] \* (n + 1)

primes[0] = primes[1] = False # 0 and 1 are not prime

for i in range(2, int(n\*\*0.5) + 1):

if primes[i]:

for j in range(i \* i, n + 1, i):

primes[j] = False

return [i for i, is\_prime in enumerate(primes) if is\_prime]

limit = 20

primes = sieve\_of\_eratosthenes(limit)

print(f"Prime numbers up to {limit}: {primes}")

9. CHECK WHETHER THE NUMBER IS PRIME OR NOT USING RECURSION

def is\_prime(n):

if n <= 1:

return False

elif n <= 3:

return True

elif n % 2 == 0 or n % 3 == 0:

return False

i = 5

while i \* i <= n:

if n % i == 0 or n % (i + 2) == 0:

return False

i += 6

return True

num = 11

if is\_prime(num):

print(f"{num} is a prime number")

else:

print(f"{num} is not a prime number")

10. CHECK WHETHER PALINDRIME OR NOT USING RECURSION

def is\_palindrome(string, start, end):

if start >= end:

return True

elif string[start] != string[end]:

return False

else:

return is\_palindrome(string, start + 1, end - 1)

text = "racecar"

if is\_palindrome(text, 0, len(text) - 1):

print(f"'{text}' is a palindrome")

else:

print(f"'{text}' is not a palindrome")