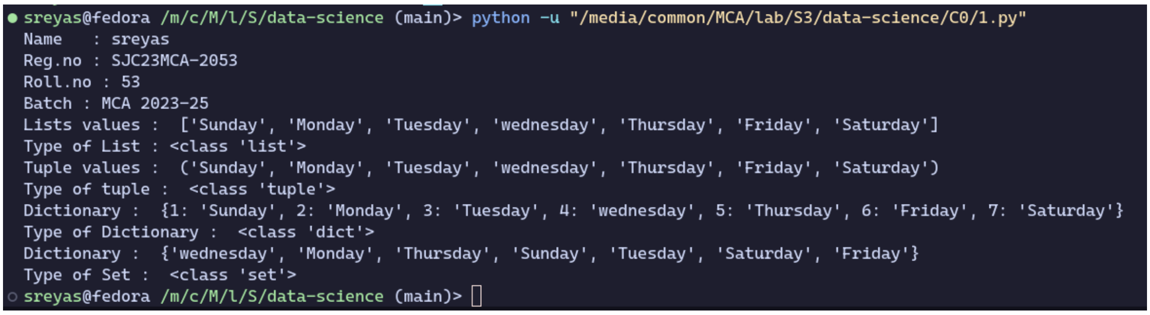
**1. Write a program to represent the days of a week using List, Tuple and Dictionary and display their types.**

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25")
day_list = ["Sunday","Monday","Tuesday","wednesday","Thursday","Friday","Saturday"]
print("Lists values : ",day_list)
print("Type of List :",type(day_list))
day_tuple =("Sunday","Monday","Tuesday","wednesday","Thursday","Friday","Saturday")
print("Tuple values : ",day_tuple)
print("Type of tuple : ", type(day_tuple))
day_dict ={1:"Sunday",2:"Monday",3:"Tuesday",4:"wednesday",5:"Thursday",6:"Friday",7:"Saturday" }
print("Dictionary : ",day_dict)
print("Type of Dictionary : ",type(day_dict))
day_set ={"Sunday","Monday","Tuesday","wednesday","Thursday","Friday","Saturday"}
print("Dictionary : ",day_set)
print("Type of Set : ",type(day_set))
```

**OUTPUT:**



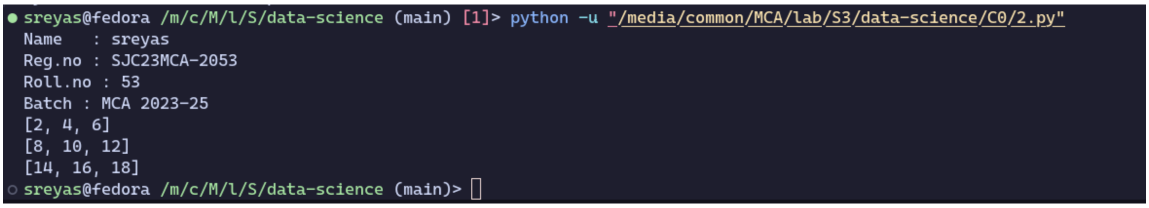
```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C0/1.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25
  Lists values :  ['Sunday', 'Monday', 'Tuesday', 'wednesday', 'Thursday', 'Friday', 'Saturday']
  Type of List : <class 'list'>
  Tuple values :  ('Sunday', 'Monday', 'Tuesday', 'wednesday', 'Thursday', 'Friday', 'Saturday')
  Type of tuple :  <class 'tuple'>
  Dictionary :  {1: 'Sunday', 2: 'Monday', 3: 'Tuesday', 4: 'wednesday', 5: 'Thursday', 6: 'Friday', 7: 'Saturday'}
  Type of Dictionary :  <class 'dict'>
  Dictionary :  {'wednesday', 'Monday', 'Thursday', 'Sunday', 'Tuesday', 'Saturday', 'Friday'}
  Type of Set :  <class 'set'>
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

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

### CODE:

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25")


X = [[1,2,3],[4,5,6],[7,8,9]]
Y = [[1,2,3,],[4,5,6],[7,8,9]]
result = [[X[i][j] + Y[i][j] for j in range(len(X[0]))] for i in range(len(X))]
for r in result:
  print(r)
```

### OUTPUT:



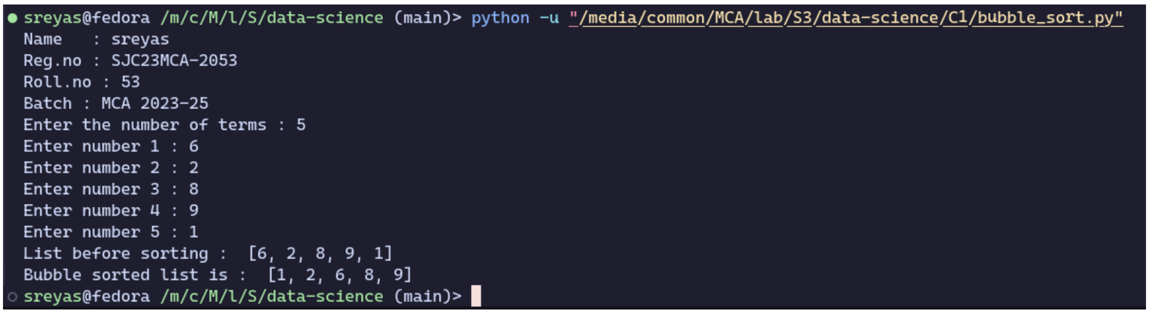
```
● sreyas@fedora /m/c/M/l/S/data-science (main) [1]> python -u "/media/common/MCA/lab/S3/data-science/C0/2.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25
  [2, 4, 6]
  [8, 10, 12]
  [14, 16, 18]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

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

### CODE:

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25")

n = int(input("Enter the number of terms : "))
a = []
for i in range(0, n):
    a.append(int(input(f"Enter number {i+1} : ")))
print("List before sorting : ", a)

for i in range(0, n-1):
    for j in range(0, n-i-1):
        if a[j] > a[j+1]:
            temp = a[j+1]
            a[j+1] = a[j]
            a[j] = temp

print("Bubble sorted list is : ", a)
```

### OUTPUT:



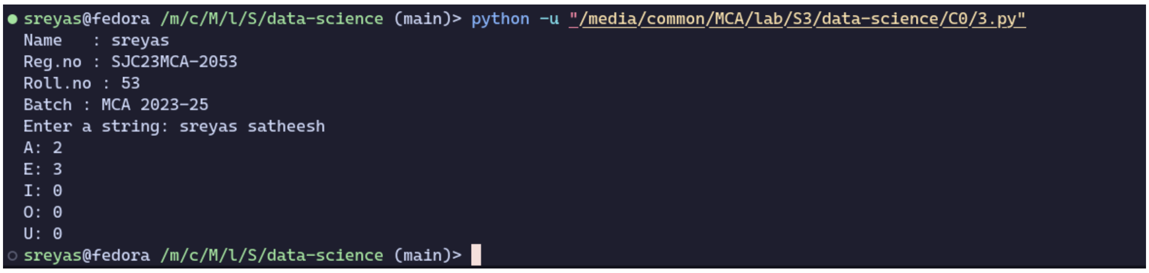
```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C1/bubble_sort.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25
  Enter the number of terms : 5
  Enter number 1 : 6
  Enter number 2 : 2
  Enter number 3 : 8
  Enter number 4 : 9
  Enter number 5 : 1
  List before sorting :  [6, 2, 8, 9, 1]
  Bubble sorted list is :  [1, 2, 6, 8, 9]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> █
```

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

## CODE:

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25")

def count_vowels(string):
  vowel_count = {'A': 0, 'E': 0, 'I': 0, 'O': 0, 'U': 0}
  string = string.upper()
  for char in string:
    if char in vowel_count:
      vowel_count[char] += 1
  return vowel_count

input_string = input("Enter a string: ")
vowel_counts = count_vowels(input_string)
for vowel, count in vowel_counts.items():
  print(f"{vowel}: {count}")
```

## OUTPUT:



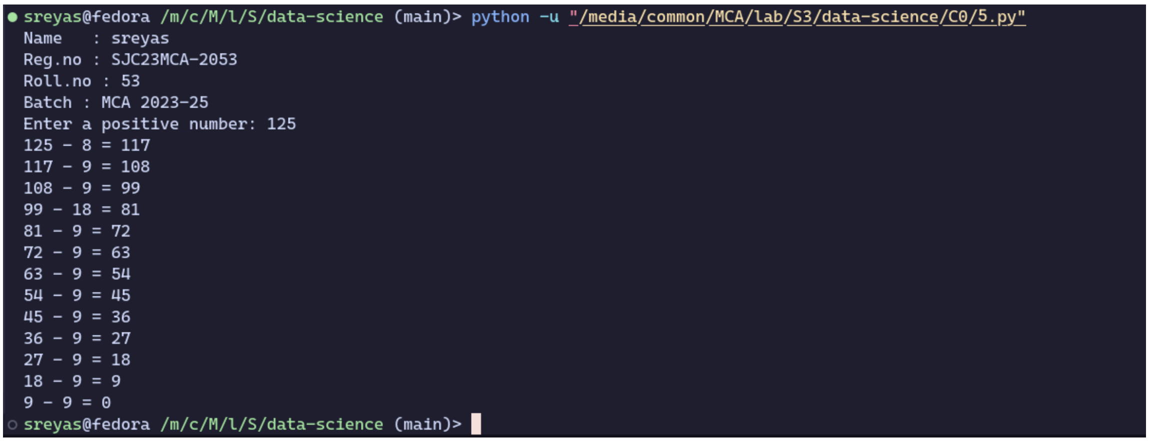
```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C0/3.py"
  Name    : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25
  Enter a string: sreyas satheesh
  A: 2
  E: 3
  I: 0
  O: 0
  U: 0
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

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

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25")


def sum_of_digits(n):
    digit_sum = 0
    while n > 0:
        digit_sum += n % 10
        n //= 10
    return digit_sum


def main():
    try:
        num = int(input("Enter a positive number: "))
        if num <= 0:
            print("Please enter a positive number.")
            return
        while num > 0:
            current_sum = sum_of_digits(num)
            print(f"{num} - {current_sum} =", num - current_sum)
            num -= current_sum
    except ValueError:
        print("Invalid input. Please enter a valid positive number.")


if __name__ == "__main__":
    main()
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C0/5.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25
  Enter a positive number: 125
  125 - 8 = 117
  117 - 9 = 108
  108 - 9 = 99
  99 - 18 = 81
  81 - 9 = 72
  72 - 9 = 63
  63 - 9 = 54
  54 - 9 = 45
  45 - 9 = 36
  36 - 9 = 27
  27 - 9 = 18
  18 - 9 = 9
  9 - 9 = 0
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

**6. 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**

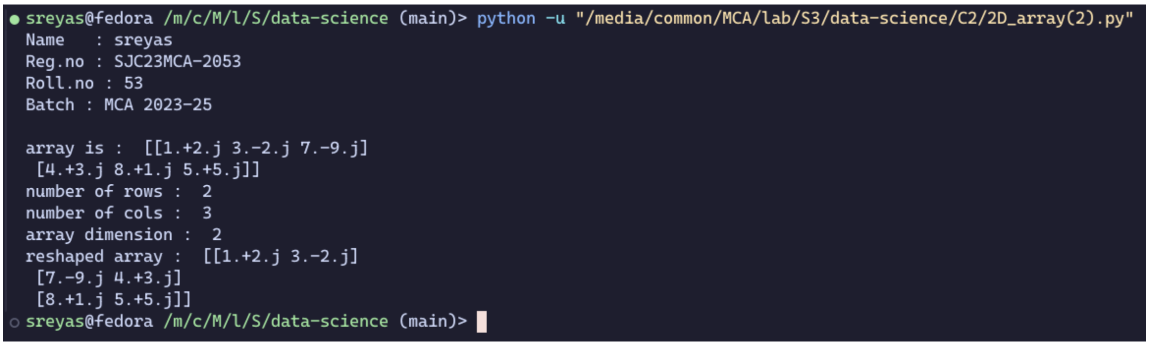
**c. reshape the same array to 3X2**

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")


from numpy import array


arr = array([
    [
        1 + 2j,
        3 - 2j,
        7 - 9j,
    ],
    [
        4 + 3j,
        8 + 1j,
        5 + 5j
    ]
], dtype=complex)

print("array is : ", arr)

# tuple destructuring (arr.shape returns a tuple with a size of 2)
(rows, cols) = arr.shape
print("number of rows : ", rows)
print("number of cols : ", cols)
```

```
dim = arr.ndim
print("array dimension : ", dim)

reshaped_arr = arr.reshape(3, 2)
print("reshaped array : ", reshaped_arr)
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/2D_array(2).py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  array is :  [[1.+2.j 3.-2.j 7.-9.j]
   [4.+3.j 8.+1.j 5.+5.j]]
  number of rows :  2
  number of cols :  3
  array dimension :  2
  reshaped array :  [[1.+2.j 3.-2.j]
   [7.-9.j 4.+3.j]
   [8.+1.j 5.+5.j]]
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

**7. Create an one dimensional array using arange function containing 10 elements.Display**

**a. First 4 elements**

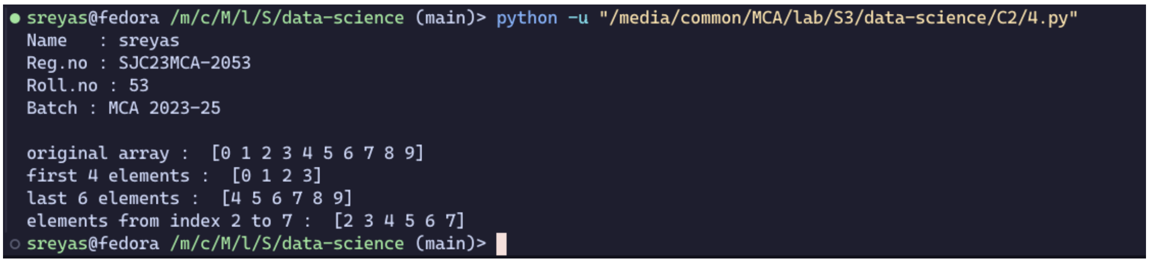
**b. Last 6 elements**

**c. Elements from index 2 to 7**

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")


import numpy as np


arr = np.arange(10)
first_4 = arr[:4]
last_6 = arr[-6:]
ele_2_to_7 = arr[2:8]


print("original array : ", arr)
print("first 4 elements : ", first_4)
print("last 6 elements : ", last_6)
print("elements from index 2 to 7 : ", ele_2_to_7)
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/4.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  original array :  [0 1 2 3 4 5 6 7 8 9]
  first 4 elements :  [0 1 2 3]
  last 6 elements :  [4 5 6 7 8 9]
  elements from index 2 to 7 :  [2 3 4 5 6 7]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

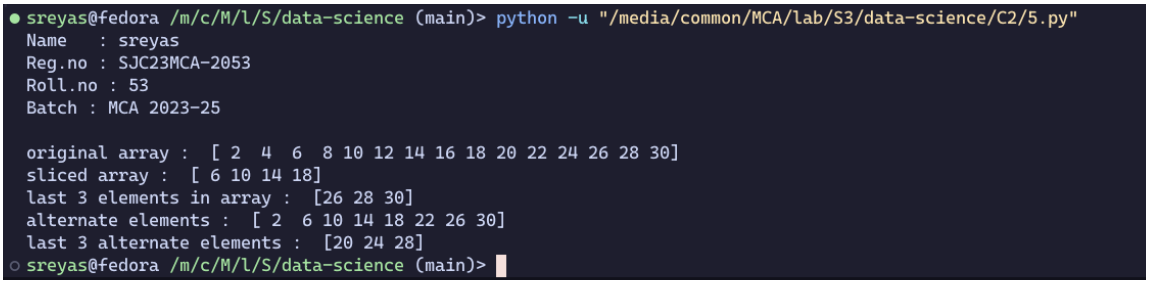
**8. Create an 1D array with arrange containing first 15 even numbers as elements**

- **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:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")


import numpy as np


arr = np.arange(2, 31, 2)
slice_arr = arr[2:9:2]
last_3 = arr[-3:]
alternate_ele = arr[::2]
last_3_alternate = arr[-3*2::2]


print("original array : ", arr)
print("sliced array : ", slice_arr)
print("last 3 elements in array : ", last_3)
print("alternate elements : ", alternate_ele)
print("last 3 alternate elements : ", last_3_alternate)
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/5.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  original array :  [ 2  4  6  8 10 12 14 16 18 20 22 24 26 28 30]
  sliced array :  [ 6 10 14 18]
  last 3 elements in array :  [26 28 30]
  alternate elements :  [ 2  6 10 14 18 22 26 30]
  last 3 alternate elements :  [20 24 28]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

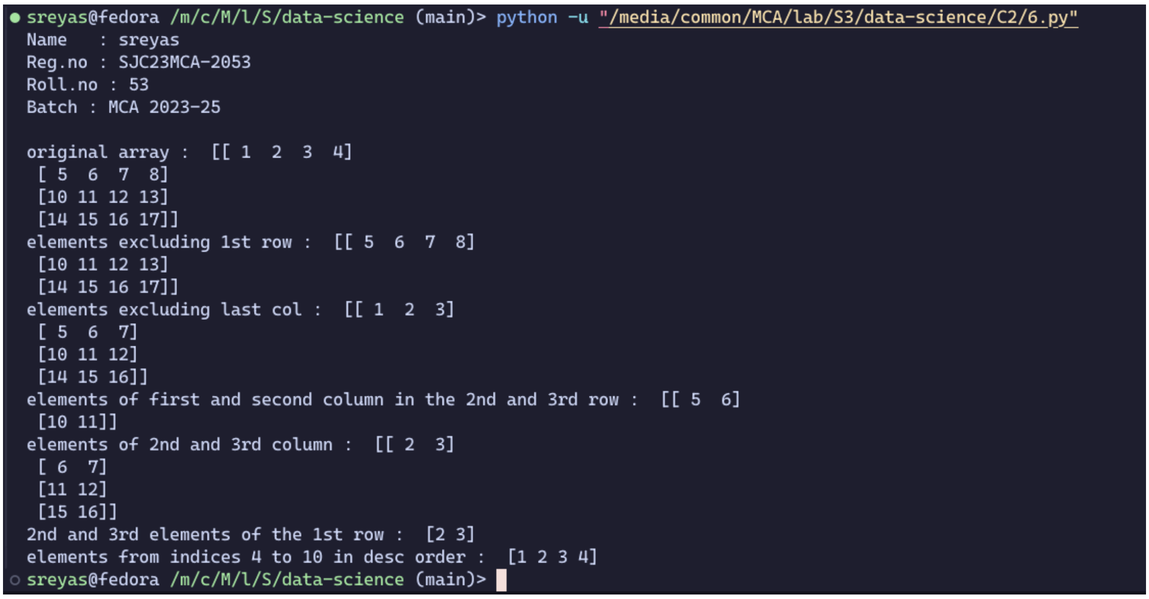
**9. Create a 2 Dimensional array with 4 rows and 4 columns.**

**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:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")

import numpy as np

arr = np.array([
   [1, 2, 3, 4],
   [5, 6, 7, 8],
   [10, 11, 12, 13],
   [14, 15, 16, 17]
])

print("original array : ", arr)
print("elements excluding 1st row : ", arr[1:])
print("elements excluding last col : ", arr[:, :-1])
print("elements of first and second column in the 2nd and 3rd row : ", arr[1:3, 0:2])
print("elements of 2nd and 3rd column : ", arr[:, 1:3])
print("2nd and 3rd elements of the 1st row : ", arr[0, 1:3])
print("elements from indices 4 to 10 in desc order : ", arr[0])
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/6.py"
  Name    : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  original array :  [[ 1  2  3  4]
   [ 5  6  7  8]
   [10 11 12 13]
   [14 15 16 17]]
  elements excluding 1st row :  [[ 5  6  7  8]
   [10 11 12 13]
   [14 15 16 17]]
  elements excluding last col :  [[ 1  2  3]
   [ 5  6  7]
   [10 11 12]
   [14 15 16]]
  elements of first and second column in the 2nd and 3rd row :  [[ 5  6]
   [10 11]]
  elements of 2nd and 3rd column :  [[ 2  3]
   [ 6  7]
   [11 12]
   [15 16]]
  2nd and 3rd elements of the 1st row :  [2 3]
  elements from indices 4 to 10 in desc order :  [1 2 3 4]
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

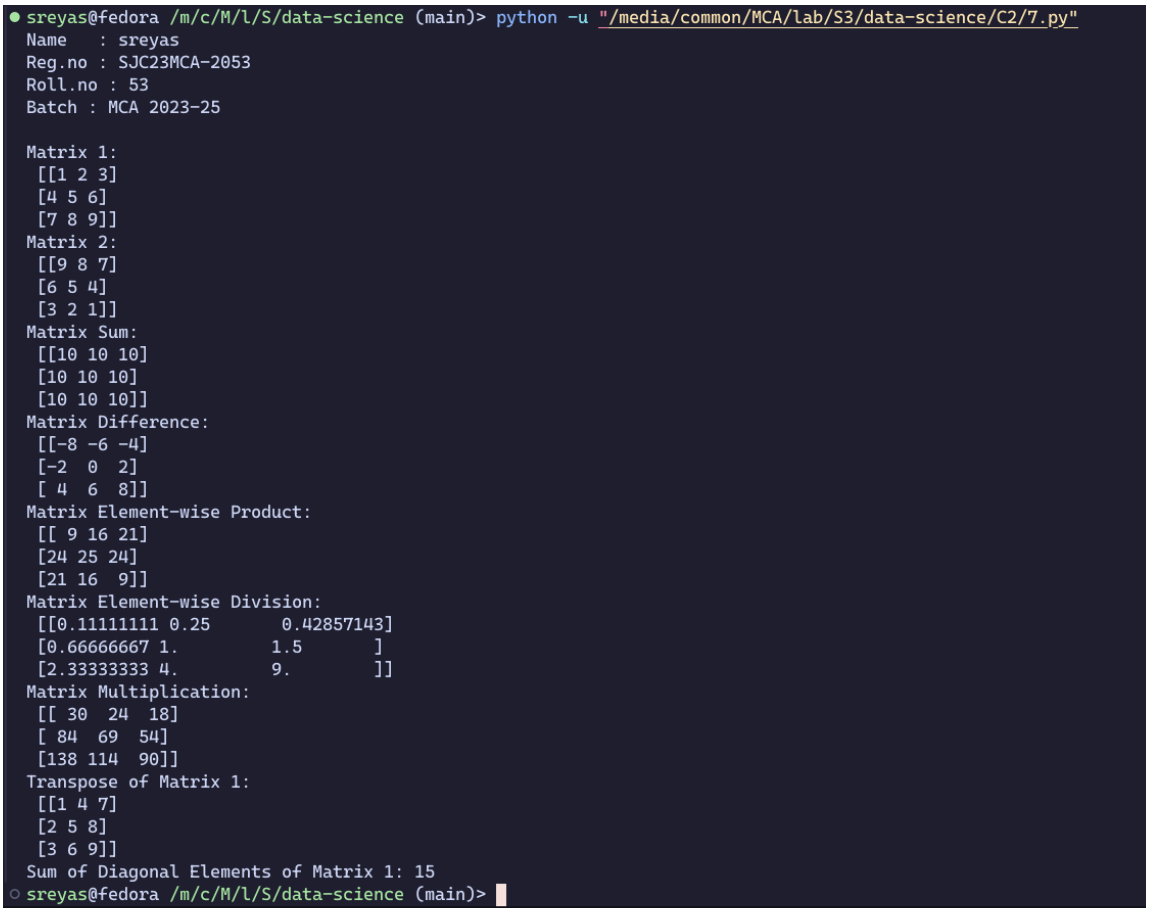
**10. Create two 2D arrays using array object and**

**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:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")
import numpy as np
matrix1 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix2 = np.array([[9, 8, 7], [6, 5, 4], [3, 2, 1]])
matrix_sum = matrix1 + matrix2
matrix_diff = matrix1 - matrix2
matrix_product = matrix1 * matrix2
matrix_divide = matrix1 / matrix2
matrix_multiply = np.dot(matrix1, matrix2)
matrix1_transpose = np.transpose(matrix1)
diagonal_sum = np.trace(matrix1)
print("Matrix 1:\n", matrix1)
print("Matrix 2:\n", matrix2)
print("Matrix Sum:\n", matrix_sum)
print("Matrix Difference:\n", matrix_diff)
print("Matrix Element-wise Product:\n", matrix_product)
print("Matrix Element-wise Division:\n", matrix_divide)
print("Matrix Multiplication:\n", matrix_multiply)
print("Transpose of Matrix 1:\n", matrix1_transpose)
print("Sum of Diagonal Elements of Matrix 1:", diagonal_sum)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/7.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Matrix 1:
   [[1 2 3]
   [4 5 6]
   [7 8 9]]
  Matrix 2:
   [[9 8 7]
   [6 5 4]
   [3 2 1]]
  Matrix Sum:
   [[10 10 10]
   [10 10 10]
   [10 10 10]]
  Matrix Difference:
   [[-8 -6 -4]
   [-2  0  2]
   [ 4  6  8]]
  Matrix Element-wise Product:
   [[ 9 16 21]
   [24 25 24]
   [21 16  9]]
  Matrix Element-wise Division:
   [[0.11111111 0.25       0.42857143]
   [0.66666667 1.         1.5       ]
   [2.33333333 4.         9.        ]]
  Matrix Multiplication:
   [[ 30  24  18]
   [ 84  69  54]
   [138 114  90]]
  Transpose of Matrix 1:
   [[1 4 7]
   [2 5 8]
   [3 6 9]]
  Sum of Diagonal Elements of Matrix 1: 15
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

**11. Create a square matrix with random integer values(use randint()) and use appropriate functions to find:**

**i. Inverse**

**ii. rank of matrix**

**iii. Determinant**

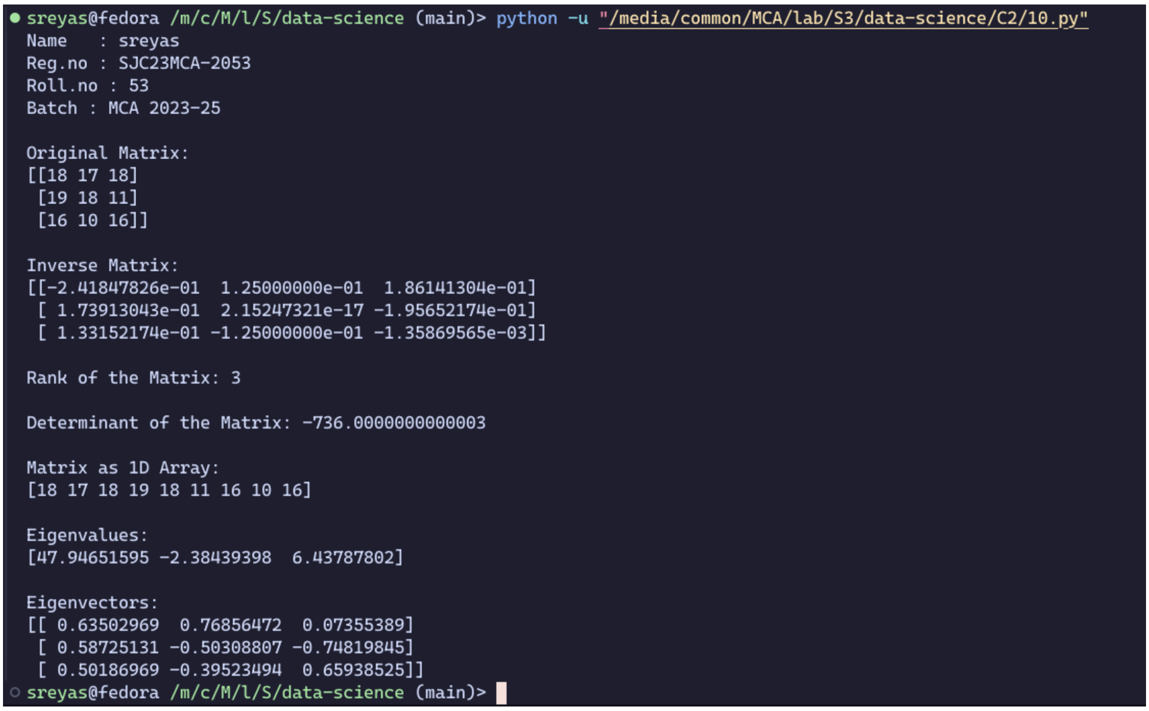
**iv. transform matrix into 1D array**

**v. eigen values and vectors**

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")

import numpy as np
matrix_size = 3
matrix = np.random.randint(10,20, size=(matrix_size, matrix_size))
print("Original Matrix:")
print(matrix)

if np.linalg.matrix_rank(matrix) == matrix_size:
    inverse_matrix = np.linalg.inv(matrix)
    print("\nInverse Matrix:")
    print(inverse_matrix)
else:
    print("\nThe matrix is not invertible (its rank is less than the size).")

rank = np.linalg.matrix_rank(matrix)
print("\nRank of the Matrix:", rank)
determinant = np.linalg.det(matrix)
print("\nDeterminant of the Matrix:", determinant)
matrix_1d = matrix.flatten()
print("\nMatrix as 1D Array:")
print(matrix_1d)
eigenvalues, eigenvectors = np.linalg.eig(matrix)
print("\nEigenvalues:")
print(eigenvalues)
print("\nEigenvectors:")
print(eigenvectors)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/10.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Original Matrix:
  [[18 17 18]
   [19 18 11]
   [16 10 16]]

  Inverse Matrix:
  [[-2.41847826e-01  1.25000000e-01  1.86141304e-01]
   [ 1.73913043e-01  2.15247321e-17 -1.95652174e-01]
   [ 1.33152174e-01 -1.25000000e-01 -1.35869565e-03]]

  Rank of the Matrix: 3

  Determinant of the Matrix: -736.0000000000003

  Matrix as 1D Array:
  [18 17 18 19 18 11 16 10 16]

  Eigenvalues:
  [47.94651595 -2.38439398  6.43787802]

  Eigenvectors:
  [[ 0.63502969  0.76856472  0.07355389]
   [ 0.58725131 -0.50308807 -0.74819845]
   [ 0.50186969 -0.39523494  0.65938525]]
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

**12. 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(),**)**

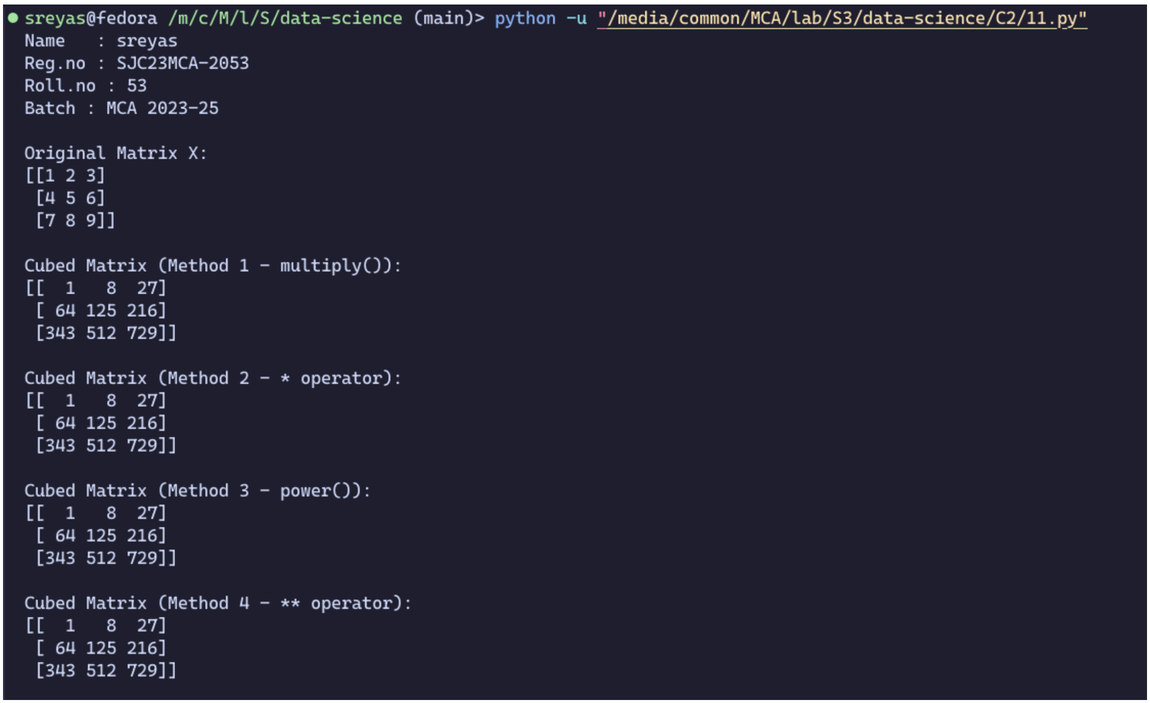
**ii. Display identity matrix of the given square matrix.**

**iii. Display each element of the matrix to different powers.**

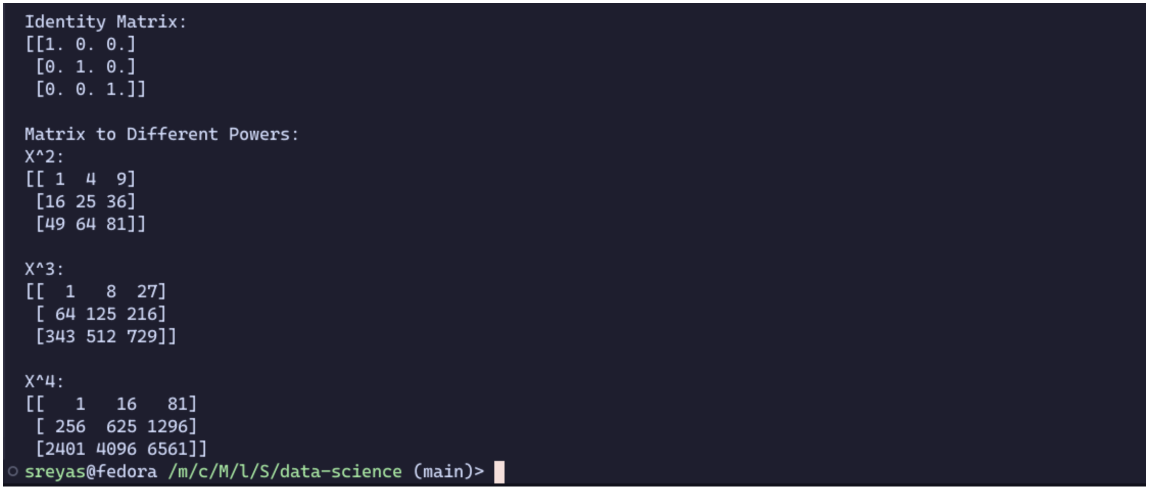
**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")
import numpy as np
X = np.array([[1, 2, 3],
[4, 5, 6],
[7, 8, 9]])
X_cube_multiply = np.multiply(X, np.multiply(X, X))
X_cube_operator = X * X * X
X_cube_power = np.power(X, 3)
X_cube_double_star = X ** 3
identity_matrix = np.identity(X.shape[0])
X_power_2 = np.power(X, 2)
X_power_3 = np.power(X, 3)
X_power_4 = np.power(X, 4)
print("Original Matrix X:")
print(X)
print("\nCubed Matrix (Method 1 - multiply()):")
print(X_cube_multiply)
print("\nCubed Matrix (Method 2 - * operator):")
print(X_cube_operator)
print("\nCubed Matrix (Method 3 - power()):")
print(X_cube_power)
print("\nCubed Matrix (Method 4 - ** operator):")
print(X_cube_double_star)
print("\nIdentity Matrix:")
print(identity_matrix)
print("\nMatrix to Different Powers:")
print("X^2:")
print(X_power_2)
print("\nX^3:")
print(X_power_3)
print("\nX^4:")
print(X_power_4)
```

**OUTPUT:**

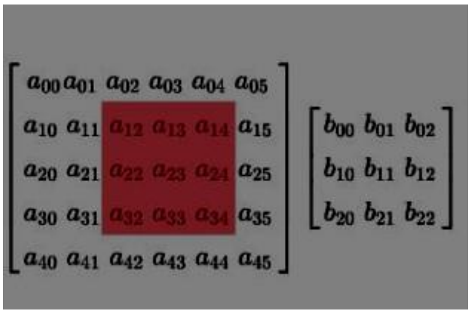


```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/11.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Original Matrix X:
  [[1 2 3]
   [4 5 6]
   [7 8 9]]

  Cubed Matrix (Method 1 - multiply()):
  [[  1   8  27]
   [ 64 125 216]
   [343 512 729]]

  Cubed Matrix (Method 2 - * operator):
  [[  1   8  27]
   [ 64 125 216]
   [343 512 729]]

  Cubed Matrix (Method 3 - power()):
  [[  1   8  27]
   [ 64 125 216]
   [343 512 729]]

  Cubed Matrix (Method 4 - ** operator):
  [[  1   8  27]
   [ 64 125 216]
   [343 512 729]]
```



```
  Identity Matrix:
  [[1. 0. 0.]
   [0. 1. 0.]
   [0. 0. 1.]]

  Matrix to Different Powers:
  X^2:
  [[ 1  4  9]
   [16 25 36]
   [49 64 81]]

  X^3:
  [[  1   8  27]
   [ 64 125 216]
   [343 512 729]]

  X^4:
  [[   1   16   81]
   [ 256  625 1296]
   [2401 4096 6561]]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

**13. 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**



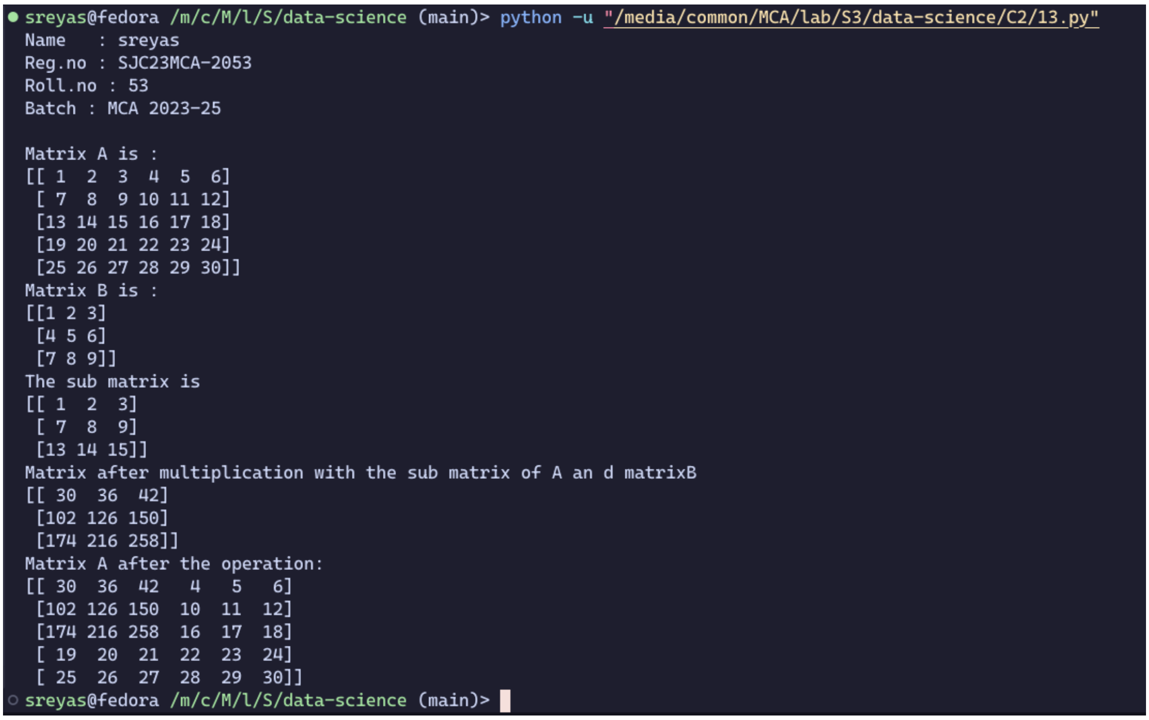
$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{bmatrix} \begin{bmatrix} b_{00} & b_{01} & b_{02} \\ b_{10} & b_{11} & b_{12} \\ b_{20} & b_{21} & b_{22} \end{bmatrix}$$

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")

import numpy as np
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]])

print("Matrix A is : ")
print(A)
B = np.array([[1,2,3,],[4,5,6],[7,8,9]])
print("Matrix B is : ")
print(B)
sub_matrix = A[:3, :3]
print("The sub matrix is ")
print(sub_matrix)
result = np.dot(sub_matrix,B)
print("Matrix after multiplication with the sub matrix of A an d matrixB")
print(result)
A[:3, :3] = result
```

```
print("Matrix A after the operation:")
print(A)
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/13.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Matrix A is :
  [[ 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]]
  Matrix B is :
  [[1 2 3]
   [4 5 6]
   [7 8 9]]
  The sub matrix is
  [[ 1  2  3]
   [ 7  8  9]
   [13 14 15]]
  Matrix after multiplication with the sub matrix of A an d matrixB
  [[ 30  36  42]
   [102 126 150]
   [174 216 258]]
  Matrix A after the operation:
  [[ 30  36  42   4   5   6]
   [102 126 150  10  11  12]
   [174 216 258  16  17  18]
   [ 19  20  21  22  23  24]
   [ 25  26  27  28  29  30]]
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

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

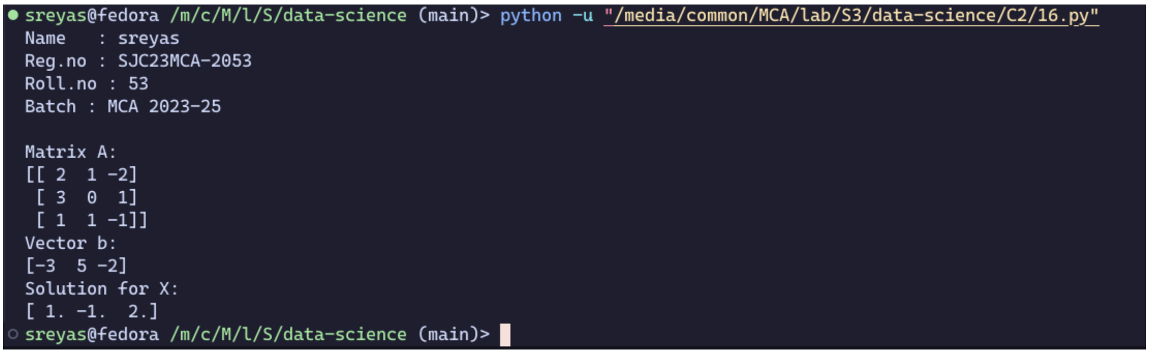
**$X=A^{-1}$ b.**

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

### CODE:

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")

import numpy as np
A = np.array([[2, 1,-2],[3,0,1],[1,1,-1]])
b = np.array([-3,5,-2])
X = np.linalg.solve(A, b)
print("Matrix A:")
print(A)
print("Vector b:")
print(b)
print("Solution for X:")
print(X)
```

### OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/16.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Matrix A:
  [[ 2  1 -2]
   [ 3  0  1]
   [ 1  1 -1]]
  Vector b:
  [-3  5 -2]
  Solution for X:
  [ 1. -1.  2.]
○ sreyas@fedora /m/c/M/l/S/data-science (main)> 
```

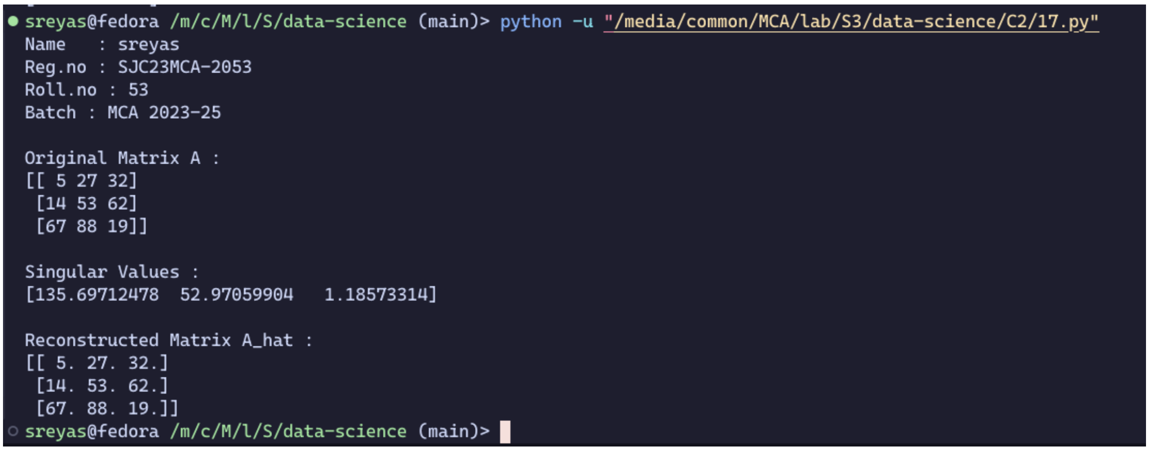
**15. Write program to perform the SVD of a given matrix A. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.**

Use the function: numpy.linalg.svd(), Singular value Decomposition Matrix decomposition, also known as matrix factorization, involves describing a given matrix using its constituent elements.The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. This approach is commonly used in reducing the no: of attributes in the given data set.The SVD of mxn matrix A is given by the formula $A = U\Sigma V^T$

**CODE:**

```
print("Name   : sreyas")
print("Reg.no : SJC23MCA-2053")
print("Roll.no : 53")
print("Batch : MCA 2023-25\n")
import numpy as np
A = np.array([[5, 27, 32], [14, 53, 62], [67, 88, 19]])
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:**



```
● sreyas@fedora /m/c/M/l/S/data-science (main)> python -u "/media/common/MCA/lab/S3/data-science/C2/17.py"
  Name   : sreyas
  Reg.no : SJC23MCA-2053
  Roll.no : 53
  Batch : MCA 2023-25

  Original Matrix A :
  [[ 5 27 32]
   [14 53 62]
   [67 88 19]]

  Singular Values :
  [135.69712478  52.97059904   1.18573314]

  Reconstructed Matrix A_hat :
  [[ 5. 27. 32.]
   [14. 53. 62.]
   [67. 88. 19.]]
○ sreyas@fedora /m/c/M/l/S/data-science (main)>
```

**16. Demonstrate creating various types of charts and plots using functions in mathplotlib library**
**Sarah bought a new car in 2001 for $24,000. The dollar value of her car changed each year as shown in the table below.**

**Value of Sarah's Car**

| Year | Value |
|---|---|
| 2001 | $24,000 |
| 2002 | $22,500 |
| 2003 | $19,700 |
| 2004 | $17,500 |
| 2005 | $14,500 |
| 2006 | $10,000 |
| 2007 | $ 5,800 |

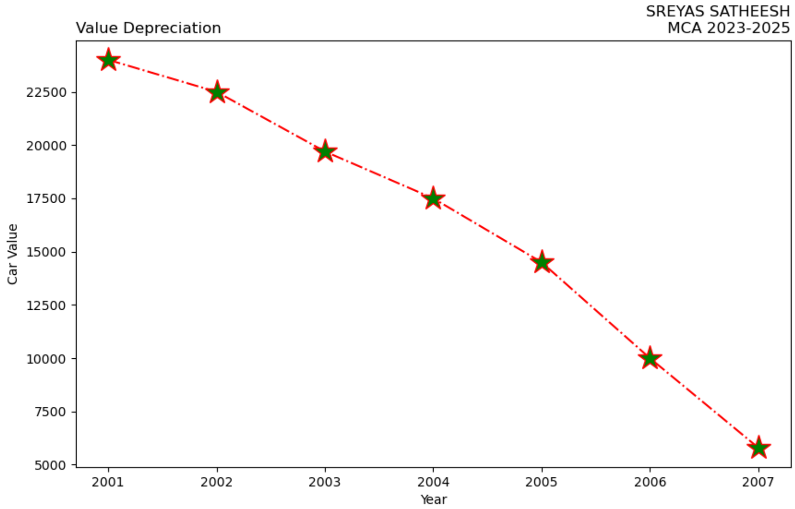
Represent the following information using a line graph with following style properties

- **X- axis - Year. Y –axis - Car Value**
- **title –Value Depreciation (left Aligned)**
- **Line Style dash dot and Line-color should be red**
- **point using * symbol with green color and size 20**

**CODE:**

```
import matplotlib.pyplot as plt

years = [2001, 2002, 2003, 2004, 2005, 2006, 2007]

car_values = [24000, 22500, 19700, 17500, 14500, 10000, 5800]

plt.figure(figsize=(10, 6))


plt.plot(years, car_values, linestyle='-.', color='red', marker='*', markersize=20,
markerfacecolor='green')

plt.title("SREYAS SATHEESH\n MCA 2023-2025", loc="right")

plt.title("Value Depreciation", loc="left")

plt.xlabel("Year")

plt.ylabel("Car Value")


plt.savefig("./Outputs/1.png")

plt.show()
```

## OUTPUT:


Value Depreciation
SREYAS SATHEESH
MCA 2023-2025
Car Value
22500
20000
17500
15000
12500
10000
7500
5000
2001
2002
2003
2004
2005
2006
2007
Year

**17. Following table gives the daily sales of the following items in a shop**

| Day | Mon | Tues | Wed | Thurs | Fri |
|---|---|---|---|---|---|
| Drinks | 300 | 450 | 150 | 400 | 650 |
| Food | 400 | 500 | 350 | 300 | 500 |

**Use subplot function to draw the line graphs with grids(color as blue and line style dotted) for the above information as 2 separate graphs in two rows**

**a) Properties for the Graph 1:**

- **X label- Days of week**
- **Y label-Sale of Drinks**
- **Title-Sales Data1 (right aligned)**
- **Line –dotted with cyan color**
- **Points- hexagon shape with color magenta and outline black**

**b) Properties for the Graph 2:**

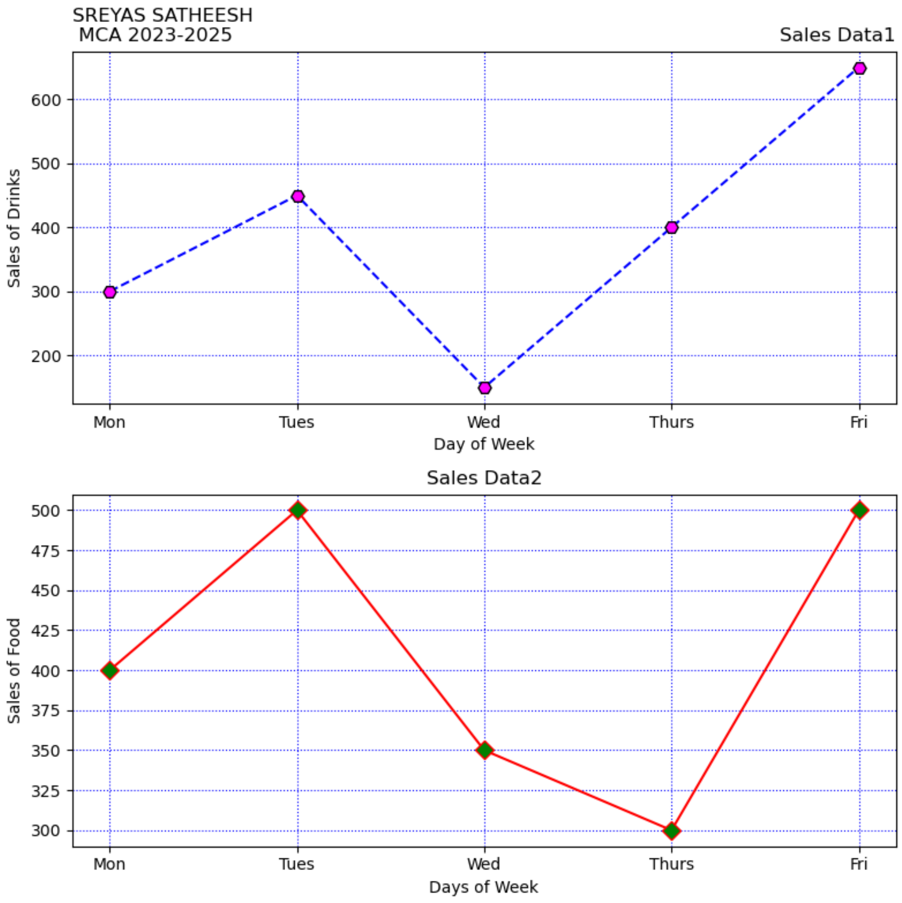
- **X label- Days of Week**
- **Y label-Sale of Food**
- **Title-Sales Data2 ( center aligned)**
- **Line –dashed with yellow color**
- **Points- diamond shape with color green and outline red**

**CODE:**

```
import matplotlib.pyplot as plt
days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri']
drinks_sales = [300, 450, 150, 400, 650]
food_sales = [400, 500, 350, 300, 500]
fig, axs = plt.subplots(2, 1, figsize=(8, 8))
axs[0].plot(days, drinks_sales, linestyle='--', color='blue', marker='H', markersize=8,
markerfacecolor='magenta', markeredgecolor='black')
axs[0].set_xlabel('Day of Week')
axs[0].set_ylabel('Sales of Drinks')
axs[0].set_title('Sales Data1', loc='right')
axs[0].set_title("SREYAS SATHEESH\n MCA 2023-2025", loc="left")
axs[0].grid(True, color='blue', linestyle='dotted')
```

```
axs[1].plot(days, food_sales, linestyle='-', color='red', marker='D', markersize=8,
markerfacecolor='green', markeredgecolor='red')
axs[1].set_xlabel('Days of Week')
axs[1].set_ylabel('Sales of Food')
axs[1].set_title('Sales Data2', loc='center')
axs[1].grid(True, color='blue', linestyle='dotted')
plt.tight_layout()
plt.savefig("./Outputs/2.png")
plt.show()
```

**OUTPUT:**



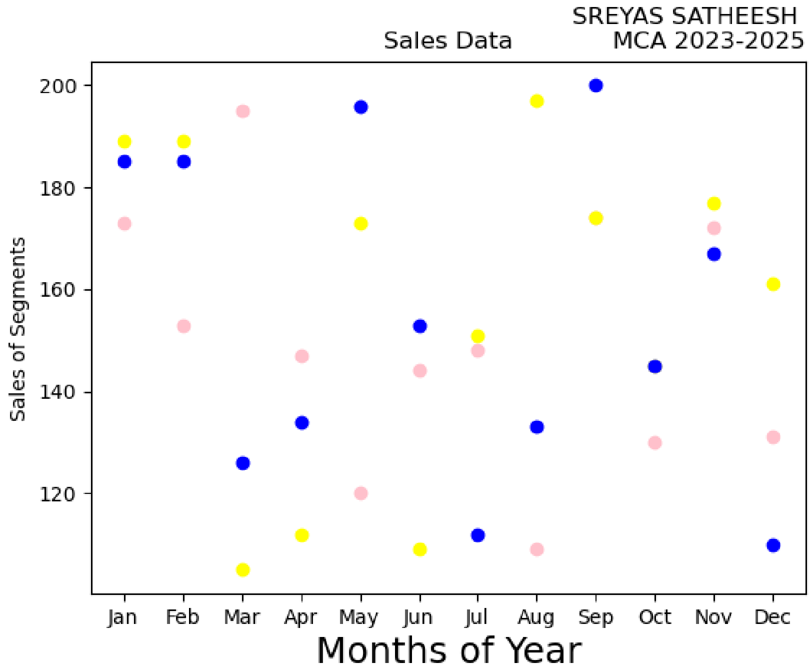
**18. Create scatter plot for the below data:(use Scatter function)**
**Create scatter plot for each Segment with following properties within one graph**

- **X Label- Months of Year with font size 18**
- **Y-Label- Sales of Segments**
- **Title –Sales Data**
- **Color for Affordable segment- pink**
- **Color for Luxury Segment- Yellow**

**CODE:**

```
import matplotlib.pyplot as plt
import numpy as np
month =np.array(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])
AS = np.array([173,153,195,147,120,144,148,109,174,130,172,131])
LS = np.array([189,189,105,112,173,109,151,197,174,145,177,161])
SLS = np.array([185,185,126,134,196,153,112,133,200,145,167,110])
plt.xlabel('Months of Year', fontsize=18)
plt.ylabel('Sales of Segments')
plt.title('Sales Data')
plt.title('SREYAS SATHEESH \nMCA 2023-2025', loc='right')
plt.scatter(month,AS, label='Affordable Segment', color='pink')
plt.scatter(month,LS, label='Luxury Segment', color='yellow')
plt.scatter(month,SLS, label='Super Luxury Segment', color='blue')
plt.savefig("./Outputs/3.png")
plt.show()
```

## OUTPUT:


Sales Data
SREYAS SATHEESH
MCA 2023-2025
Sales of Segments
200
180
160
140
120
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Months of Year

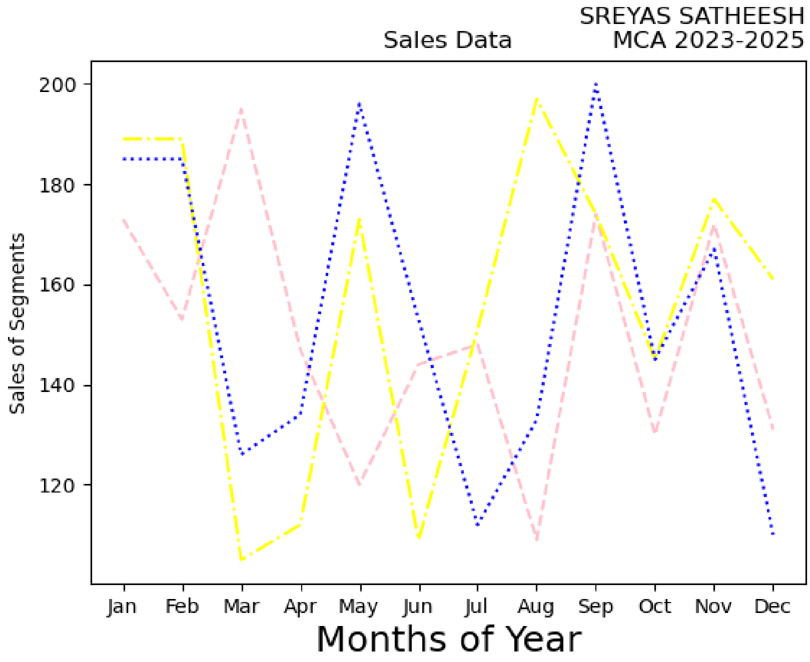
**19. Display the above data using multiline plot( 3 different lines in same graph)**

- **Display the description of the graph in upper right corner(use legend())**
- **Use different colors and line styles for 3 different lines**

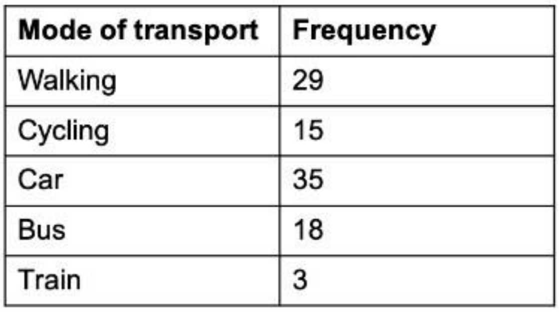
**CODE:**

```
import matplotlib.pyplot as plt
import numpy as np
month =np.array(['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec'])
AS = np.array([173,153,195,147,120,144,148,109,174,130,172,131])
LS = np.array([189,189,105,112,173,109,151,197,174,145,177,161])
SLS = np.array([185,185,126,134,196,153,112,133,200,145,167,110])
plt.plot(month,AS, label='Affordable', color='pink',linestyle='--')
plt.plot(month,LS, label='Luxury', color='yellow',linestyle='-.')
plt.plot(month,SLS, label='Super Luxury', color='blue',linestyle=':')
plt.xlabel('Months of Year', fontsize=18)
plt.ylabel('Sales of Segments')
plt.title('Sales Data')
plt.title('SREYAS SATHEESH\nMCA 2023-2025', loc='right')
plt.savefig("./Outputs/4.png")
plt.show()
```

**OUTPUT:**



**20. 100 students were asked what their primary mode of transport for getting to school was. The results of this survey are recorded in the table below. Construct a bar graph representing this information.**



| Mode of transport | Frequency |
|---|---|
| Walking | 29 |
| Cycling | 15 |
| Car | 35 |
| Bus | 18 |
| Train | 3 |

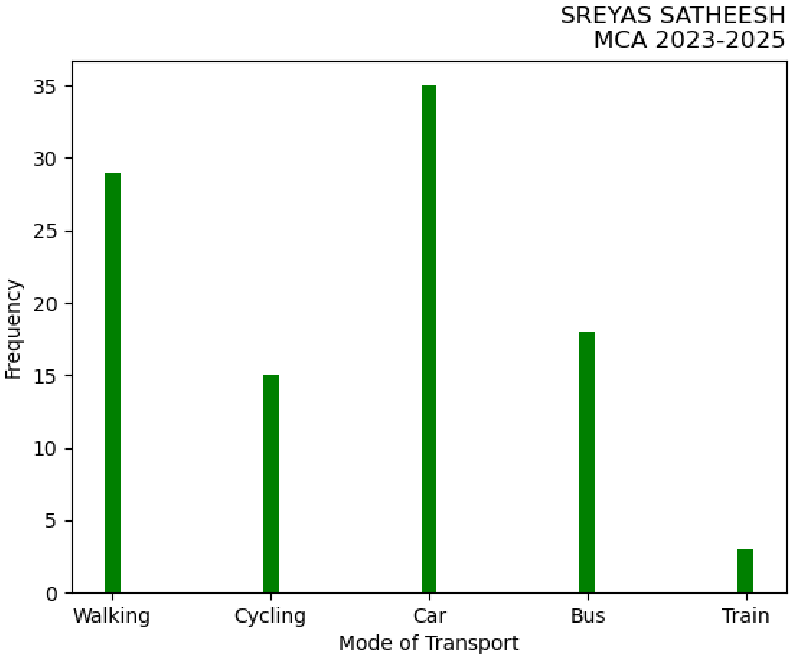
**Create a bar graph with**

- **X axis -mode of Transport and Y axis ‘frequency’**
- **Provide appropriate labels and title**
- **Width .1, color green**

**CODE:**

```
import matplotlib.pyplot as plt
import numpy as np
mode_transport = np.array(['Walking','Cycling','Car','Bus','Train'])
feq = np.array([29,15,35,18,3])
plt.xlabel('Mode of Transport')
plt.ylabel('Frequency')
plt.title('SREYAS SATHEESH\nMCA 2023-2025', loc='right')
plt.bar(mode_transport,feq, width=0.1, color='green')
plt.savefig("./Outputs/5.png")
plt.show()
```

**OUTPUT:**

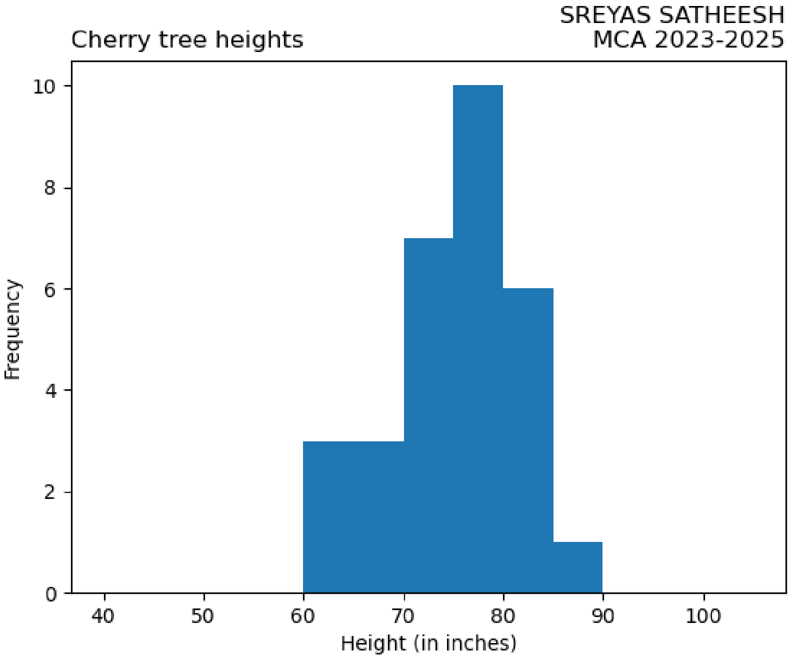

SREYAS SATHEESH
MCA 2023-2025
Frequency
0
5
10
15
20
25
30
35
Walking
Cycling
Car
Bus
Train
Mode of Transport

**21. We are provided with the height of 30 cherry trees.The height of the trees (in inches): 61,63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74, 74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87.Create a histogram with a bin size of 5**

**CODE:**

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal([61, 63, 64, 66, 68, 69, 71, 71.5, 72, 72.5, 73, 73.5, 74,
74.5, 76, 76.2, 76.5, 77, 77.5, 78, 78.5, 79, 79.2, 80, 81, 82, 83, 84, 85, 87])
plt.hist(x, bins=range(40,110,5), )
plt.title('Cherry tree heights',loc='left')
plt.title('SREYAS SATHEESH\nMCA 2023-2025', loc='right')
plt.xlabel('Height (in inches)')
plt.ylabel('Frequency')
plt.savefig("./Outputs/6.png")
plt.show()
```

**OUTPUT:**



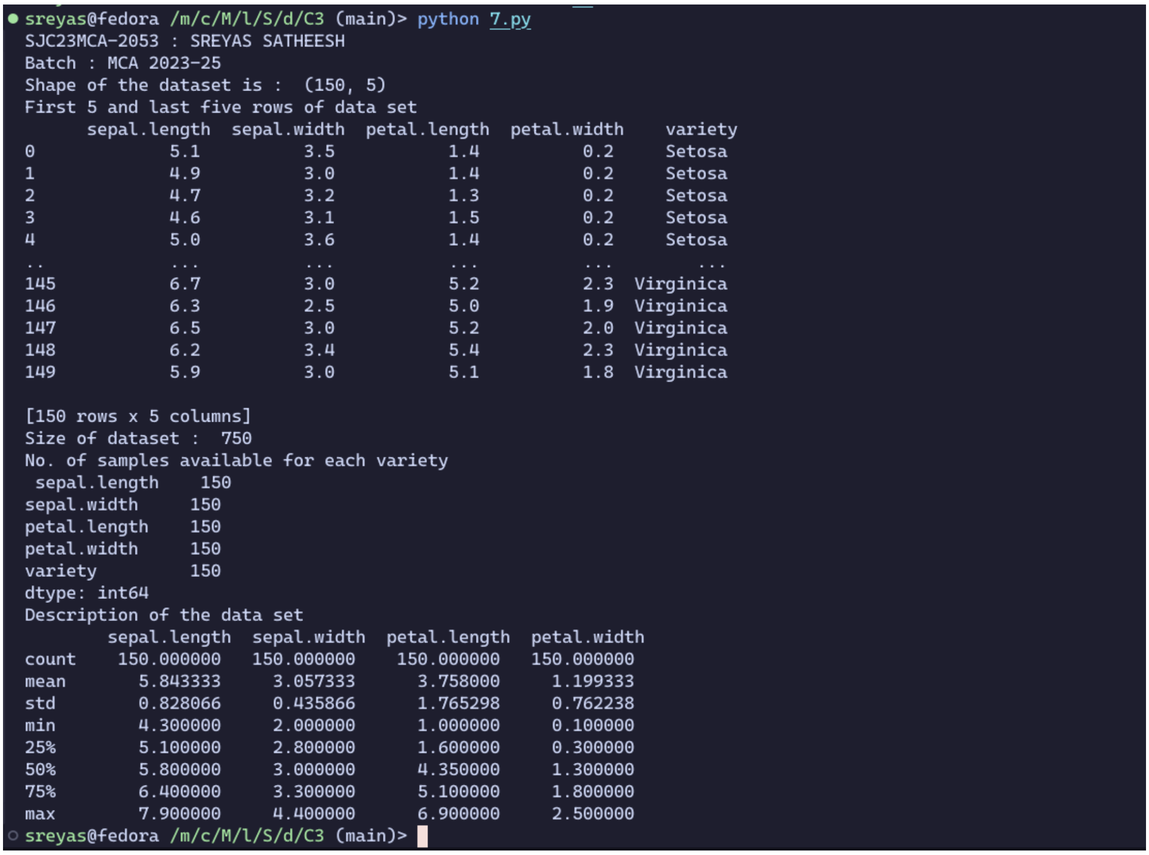
**22. Using the pandas function read_csv(), read the given 'iris' data set.**

**i. Shape of the data set.**
**ii. First 5 and last five rows of data set(head and tail)**
**iii. Size of dataset.**
**iv. No. of samples available for each variety.**
**v. Description of the data set( use describe ).**

**CODE:**

```
print("SJC23MCA-2053 : SREYAS SATHEESH")
print("Batch : MCA 2023-25")
import pandas as pd
df = pd.read_csv('iris.csv')
print("Shape of the dataset is : ",df.shape)
print("First 5 and last five rows of data set\n",df)
print("Size of dataset : ",df.size)
print("No. of samples available for each variety\n",df.count())
print("Description of the data set\n",df.describe())
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/d/C3 (main)> python 7.py
  SJC23MCA-2053 : SREYAS SATHEESH
  Batch : MCA 2023-25
  Shape of the dataset is :  (150, 5)
  First 5 and last five rows of data set
        sepal.length  sepal.width  petal.length  petal.width    variety
  0              5.1          3.5           1.4          0.2     Setosa
  1              4.9          3.0           1.4          0.2     Setosa
  2              4.7          3.2           1.3          0.2     Setosa
  3              4.6          3.1           1.5          0.2     Setosa
  4              5.0          3.6           1.4          0.2     Setosa
  ..             ...          ...           ...          ...        ...
  145            6.7          3.0           5.2          2.3  Virginica
  146            6.3          2.5           5.0          1.9  Virginica
  147            6.5          3.0           5.2          2.0  Virginica
  148            6.2          3.4           5.4          2.3  Virginica
  149            5.9          3.0           5.1          1.8  Virginica

  [150 rows x 5 columns]
  Size of dataset :  750
  No. of samples available for each variety
   sepal.length    150
  sepal.width     150
  petal.length    150
  petal.width     150
  variety         150
  dtype: int64
  Description of the data set
          sepal.length  sepal.width  petal.length  petal.width
  count    150.000000   150.000000    150.000000   150.000000
  mean       5.843333     3.057333      3.758000     1.199333
  std        0.828066     0.435866      1.765298     0.762238
  min        4.300000     2.000000      1.000000     0.100000
  25%        5.100000     2.800000      1.600000     0.300000
  50%        5.800000     3.000000      4.350000     1.300000
  75%        6.400000     3.300000      5.100000     1.800000
  max        7.900000     4.400000      6.900000     2.500000
○ sreyas@fedora /m/c/M/l/S/d/C3 (main)>
```

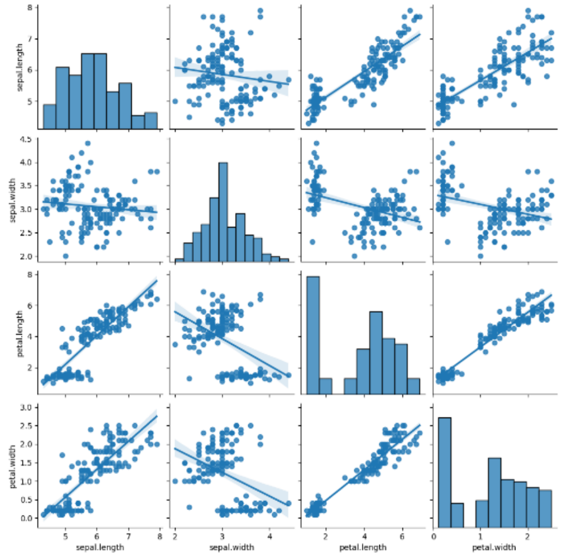
**23. Use the pairplot() function in seaborn to display pairwise relationships between attributes.**

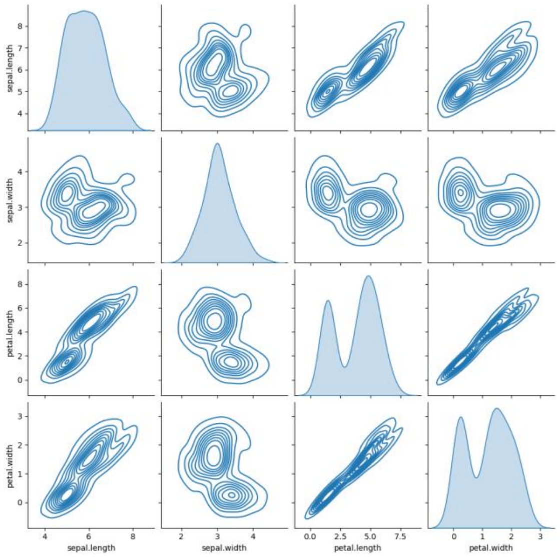
**Try different kind of plots {‘scatter’, ‘kde’, ‘hist’, ‘reg’} and different kind of markers.**

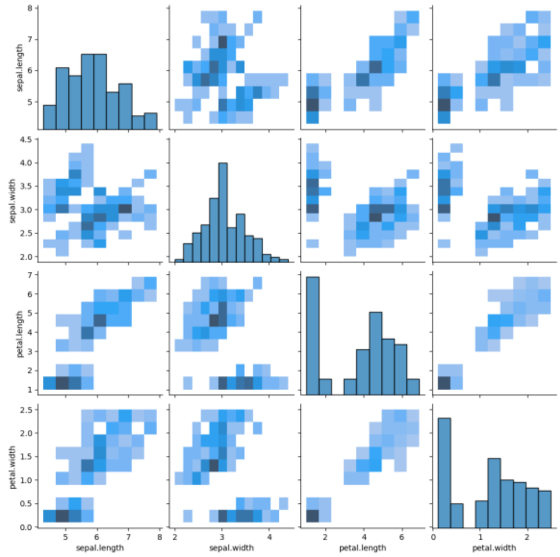
## CODE:

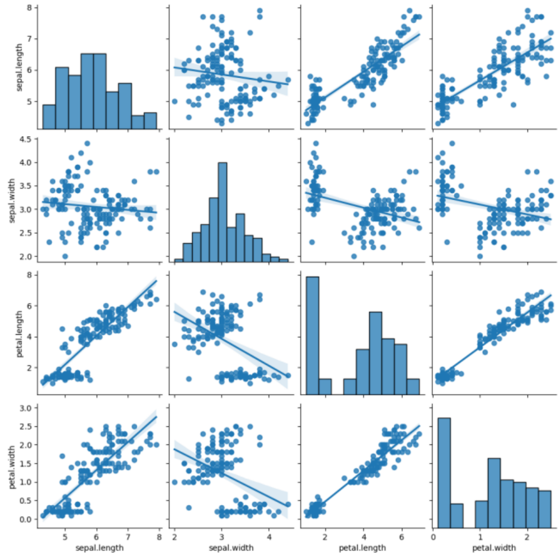
```
import pandas
import seaborn
import matplotlib.pyplot as plt

# Reading dataset
dataset = pandas.read_csv("iris.csv")

seaborn.pairplot(dataset, kind="scatter")
plt.savefig("./Outputs/8_1.png")

seaborn.pairplot(dataset, kind="kde")
plt.savefig("./Outputs/8_2.png")

seaborn.pairplot(dataset, kind="hist")
plt.savefig("./Outputs/8_3.png")

seaborn.pairplot(dataset, kind="reg")
plt.savefig("./Outputs/8_4.png")

plt.show()
```

## OUTPUT:


sepal.length
sepal.width
petal.length
petal.width
sepal.length
sepal.width
petal.length
petal.width


sepal.length
sepal.width
petal.length
petal.width
sepal.length
sepal.width
petal.length
petal.width


sepal.length
sepal.width
petal.length
petal.width
sepal.length
sepal.width
petal.length
petal.width


sepal.length
sepal.width
petal.length
petal.width
sepal.length
sepal.width
petal.length
petal.width

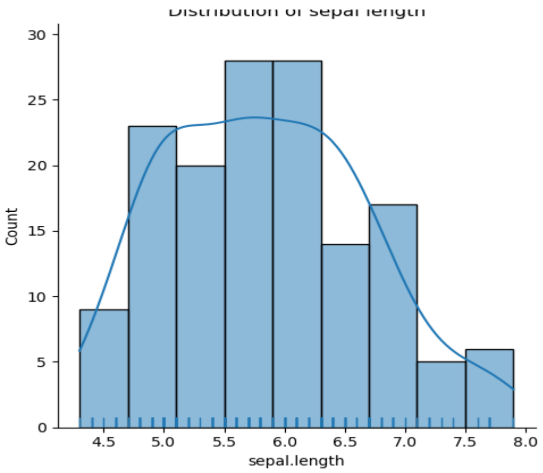
**24. Using the iris data set,get familiarize with functions:**

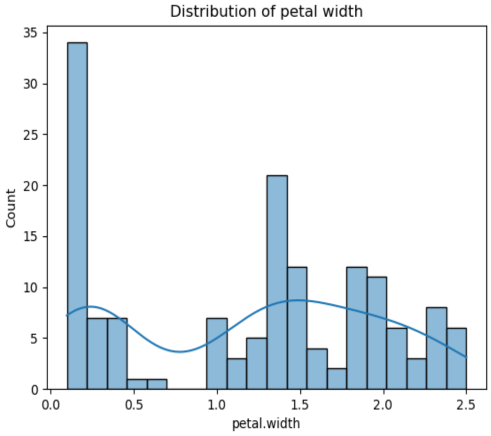
1) **displot()**
2) **histplot()**
3) **relplot()**

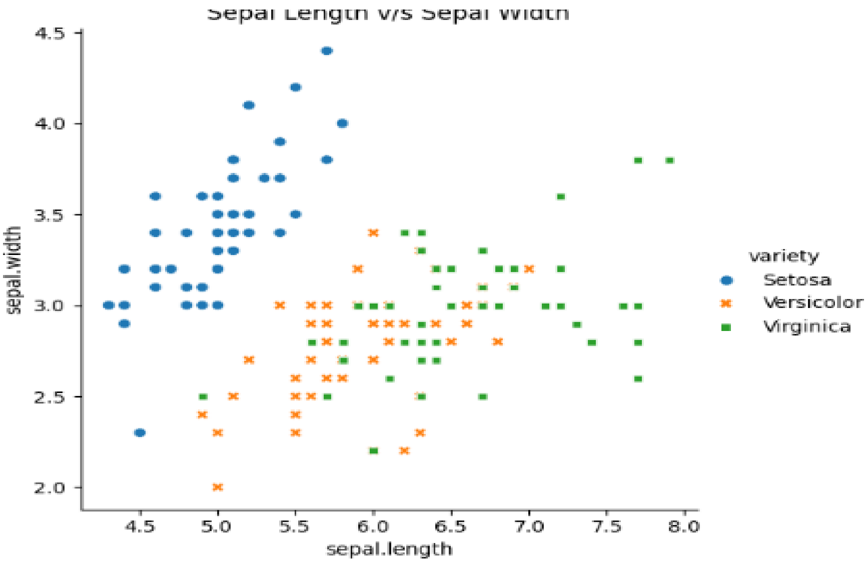
**CODE:**

```
import pandas
import matplotlib.pyplot as plt
import seaborn

iris_dataset = pandas.read_csv("iris.csv")

seaborn.displot(iris_dataset['sepal.length'], kde=True, rug=True)
plt.title("Distribution of sepal length")
plt.savefig("./Outputs/9_1.png")
plt.show()

seaborn.histplot(iris_dataset['petal.width'], kde=True, bins=20)
plt.title("Distribution of petal width")
plt.savefig("./Outputs/9_2.png")
plt.show()

seaborn.relplot(x="sepal.length", y="sepal.width", data=iris_dataset, hue="variety",
style="variety")
plt.title("sepal length vs sepal width")
plt.savefig("./Outputs/9_3.png")
plt.show()
```

## OUTPUT:


Distribution of sepal length
Count
sepal.length
0
5
10
15
20
25
30
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Distribution of petal width
Count
petal.width
0
5
10
15
20
25
30
35
0.0
0.5
1.0
1.5
2.0
2.5


Sepal Length v/s Sepal Width
sepal.width
sepal.length
2.0
2.5
3.0
3.5
4.0
4.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
variety
Setosa
Versicolor
Virginica

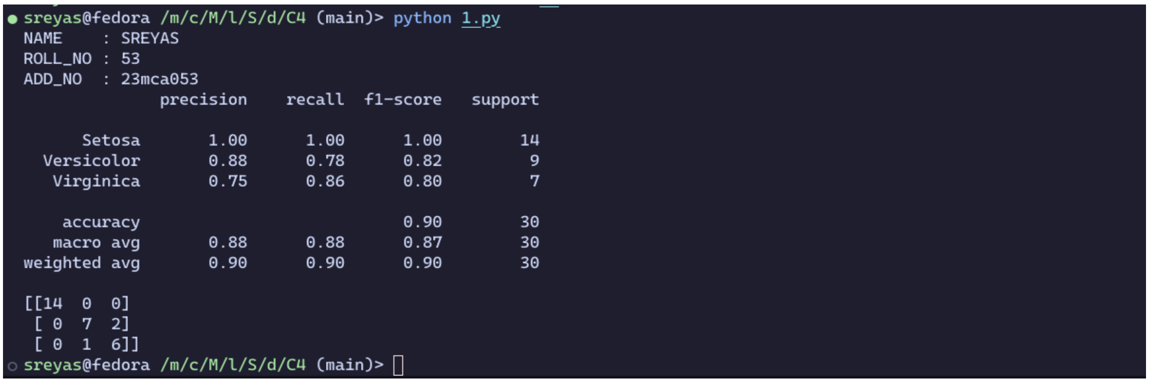
**25. Using the iris data set, implement the KNN algorithm. Take different values for the Test and training data set .Also use different values for k. Also find the accuracy level.**

## CODE:

```
print("NAME    : SREYAS")
print("ROLL_NO : 53")
print("ADD_NO  : 23mca053")
import pandas as pd
dataset = pd.read_csv("iris.csv")
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
```

## OUTPUT:

K=5, TEST= 0.20, TRAIN= 0.80
K=3, TEST= 0.20, TRAIN= 0.80



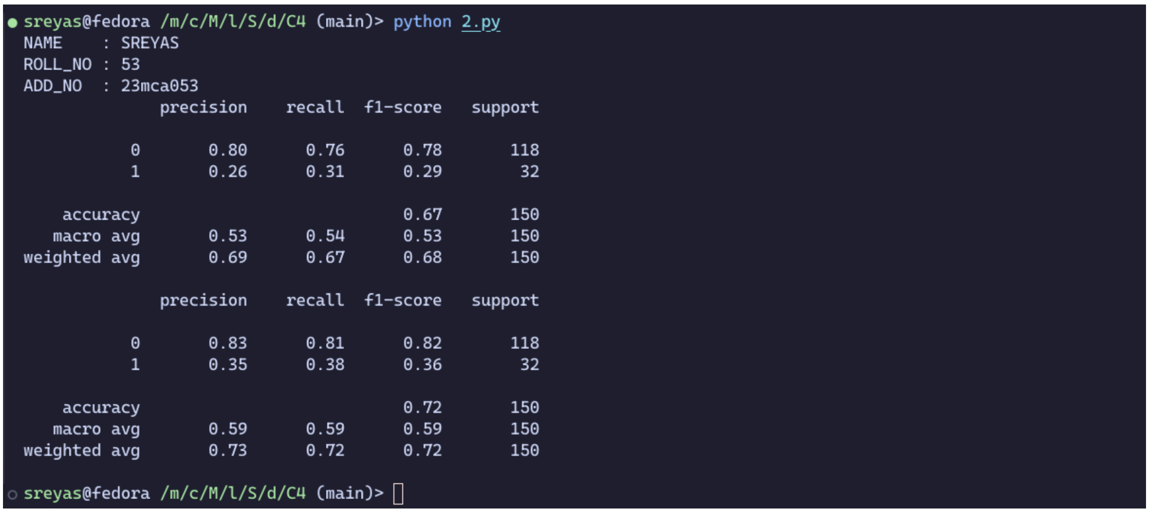
```
● sreyas@fedora /m/c/M/l/S/d/C4 (main)> python 1.py
  NAME    : SREYAS
  ROLL_NO : 53
  ADD_NO  : 23mca053
                precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        14
  Versicolor       0.88      0.78      0.82         9
   Virginica       0.75      0.86      0.80         7

    accuracy                           0.90        30
   macro avg       0.88      0.88      0.87        30
weighted avg       0.90      0.90      0.90        30

  [[14  0  0]
   [ 0  7  2]
   [ 0  1  6]]
○ sreyas@fedora /m/c/M/l/S/d/C4 (main)> 
```

**26. Using ‘blood_transfusion dataset’ implement KNN algorithm.**

**CODE:**

```
print("NAME    : SREYAS")
print("ROLL_NO : 53")
print("ADD_NO  : 23mca053")

import pandas as pd
dataset = pd.read_csv("transfusion.csv")
x = dataset.iloc[:,:-1].values
y = dataset.iloc[:,4].values
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
from sklearn.neighbors import KNeighborsClassifier

# k values as 5
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))

# k value as 2
classifier = KNeighborsClassifier(n_neighbors=3)
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,y_pred))
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C4 (main)> python 2.py
  NAME    : SREYAS
  ROLL_NO : 53
  ADD_NO  : 23mca053
              precision    recall  f1-score   support

           0       0.80      0.76      0.78       118
           1       0.26      0.31      0.29        32

    accuracy                           0.67       150
   macro avg       0.53      0.54      0.53       150
weighted avg       0.69      0.67      0.68       150

              precision    recall  f1-score   support

           0       0.83      0.81      0.82       118
           1       0.35      0.38      0.36        32

    accuracy                           0.72       150
   macro avg       0.59      0.59      0.59       150
weighted avg       0.73      0.72      0.72       150

○ sreyas@fedora /m/c/M/l/S/d/C4 (main)>
```

**27. Using iris data set, implement naive bayes classification for different naive Bayes classification algorithms.( (i) gaussian (ii) bernoulli etc)**

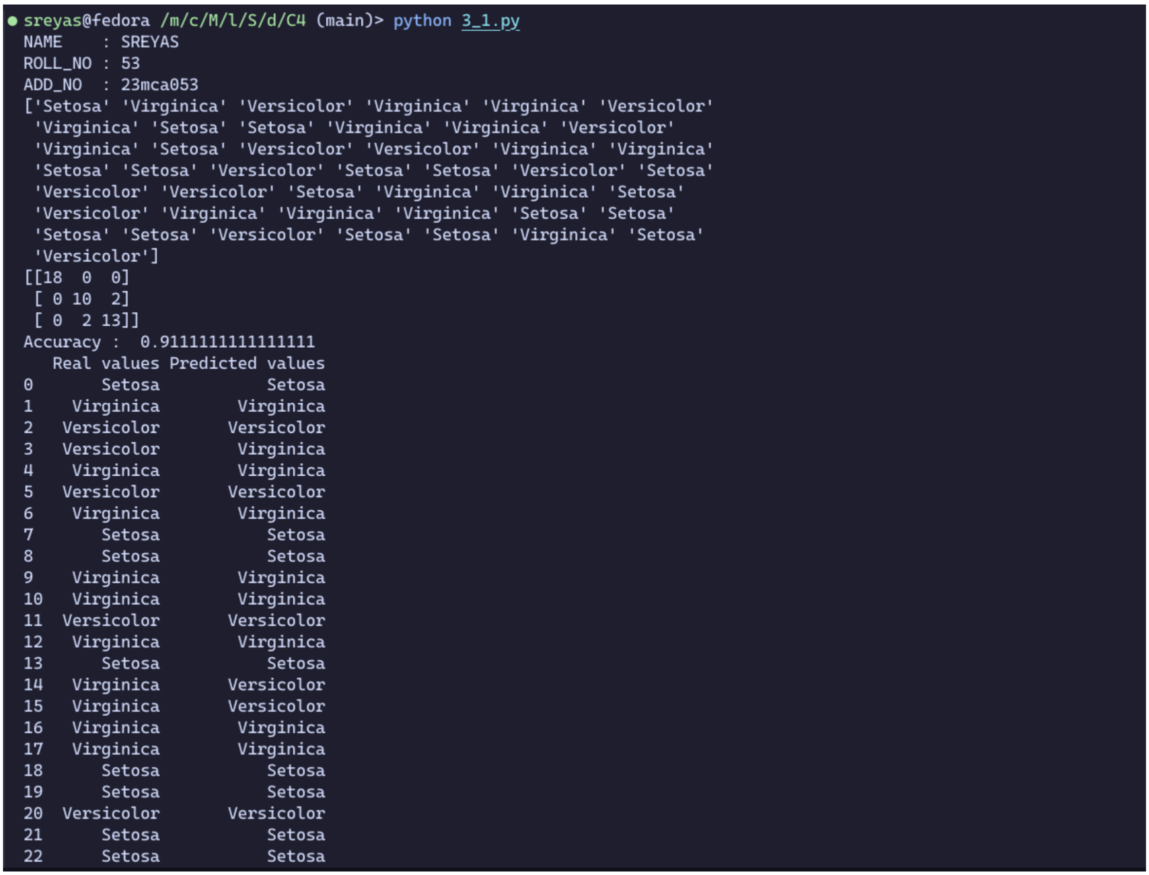
- **Find out the accuracy level w.r.t to each algorithm**
- **Display the no:of mislabeled classification from test data set**
- **List out the class labels of the mismatching records**

**I. Gaussian**

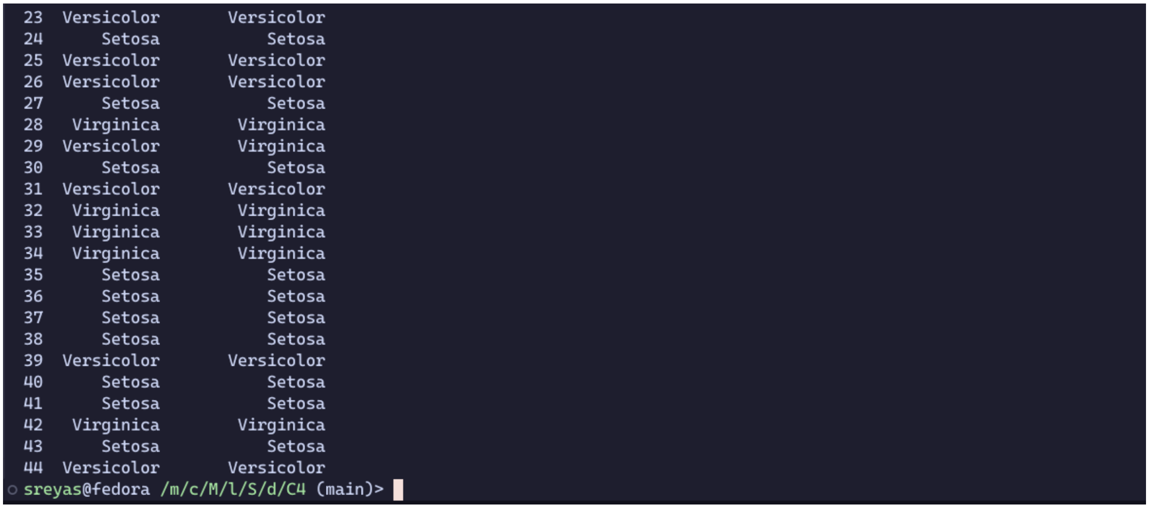
**CODE:**

```
iprint("NAME    : SREYAS")
print("ROLL_NO : 53")
print("ADD_NO  : 23mca053")


import pandas as pd
dataset=pd.read_csv('iris.csv')
x=dataset.iloc[:,:4].values
y=dataset['variety'].values
dataset.head(5)
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
df=pd.DataFrame({'Real values':y_test,'Predicted values':y_pred})
print(df)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C4 (main)> python 3_1.py
NAME     : SREYAS
ROLL_NO  : 53
ADD_NO   : 23mca053
['Setosa' 'Virginica' 'Versicolor' 'Virginica' 'Virginica' 'Versicolor'
 'Virginica' 'Setosa' 'Setosa' 'Virginica' 'Virginica' 'Versicolor'
 'Virginica' 'Setosa' 'Versicolor' 'Versicolor' 'Virginica' 'Virginica'
 'Setosa' 'Setosa' 'Versicolor' 'Setosa' 'Setosa' 'Versicolor' 'Setosa'
 'Versicolor' 'Versicolor' 'Setosa' 'Virginica' 'Virginica' 'Setosa'
 'Versicolor' 'Virginica' 'Virginica' 'Virginica' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Versicolor' 'Setosa' 'Setosa' 'Virginica' 'Setosa'
 'Versicolor']
[[18  0  0]
 [ 0 10  2]
 [ 0  2 13]]
Accuracy :  0.9111111111111111
   Real values Predicted values
0       Setosa           Setosa
1    Virginica        Virginica
2   Versicolor       Versicolor
3   Versicolor        Virginica
4    Virginica        Virginica
5   Versicolor       Versicolor
6    Virginica        Virginica
7       Setosa           Setosa
8       Setosa           Setosa
9    Virginica        Virginica
10   Virginica        Virginica
11  Versicolor       Versicolor
12   Virginica        Virginica
13      Setosa           Setosa
14   Virginica       Versicolor
15   Virginica       Versicolor
16   Virginica        Virginica
17   Virginica        Virginica
18      Setosa           Setosa
19      Setosa           Setosa
20  Versicolor       Versicolor
21      Setosa           Setosa
22      Setosa           Setosa
```



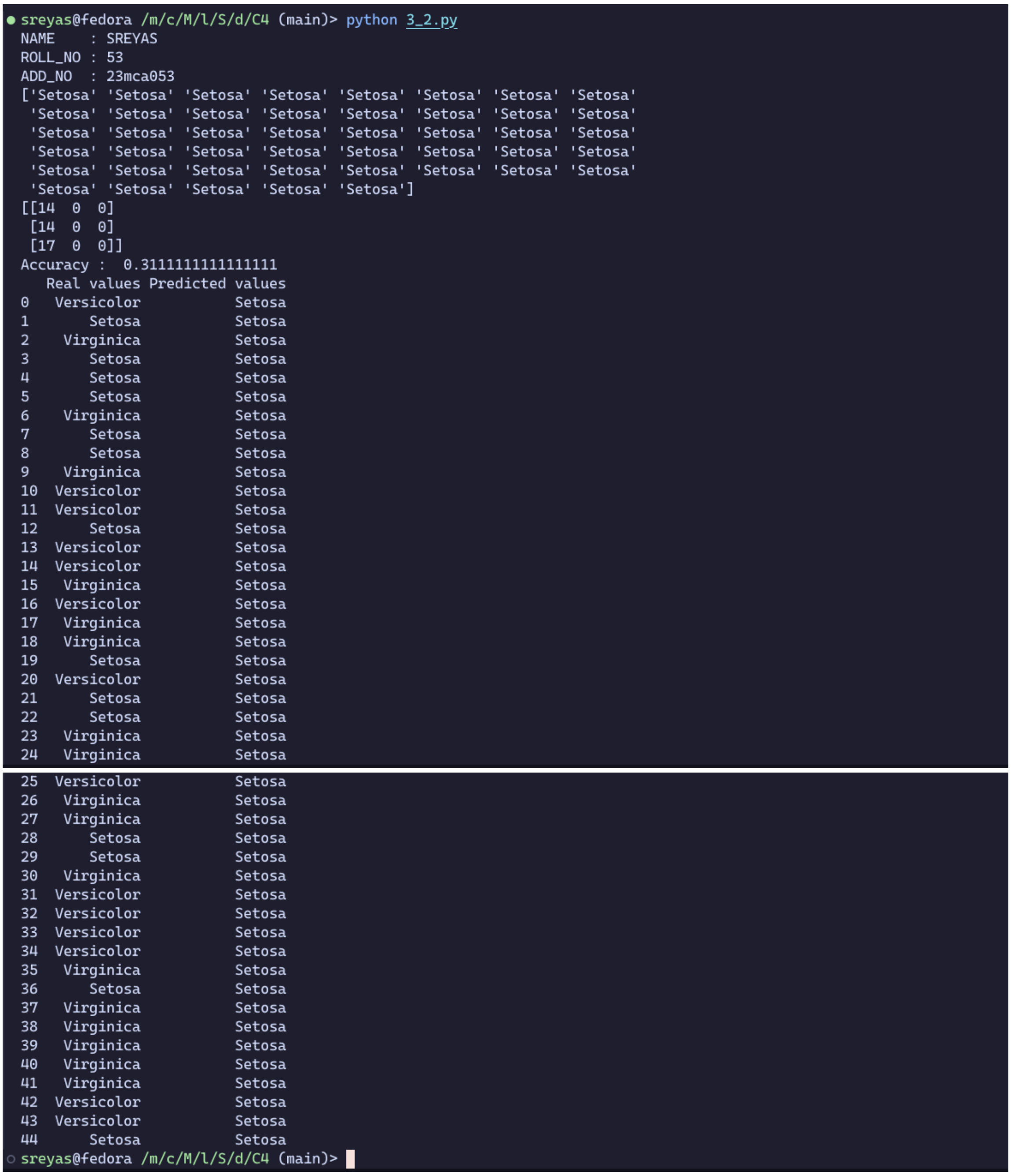
```
23  Versicolor       Versicolor
24      Setosa           Setosa
25  Versicolor       Versicolor
26  Versicolor       Versicolor
27      Setosa           Setosa
28   Virginica        Virginica
29  Versicolor        Virginica
30      Setosa           Setosa
31  Versicolor       Versicolor
32   Virginica        Virginica
33   Virginica        Virginica
34   Virginica        Virginica
35      Setosa           Setosa
36      Setosa           Setosa
37      Setosa           Setosa
38      Setosa           Setosa
39  Versicolor       Versicolor
40      Setosa           Setosa
41      Setosa           Setosa
42   Virginica        Virginica
43      Setosa           Setosa
44  Versicolor       Versicolor
○ sreyas@fedora /m/c/M/l/S/d/C4 (main)>
```

## II.Bernoulli

## CODE:

```
print("NAME    : SREYAS")
print("ROLL_NO : 53")
print("ADD_NO  : 23mca053")

import pandas as pd
dataset=pd.read_csv('iris.csv')
x=dataset.iloc[:,:4].values
y=dataset['variety'].values
dataset.head(5)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
from sklearn.naive_bayes import BernoulliNB
classifier=BernoulliNB()
classifier.fit(x_train,y_train)
y_pred=classifier.predict(x_test)
print(y_pred)
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy : ",accuracy_score(y_test,y_pred))
df=pd.DataFrame({'Real values':y_test,'Predicted values':y_pred})
print(df)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C4 (main)> python 3_2.py
NAME    : SREYAS
ROLL_NO : 53
ADD_NO  : 23mca053
['Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa'
 'Setosa' 'Setosa' 'Setosa' 'Setosa' 'Setosa']
[[14  0  0]
 [14  0  0]
 [17  0  0]]
Accuracy :  0.3111111111111111
   Real values Predicted values
0   Versicolor           Setosa
1       Setosa           Setosa
2    Virginica           Setosa
3       Setosa           Setosa
4       Setosa           Setosa
5       Setosa           Setosa
6    Virginica           Setosa
7       Setosa           Setosa
8       Setosa           Setosa
9    Virginica           Setosa
10  Versicolor           Setosa
11  Versicolor           Setosa
12      Setosa           Setosa
13  Versicolor           Setosa
14  Versicolor           Setosa
15   Virginica           Setosa
16  Versicolor           Setosa
17   Virginica           Setosa
18   Virginica           Setosa
19      Setosa           Setosa
20  Versicolor           Setosa
21      Setosa           Setosa
22      Setosa           Setosa
23   Virginica           Setosa
24   Virginica           Setosa
25  Versicolor           Setosa
26   Virginica           Setosa
27   Virginica           Setosa
28      Setosa           Setosa
29      Setosa           Setosa
30   Virginica           Setosa
31  Versicolor           Setosa
32  Versicolor           Setosa
33  Versicolor           Setosa
34  Versicolor           Setosa
35   Virginica           Setosa
36      Setosa           Setosa
37   Virginica           Setosa
38   Virginica           Setosa
39   Virginica           Setosa
40   Virginica           Setosa
41   Virginica           Setosa
42  Versicolor           Setosa
43  Versicolor           Setosa
44      Setosa           Setosa
○ sreyas@fedora /m/c/M/l/S/d/C4 (main)>
```

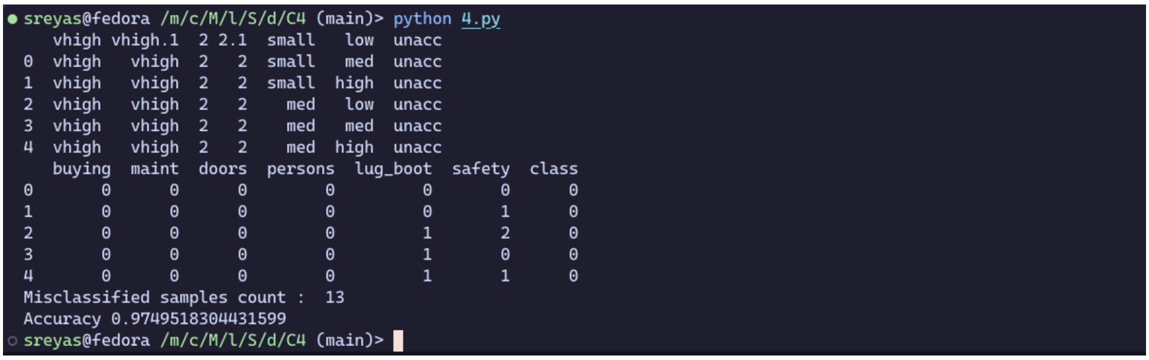
## 28. Use car details CSV file and implement decision tree algorithm

- **Find out the accuracy level.**
- **Display the no: of mislabelled classification from test data set**
- **List out the class labels of the mismatching records**

**CODE:**

```
import pandas as pd
data = pd.read_csv('car.csv')
print(data.head())
data.columns = ['buying','maint','doors','persons','lug_boot','safety','class']
data['class'],_ = pd.factorize(data['class'])
data['buying'],_ = pd.factorize(data['buying'])
data['maint'],_ = pd.factorize(data['maint'])
data['doors'],_ = pd.factorize(data['doors'])
data['persons'],_ = pd.factorize(data['persons'])
data['lug_boot'],_ = pd.factorize(data['lug_boot'])
data['safety'],_ = pd.factorize(data['safety'])

print(data.head())
x = data.iloc[:, :-1]
y = data.iloc[:, -1]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.3)
from sklearn.tree import DecisionTreeClassifier
tree1 = DecisionTreeClassifier()
tree1.fit(x_train,y_train)
y_pred = tree1.predict(x_test)
#how did our model perform?

count_missclassified = (y_test != y_pred).sum()
print('Misclassified samples count : ',count_missclassified)
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test,y_pred)
print("Accuracy",accuracy)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C4 (main)> python 4.py
     vhigh vhigh.1  2 2.1  small   low  unacc
  0  vhigh   vhigh  2   2  small   med  unacc
  1  vhigh   vhigh  2   2  small  high  unacc
  2  vhigh   vhigh  2   2    med   low  unacc
  3  vhigh   vhigh  2   2    med   med  unacc
  4  vhigh   vhigh  2   2    med  high  unacc
     buying  maint  doors  persons  lug_boot  safety  class
  0       0      0      0        0         0       0      0
  1       0      0      0        0         0       1      0
  2       0      0      0        0         1       2      0
  3       0      0      0        0         1       0      0
  4       0      0      0        0         1       1      0
  Misclassified samples count :  13
  Accuracy 0.9749518304431599
○ sreyas@fedora /m/c/M/l/S/d/C4 (main)> 
```

## 29. Implement Simple and multiple linear regression for the data sets ‘student_score.csv’ and ‘company_data .csv’ respectively

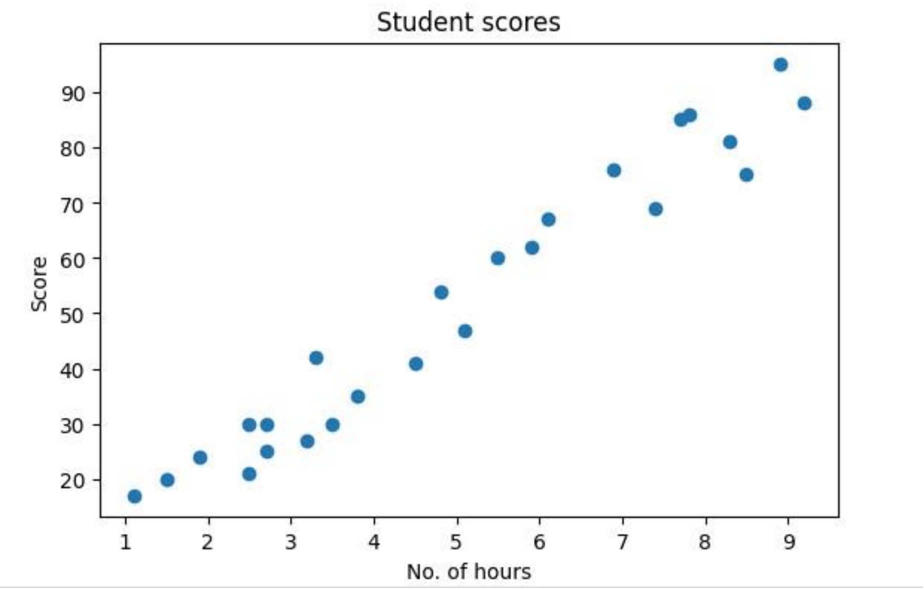
### Single linear Regression

### CODE :

```
import numpy as np
import pandas as pd
import sklearn as sk
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
student = pd.read_csv("student_scores.csv")
print(student.head())
student.describe()
student.info()
x_axis = student.iloc[:,0]
y_axis = student.iloc[:,1]
plt.scatter(x_axis, y_axis)
plt.xlabel("no.of hours")
plt.ylabel("scores")
plt.show()
x = student.iloc[:, :-1]
y = student.iloc[:, 1]
print("x values", x)
print("y values", y)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
print(x_train)
regression = LinearRegression()
regression.fit(x_train, y_train)
print("intercept : ", regression.intercept_)
print("co-efficient : ", regression.coef_)
```

```
y_pred = regression.predict(x_test)
for (i, j) in zip(y_test, y_pred):
    if(i!=j):
        print("actual value : ", i, "\npredicted value : ", j)
print("mislabeld : ", (y_test != y_pred).sum())
from sklearn.metrics import mean_squared_error, mean_absolute_error
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print("mean absolute error : ", mae)
print("mean square error : ", mse)
print("root mean square error : ", rmse)
```

**OUTPUT :**

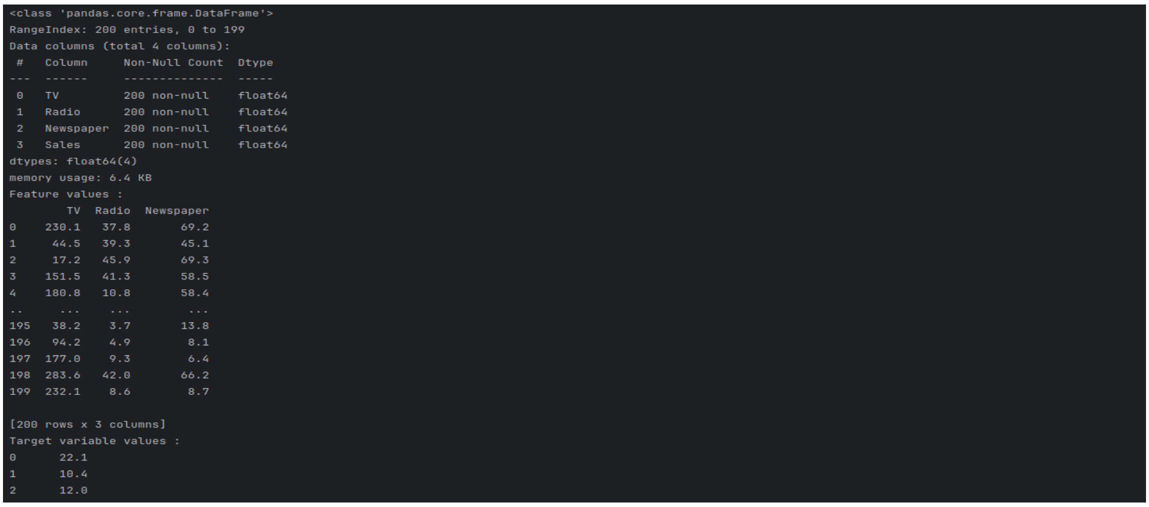


## Multiple linear regression

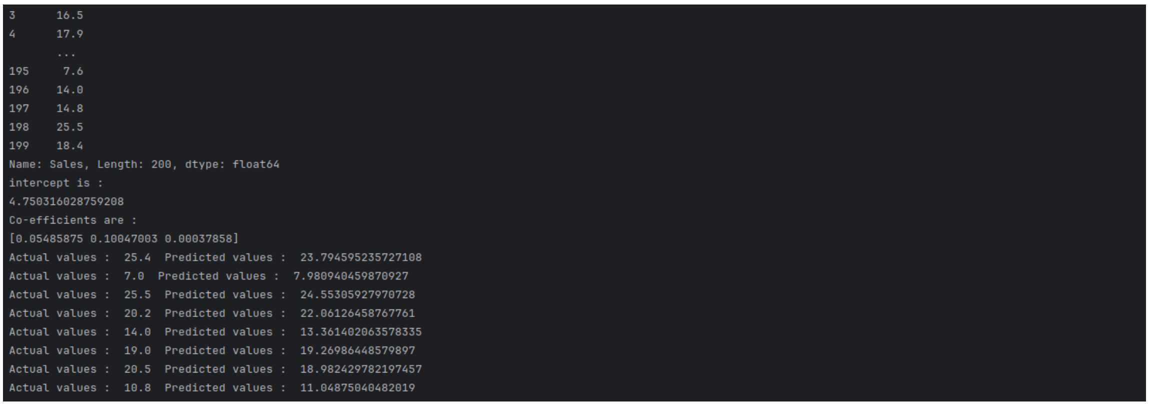
## CODE :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
advertising =pd.read_csv('Company_data.csv')
advertising.head()
advertising.describe()
advertising.info()
print("Feature values : ")
x = advertising.iloc[:, :-1]
print(x)
print("Target variable values : ") y = advertising.iloc[:, -1] print(y)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.3) from
sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(x_train,y_train)
print("intercept is : ")
print(regressor.intercept_)
print("Co-efficients are : ")
print(regressor.coef_)
y_pred = regressor.predict(x_test)
for(i,j) in zip(y_test,y_pred):
          if i!=j:
        print("Actual values : ",i," Predicted values : ",j)
print("Number of mislabeled points from test data set : ",(y_test != y_pred).sum())
from sklearn import metrics print("Mean Absolute error :",
metrics.mean_absolute_error(y_test,y_pred))
print("Mean Squared error :", metrics.mean_squared_error(y_test,y_pred))
print("Root Mean Squared error :", np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

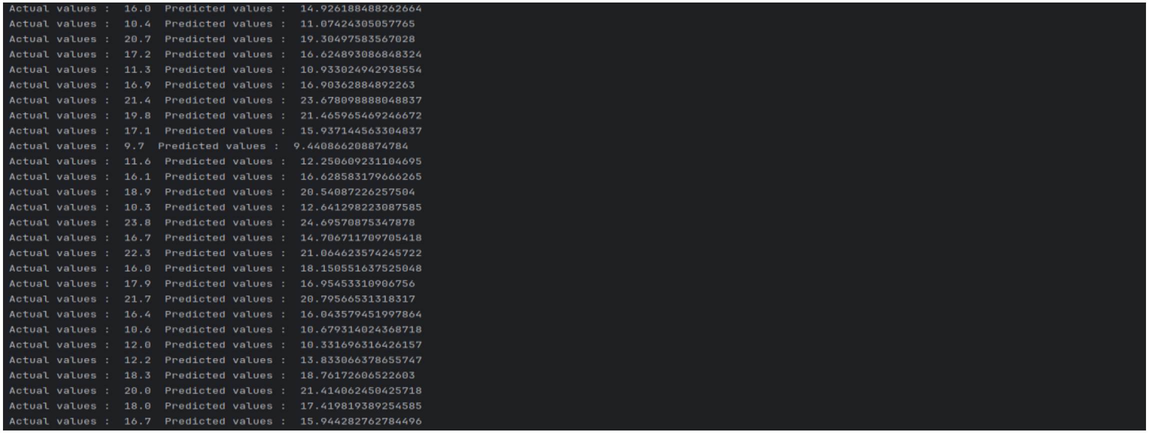
## OUTPUT :



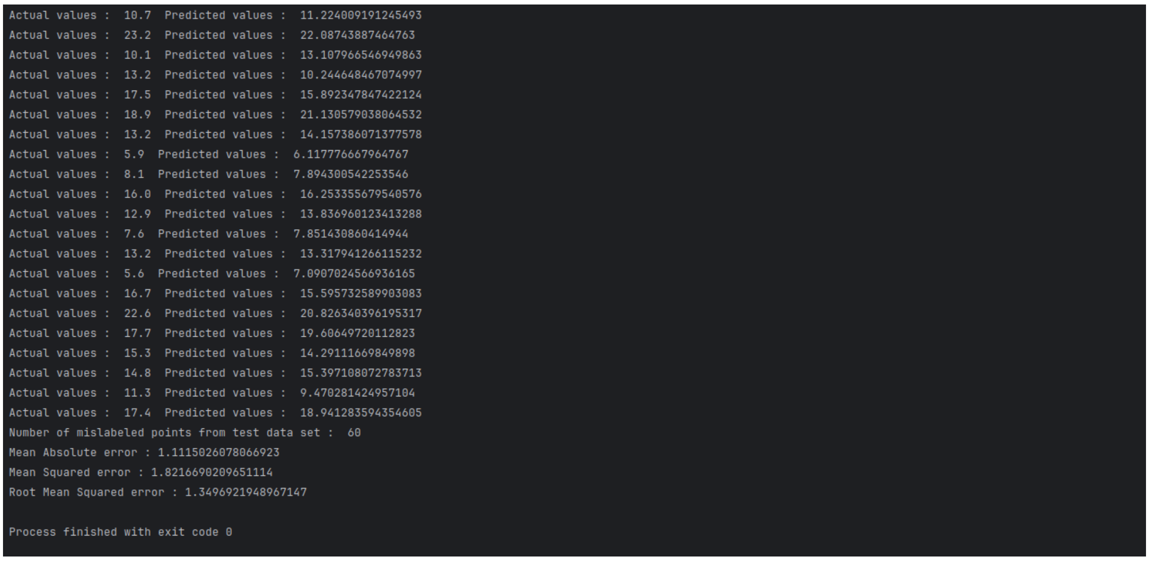
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
Feature values :
        TV  Radio  Newspaper
0    230.1   37.8       69.2
1     44.5   39.3       45.1
2     17.2   45.9       69.3
3    151.5   41.3       58.5
4    180.8   10.8       58.4
..     ...    ...        ...
195   38.2    3.7       13.8
196   94.2    4.9        8.1
197  177.0    9.3        6.4
198  283.6   42.0       66.2
199  232.1    8.6        8.7

[200 rows x 3 columns]
Target variable values :
0      22.1
1      10.4
2      12.0
```



```
3      16.5
4      17.9
       ...
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
intercept is :
4.750316028759208
Co-efficients are :
[0.05485875 0.10047003 0.00037858]
Actual values :  25.4  Predicted values :  23.794595235727108
Actual values :  7.0  Predicted values :  7.980940459870927
Actual values :  25.5  Predicted values :  24.55305927970728
Actual values :  20.2  Predicted values :  22.06126458767761
Actual values :  14.0  Predicted values :  13.361402063578335
Actual values :  19.0  Predicted values :  19.26986448579897
Actual values :  20.5  Predicted values :  18.982429782197457
Actual values :  10.8  Predicted values :  11.04875040482019
```



```
Actual values :  16.0  Predicted values :  14.926188488262664
Actual values :  10.4  Predicted values :  11.07424305057765
Actual values :  20.7  Predicted values :  19.30497583567028
Actual values :  17.2  Predicted values :  16.624893086848324
Actual values :  11.3  Predicted values :  10.933024942938554
Actual values :  16.9  Predicted values :  16.90362884892263
Actual values :  21.4  Predicted values :  23.678098888048837
Actual values :  19.8  Predicted values :  21.465965469246672
Actual values :  17.1  Predicted values :  15.937144563304837
Actual values :  9.7  Predicted values :  9.440866208874784
Actual values :  11.6  Predicted values :  12.250609231104695
Actual values :  16.1  Predicted values :  16.628583179666265
Actual values :  18.9  Predicted values :  20.54087226257504
Actual values :  10.3  Predicted values :  12.641298223087585
Actual values :  23.8  Predicted values :  24.69570875347878
Actual values :  16.7  Predicted values :  14.706711709705418
Actual values :  22.3  Predicted values :  21.064623574245722
Actual values :  16.0  Predicted values :  18.150551637525048
Actual values :  17.9  Predicted values :  16.95453310906756
Actual values :  21.7  Predicted values :  20.79566531318317
Actual values :  16.4  Predicted values :  16.043579451997864
Actual values :  10.6  Predicted values :  10.679314024368718
Actual values :  12.0  Predicted values :  10.331696316426157
Actual values :  12.2  Predicted values :  13.833066378655747
Actual values :  18.3  Predicted values :  18.76172606522603
Actual values :  20.0  Predicted values :  21.414062450425718
Actual values :  18.0  Predicted values :  17.419819389254585
Actual values :  16.7  Predicted values :  15.944282762784496
```



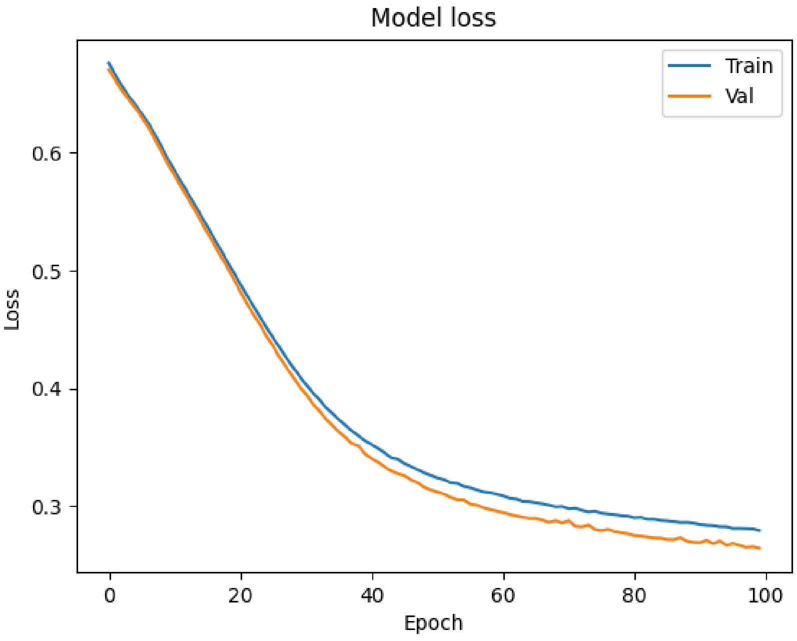
```
Actual values :  10.7  Predicted values :  11.224009191245493
Actual values :  23.2  Predicted values :  22.08743887464763
Actual values :  10.1  Predicted values :  13.107966546949863
Actual values :  13.2  Predicted values :  10.244648467074997
Actual values :  17.5  Predicted values :  15.892347847422124
Actual values :  18.9  Predicted values :  21.130579038064532
Actual values :  13.2  Predicted values :  14.157386071377578
Actual values :  5.9  Predicted values :  6.117776667964767
Actual values :  8.1  Predicted values :  7.894300542253546
Actual values :  16.0  Predicted values :  16.253355679540576
Actual values :  12.9  Predicted values :  13.836960123413288
Actual values :  7.6  Predicted values :  7.851430860414944
Actual values :  13.2  Predicted values :  13.317941266115232
Actual values :  5.6  Predicted values :  7.0907024566936165
Actual values :  16.7  Predicted values :  15.595732589903083
Actual values :  22.6  Predicted values :  20.826340396195317
Actual values :  17.7  Predicted values :  19.60649720112823
Actual values :  15.3  Predicted values :  14.29111669849898
Actual values :  14.8  Predicted values :  15.397108072783713
Actual values :  11.3  Predicted values :  9.470281424957104
Actual values :  17.4  Predicted values :  18.941283594354605
Number of mislabeled points from test data set :  60
Mean Absolute error : 1.1115026078066923
Mean Squared error : 1.8216690209651114
Root Mean Squared error : 1.3496921948967147

Process finished with exit code 0
```

## 30. Create a neural network for the given ‘houseprice.csv’ to predict the weather price of the house is above or below median value or not

**CODE:**

```
import tensorflow as tf

import keras import pandas

import sklearn import matplotlib

import pandas as pd

df = pd.read_csv('housepricedata.csv')

print(df.head()) dataset = df.values X = dataset[:,0:10]

Y = dataset[:,10]

from sklearn import preprocessing min_max_scaler = preprocessing.MinMaxScaler()
X_scale = min_max_scaler.fit_transform(X)

print(X_scale)

from sklearn.model_selection import train_test_split

X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale,

Y, test_size=0.3)

X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test,
test_size=0.5)

print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape,
Y_test.shape)

from keras.models import Sequential from keras.layers import Dense

model = Sequential([Dense(32, activation='relu', input_shape=(10,)), Dense(32,
activation='relu'),Dense(1, activation='sigmoid'),]) model.compile(optimizer='sgd',
loss='binary_crossentropy', metrics=['accuracy'])


hist = model.fit(X_train, Y_train, batch_size=32, epochs=100,
validation_data=(X_val, Y_val))
model.evaluate(X_test, Y_test)[1]
import matplotlib.pyplot as plt
 plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Val'], loc='upper right')
plt.savefig("./6.png")
plt.show()
```

**OUTPUT:**

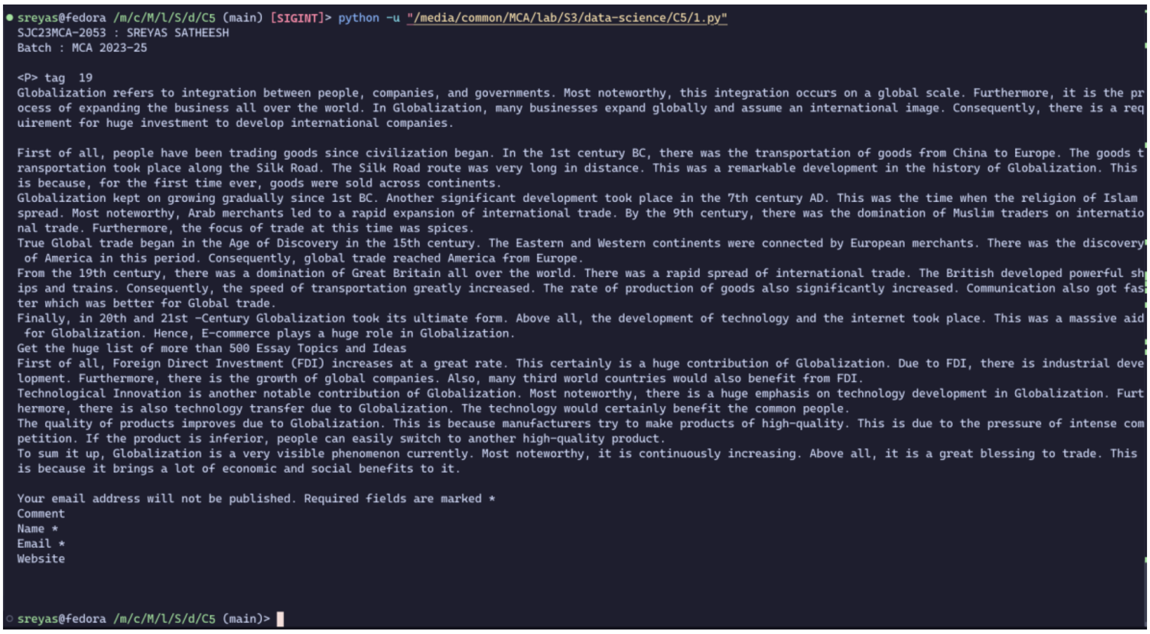


### 31. Write a program to implement a simple web crawler using Python. Extract and display the content of the page(p tag)

**CODE:**

```
import requests
from bs4 import BeautifulSoup
print("SJC23MCA-2053 : SREYAS SATHEESH")
print("Batch : MCA 2023-25\n")
def getdata(url):
    r = requests.get(url)
    return r.content
htmldata = getdata("https://www.toppr.com/guides/essays/globalization-essay/")
soup = BeautifulSoup(htmldata, 'html.parser')
data = ''
pr = len(soup.find_all('p'))
print("<P> tag ", pr)
for data in soup.find_all('p'):
    print(data.get_text())
```

**OUTPUT:**



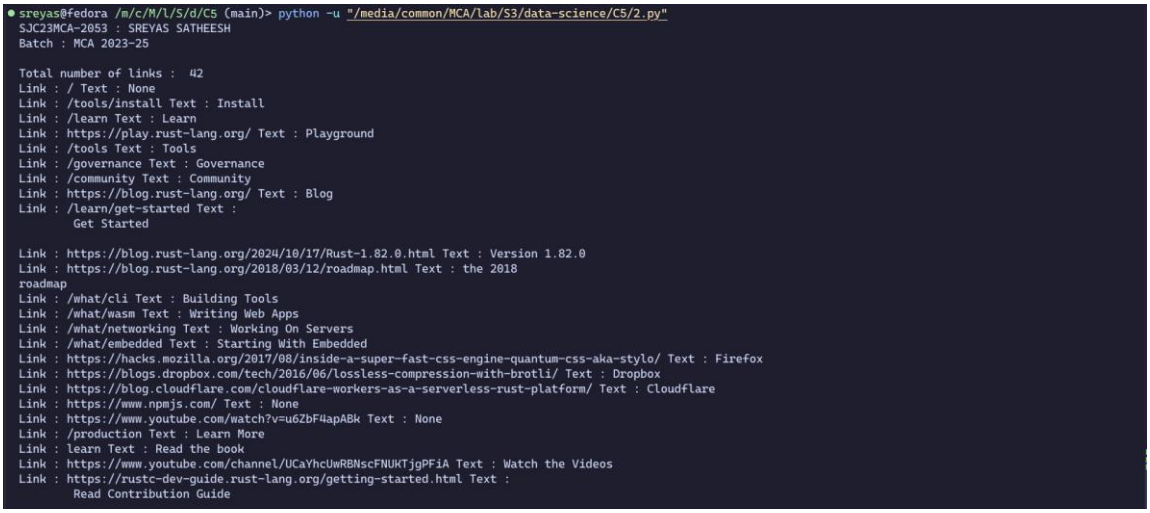
```
● sreyas@fedora /m/c/M/l/S/d/C5 (main) [SIGINT]> python -u "/media/common/MCA/lab/S3/data-science/C5/1.py"
 SJC23MCA-2053 : SREYAS SATHEESH
 Batch : MCA 2023-25

 <P> tag  19
 Globalization refers to integration between people, companies, and governments. Most noteworthy, this integration occurs on a global scale. Furthermore, it is the pr
 ocess of expanding the business all over the world. In Globalization, many businesses expand globally and assume an international image. Consequently, there is a req
 uirement for huge investment to develop international companies.

 First of all, people have been trading goods since civilization began. In the 1st century BC, there was the transportation of goods from China to Europe. The goods t
 ransportation took place along the Silk Road. The Silk Road route was very long in distance. This was a remarkable development in the history of Globalization. This
 is because, for the first time ever, goods were sold across continents.
 Globalization kept on growing gradually since 1st BC. Another significant development took place in the 7th century AD. This was the time when the religion of Islam
 spread. Most noteworthy, Arab merchants led to a rapid expansion of international trade. By the 9th century, there was the domination of Muslim traders on internatio
 nal trade. Furthermore, the focus of trade at this time was spices.
 True Global trade began in the Age of Discovery in the 15th century. The Eastern and Western continents were connected by European merchants. There was the discovery
  of America in this period. Consequently, global trade reached America from Europe.
 From the 19th century, there was a domination of Great Britain all over the world. There was a rapid spread of international trade. The British developed powerful sh
 ips and trains. Consequently, the speed of transportation greatly increased. The rate of production of goods also significantly increased. Communication also got fas
 ter which was better for Global trade.
 Finally, in 20th and 21st -Century Globalization took its ultimate form. Above all, the development of technology and the internet took place. This was a massive aid
  for Globalization. Hence, E-commerce plays a huge role in Globalization.
 Get the huge list of more than 500 Essay Topics and Ideas
 First of all, Foreign Direct Investment (FDI) increases at a great rate. This certainly is a huge contribution of Globalization. Due to FDI, there is industrial deve
 lopment. Furthermore, there is the growth of global companies. Also, many third world countries would also benefit from FDI.
 Technological Innovation is another notable contribution of Globalization. Most noteworthy, there is a huge emphasis on technology development in Globalization. Furt
 hermore, there is also technology transfer due to Globalization. The technology would certainly benefit the common people.
 The quality of products improves due to Globalization. This is because manufacturers try to make products of high-quality. This is due to the pressure of intense com
 petition. If the product is inferior, people can easily switch to another high-quality product.
 To sum it up, Globalization is a very visible phenomenon currently. Most noteworthy, it is continuously increasing. Above all, it is a great blessing to trade. This
 is because it brings a lot of economic and social benefits to it.

 Your email address will not be published. Required fields are marked *
 Comment
 Name *
 Email *
 Website


○ sreyas@fedora /m/c/M/l/S/d/C5 (main>
```

**32. Write a program to implement a simple web crawler using Python. Display all hyperlinks in the page**

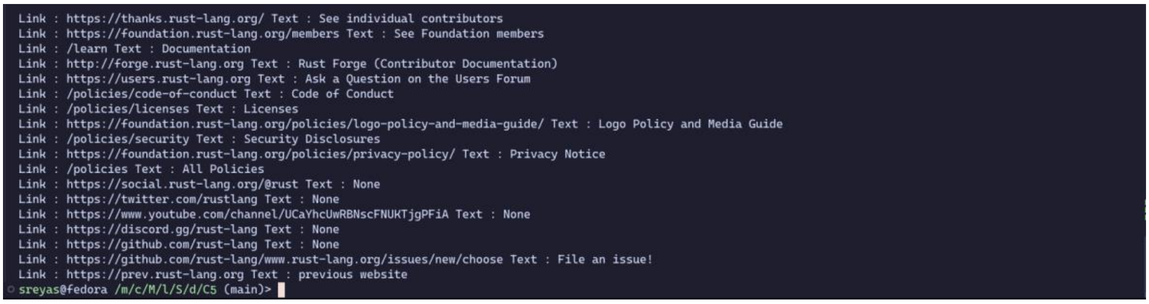
**CODE:**

```
import requests
from bs4 import BeautifulSoup

def getdata(url):
    r = requests.get(url)
    return r.content
htmldata = getdata("https://www.rust-lang.org/")
soup = BeautifulSoup(htmldata,'html.parser')

print("SJC23MCA-2053 : SREYAS SATHEESH")
print("Batch : MCA 2023-25\n")

links = soup.find_all("a")
print("Total number of links : ",len(links))
for link in links:
    if link.get("href") != "":
        print("Link :",link.get("href"),"Text :",link.string)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C5 (main)> python -u "/media/common/MCA/lab/S3/data-science/C5/2.py"
  SJC23MCA-2053 : SREYAS SATHEESH
  Batch : MCA 2023-25

  Total number of links :  42
  Link : / Text : None
  Link : /tools/install Text : Install
  Link : /learn Text : Learn
  Link : https://play.rust-lang.org/ Text : Playground
  Link : /tools Text : Tools
  Link : /governance Text : Governance
  Link : /community Text : Community
  Link : https://blog.rust-lang.org/ Text : Blog
  Link : /learn/get-started Text :
          Get Started

  Link : https://blog.rust-lang.org/2024/10/17/Rust-1.82.0.html Text : Version 1.82.0
  Link : https://blog.rust-lang.org/2018/03/12/roadmap.html Text : the 2018
  roadmap
  Link : /what/cli Text : Building Tools
  Link : /what/wasm Text : Writing Web Apps
  Link : /what/networking Text : Working On Servers
  Link : /what/embedded Text : Starting With Embedded
  Link : https://hacks.mozilla.org/2017/08/inside-a-super-fast-css-engine-quantum-css-aka-stylo/ Text : Firefox
  Link : https://blogs.dropbox.com/tech/2016/06/lossless-compression-with-brotli/ Text : Dropbox
  Link : https://blog.cloudflare.com/cloudflare-workers-as-a-serverless-rust-platform/ Text : Cloudflare
  Link : https://www.npmjs.com/ Text : None
  Link : https://www.youtube.com/watch?v=u6ZbF4apABk Text : None
  Link : /production Text : Learn More
  Link : learn Text : Read the book
  Link : https://www.youtube.com/channel/UCaYhcUwRBNscFNUKTjgPFiA Text : Watch the Videos
  Link : https://rustc-dev-guide.rust-lang.org/getting-started.html Text :
          Read Contribution Guide
```



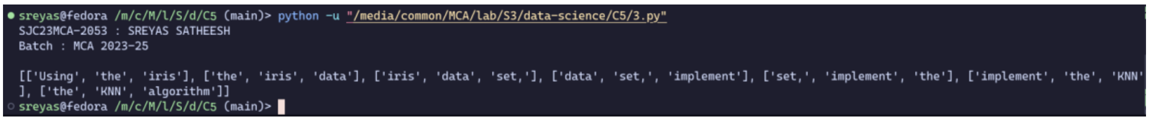
```
  Link : https://thanks.rust-lang.org/ Text : See individual contributors
  Link : https://foundation.rust-lang.org/members Text : See Foundation members
  Link : /learn Text : Documentation
  Link : http://forge.rust-lang.org Text : Rust Forge (Contributor Documentation)
  Link : https://users.rust-lang.org Text : Ask a Question on the Users Forum
  Link : /policies/code-of-conduct Text : Code of Conduct
  Link : /policies/licenses Text : Licenses
  Link : https://foundation.rust-lang.org/policies/logo-policy-and-media-guide/ Text : Logo Policy and Media Guide
  Link : /policies/security Text : Security Disclosures
  Link : https://foundation.rust-lang.org/policies/privacy-policy/ Text : Privacy Notice
  Link : /policies Text : All Policies
  Link : https://social.rust-lang.org/@rust Text : None
  Link : https://twitter.com/rustlang Text : None
  Link : https://www.youtube.com/channel/UCaYhcUwRBNscFNUKTjgPFiA Text : None
  Link : https://discord.gg/rust-lang Text : None
  Link : https://github.com/rust-lang Text : None
  Link : https://github.com/rust-lang/www.rust-lang.org/issues/new/choose Text : File an issue!
  Link : https://prev.rust-lang.org Text : previous website
○ sreyas@fedora /m/c/M/l/S/d/C5 (main)>
```

## 33. Program for Natural Language Processing which performs n-grams(without using library)

### CODE:

```
print("SJC23MCA-2053 : SREYAS SATHEESH")
print("Batch : MCA 2023-25\n")

def gen_ngrams(text, wordsToCombine):
    words = text.split()
    output = []
    for i in range(len(words)-wordsToCombine+1):
        output.append(words[i:i+wordsToCombine])
    return output

x = gen_ngrams(text='Using the iris data set, implement the KNN
algorithm',wordsToCombine=3)
print(x)
```

### OUTPUT:



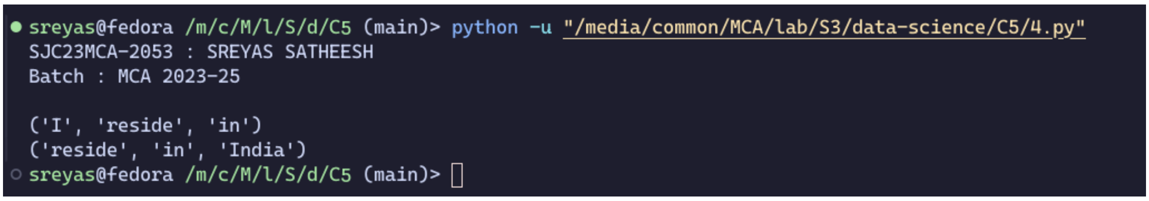
```
● sreyas@fedora /m/c/M/l/S/d/C5 (main)> python -u "/media/common/MCA/lab/S3/data-science/C5/3.py"
  SJC23MCA-2053 : SREYAS SATHEESH
  Batch : MCA 2023-25

  [['Using', 'the', 'iris'], ['the', 'iris', 'data'], ['iris', 'data', 'set,'], ['data', 'set,', 'implement'], ['set,', 'implement', 'the'], ['implement', 'the', 'KNN'
  ], ['the', 'KNN', 'algorithm']]
○ sreyas@fedora /m/c/M/l/S/d/C5 (main)>
```

### 34. Program for Natural Language Processing which performs n-grams(using nltk library)

**CODE:**

```
print("SJC23MCA-2053 : SREYAS SATHEESH")
print("Batch : MCA 2023-25\n")


from nltk import ngrams
sentence = 'I reside in India'
n = 3
trigrams = ngrams(sentence.split(),n)
for grams in trigrams:
   print(grams)
```

**OUTPUT:**



```
● sreyas@fedora /m/c/M/l/S/d/C5 (main)> python -u "/media/common/MCA/lab/S3/data-science/C5/4.py"
  SJC23MCA-2053 : SREYAS SATHEESH
  Batch : MCA 2023-25

  ('I', 'reside', 'in')
  ('reside', 'in', 'India')
○ sreyas@fedora /m/c/M/l/S/d/C5 (main)> 
```

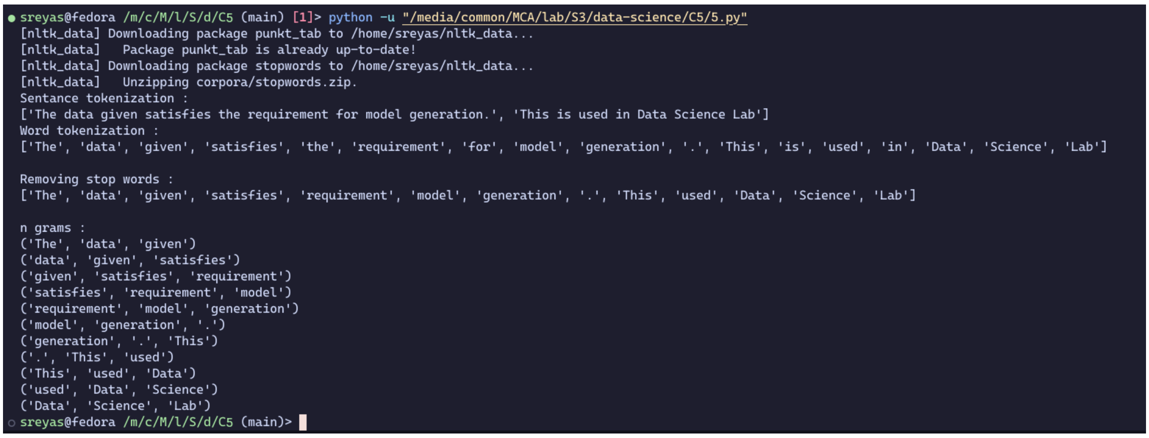
**35. For given text, perform the following Natural Language Processing tasks:**

- **perform word tokenization**
- **sentence tokenization**
- **Remove the stop words from the given text**
- **create n-grams**

**CODE:**

```
import nltk
from nltk import ngrams
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize,word_tokenize

nltk.download('punkt')

text1 = "The data given satisfies the requirement for model generation. This is used in
Data Science Lab"
print("Sentance tokenization : ")
print(sent_tokenize(text1))
print("Word tokenization : ")
print(word_tokenize(text1))

text = word_tokenize(text1)
text2 = [word for word in text if word not in stopwords.words('english')]
print("")
print("Removing stop words : ")
print(text2)
print("")
print("n grams : ")
unigrams = ngrams(text2,3)

for grams in unigrams:
    print(grams)
```

## OUTPUT:



```
● sreyas@fedora /m/c/M/l/S/d/C5 (main) [1]> python -u "/media/common/MCA/lab/S3/data-science/C5/5.py"
  [nltk_data] Downloading package punkt_tab to /home/sreyas/nltk_data...
  [nltk_data]   Package punkt_tab is already up-to-date!
  [nltk_data] Downloading package stopwords to /home/sreyas/nltk_data...
  [nltk_data]   Unzipping corpora/stopwords.zip.
  Sentance tokenization :
  ['The data given satisfies the requirement for model generation.', 'This is used in Data Science Lab']
  Word tokenization :
  ['The', 'data', 'given', 'satisfies', 'the', 'requirement', 'for', 'model', 'generation', '.', 'This', 'is', 'used', 'in', 'Data', 'Science', 'Lab']

  Removing stop words :
  ['The', 'data', 'given', 'satisfies', 'requirement', 'model', 'generation', '.', 'This', 'used', 'Data', 'Science', 'Lab']

  n grams :
  ('The', 'data', 'given')
  ('data', 'given', 'satisfies')
  ('given', 'satisfies', 'requirement')
  ('satisfies', 'requirement', 'model')
  ('requirement', 'model', 'generation')
  ('model', 'generation', '.')
  ('generation', '.', 'This')
  ('.', 'This', 'used')
  ('This', 'used', 'Data')
  ('used', 'Data', 'Science')
  ('Data', 'Science', 'Lab')
○ sreyas@fedora /m/c/M/l/S/d/C5 (main)>
```

**36. Given dataset contains 200 records and five columns, two of which describe the customer's annual income and spending score. The latter is a value from 0 to 100. The higher the number, the more this customer has spent with the company in the past:**

**Using k means clustering creates 6 clusters of customers based on their spending pattern.**

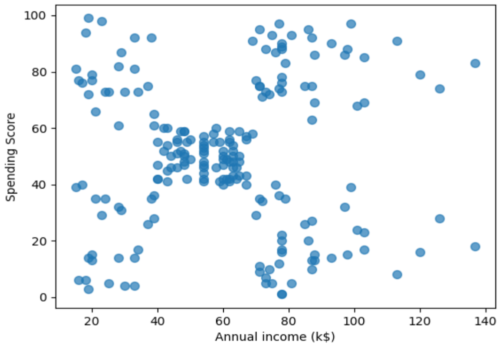
- **Visualize the same in a scatter plot with each cluster in a different color scheme.**
- **Display the cluster labels of each point.(print cluster indexes)**
- **Display the cluster centers.**
- **Use different values of K and visualize the same using scatter plot**

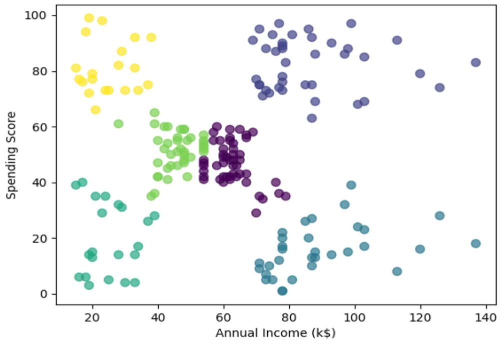
**CODE:**

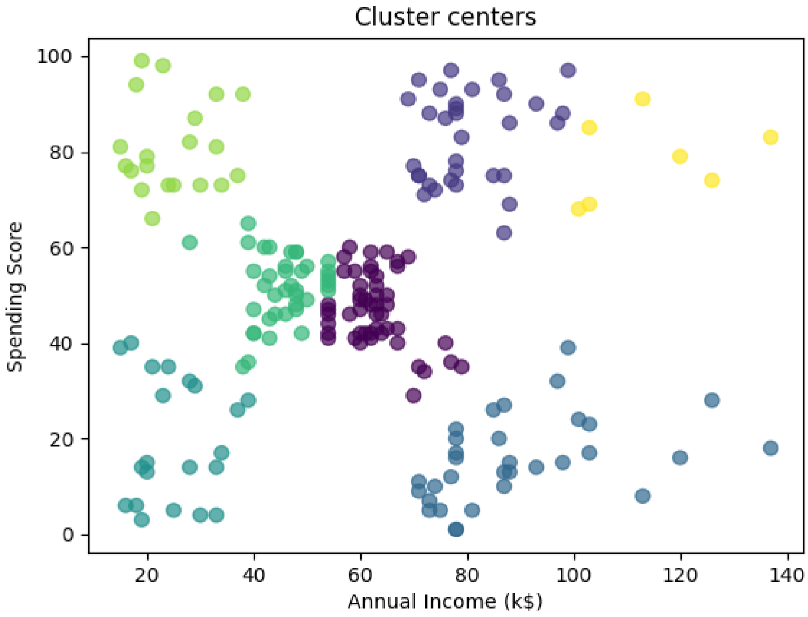
```
import pandas as pd


customer = pd.read_csv('customer_data.csv')

customer.head()

import matplotlib.pyplot as plt


point = customer.iloc[:,3:5].values

x = point[:,0]

y = point[:,1]

plt.scatter(x,y,s=50,alpha=0.7)

plt.xlabel('Annual income (k$)')

plt.ylabel('Spending Score')

plt.show()


from sklearn.cluster import KMeans


kmeans = KMeans(n_clusters=6,random_state=0)
```

```
kmeans.fit(point)
predicted_cluster_indexes = kmeans.predict(point)

plt.scatter(x,y,c=predicted_cluster_indexes,s=50,alpha=0.7,cmap='viridis')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.show()

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=7,random_state=0)
kmeans.fit(point)
predicted_cluster_indexes = kmeans.predict(point)
plt.scatter(x,y,c=predicted_cluster_indexes,s=50,alpha=0.7,cmap='viridis')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score')
plt.title('Cluster centers')
plt.show()
```

## OUTPUT:


Spending Score
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual income (k$)


Spending Score
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income (k$)


Cluster centers
Spending Score
100
80
60
40
20
0
20
40
60
80
100
120
140
Annual Income (k$)