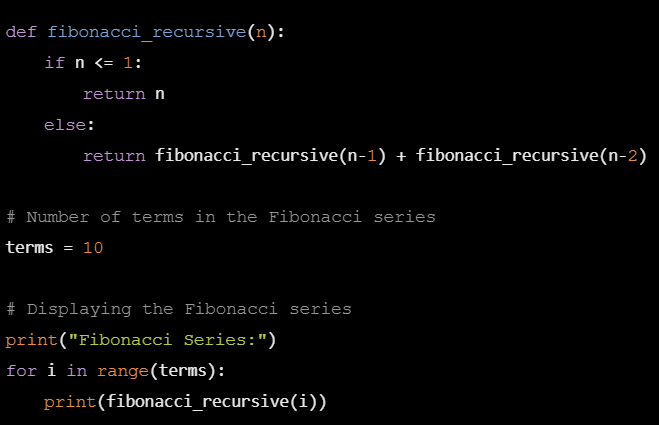
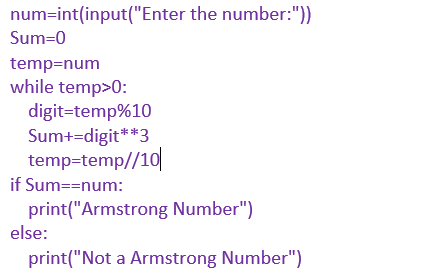
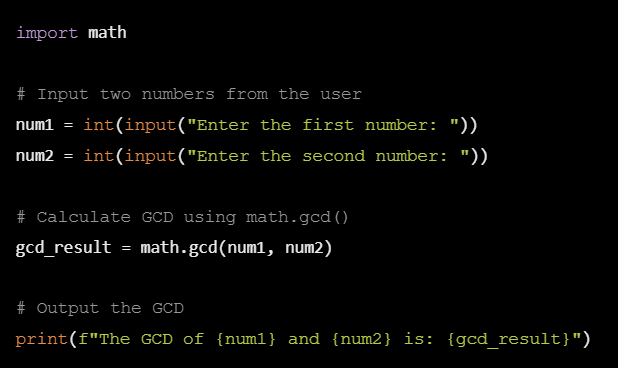
**Write a program to Print Fibonacci Series using recursion.**

****

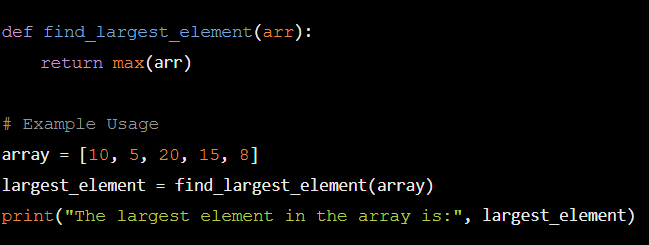
**Armstrong number:**

****

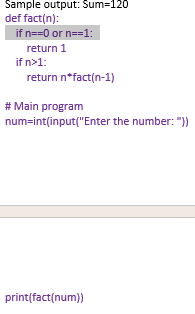
**Write a program to find the GCD of two numbers .**

****

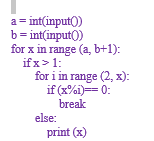
**Write a program to get the largest element of an array**

****

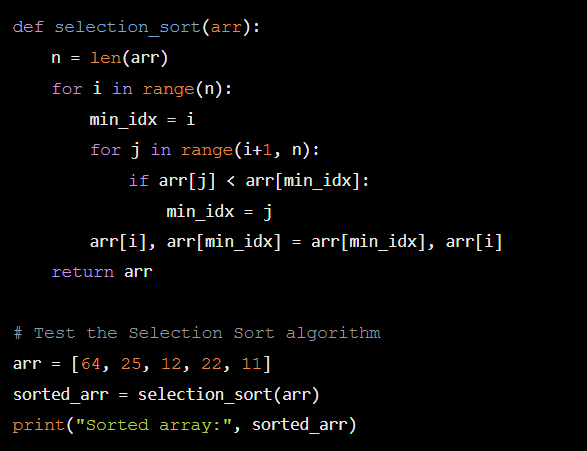
**Factorial number**

****

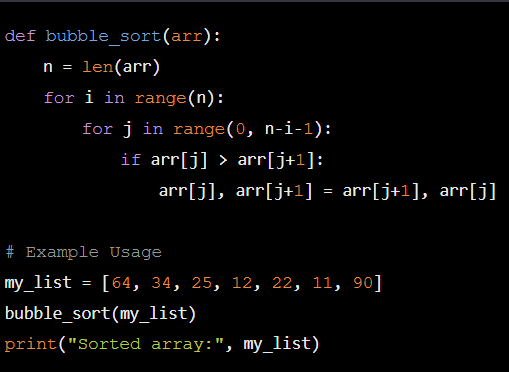
**Write a program to check a number is a prime number or not .**

****

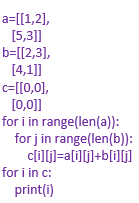
**Selection sort**

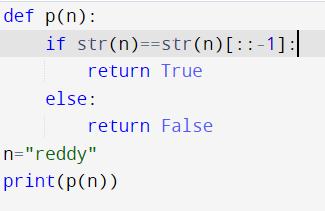
****

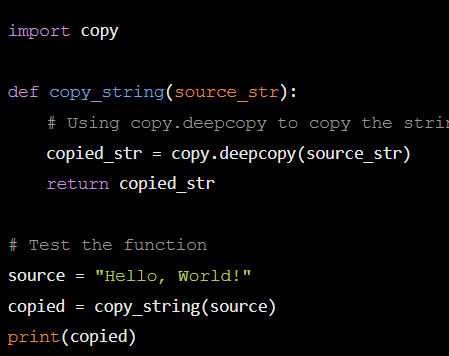
**Write a program to perform Bubble sort**

****

**Write a program for to multiply two Matrix**

****

**Write a program for to check whether a given String is Palindrome or not **

****

**Binary search:**

**import bisect**

**arr = [2, 4, 6, 8, 10, 12, 14, 16]**

**target = 6**

**index = bisect.bisect\_left(arr, target)**

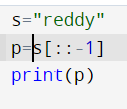
**if index < len(arr) and arr[index] == target:**

**print(f"Target found at index: {index}")**

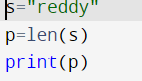
**else:**

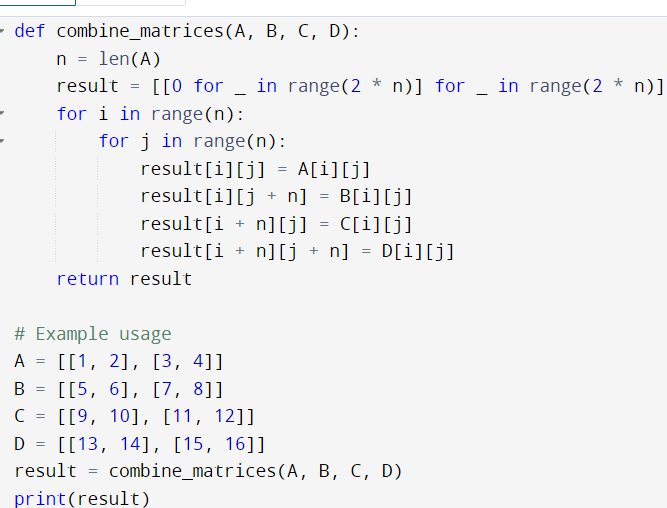
**print("Target not found")**

**reverse string:**

****

**Length of string:**

****

****

**Merge sort**

**import random**

**def generate\_random\_list(size):**

**return [random.randint(1, 100) for \_ in range(size)]**

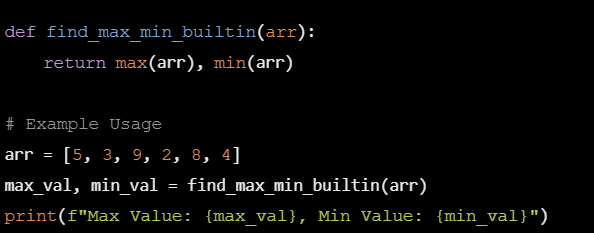
**arr = generate\_random\_list(10)**

**print("Original array:", arr)**

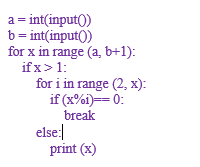
**sorted\_arr = merge\_sort(arr)**

**print("Sorted array:", sorted\_arr)**

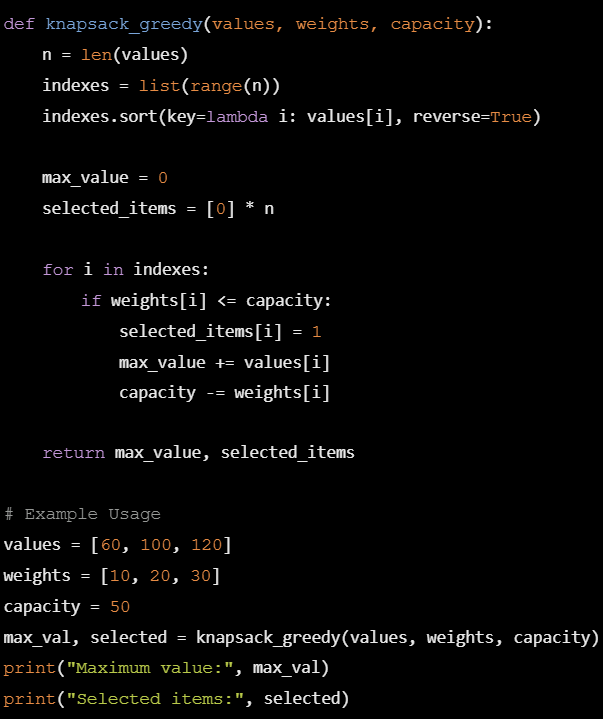
**Using Divide and Conquer strategy to find Max and Min value in the list**

****

**. Write a program to generate all the prime numbers.**

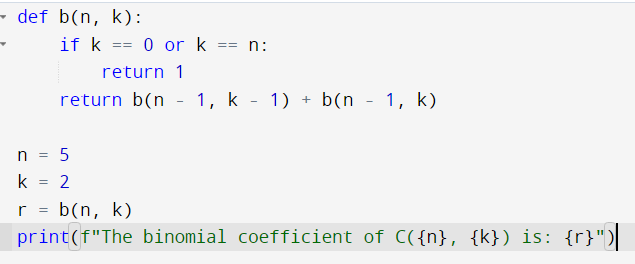
****

**to perform Knapsack problem using greedy techniques.**

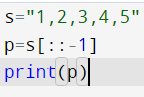
****

**Using Dynamic programming concept to find out Optimal binary search tree.**

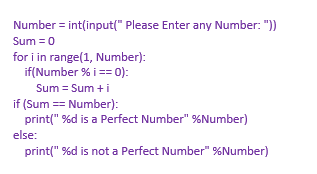
**Using Dynamic programming techniques to find binomial coefficient of a given number**

****

**Reverse number:**

****

**. Write a program to find the perfect number.**

****

**Write a program to perform travelling salesman problem using dynamic programming**

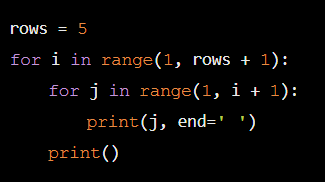
**1**

**1 2**

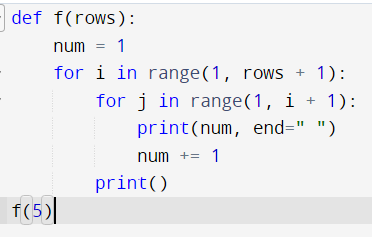
**1 2 3**

**1 2 3 4**

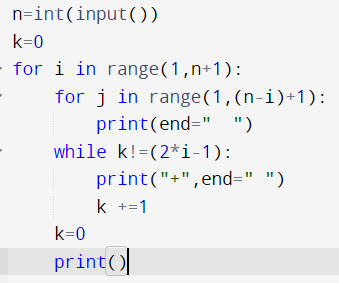
**1 2 3 4 5**

****

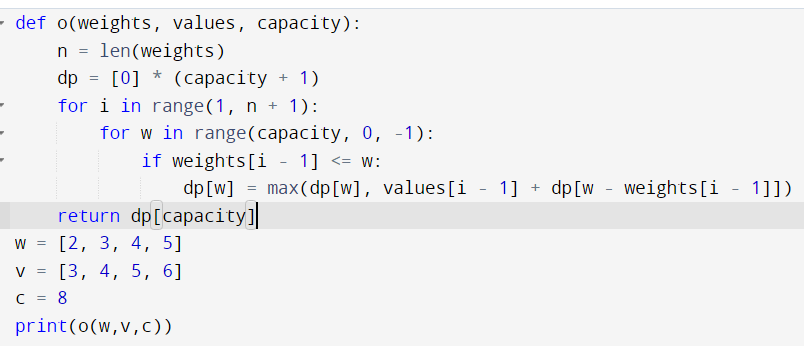
**Floyds triangle: algorthim**

****

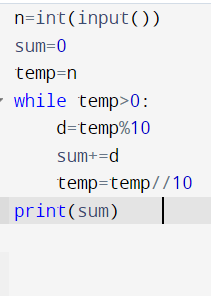
**Pascal triangle:**

****

**Write a program to find the optimal cost by using appropriate algorithm**

****

**Write a program to find the sum of digits.**

****

**Write a program for to perform liner search.**

def linear\_search\_string(words, target\_word):

for i in range(len(words)):

if words[i] == target\_word:

return i # Return the index if the target word is found

return -1 # Return -1 if the word is not found

word\_list = ["apple", "banana", "cherry", "date"]

target\_word = "banana"

result = linear\_search\_string(word\_list, target\_word)

if result != -1:

print(f"Word found at index: {result}")

else:

print("Word not found in the list")

**Write a program to generate the list of all factor for n value.**

def find\_factors(n):

factors = []

for i in range(1, n + 1):

if n % i == 0:

factors.append(i)

return factors

n = 24

print(f"Factors of {n}: {find\_factors(n)}")