#Bài 12

def hoan\_vi(a, b):

    temp = a

    a = b

    b = temp

    return a, b

def interchange\_sort(arr):

    n = len(arr)

    for i in range(n - 1):

        for j in range(i + 1, n):

            if arr[i] > arr[j]:

                arr[i], arr[j] = hoan\_vi(arr[i], arr[j])

def bubble\_sort(arr):

    n = len(arr)

    for i in range(n - 1):

        for j in range(n - 1):

            if arr[j] > arr[j + 1]:

                arr[j], arr[j + 1] = hoan\_vi(arr[j], arr[j + 1])

# Nhập danh sách các số nguyên từ người dùng

DaySo = input("Nhập một danh sách các số nguyên, cách nhau bằng dấu cách: ")

PhanTu = list(map(int, DaySo.split(',')))

# Tạo bản sao của danh sách để sắp xếp bằng các thuật toán khác nhau

PhanTu\_interchange = PhanTu[:]

PhanTu\_bubble = PhanTu[:]

# Sắp xếp bằng Interchange Sort

interchange\_sort(PhanTu\_interchange)

print(f"Danh sách sau khi sắp xếp bằng Interchange Sort: {PhanTu\_interchange}")

# Sắp xếp bằng Bubble Sort

bubble\_sort(PhanTu\_bubble)

print(f"Danh sách sau khi sắp xếp bằng Bubble Sort: {PhanTu\_bubble}")

# Bài 13

def sequential\_search(arr, x):

    for i in range(len(arr)):

        if arr[i] == x:

            return i

    return -1

def binary\_search(arr, x):

    left = 0

    right = len(arr) - 1

    while left <= right:

        mid = (left + right) // 2

        if arr[mid] == x:

            return mid

        elif arr[mid] < x:

            left = mid + 1

        else:

            right = mid - 1

    return -1

# Nhập danh sách các số nguyên từ người dùng

DaySo = input("Nhập một danh sách các số nguyên: ")

PhanTu = list(map(int, DaySo.split(',')))

# Nhập số cần tìm

x = int(input("Nhập số cần tìm: "))

# Tìm kiếm tuần tự

index\_seq = sequential\_search(PhanTu, x)

if index\_seq != -1:

    print(f"Số {x} được tìm thấy tại vị trí {index\_seq} bằng tìm kiếm tuần tự.")

else:

    print(f"Số {x} không có trong danh sách theo tìm kiếm tuần tự.")

# Sắp xếp danh sách trước khi thực hiện tìm kiếm nhị phân

PhanTu.sort()

# Tìm kiếm nhị phân

index\_bin = binary\_search(PhanTu, x)

if index\_bin != -1:

    print(f"Số {x} được tìm thấy tại vị trí {index\_bin} trong danh sách đã sắp xếp bằng tìm kiếm nhị phân.")

else:

    print(f"Số {x} không có trong danh sách đã sắp xếp theo tìm kiếm nhị phân.")

#Bài 14

alphabet=input("Nhập dãy alphabet:") #Nhập A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z vào

phanTu=alphabet.split(',')

def column\_to\_number(column):

    result = 0

    for char in column:

        value = phanTu.index(char) + 1

        result = result \* 26 + value

    return result

def number\_to\_column(number):

    result = ""

    while number > 0:

        number -= 1

        du = number % 26

        char = phanTu[du]  # Chuyển đổi phần dư thành ký tự

        result = char + result  # Thêm ký tự vào kết quả

        number //= 26

    return result

# Nhập chuỗi cột từ người dùng

column = input("Nhập ký tự cột: ")

to\_number = column\_to\_number(column)

print(f"Ký tự cột '{column}' tương ứng với số {to\_number}")

# Nhập số cột từ người dùng

number = int(input("Nhập số cột: "))

to\_column = number\_to\_column(number)

print(f"Số {number} tương ứng với ký tự cột '{to\_column}'")

#Bài 15 (Cài numpy trước)

import numpy as np

# Nhập số nguyên dương n từ người dùng

n = int(input("Nhập số nguyên dương n: "))

# Tạo ma trận đơn vị cấp n

identity\_matrix = np.eye(n)

# Xuất ma trận đơn vị

print("Ma trận đơn vị cấp", n, "là:")

print(identity\_matrix)

#-----------------------------------------------------------------------------

#Lab02

#Bài 01

suits = input("Nhập từng thuộc tính bài (phân tách bằng dấu phẩy): ")

suit = suits.split(',')

print("List suits sau khi nhập từ người dùng:", suit)

values = input(("Nhập số thứ tự của từng lá bài:"))

value = values.split(',')

print("List values sau khi nhập từ người dùng:", value)

# Duyệt qua từng giá trị bằng biến j và từng chất bằng biến i để in ra tên tất cả các quân bài

for i in suit:

    for j in value:

        print(f"{j} of {i}")

#Vì sao lại là "i in suit, j in value" thay vì "suit in suits, value in values": Chúng ta gặp hiểu nhầm khi sử dụng suit và value như biến của chuỗi, chứ không phải là list. Khi duyệt qua các ký tự của suits và values, chúng ta chỉ lặp qua từng ký tự một trong mỗi chuỗi, và do đó, kết quả sẽ không chính xác như mong đợi.

#Trường hợp "i in suit, j in value": chúng ta sử dụng suit để lặp qua các phần tử của list suit, và value để lặp qua các phần tử của list value. Với mỗi giá trị của suit, chúng ta duyệt qua tất cả các giá trị của value và in ra tên của tất cả các quân bài.

#Bài 2

# Nhập dãy kí tự.

DayKiTu = input("Nhập dãy kí tự: ")

PhanTu=DayKiTu.split(',')

# Nhập số hàng và số cột.

n = int(input("Nhập số hàng: "))

m = int(input("Nhập số cột: "))

# Tách dữ liệu thành các hàng của ma trận.

matrix = [PhanTu[i:i+m] for i in range(0, len(PhanTu), m)]

# Đếm số phần tử trong list.

numItems = len(PhanTu)

# Đếm tần số xuất hiện của từng giá trị và lưu chúng trong một biến kiểu dictionary.

frequency\_dict = {}

for row in matrix:

    for item in row:

        if item in frequency\_dict:

            frequency\_dict[item] += 1

        else:

            frequency\_dict[item] = 1

# In ra ma trận.

print("Ma trận:")

for row in matrix:

    print(row)

print()

# In số hàng của ma trận.

print(f"Số phần tử của ma trận: {numItems}")

# In tần số xuất hiện của từng giá trị.

print("Tần số xuất hiện của từng giá trị:")

for key, value in frequency\_dict.items():

    print(f"{key}: {value}")

#Bài 3

width = int(input("Nhập chiều rộng:"))

length = int(input("Nhập chiều dài:"))

for i in range(width):

        print('\*' \* length)

#Bài 4

height = int(input("Nhập chiều cao:"))

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

    if i == 1 or i == height:

        print('\*' \* i)

    else: #Nếu i == 2 trở lên đến height - 1

        print('\*' + ' ' \* (i - 2) + '\*') #i == 2 => 2 - 2 = 0 => print('\*' + ' ' \* 0 + '\*') => print('\*' + '\*')

#Bài 5

def FindFibonacci(n):

    # Xác định hai số đầu tiên của dãy Fibonacci

    fibo = [0, 1]

    # Tính toán các số Fibonacci tiếp theo và thêm vào list fib cho đến khi đạt được n

    while len(fibo) < n:

        next\_fibo = fibo[-1] + fibo[-2]

        fibo.append(next\_fibo)

    # Trả về list các số Fibonacci

    return fibo

def FindNumberFibonacci(n):

    # Xác định hai số đầu tiên của dãy Fibonacci

    fibo = [0, 1]

    # Tính toán các số Fibonacci tiếp theo và thêm vào list fib

    for i in range(2, n):

        next\_fibo = fibo[-1] + fibo[-2]

        fibo.append(next\_fibo)

    # Trả về giá trị Fibonacci thứ n

    if n > 0:

        return fibo[n - 1]

    else: print("Không có giá trị Fibonacci cho n <= 0")

n = int(input("Nhập n:"))

print(f"Dãy số fibonacci như sau:", FindFibonacci(n))

print(f"Giá trị Fibonacci thứ {n} là:", FindNumberFibonacci(n))

#Bài 6

import numpy as np

def create\_random\_array(n):

    # Tạo mảng 1 chiều gồm n phần tử có giá trị ngẫu nhiên

    if n % 5 == 0 and n > 20:

        return np.random.randint(0, 100, n)

    else:

        print("n không chia hết cho 5 hoặc n không lớn hơn 20")

        return None #None == Null trong python

def print\_array(arr):

    # Xuất mảng

    print("Mảng 1 chiều:")

    print(arr)

def reshape\_array(arr):

    # Chuyển mảng 1 chiều thành mảng 2 chiều số cột là 5

    return arr.reshape(-1, 5)

def print\_array\_info(arr):

    # In các thông tin: shape, ndim, dtype, itemsize, size

    print("Shape:", arr.shape)

    print("Number of dimensions:", arr.ndim)

    print("Data type:", arr.dtype)

    print("Size of each element (in bytes):", arr.itemsize)

    print("Total number of elements:", arr.size)

def count\_occurrences(arr\_2d, x):

    # Đếm tần số xuất hiện của giá trị x trong mảng 2 chiều

    return np.count\_nonzero(arr\_2d == x)

def array\_sum(arr1, arr2):

    # Tính tổng 2 mảng

    return np.add(arr1, arr2)

def array\_product(arr1, arr2):

    # Tính tích 2 mảng

    return np.multiply(arr1, arr2)

# Kiểm tra các hàm

n = int(input("Nhập n:"))

a = create\_random\_array(n)

if a is not None:

    print\_array(a)

    mang2chieu = reshape\_array(a)

    print\_array(mang2chieu)

    print\_array\_info(mang2chieu)

    x = 5

    print(f"Số lần xuất hiện của giá trị {x} trong mảng 2 chiều:", count\_occurrences(mang2chieu, x))

    a1 = np.random.randint(0, 100, mang2chieu.shape)

    a2 = np.random.randint(0, 100, mang2chieu.shape)

    print("Tổng của hai mảng:")

    print(array\_sum(a1, a2))

    print("Tích của hai mảng:")

    print(array\_product(a1, a2))