**DATA SCIENCE & MACHINE LEARNING**

**LAB CYCLE 1**

1. **Program to Print all non-Prime Numbers in an Interval.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def is\_prime(num):

if num <= 1:

return False

for i in range (2,num):

if num % i == 0:

return False

return True

f=int(input("Enter the starting number: "))

l=int(input("Enter the end number: "))

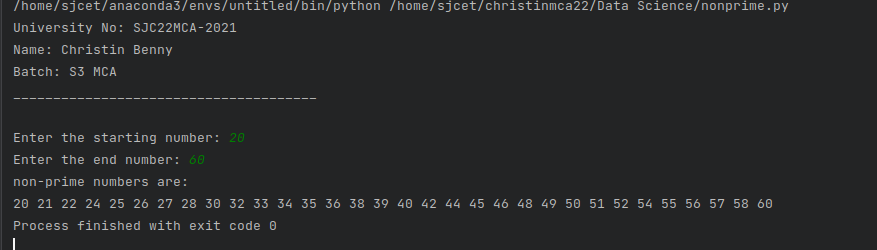
print("non-prime numbers are: ")

for num in range(f,l+1):

if not is\_prime(num):

print(num,end=" ")

**Output:**

****

1. **Program to print the first N Fibonacci numbers.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

num=int(input("enter the range:"))

n1, n2 = 0, 1

print("Fibonacci Series:", n1, n2, end=" ")

for i in range(2, num):

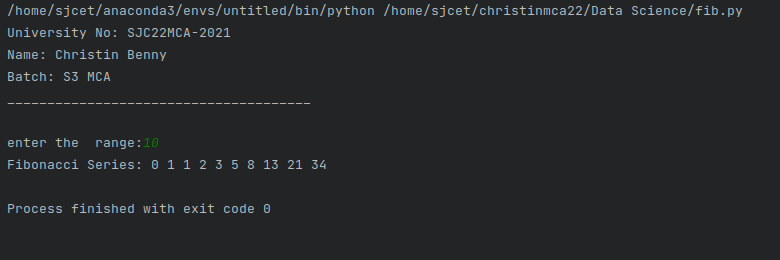
n3 = n1 + n2

n1 = n2

n2 = n3

print(n3, end=" ")

print()

**Output:**

1. **Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

print("Enter the lengths of the triangle sides: ")

s1 = int(input("s1: "))

s2 = int(input("s2: "))

s3 = int(input("s3: "))

if s1 == s2 == s3:

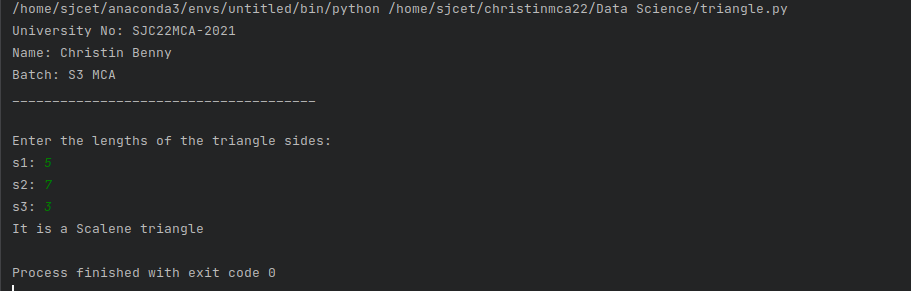
print("It is an Equilateral triangle")

elif s1 == s2 or s2 == s3 or s3 == s1:

print("It is an Isosceles triangle")

else:

print("It is a Scalene triangle")

**Output:**

1. **Program to check whether given pair of number is coprime.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def gcd(a, b):

while b:

a, b = b, a % b

return a

def are\_coprime(num1, num2):

return gcd(num1, num2) == 1

# Input two numbers

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

if are\_coprime(num1, num2):

print(f"{num1} and {num2} are coprime.")

else:

print(f"{num1} and {num2} are not coprime.")

**Output:**

1. **Program to find the roots of a quadratic equation(rounded to 2 decimal places).**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

import math

a = float(input("Enter the coefficient a: "))

b = float(input("Enter the coefficient b: "))

c = float(input("Enter the coefficient c: "))

discriminant = b\*\*2 - 4\*a\*c

if discriminant > 0:

root1 = (-b + math.sqrt(discriminant)) / (2\*a)

root2 = (-b - math.sqrt(discriminant)) / (2\*a)

print(f"Root 1: {round(root1, 2)}")

print(f"Root 2: {round(root2, 2)}")

elif discriminant == 0:

root = -b / (2\*a)

print(f"Root: {round(root, 2)}")

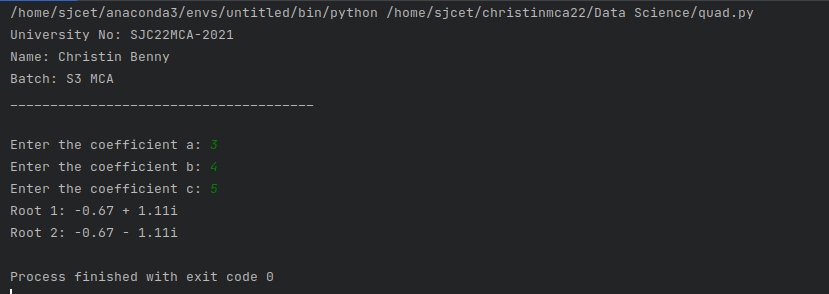
else:

real = -b / (2\*a)

imag = math.sqrt(-discriminant) / (2\*a)

print(f"Root 1: {round(real, 2)} + {round(imag, 2)}i")

print(f"Root 2: {round(real, 2)} - {round(imag, 2)}i")

**Output:**

1. **Program to check whether a given number is perfect number or not(sum of factors =number).**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def is\_perfect\_number(num):

if num <= 0:

return False

sum = 0

for i in range(1, num):

if num % i == 0:

sum += i

print("The sum of factors:", sum)

return sum == num

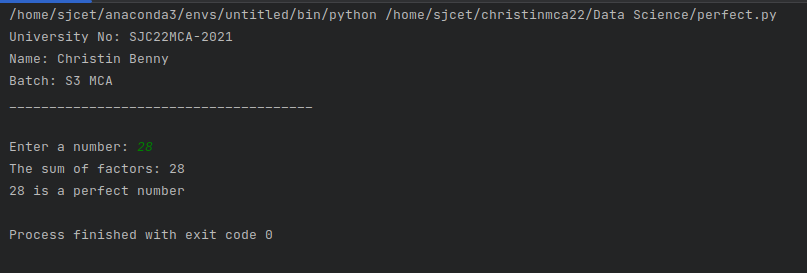
num = int(input("Enter a number: "))

if is\_perfect\_number(num):

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

else:

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

**Output:**

1. **Program to display amstrong numbers upto 1000.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def is\_armstrong\_number(num):

num\_str = str(num)

num\_digits = len(num\_str)

digit\_sum = sum(int(digit) \*\* num\_digits for digit in num\_str)

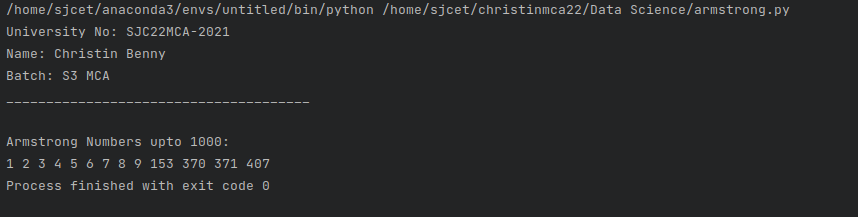
return num == digit\_sum

print("Armstrong Numbers upto 1000:")

for i in range(1, 1001):

if is\_armstrong\_number(i):

print(i,end=" ")

**Output:**

1. **Store and display the days of a week as a List, Tuple, Dictionary, Set. Also demonstrate different ways to store values in each of them. Display its type also.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

days\_list = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

print("List:", days\_list)

print("Type of List:", type(days\_list))

days\_tuple = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

print("Tuple:", days\_tuple)

print("Type of Tuple:", type(days\_tuple))

days\_dict = {1: "Monday", 2: "Tuesday", 3: "Wednesday", 4: "Thursday", 5: "Friday", 6: "Saturday", 7: "Sunday"}

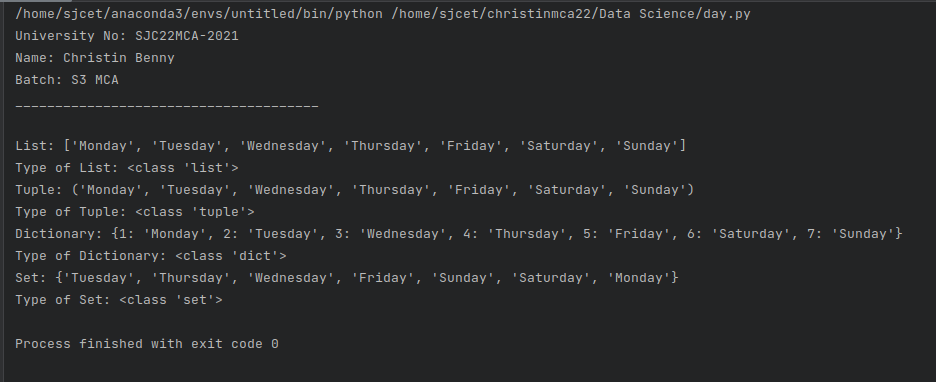
print("Dictionary:", days\_dict)

print("Type of Dictionary:", type(days\_dict))

days\_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}

print("Set:", days\_set)

print("Type of Set:", type(days\_set))

**Output:**

1. **Write a program to add elements of given 2 lists.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

list1 = [2, 4, 6, 8, 10]

list2 = [1, 3, 5, 7, 9]

print("The First list is:", list1)

print("The Second list is:", list2)

if len(list1) != len(list2):

print("Lists must have the same length for addition.")

else:

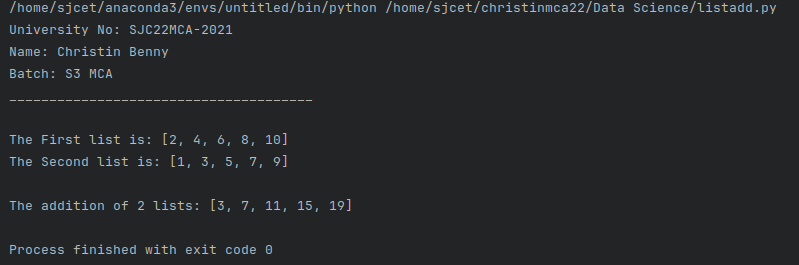
result = []

for i in range(len(list1)):

sum\_element = list1[i] + list2[i]

result.append(sum\_element)

print("\nThe addition of 2 lists:", result)

**Output:**

1. **Write a program to find the sum of 2 matrices using nested List.**

**Code:**

def add\_matrices(matrix1, matrix2):

if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):

return None

result = []

for i in range(len(matrix1)):

row = []

for j in range(len(matrix1[0])):

row.append(matrix1[i][j] + matrix2[i][j])

result.append(row)

return result

def print\_matrix(matrix):

for row in matrix:

print(row)

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

rows = int(input("Enter the number of rows: "))

cols = int(input("Enter the number of columns: "))

matrix1 = []

matrix2 = []

print("Enter elements of the first matrix:")

for i in range(rows):

row = [int(x) for x in input().split()]

matrix1.append(row)

print("Enter elements of the second matrix:")

for i in range(rows):

row = [int(x) for x in input().split()]

matrix2.append(row)

sum\_matrix = add\_matrices(matrix1, matrix2)

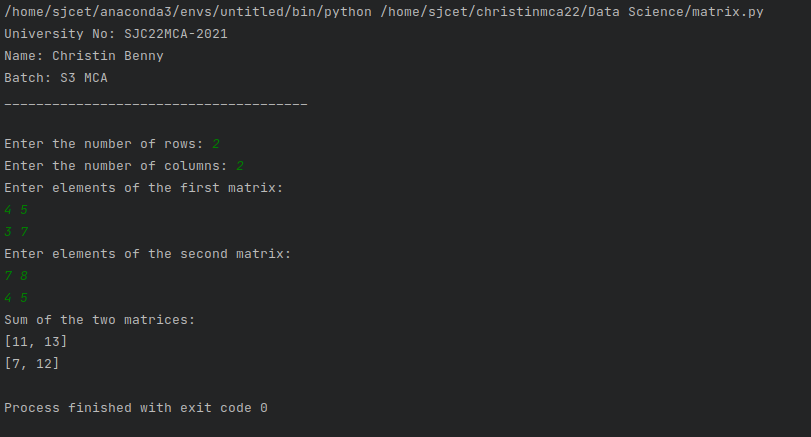
if sum\_matrix is not None:

print("Sum of the two matrices:")

print\_matrix(sum\_matrix)

else:

print("Matrices have different dimensions. Cannot perform addition.")

**Output:**

1. **Write a program to perform bubble sort on a given set of elements.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def bubble\_sort(arr):

n = len(arr)

for i in range(n - 1):

swapped = False

for j in range(n - 1 - i):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

swapped = True

if not swapped:

break

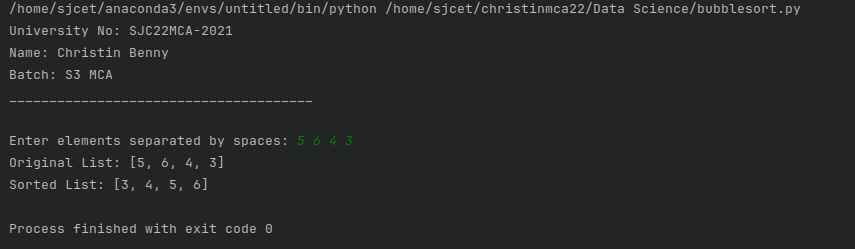
input\_str = input("Enter elements separated by spaces: ")

elements = [int(x) for x in input\_str.split()]

print("Original List:", elements)

bubble\_sort(elements)

print("Sorted List:", elements)

**Output:**

1. **Program to find the count of each vowel in a string(use dictionary).**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def count\_vowels(input\_string):

vowel\_counts = {'a': 0, 'e': 0, 'i': 0, 'o': 0, 'u': 0}

input\_string = input\_string.lower()

for char in input\_string:

if char in vowel\_counts:

vowel\_counts[char] += 1

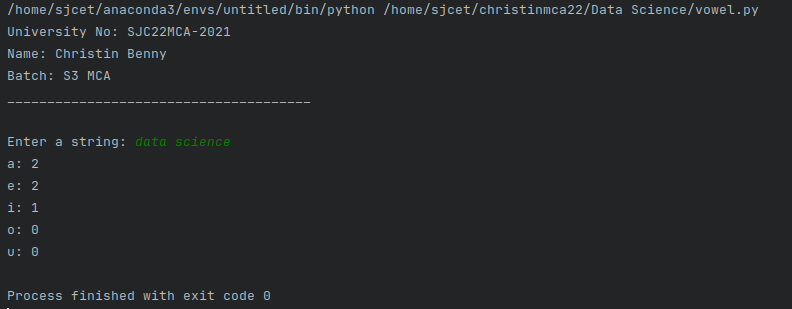
return vowel\_counts

input\_string = input("Enter a string: ")

vowel\_count = count\_vowels(input\_string)

for vowel, count in vowel\_count.items():

print(f"{vowel}: {count}")

**Output:**

1. **Write a Python program that accept a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive(eg: 256->2+5+6=13**

**256-13=243**

**243-9=232……..**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def sum\_of\_digits(n):

digit\_sum = 0

while n > 0:

digit\_sum += n % 10

n //= 10

return digit\_sum

num = int(input("Enter a positive number: "))

while num > 0:

digit\_sum = sum\_of\_digits(num)

num -= digit\_sum

print(f"{num + digit\_sum} - {digit\_sum} = {num}")

print("The number is now positive or zero.")

**Output:**

1. **Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number.**

**Code:**

print("University No: SJC22MCA-2021 \nName: Christin Benny \nBatch: S3 MCA \n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n")

def find\_absent\_digits(mobile\_number):

all\_digits = set("0123456789")

number\_digits = set(mobile\_number)

absent\_digits = all\_digits - number\_digits

return absent\_digits

mobile\_number = input("Enter a 10-digit mobile number: ")

if len(mobile\_number) == 10 and mobile\_number.isdigit():

absent\_digits = find\_absent\_digits(mobile\_number)

if absent\_digits:

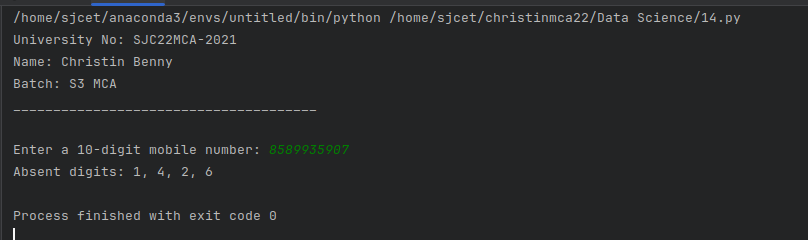
print("Absent digits:", ', '.join(absent\_digits))

else:

print("All digits are present in the mobile number.")

else:

print("Invalid input. Please enter a 10-digit mobile number.")

**Output:**