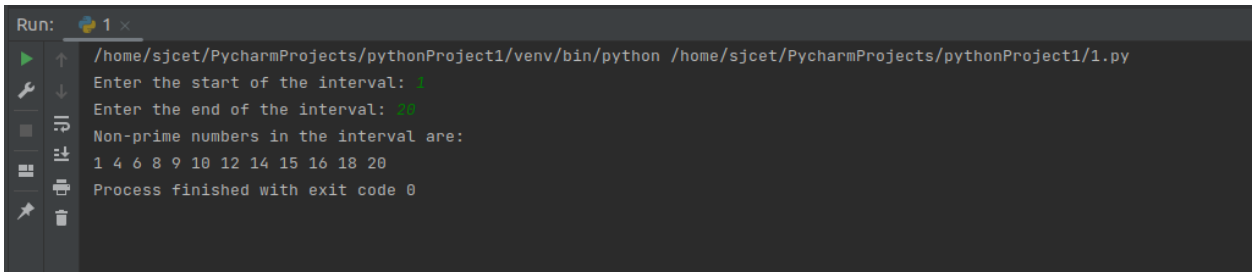
CYCLE 1

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

CODE

```
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 print_non_prime_in_interval(start, end):
    for number in range(start, end + 1):
        if not is_prime(number):
            print(number, end=' ')

start = int(input("Enter the start of the interval: "))
end = int(input("Enter the end of the interval: "))

print("Non-prime numbers in the interval:", start, "to", end, "are:")
print_non_prime_in_interval(start, end)
```

OUTPUT

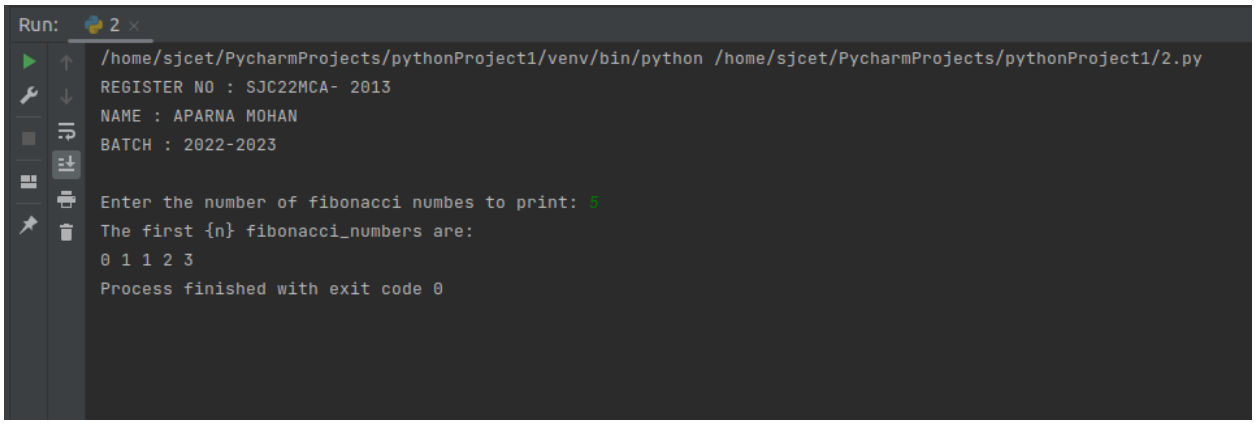


2. Program to print the first N Fibonacci numbers.

CODE

```
def print_fibonacci_numbers(n):
    fib_sequence = []
    a, b = 0, 1
    for _ in range(n):
        fib_sequence.append(a)
        a, b = b, a + b
    return fib_sequence

print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

n = int(input("Enter the number of Fibonacci numbers to print: "))

if n <= 0:
    print("Please enter a positive integer.")
else:
    fibonacci_numbers = print_fibonacci_numbers(n)
    print("The first", n, "Fibonacci numbers are:")
    print(fibonacci_numbers)
```

OUTPUT

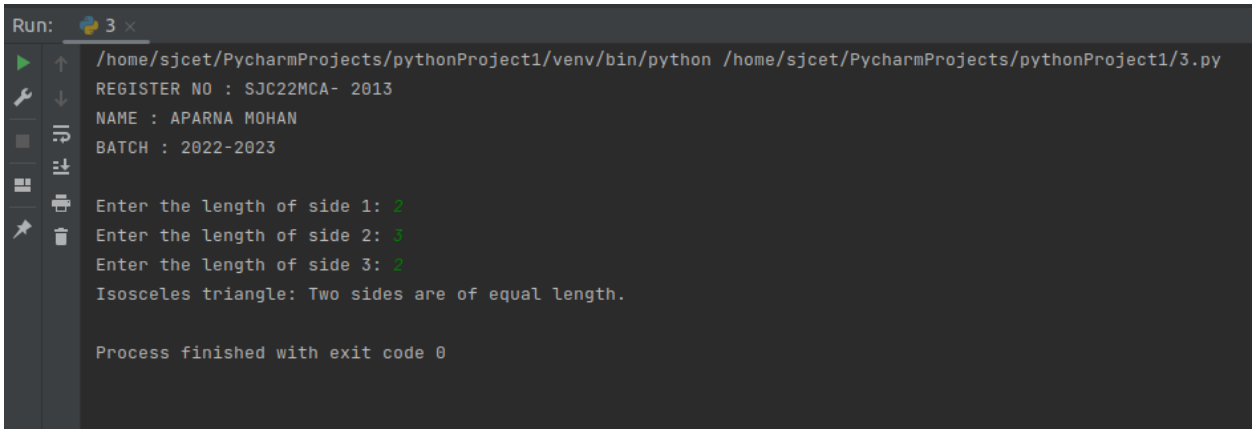


3.Given sides of a triangle, write a program to check whether given triangle is an isosceles, equilateral or scalene.

CODE

```
def check_triangle_type(a, b, c):
    if a == b == c:
        return "Equilateral"
    elif a == b or b == c or a == c:
        return "Isosceles"
    else:
        return "Scalene"

print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

a = float(input("Enter the length of side 'a': "))
b = float(input("Enter the length of side 'b': "))
c = float(input("Enter the length of side 'c': "))

if a <= 0 or b <= 0 or c <= 0:
    print("Invalid input. Side lengths must be positive.")
else:
    triangle_type = check_triangle_type(a, b, c)
    print("The given triangle is:", triangle_type)
```

OUTPUT



4. Program to check whether given pair of number is coprime.

CODE

```
import math

def are_coprime(a, b):
    gcd = math.gcd(a, b)
    return gcd == 1
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

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.")
```

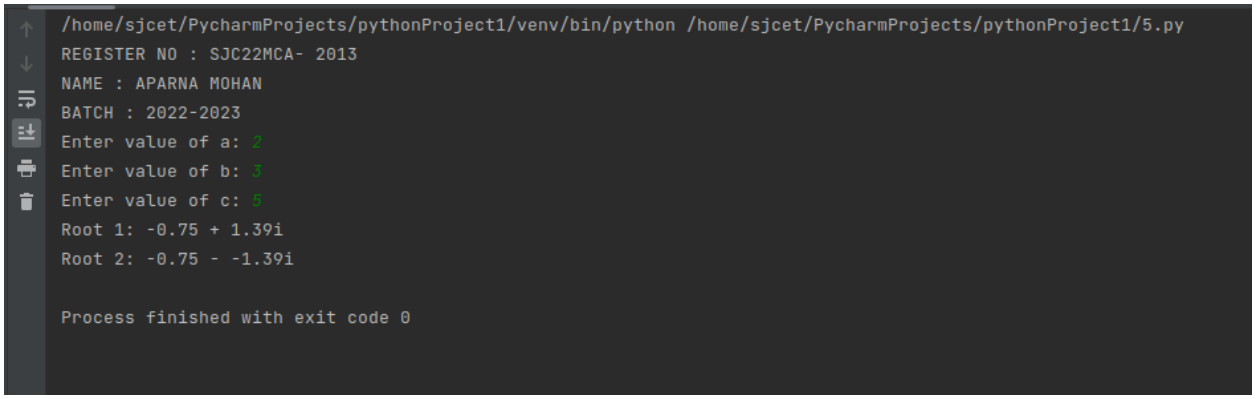
OUTPUT

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

CODE

```
import math
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

a = float(input("Enter value of a: "))
b = float(input("Enter value of b: "))
c = float(input("Enter value of c: "))
discri = b**2 - 4*a*c

if discri > 0:
    root1 = (-b + math.sqrt(discri)) / (2*a)
    root2 = (-b - math.sqrt(discri)) / (2*a)
    print(f"Root 1: {round(root1, 2)}")
    print(f"Root 2: {round(root2, 2)}")
elif discri == 0:
    root = -b / (2*a)
    print(f"Root: {round(root, 2)}")
else:
    real_part = -b / (2*a)
    img_part = math.sqrt(-discri) / (2*a)
    root1 = complex(real_part, img_part)
    root2 = complex(real_part, -img_part)
    print(f"Root 1: {root1.real:.2f} + {root1.imag:.2f}i")
    print(f"Root 2: {root2.real:.2f} - {root2.imag:.2f}i")
```

OUTPUT

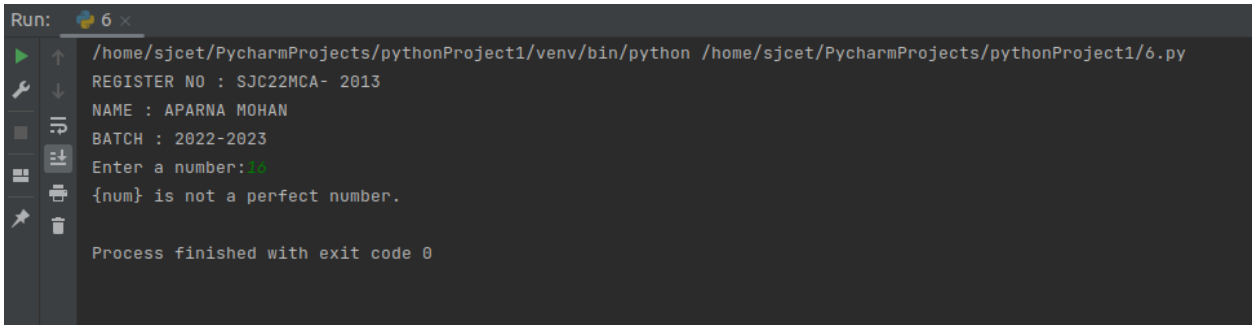


6. Program to check whether a given number is perfect number or not(sum of factor =number).

CODE

```
def is_perfect_number(num):
    if num <= 0:
        return False
    sum_of_divisors = 0
    for i in range(1, num):
        if num % i == 0:
            sum_of_divisors += i
    return sum_of_divisors == num
try:
  print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

    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.")
except ValueError:
    print("Invalid input. Please enter a valid number.")
```

OUTPUT

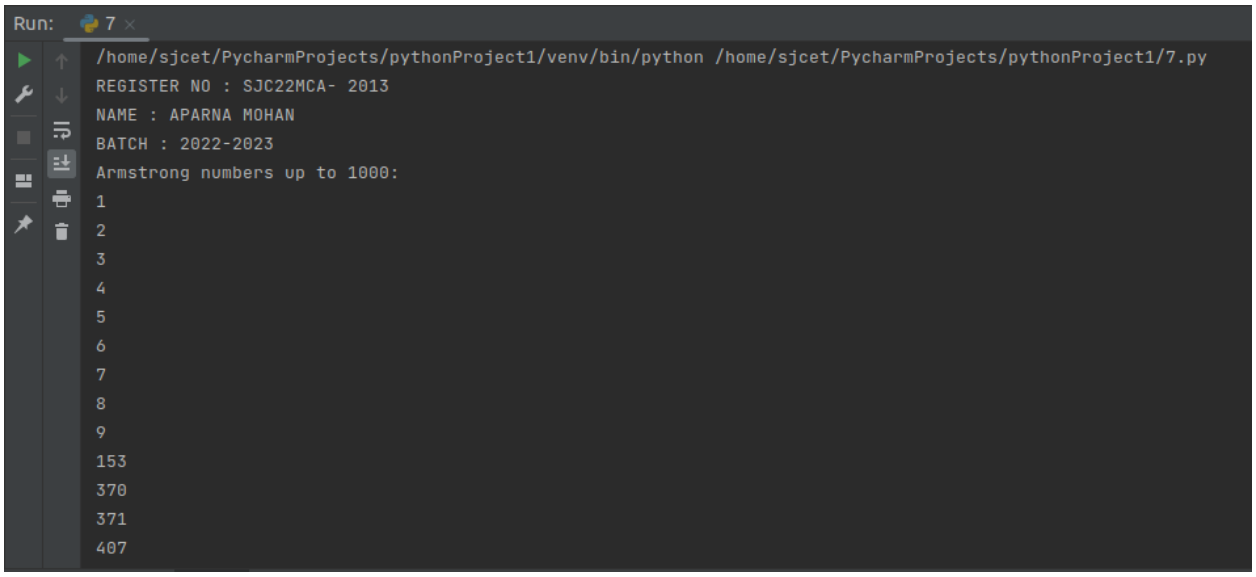


7. Program to display armstrong numbers upto 1000.

CODE

```
def is_armstrong_number(num):
    num_str = str(num)
    num_digits = len(num_str)
    armstrong_sum = sum(int(digit) ** num_digits for digit in num_str)
    return armstrong_sum == num
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

print("Armstrong numbers up to 1000:")
for num in range(1, 1001):
    if is_armstrong_number(num):
        print(num)
```

OUTPUT

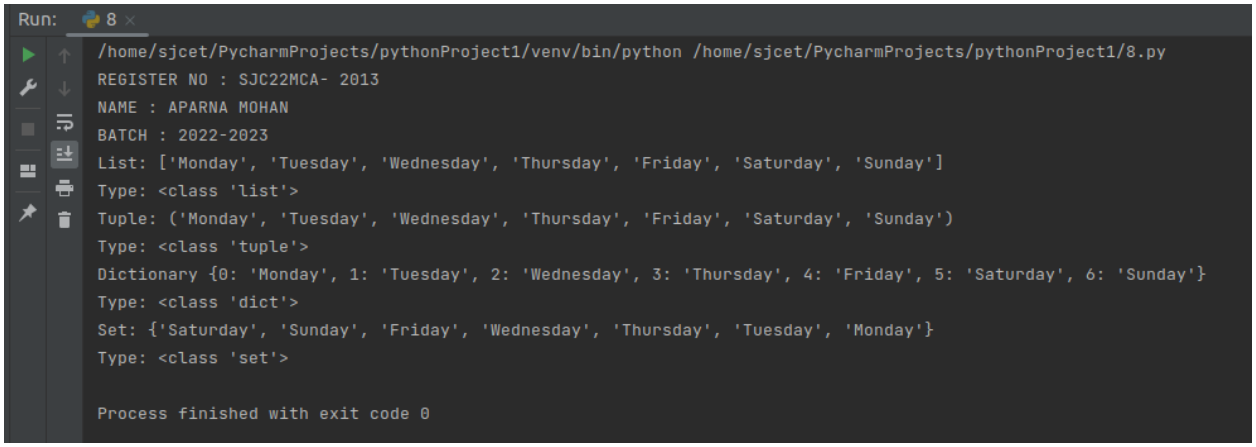


8. Store and display the days of a week as a List, Tuple, Dictionary, Set. Also demonstrate different ways to store values in each of them. Display its type also.

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

days_list = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
print("List:", days_list)
print("Type:", type(days_list))

days_tuple = ("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")
print("Tuple:", days_tuple)
print("Type:", type(days_tuple))

days_dict = {0: "Monday", 1: "Tuesday", 2: "Wednesday", 3: "Thursday", 4: "Friday", 5: "Saturday", 6: "Sunday"}
print("Dictionary:", days_dict)
print("Type:", type(days_dict))

days_set = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}
print("Set:", days_set)
print("Type:", type(days_set))
```

OUTPUT

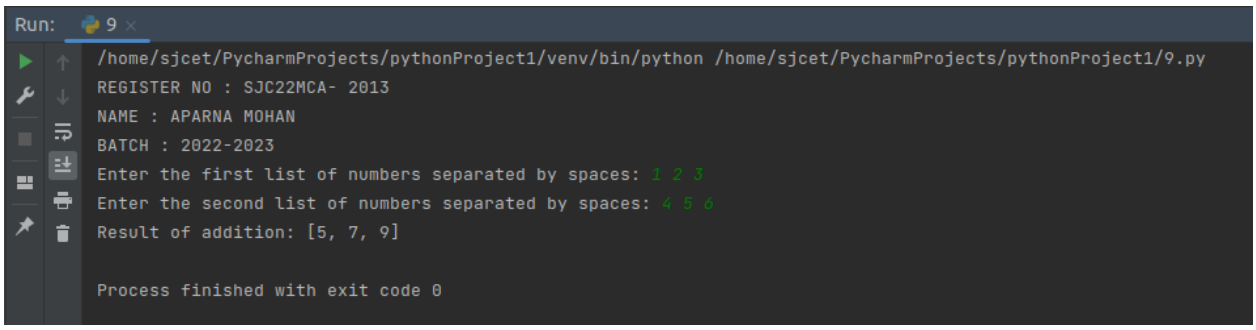


9. Write a program to add elements of given 2 lists.

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

def add_lists(list1, list2):

    if len(list1) != len(list2):
        return None

    result = []

    for i in range(len(list1)):
        result.append(list1[i] + list2[i])

    return result

try:
    list1 = input("Enter the first list of numbers separated by spaces: ").split()
    list1 = [int(x) for x in list1]

    list2 = input("Enter the second list of numbers separated by spaces: ").split()
    list2 = [int(x) for x in list2]

    result = add_lists(list1, list2)

    if result is None:
        print("The lists have different lengths and cannot be added.")
    else:
        print("Result of addition:", result)
except ValueError:
    print("Invalid input. Please enter valid numbers separated by spaces.")
```

OUTPUT



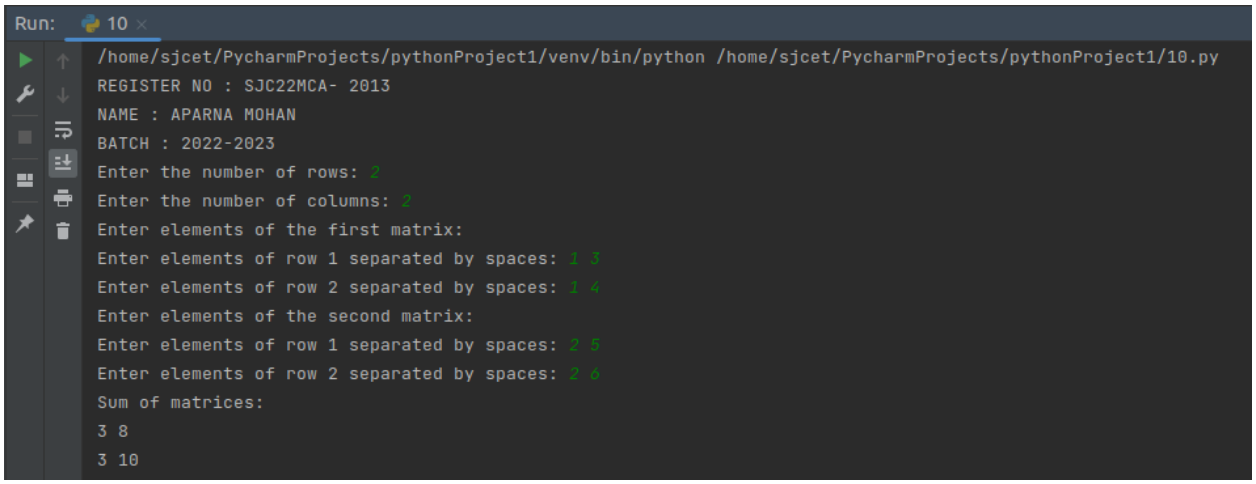
10. Write a program to find the sum of 2 matrices using nested List.

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

def add_matrices(matrix1, matrix2):

    if len(matrix1) != len(matrix2) or len(matrix1[0]) != len(matrix2[0]):
        return None


    result = [[0 for _ in range(len(matrix1[0]))] for _ in range(len(matrix1))]

    for i in range(len(matrix1)):
        for j in range(len(matrix1[0])):
            result[i][j] = matrix1[i][j] + matrix2[i][j]

    return result

try:

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

    print("Enter elements of the first matrix:")
    matrix1 = []
    for i in range(rows):
        row = input(f"Enter elements of row {i + 1} separated by spaces: ").split()
        matrix1.append([int(x) for x in row])

    print("Enter elements of the second matrix:")
    matrix2 = []
    for i in range(rows):
        row = input(f"Enter elements of row {i + 1} separated by spaces: ").split()
        matrix2.append([int(x) for x in row])


    result = add_matrices(matrix1, matrix2)

    if result is None:
        print("Matrix dimensions are not compatible for addition.")
    else:
        print("Sum of matrices:")
```

```
        for row in result:
            print(" ".join(map(str, row)))
except ValueError:
    print("Invalid input. Please enter valid numbers.")
```

OUTPUT

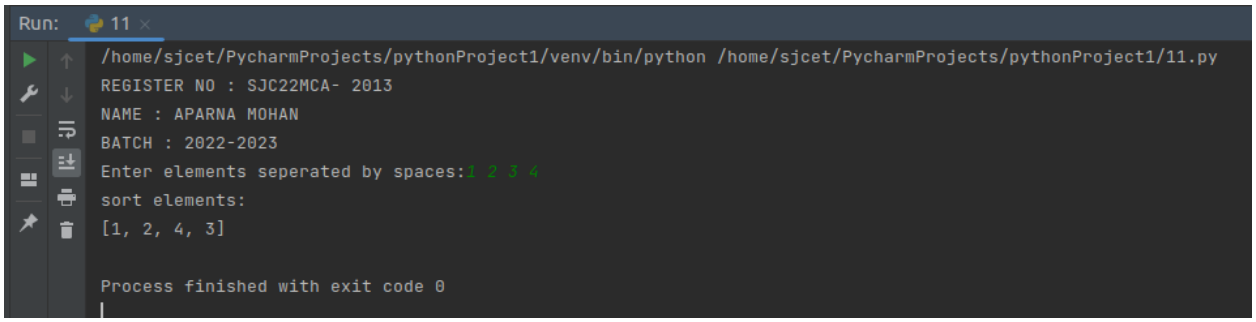


11. Write a program to perform bubble sort on a given set of elements.

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

def bubble_sort(arr):
    n = len(arr)

     for i in range(n):

        swapped = False

        for j in range(0, n - i - 1):

            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
                swapped = True

        if not swapped:
            break
try:

    elements = input("Enter elements separated by spaces: ").split()
    elements = [int(x) for x in elements]

    bubble_sort(elements)
    print("Sorted elements:")
    print(elements)
except ValueError:
    print("Invalid input. Please enter valid numbers separated by spaces.")
```

OUTPUT

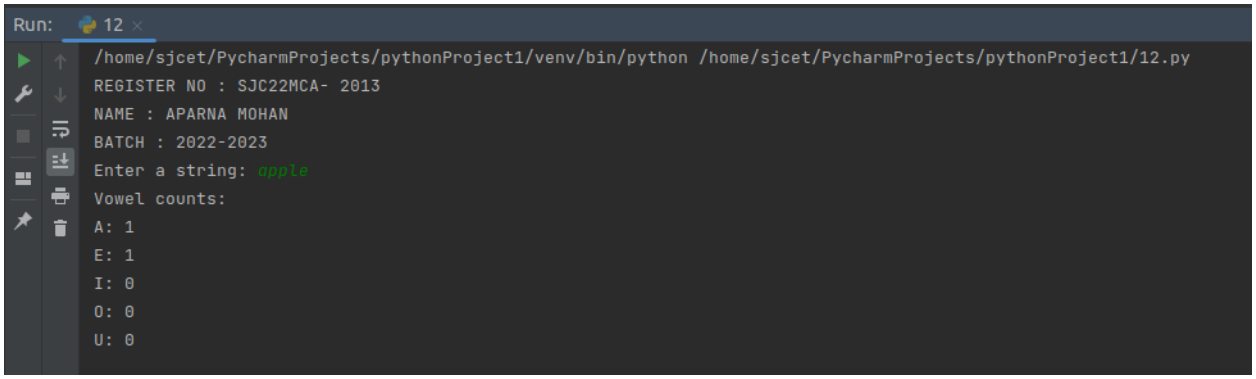


12. Program to find the count of each vowel in a string(use dictionary).

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

def count_vowels(string):
    vowel_counts = {'A': 0, 'E': 0, 'I': 0, 'O': 0, 'U': 0}
    string = string.upper()
    for char in string:

        if char in vowel_counts:

            vowel_counts[char] += 1

    return vowel_counts

try:

    input_string = input("Enter a string: ")

    vowel_counts = count_vowels(input_string)

    print("Vowel counts:")
    for vowel, count in vowel_counts.items():
        print(f"{vowel}: {count}")
except ValueError:
    print("Invalid input. Please enter a valid string.")
```

OUTPUT

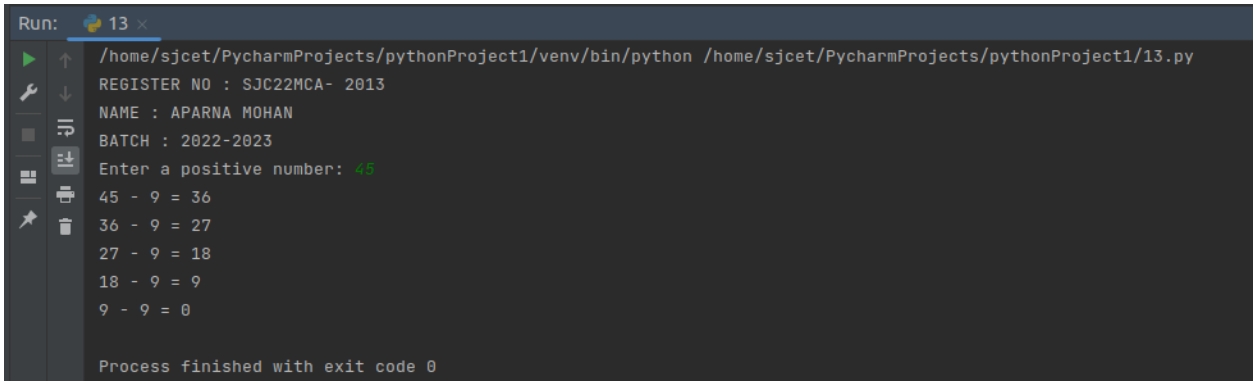


13. Write a Python program that accepts a positive number and subtract from this number the sum of its digits and so on. Continues this operation until the number is positive.
(eg: 256->2+5+6=13,256-13=243,243-9=232).

CODE

```
def sum_of_digits(n):

    digit_sum = 0


    while n > 0:
        digit_sum += n % 10
        n //= 10
    return digit_sum
try:
  print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

    num = int(input("Enter a positive number: "))
    if num <= 0:
        print("Please enter a positive number.")
    else:
        while num > 0:
            digit_sum = sum_of_digits(num)
            print(f"{num} - {digit_sum} = {num - digit_sum}")
            num -= digit_sum
except ValueError:
    print("Invalid input. Please enter a valid positive number.")
```

OUTPUT

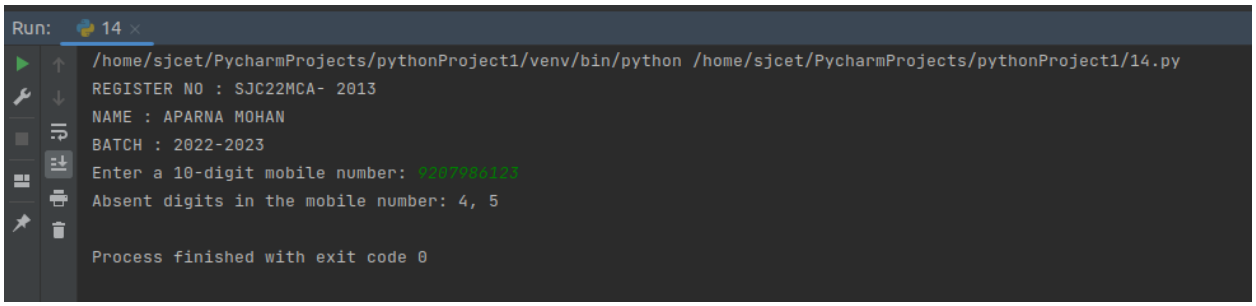


14. Write a Python program that accepts a 10 digit mobile number, and find the digits which are absent in a given mobile number.

CODE

```
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023 ")

def find_absent_digits(mobile_number):

    all_digits = set("0123456789")


    mobile_digits = set(mobile_number)


    absent_digits = all_digits - mobile_digits
    return sorted(list(absent_digits))
try:
    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 in the mobile number:", ', '.join(absent_digits))
        else:
            print("The mobile number contains all digits from 0 to 9.")
    else:
        print("Invalid input. Please enter a valid 10-digit mobile number.")
except ValueError:
    print("Invalid input. Please enter a valid 10-digit mobile number.")
```

OUTPUT



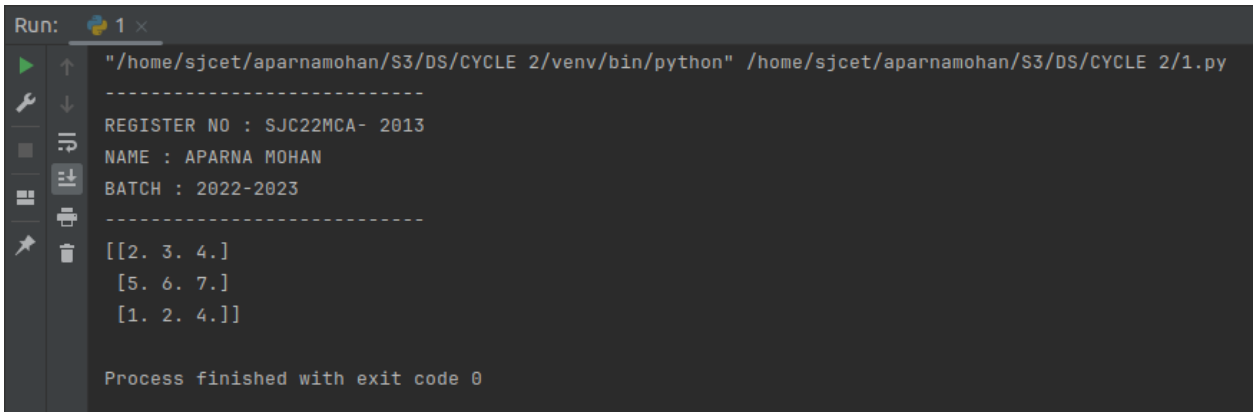
CYCLE 2

1.Create a three dimensional array specifying float data type and print it.

CODE

```
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")

import numpy as np
a= np.array([[2,3,4],[5,6,7],[1,2,4]]).astype('f')
print(a)
```

OUTPUT



2. Create a 2 dimensional array (2X3) with elements belonging to complex data type and print it. Also display

a. the no: of rows and columns

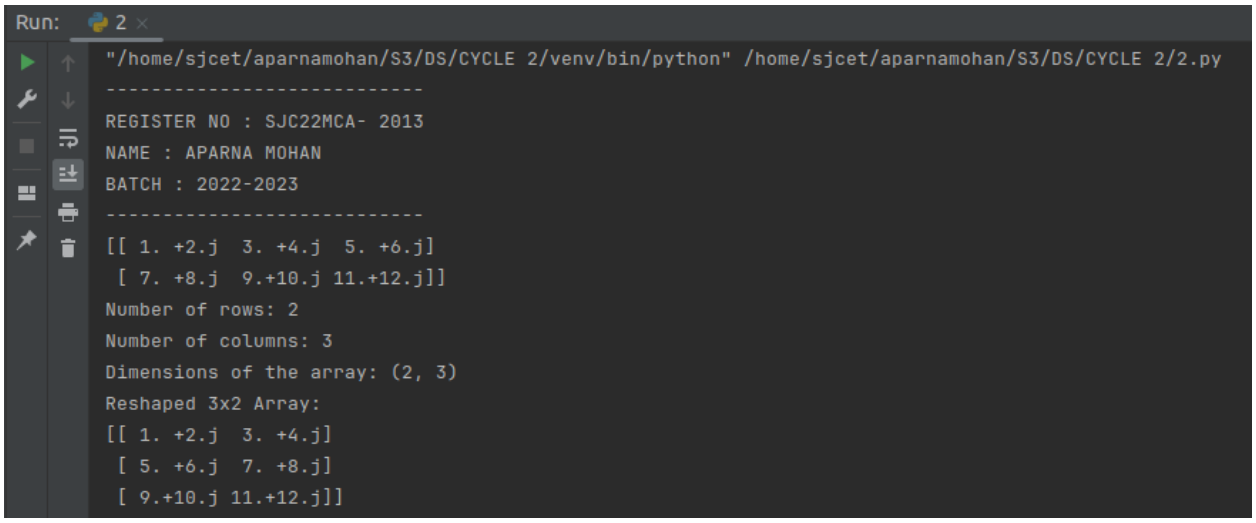
b. dimension of an array

c. reshape the same array to 3X2

CODE

```
import nprint("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")

import numpy as np

# Create a 2x3 array with complex data type elements
complex_array = np.array([[1 + 2j, 3 + 4j, 5 + 6j],
                          [7 + 8j, 9 + 10j, 11 + 12j]])

# Print the complex array
print(complex_array)

num_rows, num_columns = complex_array.shape
print(f"Number of rows: {num_rows}")
print(f"Number of columns: {num_columns}")

dimensions = complex_array.shape
print(f"Dimensions of the array: {dimensions}")

reshaped_array = complex_array.reshape(3, 2)
print("Reshaped 3x2 Array:")
print(reshaped_array)umpy as np
```

OUTPUT


Run:
2
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/2.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

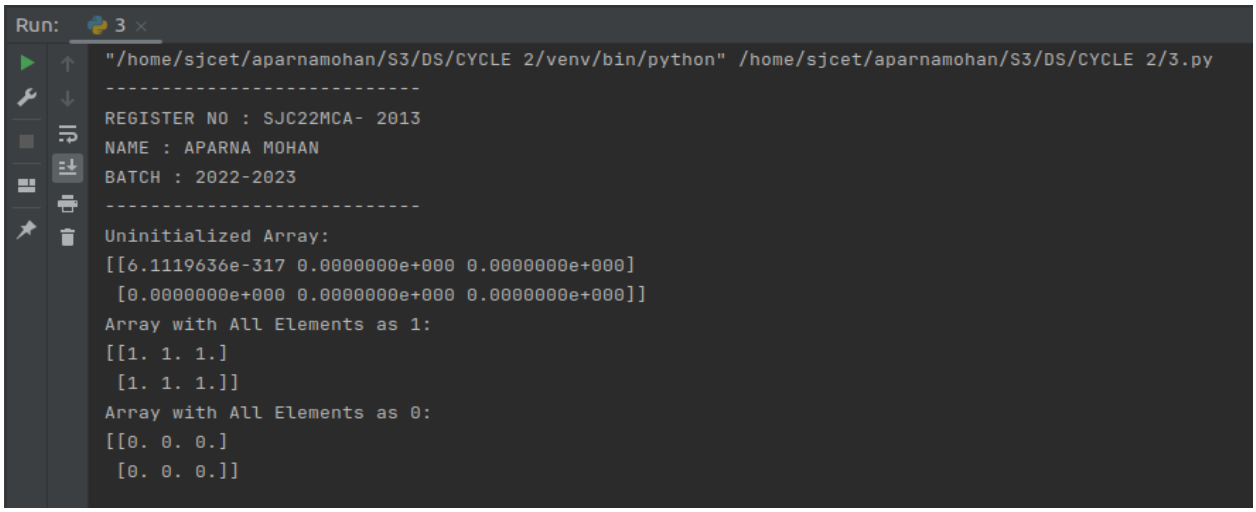
[[1. +2.j 3. +4.j 5. +6.j]
[7. +8.j 9.+10.j 11.+12.j]]
Number of rows: 2
Number of columns: 3
Dimensions of the array: (2, 3)
Reshaped 3x2 Array:
[[1. +2.j 3. +4.j]
[5. +6.j 7. +8.j]
[9.+10.j 11.+12.j]]

3. Familiarize with the functions to create
a) an uninitialized array
b) array with all elements as 1,
c) all elements as 0

CODE

```
import numpy as np

print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")

# Create an uninitialized array
uninitialized_array = np.empty(shape=(2, 3))
print("Uninitialized Array:")
print(uninitialized_array)


# Create an array with all elements as 1
ones_array = np.ones(shape=(2, 3))
print("Array with All Elements as 1:")
print(ones_array)



# Create an array with all elements as 0
zeros_array = np.zeros(shape=(2, 3))
print("Array with All Elements as 0:")
print(zeros_array)
```

OUTPUT


Run:
3
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/3.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

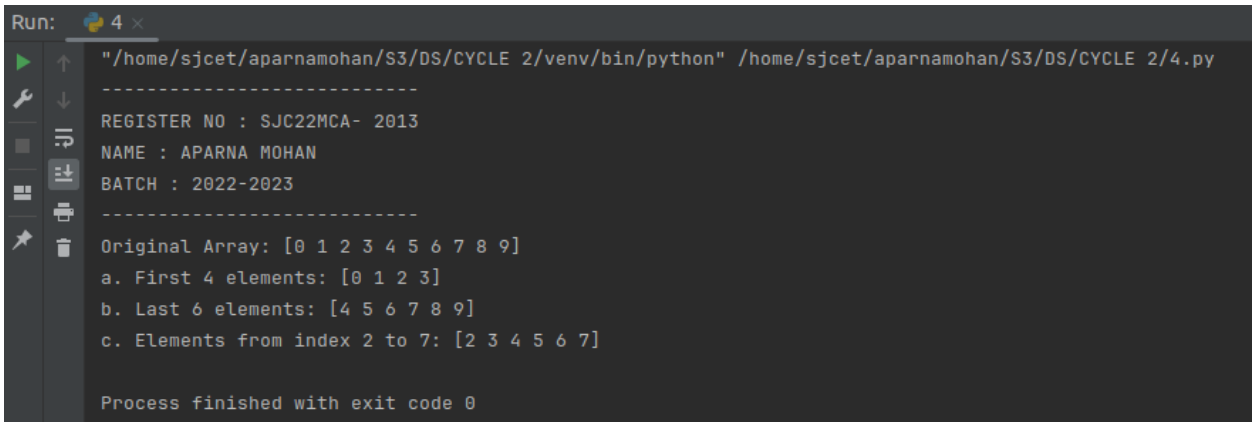
Uninitialized Array:
[[6.1119636e-317 0.0000000e+000 0.0000000e+000]
[0.0000000e+000 0.0000000e+000 0.0000000e+000]]
Array with All Elements as 1:
[[1. 1. 1.]
[1. 1. 1.]]
Array with All Elements as 0:
[[0. 0. 0.]
[0. 0. 0.]]

4. Create an one dimensional array using arange function containing 10 elements.
Display
a. First 4 elements
b. Last 6 elements
c. Elements from index 2 to 7import numpy as np

CODE

```
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")

# Create a one-dimensional array with 10 elements using arange
one_dimensional_array = np.arange(10)

# a. Display the first 4 elements
first_4_elements = one_dimensional_array[:4]

# b. Display the last 6 elements
last_6_elements = one_dimensional_array[-6:]

# c. Display elements from index 2 to 7
elements_2_to_7 = one_dimensional_array[2:8]

# Display the results
print("Original Array:", one_dimensional_array)
print("a. First 4 elements:", first_4_elements)
print("b. Last 6 elements:", last_6_elements)
print("c. Elements from index 2 to 7:", elements_2_to_7)
```

OUTPUT


Run:
4
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/4.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

Original Array: [0 1 2 3 4 5 6 7 8 9]
a. First 4 elements: [0 1 2 3]
b. Last 6 elements: [4 5 6 7 8 9]
c. Elements from index 2 to 7: [2 3 4 5 6 7]
Process finished with exit code 0

5. Create an 1D array with arange containing first 15 even numbers as elements

a. Elements from index 2 to 8 with step 2(also demonstrate the same using slice function)

b. Last 3 elements of the array using negative index

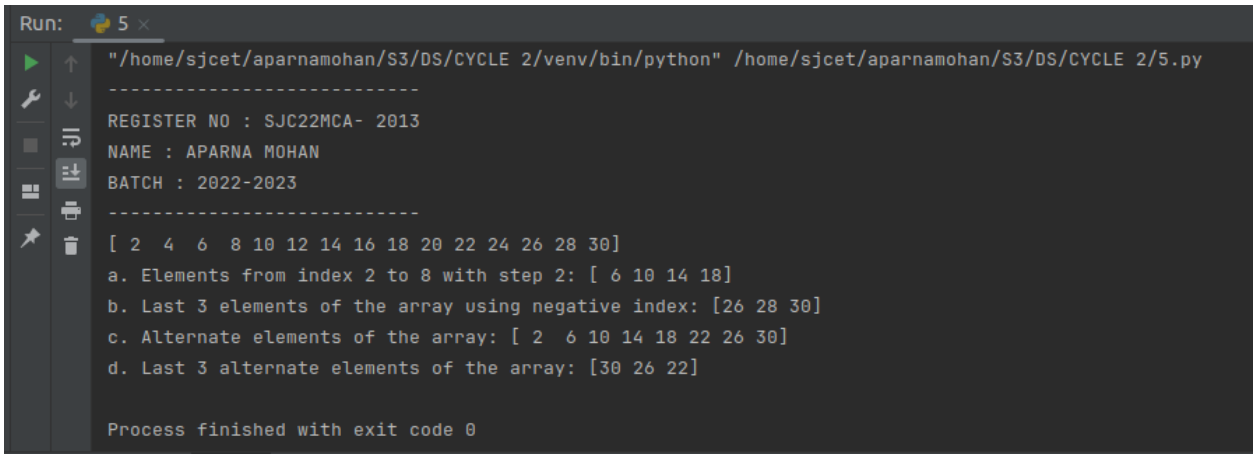
c. Alternate elements of the array

d. Display the last 3 alternate elements

CODE

```
import numpy as np

print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")
# Create a one-dimensional array with the first 15 even numbers
even_numbers = np.arange(2, 32, 2)
print(even_numbers)

# Display elements from index 2 to 8 with a step of 2
elements_from_2_to_8_step_2 = even_numbers[2:9:2]

print("a. Elements from index 2 to 8 with step 2:", elements_from_2_to_8_step_2)
# Display the last 3 elements using negative index
last_3_elements = even_numbers[-3:]

print("b. Last 3 elements of the array using negative index:", last_3_elements)
# Display alternate elements of the array
alternate_elements = even_numbers[::2]

print("c. Alternate elements of the array:", alternate_elements)
# Display the last 3 alternate elements

last_3_alternate_elements = even_numbers[-1::-2][:3]

print("d. Last 3 alternate elements of the array:", last_3_alternate_elements)
```

OUTPUT


Run:
5
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/5.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

[2 4 6 8 10 12 14 16 18 20 22 24 26 28 30]
a. Elements from index 2 to 8 with step 2: [6 10 14 18]
b. Last 3 elements of the array using negative index: [26 28 30]
c. Alternate elements of the array: [2 6 10 14 18 22 26 30]
d. Last 3 alternate elements of the array: [30 26 22]
Process finished with exit code 0

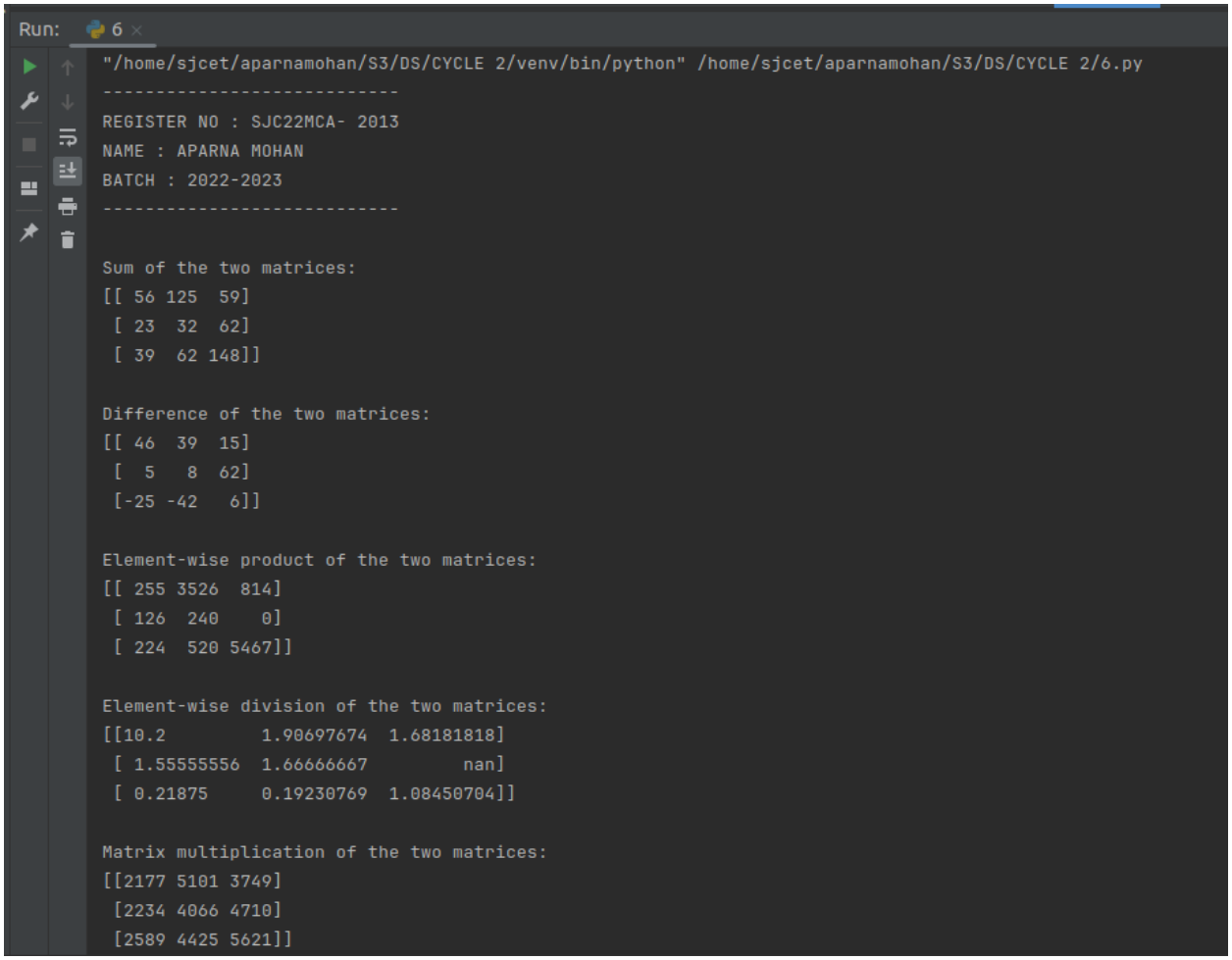
6. Create a 2 Dimensional array with 4 rows and 4 columns.
a. Display all elements excluding the first row
b. Display all elements excluding the last column
c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row
d. Display the elements of 2 nd and 3 rd column
e. Display 2 nd and 3 rd element of 1 st row
f. Display the elements from indices 4 to 10 in descending order(use –values)

CODE

```
import numpy as np
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")
print()
matrix1 = np.array([[51, 82, 37], [14, 20, 62], [7, 10, 77]])
matrix2 = np.array([[5, 43, 22], [9, 12, 0], [32, 52, 71]])

matrix_sum = matrix1 + matrix2
print("Sum of the two matrices:")
print(matrix_sum)

matrix_diff = matrix1 - matrix2
print("\nDifference of the two matrices:")
print(matrix_diff)

matrix_product = matrix1 * matrix2
print("\nElement-wise product of the two matrices:")
print(matrix_product)

with np.errstate(divide='ignore', invalid='ignore'):
    matrix_division = np.true_divide(matrix1, matrix2)
    matrix_division[~np.isfinite(matrix_division)] = np.nan
print("\nElement-wise division of the two matrices:")
print(matrix_division)

matrix_mult = np.dot(matrix1, matrix2)
print("\nMatrix multiplication of the two matrices:")
print(matrix_mult)
```

```
matrix1_transpose = np.transpose(matrix1)
print("\nTranspose of matrix1:")
print(matrix1_transpose)

diagonal_sum = np.trace(matrix1)
print("\nSum of diagonal elements of matrix1:")
print(diagonal_sum)
```

OUTPUT



```
Transpose of matrix1:
[[51 14  7]
 [82 20 10]
 [37 62 77]]

Sum of diagonal elements of matrix1:
148

Process finished with exit code 0
```

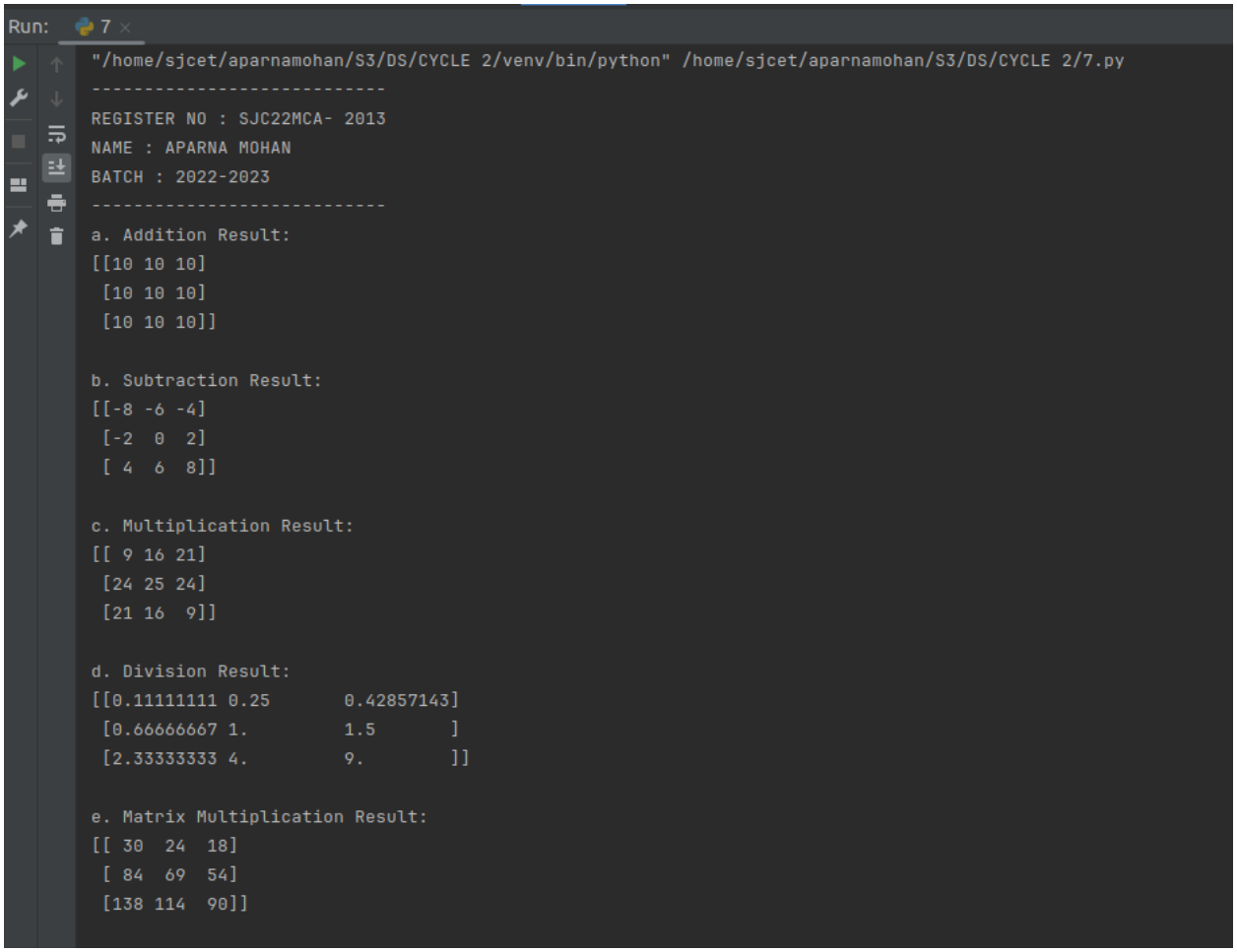
7. Create two 2D arrays using array object and
a. Add the 2 matrices and print it
b. Subtract 2 matrices
c. Multiply the individual elements of matrix
d. Divide the elements of the matrices
e. Perform matrix multiplication
f. Display transpose of the matrix
g. Sum of diagonal elements of a matrix

CODE

```
import numpy as np
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")


matrix1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix2 = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])

addition_result = matrix1 + matrix2
print("a. Addition Result:")
print(addition_result)

subtraction_result = matrix1 - matrix2
print("\nb. Subtraction Result:")
print(subtraction_result)

multiplication_result = matrix1 * matrix2
print("\nc. Multiplication Result:")
print(multiplication_result)

epsilon = 1e-15
division_result = np.divide(matrix1, matrix2 + epsilon)
print("\nd. Division Result:")
print(division_result)

matrix_multiplication_result = np.dot(matrix1, matrix2)
print("\ne. Matrix Multiplication Result:")
print(matrix_multiplication_result)
```

```
transpose_result = np.transpose(matrix1)
print("\nf. Transpose of the Matrix:")
print(transpose_result)

diagonal_sum = np.trace(matrix1)
print("\ng. Sum of Diagonal Elements:")
print(diagonal_sum)
```

OUTPUT



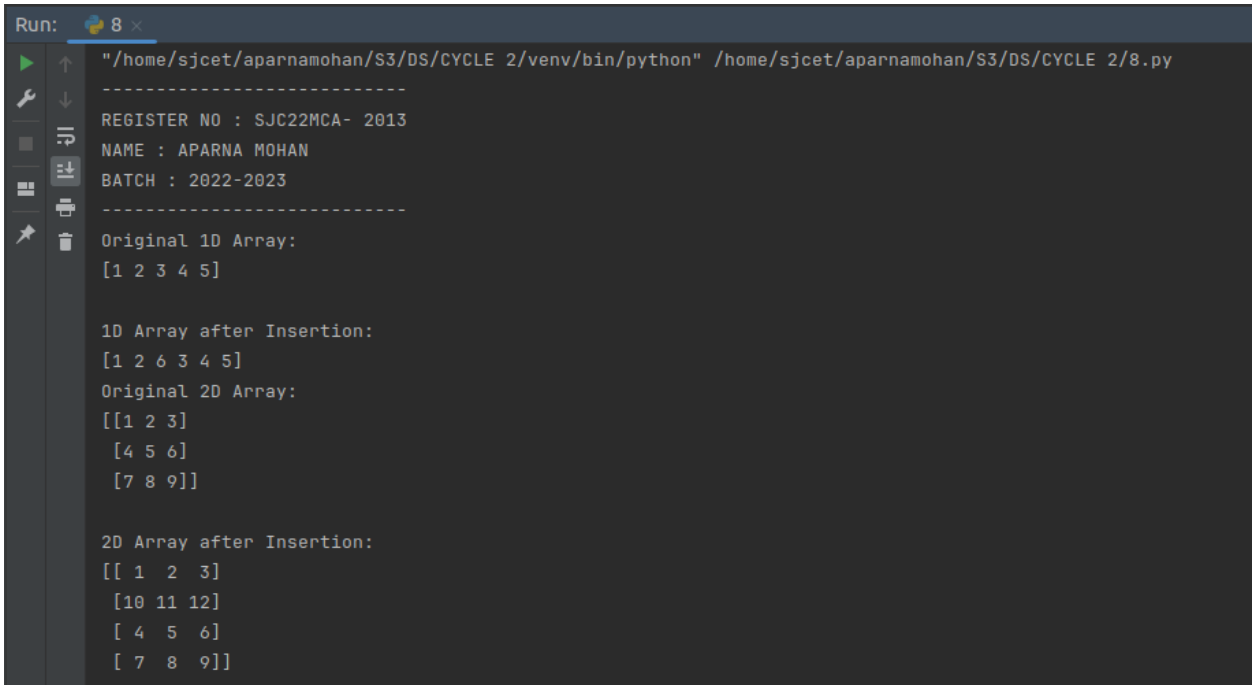
```
f. Transpose of the Matrix:
[[1 4 7]
 [2 5 8]
 [3 6 9]]

g. Sum of Diagonal Elements:
15

Process finished with exit code 0
```

8. Demonstrate the use of insert() function in 1D and 2D array

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")


arr1d = np.array([1, 2, 3, 4, 5])
inserted_arr = np.insert(arr1d, 2, 6)

print("Original 1D Array:")
print(arr1d)

print("\n1D Array after Insertion:")
print(inserted_arr)

import numpy as np


arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

inserted_arr = np.insert(arr2d, 1, [10, 11, 12], axis=0)

print("Original 2D Array:")
print(arr2d)

print("\n2D Array after Insertion:")
print(inserted_arr)
```

OUTPUT


Run:
8
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/8.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

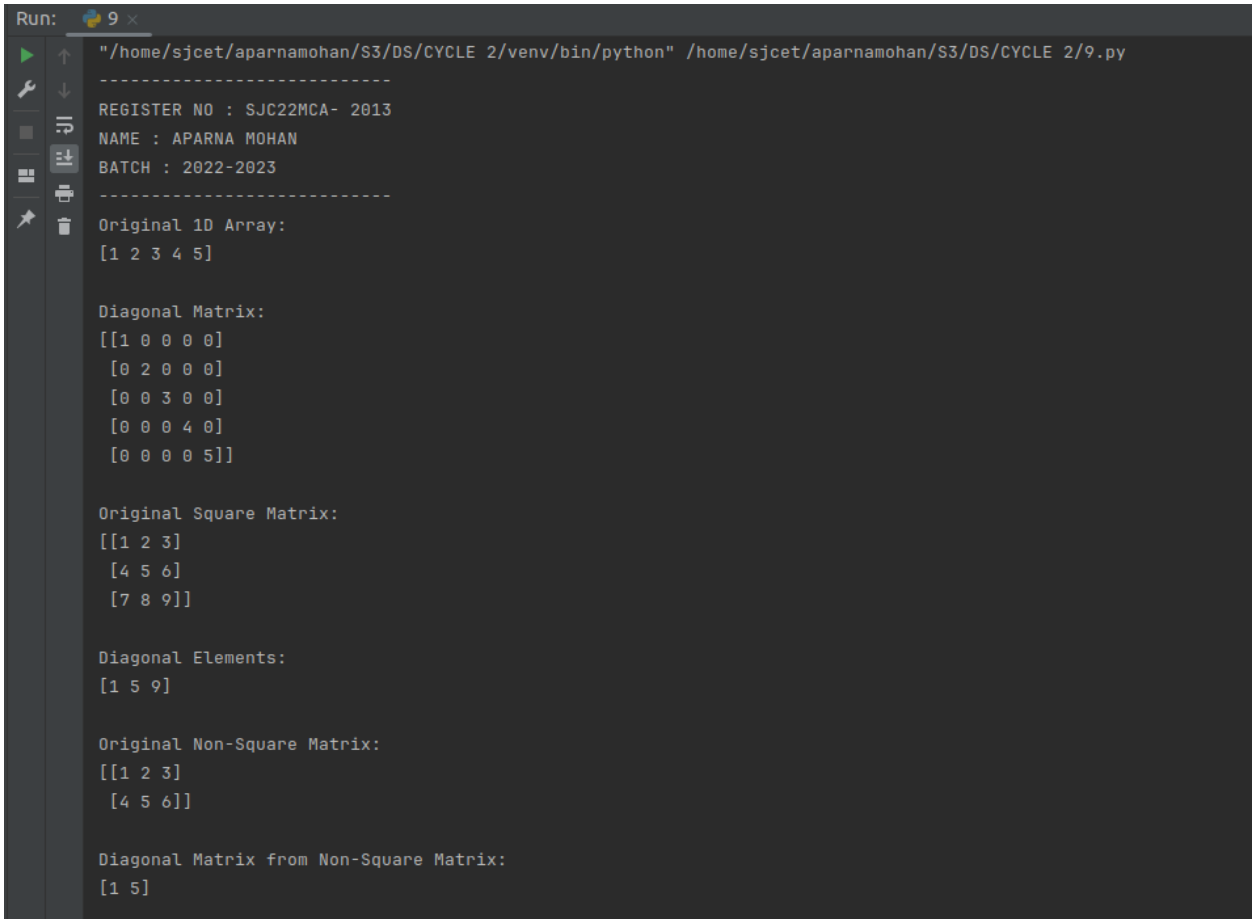
Original 1D Array:
[1 2 3 4 5]
1D Array after Insertion:
[1 2 6 3 4 5]
Original 2D Array:
[[1 2 3]
[4 5 6]
[7 8 9]]
2D Array after Insertion:
[[1 2 3]
[10 11 12]
[4 5 6]
[7 8 9]]

9. Demonstrate the use of diag() function in 1D and 2D array.(use both square matrix and matrix with different dimensions)

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")

A = np.array([1, 2, 3, 4, 5])


D = np.diag(A)


print("Original 1D Array:")
print(A)

print("\nDiagonal Matrix:")
print(D)

B = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

D_square = np.diag(B)

print("\nOriginal Square Matrix:")
print(B)

print("\nDiagonal Elements:")
print(D_square)

C = np.array([[1, 2, 3],
              [4, 5, 6]])

D_nonsquare = np.diag(C)


print("\nOriginal Non-Square Matrix:")
print(C)

print("\nDiagonal Matrix from Non-Square Matrix:")
print(D_nonsquare)
```

OUTPUT


Run: 9
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/9.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

Original 1D Array:
[1 2 3 4 5]
Diagonal Matrix:
[[1 0 0 0 0]
[0 2 0 0 0]
[0 0 3 0 0]
[0 0 0 4 0]
[0 0 0 0 5]]
Original Square Matrix:
[[1 2 3]
[4 5 6]
[7 8 9]]
Diagonal Elements:
[1 5 9]
Original Non-Square Matrix:
[[1 2 3]
[4 5 6]]
Diagonal Matrix from Non-Square Matrix:
[1 5]

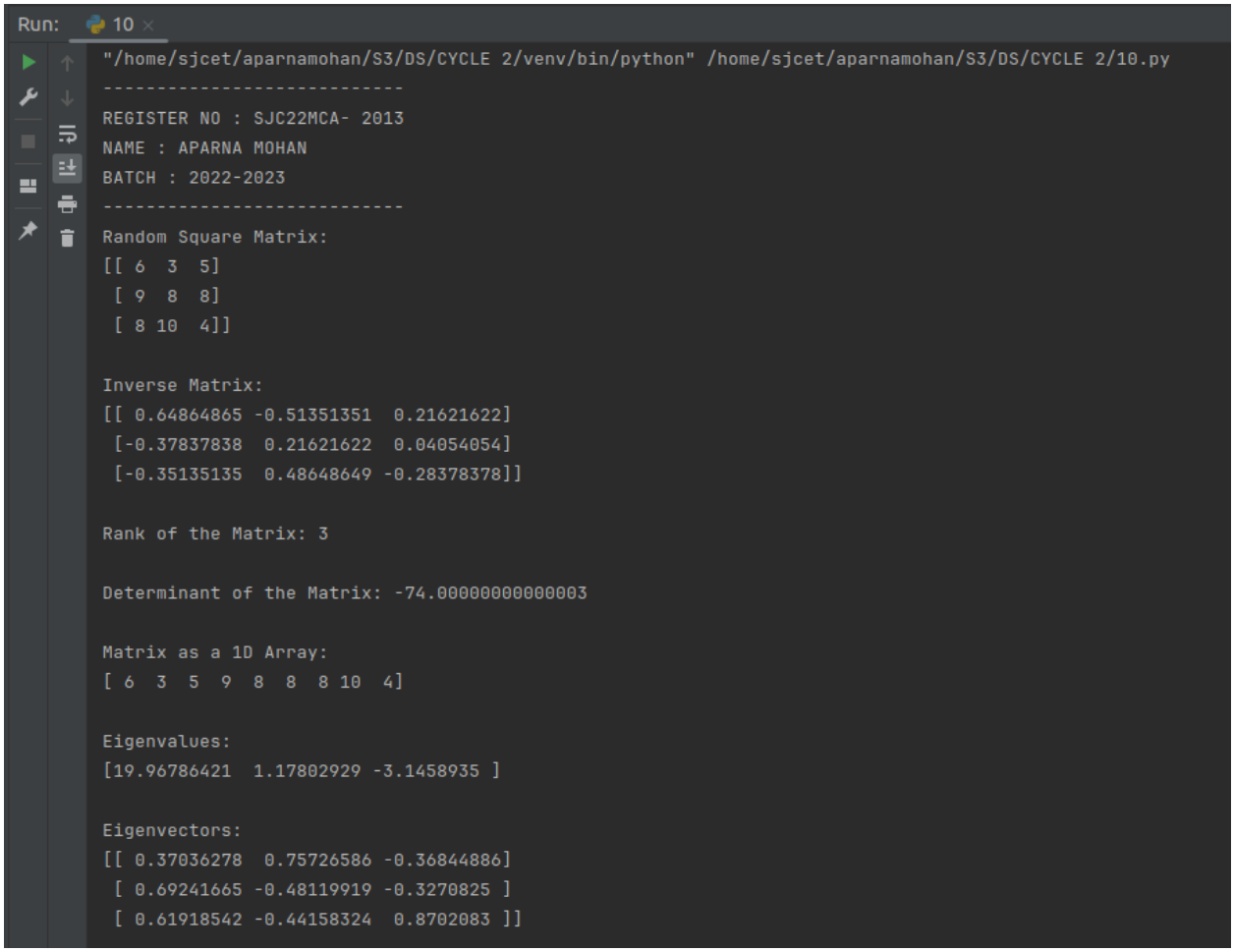
10. Create a square matrix with random integer values(use randint()) and use appropriate functions to find:
i) inverse
ii) rank of matrix
iii) Determinant
iv) transform matrix into 1D array
v) eigen values and vectors.

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")

matrix_size = 3

random_matrix = np.random.randint(1, 11, size=(matrix_size, matrix_size))

print("Random Square Matrix:")
print(random_matrix)

try:
    inverse_matrix = np.linalg.inv(random_matrix)
    print("\nInverse Matrix:")
    print(inverse_matrix)
except np.linalg.LinAlgError:
    print("\nInverse does not exist for this matrix.")

rank = np.linalg.matrix_rank(random_matrix)
print("\nRank of the Matrix:", rank)

determinant = np.linalg.det(random_matrix)
print("\nDeterminant of the Matrix:", determinant)

matrix_1d = random_matrix.flatten()
print("\nMatrix as a 1D Array:")
print(matrix_1d)

eigenvalues, eigenvectors = np.linalg.eig(random_matrix)
```

```
print("\nEigenvalues:")
print(eigenvalues)
print("\nEigenvectors:")
print(eigenvectors)
```

OUTPUT



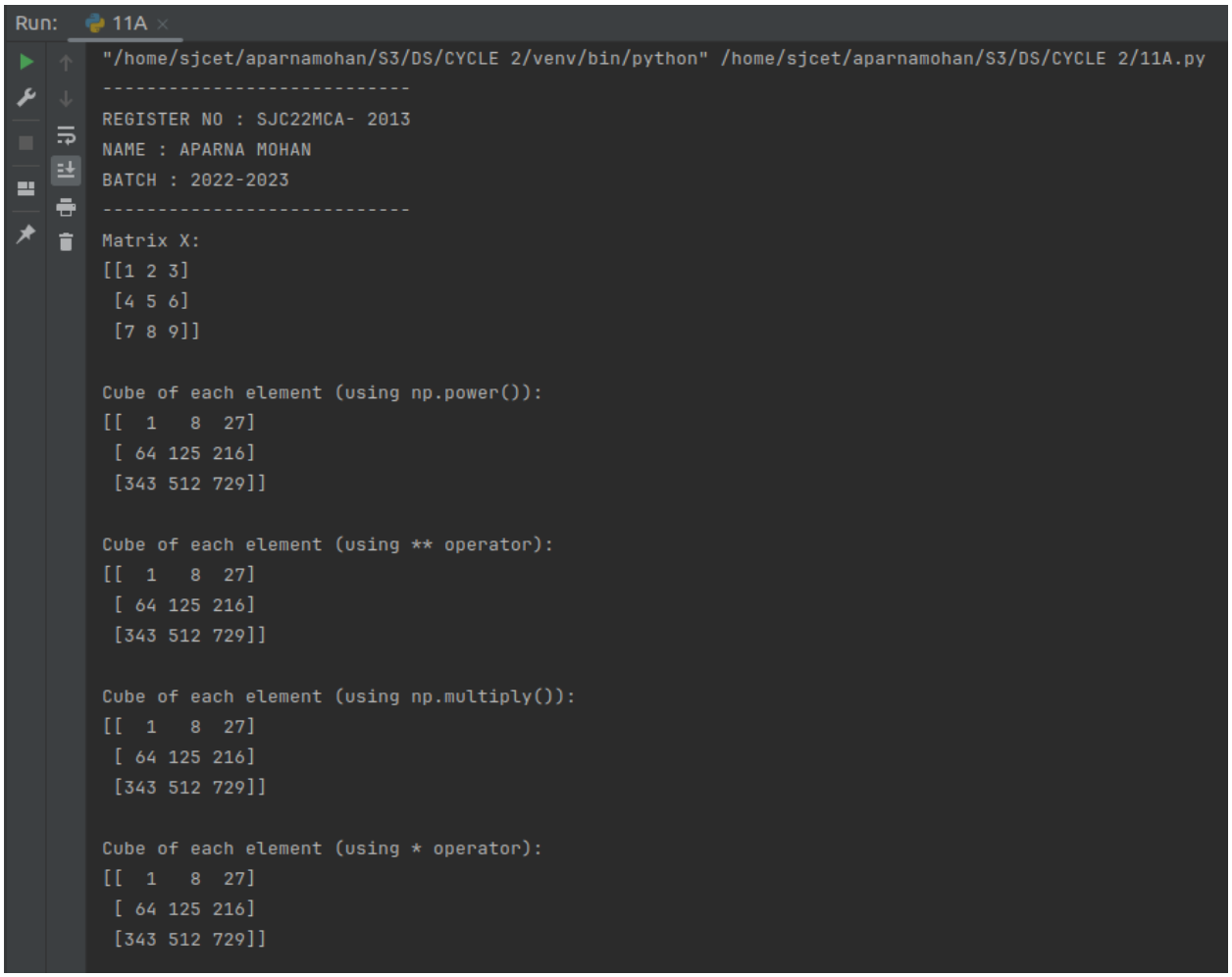
11.. Create a matrix X with suitable rows and columns
i) Display the cube of each element of the matrix using different methods(use multiply(), *, power(),**)
ii) Display identity matrix of the given square matrix.
iii) Display each element of the matrix to different powers.

CODE

```
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")
import numpy as np

X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# i) Display the cube of each element of the matrix using different methods

# Using np.power() to calculate the cube
cubed_matrix1 = np.power(X, 3)

# Using the ** operator to calculate the cube
cubed_matrix2 = X ** 3

# Using np.multiply() to calculate the cube
cubed_matrix3 = np.multiply(X, np.multiply(X, X))

# Using the * operator to calculate the cube
cubed_matrix4 = X * X * X

print("Matrix X:")
print(X)

print("\nCube of each element (using np.power()):")
print(cubed_matrix1)

print("\nCube of each element (using ** operator):")
print(cubed_matrix2)

print("\nCube of each element (using np.multiply()):")
print(cubed_matrix3)

print("\nCube of each element (using * operator):")
```

```
print(cubed_matrix4)

# ii) Display the identity matrix of the given square matrix
identity_matrix = np.identity(X.shape[0])
print("\nIdentity Matrix of X:")
print(identity_matrix)

# iii) Display each element of the matrix to different powers
exponentials = [2, 3, 4]

powered_matrices = [np.power(X, exp) for exp in exponentials]

for i, exp in enumerate(exponentials):
    print(f"\nMatrix X to the power of {exp}:")
    print(powered_matrices[i])
```

OUTPUT



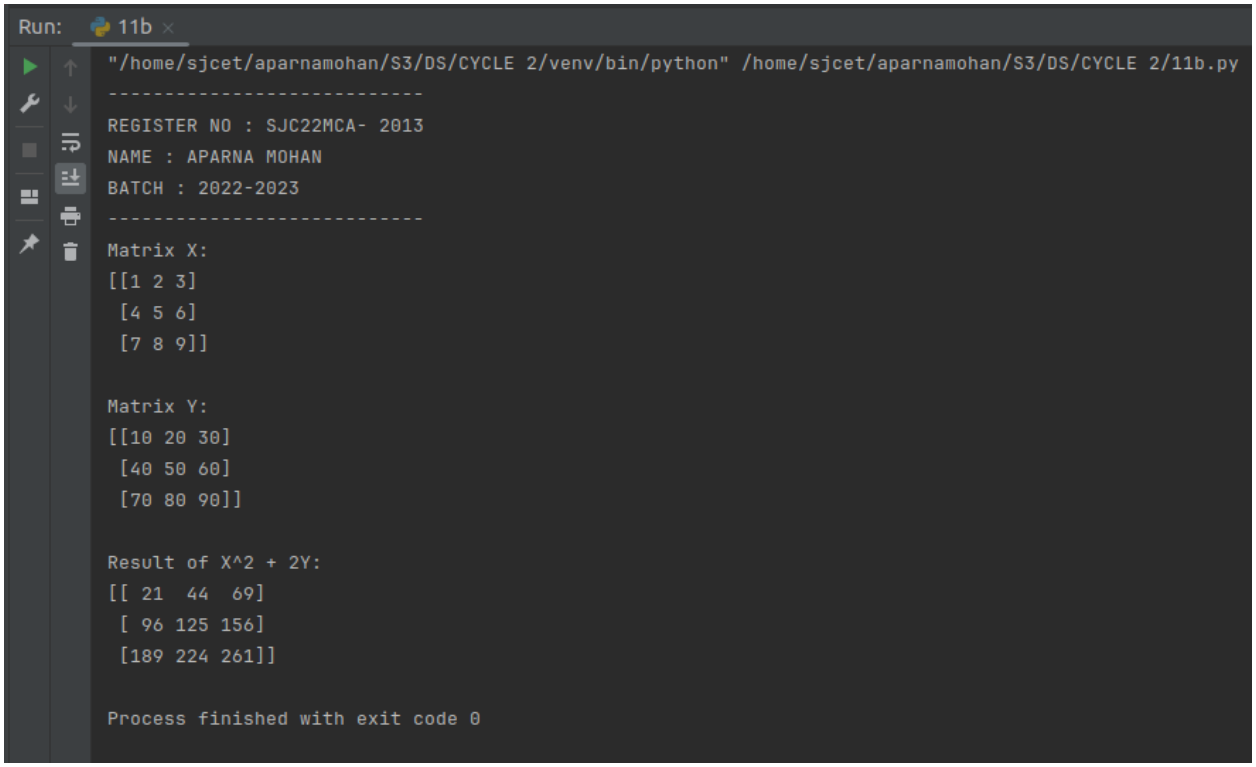
```
Cube of each element (using * operator):
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Identity Matrix of X:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Matrix X to the power of 2:
[[ 1  4  9]
 [16 25 36]
 [49 64 81]]

Matrix X to the power of 3:
[[  1   8  27]
 [ 64 125 216]
 [343 512 729]]

Matrix X to the power of 4:
[[   1   16   81]
 [ 256  625 1296]
 [2401 4096 6561]]

Process finished with exit code 0
```

11. Create a matrix Y with same dimension as X and perform the operation X 2 +2Y

CODE

```
import numpy as np
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")
X = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

Y = np.array([[10, 20, 30],
              [40, 50, 60],
              [70, 80, 90]])

result = np.power(X, 2) + 2 * Y

print("Matrix X:")
print(X)

print("\nMatrix Y:")
print(Y)

print("\nResult of X^2 + 2Y:")
print(result)
```

OUTPUT


Run:
11b
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/11b.py
---
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023

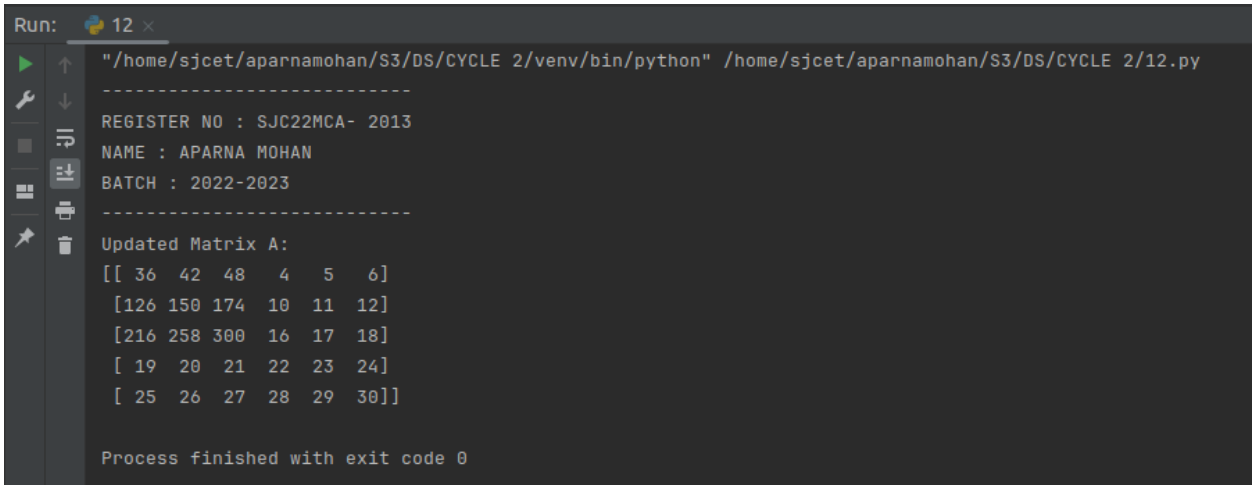
Matrix X:
[[1 2 3]
[4 5 6]
[7 8 9]]
Matrix Y:
[[10 20 30]
[40 50 60]
[70 80 90]]
Result of X^2 + 2Y:
[[21 44 69]
[96 125 156]
[189 224 261]]
Process finished with exit code 0

12. Define matrices A with dimension 5x6 and B with dimension 3x3. Extract a sub matrix of dimension 3x3 from A and multiply it with B. Replace the extracted sub matrix in A with the matrix obtained after multiplication

CODE

```
import numpy as np
print("-----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("-----------------------------")
A = np.array([[1, 2, 3, 4, 5, 6],
              [7, 8, 9, 10, 11, 12],
              [13, 14, 15, 16, 17, 18],
              [19, 20, 21, 22, 23, 24],
              [25, 26, 27, 28, 29, 30]])

B = np.array([[2, 3, 4],
              [5, 6, 7],
              [8, 9, 10]])

submatrix_A = A[:3, :3]

result = np.dot(submatrix_A, B)

A[:3, :3] = result

# Display the updated matrix A
print("Updated Matrix A:")
print(A)
```

OUTPUT

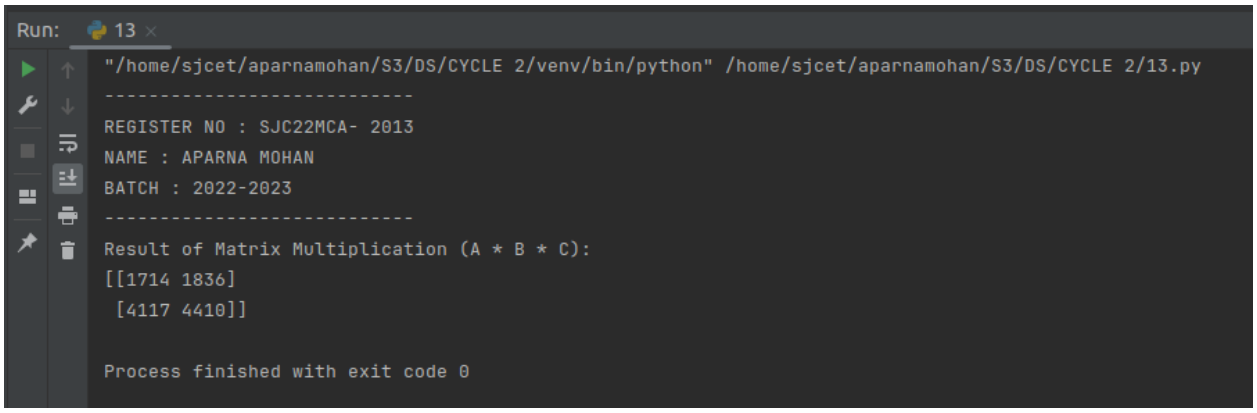

Run:
12
"/home/sjcet/aparnamohan/S3/DS/CYCLE 2/venv/bin/python" /home/sjcet/aparnamohan/S3/DS/CYCLE 2/12.py
REGISTER NO : SJC22MCA- 2013
NAME : APARNA MOHAN
BATCH : 2022-2023
Updated Matrix A:
[[36 42 48 4 5 6]
[126 150 174 10 11 12]
[216 258 300 16 17 18]
[19 20 21 22 23 24]
[25 26 27 28 29 30]]
Process finished with exit code 0

13. Given 3 Matrices A, B and C. Write a program to perform matrix multiplication of the 3 matrices.

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")
A = np.array([[1, 2, 3],
              [4, 5, 6]])

B = np.array([[7, 8],
              [9, 10],
              [11, 12]])

C = np.array([[13, 14],
              [15, 16]])

result = np.dot(np.dot(A, B), C)

print("Result of Matrix Multiplication (A * B * C):")
print(result)
```

OUTPUT



14. Write a program to check whether given matrix is symmetric or Skew Symmetric.

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")
def is_symmetric(matrix):
    transpose = np.transpose(matrix)
    return np.array_equal(matrix, transpose)

def is_skew_symmetric(matrix):
    transpose = np.transpose(matrix)
    return np.array_equal(matrix, -transpose)

matrix = np.array([[0, 1, -2],
                   [-1, 0, 3],
                   [2, -3, 0]])

if is_symmetric(matrix):
    print("The matrix is symmetric.")
elif is_skew_symmetric(matrix):
    print("The matrix is skew-symmetric (antisymmetric).")
else:
    print("The matrix is neither symmetric nor skew-symmetric.")
```

OUTPUT

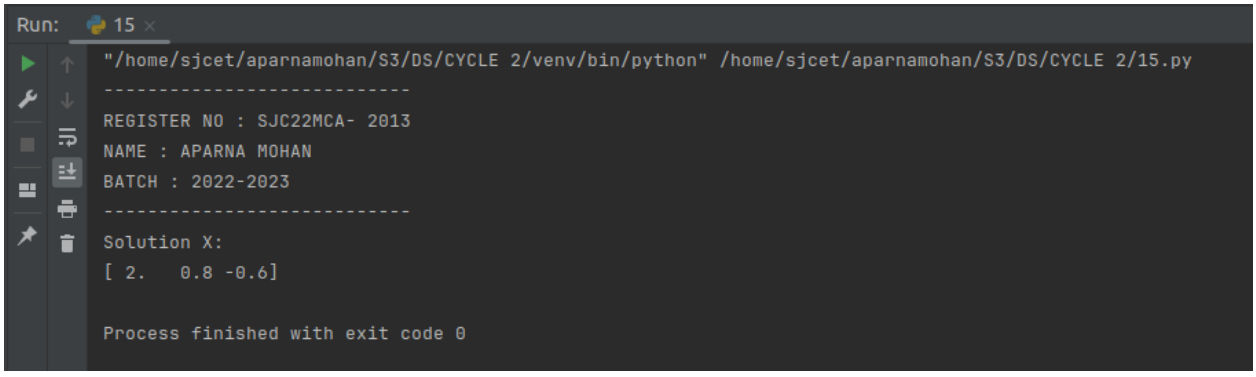
15. Given a matrix-vector equation AX=b. Write a program to find out the value of X using solve(), given A and b as below
X=A -1 b.

Note: Numpy provides a function called solve for solving such equations.

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")
A = np.array([[2, 3, -1],
              [1, 2, 1],
              [3, 1, -2]])

b = np.array([7, 3, 8])

try:

    X = np.linalg.solve(A, b)


    print("Solution X:")
    print(X)
except np.linalg.LinAlgError:
    print("Matrix A is singular. The system of equations may not have a unique solution.")
```

OUTPUT



16. Write a program to perform the SVD of a given matrix A. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.

CODE

```
import numpy as np
print("----------------------------")
print("REGISTER NO : SJC22MCA- 2013")
print("NAME : APARNA MOHAN")
print("BATCH : 2022-2023")
print("----------------------------")
A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

U, S, Vt = np.linalg.svd(A)

A_hat = U @ np.diag(S) @ Vt

print("Original Matrix A:")
print(A)
print("\nSingular Values:")
print(S)
print("\nReconstructed Matrix A_hat:")
print(A_hat)
```

OUTPUT