**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:**



**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]]

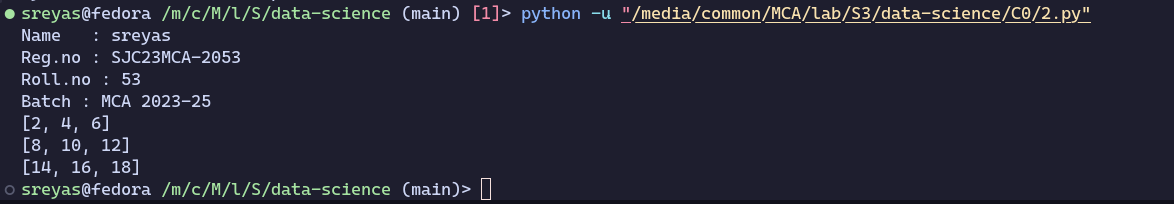
Y = [[1,2,3,],[4,5,6],[7,8,9]]

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:**



**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:**



**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:**



**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:**



**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)

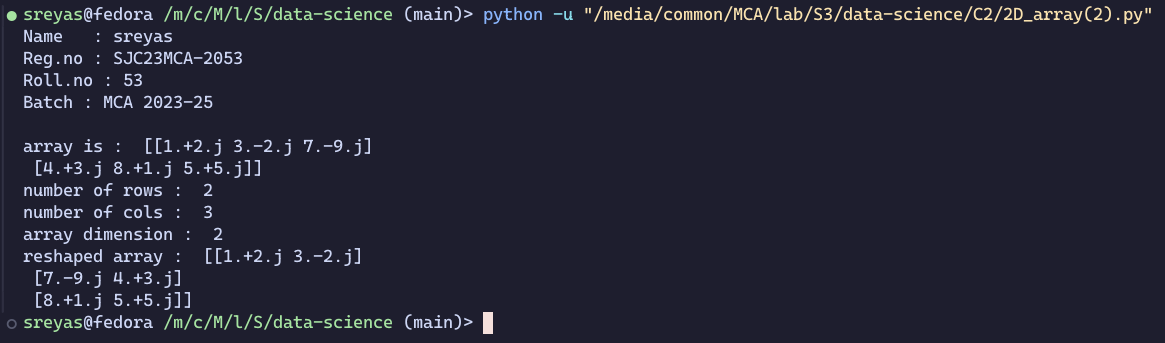
dim = arr.ndim

print("array dimension : ", dim)

reshaped\_arr = arr.reshape(3, 2)

print("reshaped array : ", reshaped\_arr)

**OUTPUT:**



**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]

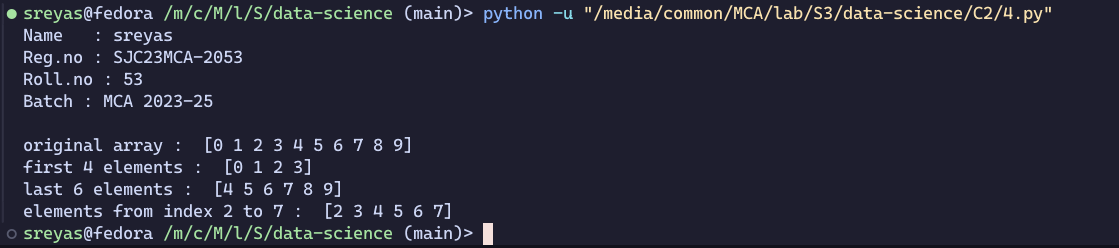
print("original array : ", arr)

print("first 4 elements : ", first\_4)

print("last 6 elements : ", last\_6)

print("elements from index 2 to 7 : ", ele\_2\_to\_7)

**OUTPUT:**



**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)

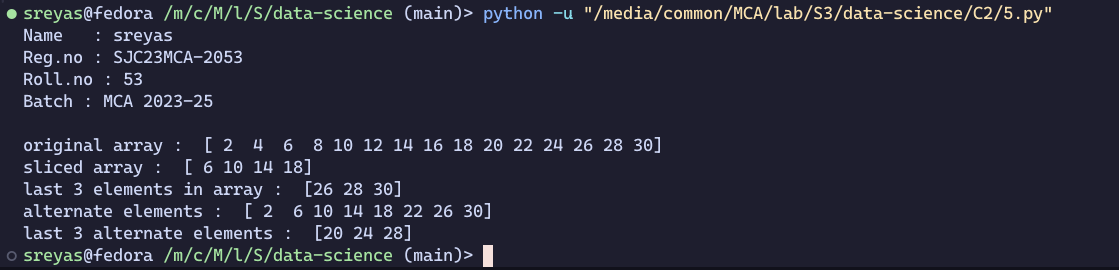
print("sliced array : ", slice\_arr)

print("last 3 elements in array : ", last\_3)

print("alternate elements : ", alternate\_ele)

print("last 3 alternate elements : ", last\_3\_alternate)

**OUTPUT:**



**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])

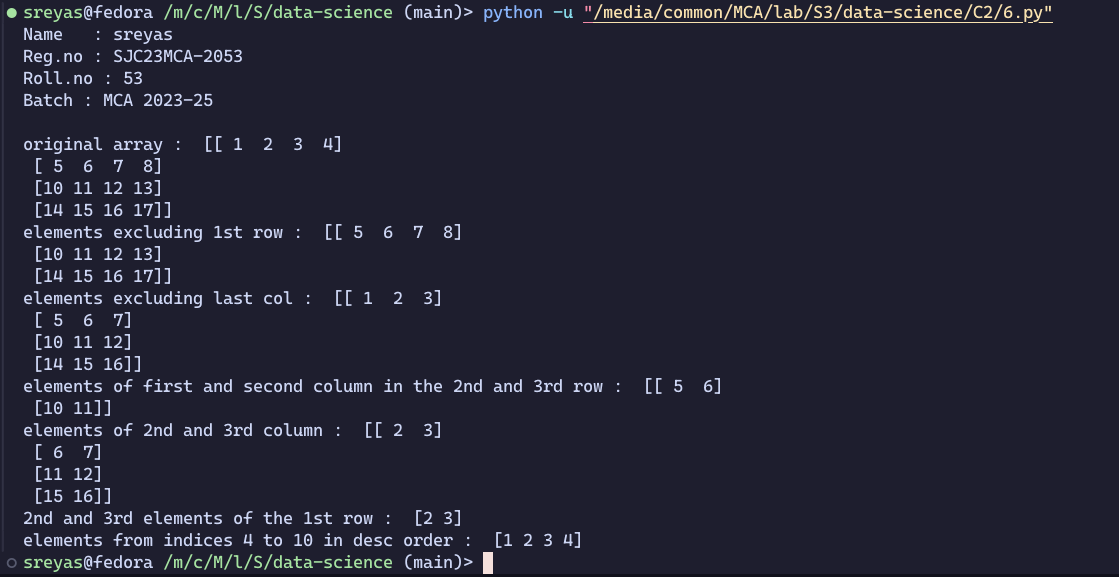
print("elements of first and second column in the 2nd and 3rd row : ", arr[1:3, 0:2])

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:**



**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)

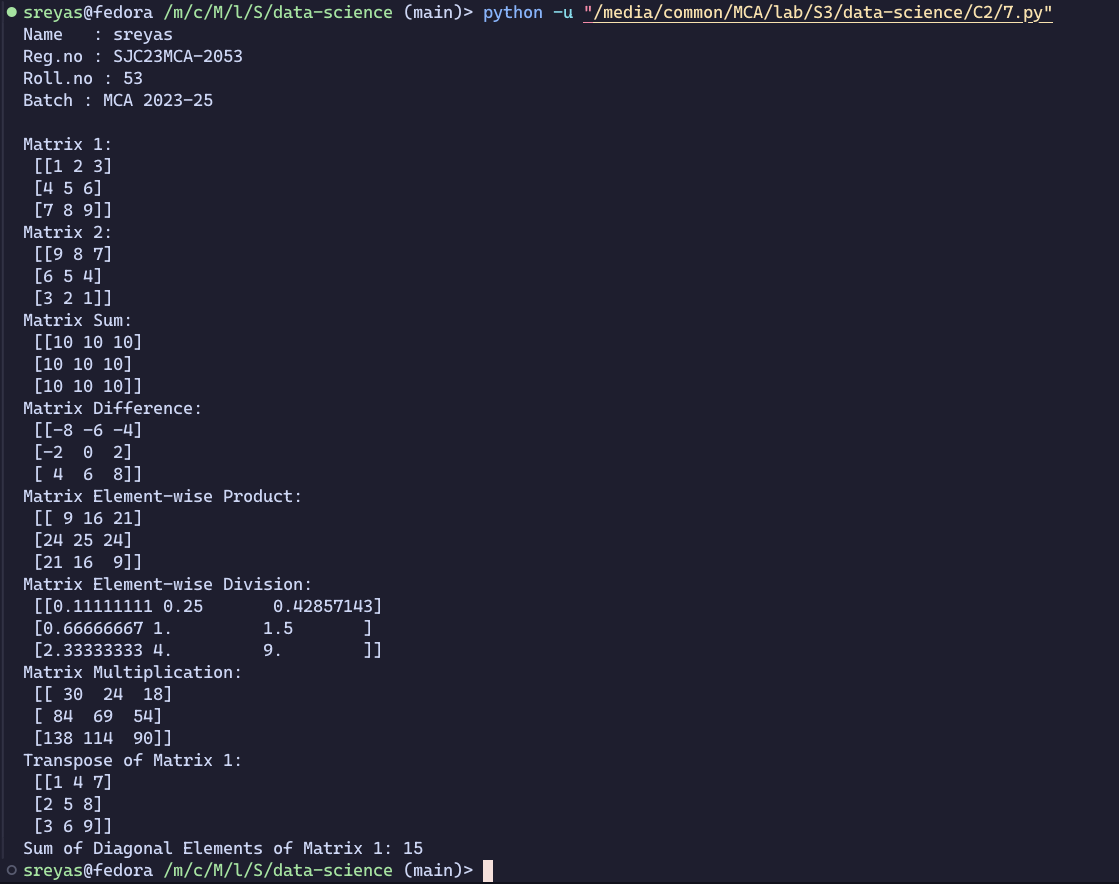
print("Matrix Element-wise Division:\n", matrix\_divide)

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:**



**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)

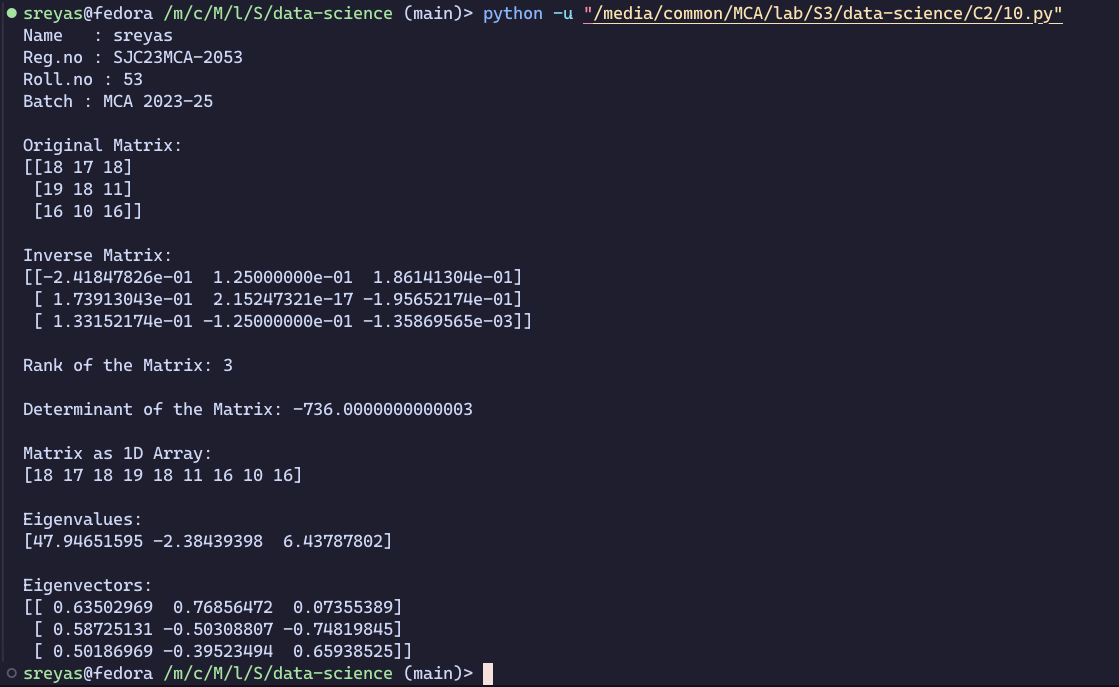
print("\nEigenvalues:")

print(eigenvalues)

print("\nEigenvectors:")

print(eigenvectors)

**OUTPUT:**



**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)

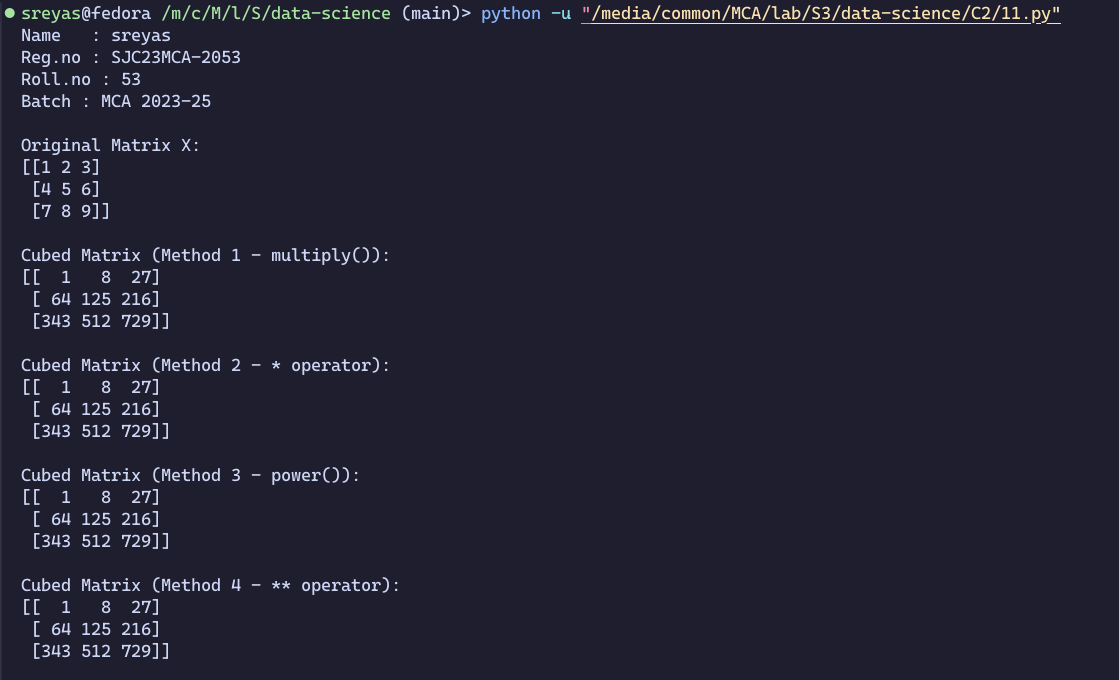
print("\nX^3:")

print(X\_power\_3)

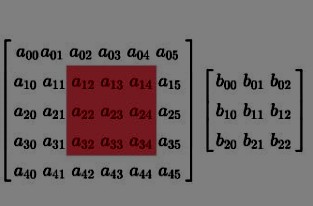
print("\nX^4:")

print(X\_power\_4)

**OUTPUT:**



**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**



**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")

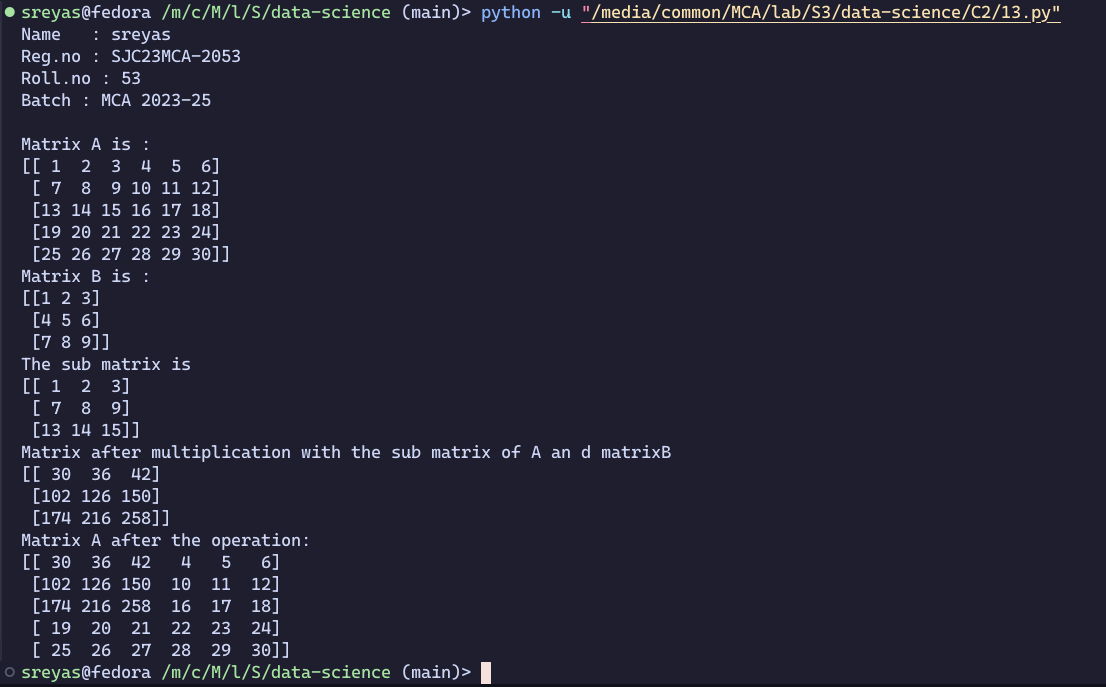
print(result)

A[:3, :3] = result

print("Matrix A after the operation:")

print(A)

**OUTPUT:**



**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:**



**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 

**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:**



**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")

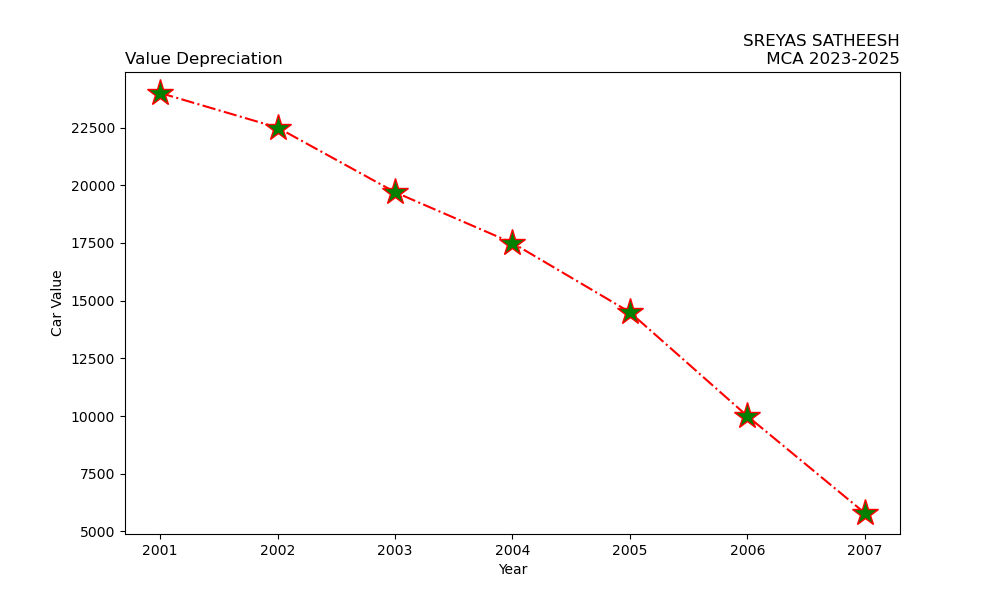
plt.xlabel("Year")

plt.ylabel("Car Value")

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

plt.show()

**OUTPUT:**

****

**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')

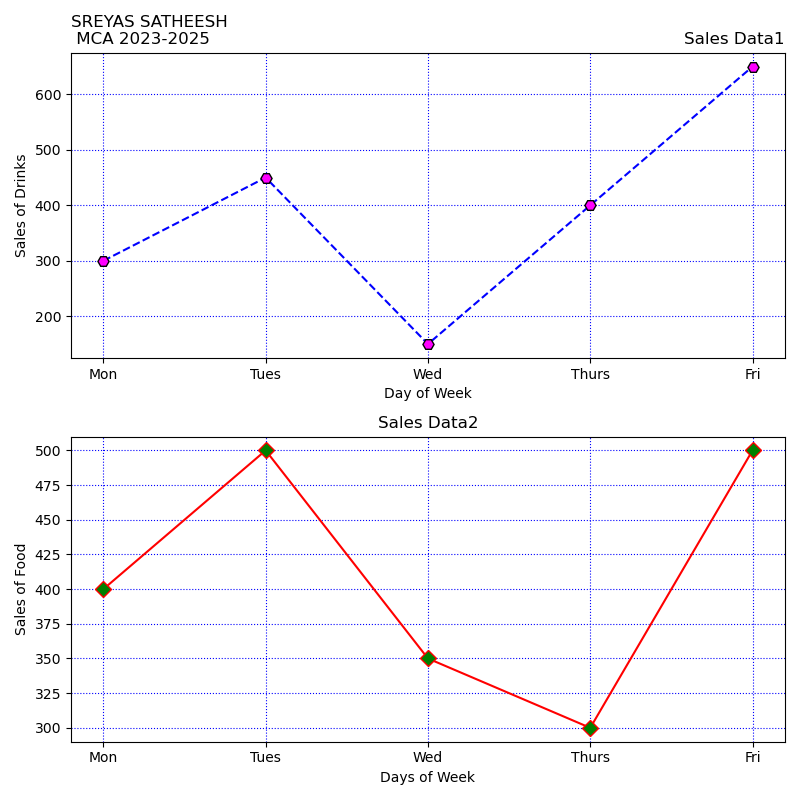
axs[1].grid(True, color='blue', linestyle='dotted')

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')

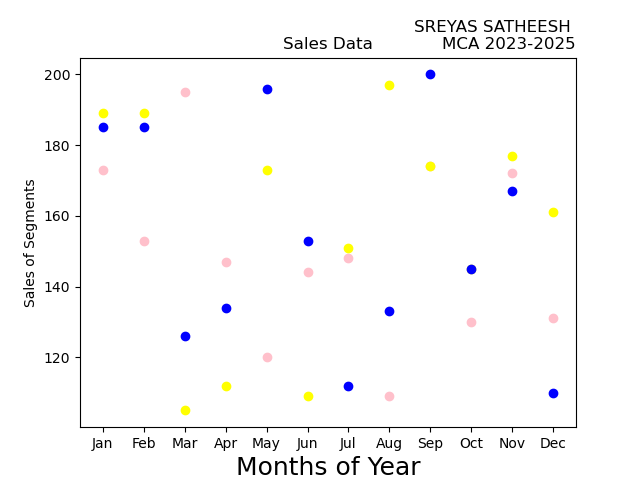
plt.scatter(month,LS, label='Luxury Segment', color='yellow')

plt.scatter(month,SLS, label='Super Luxury Segment', color='blue')

plt.savefig("./Outputs/3.png")

plt.show()

**OUTPUT:**



**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')

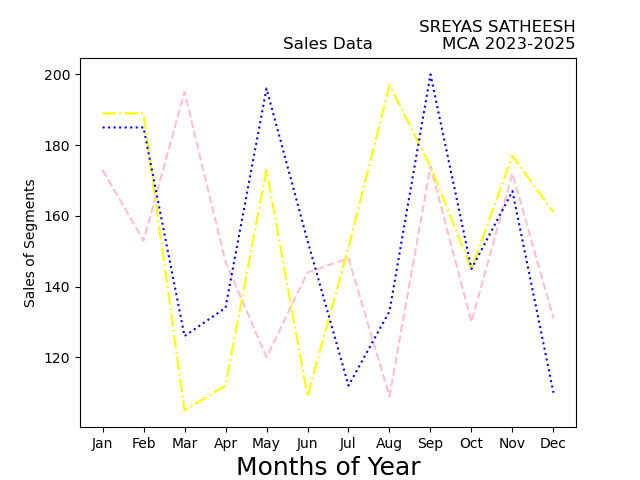
plt.title('Sales Data')

plt.title('SREYAS SATHEESH\nMCA 2023-2025', loc='right')

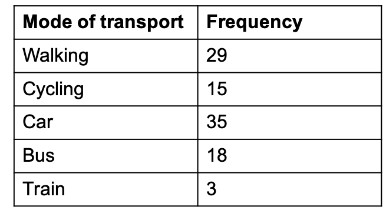
plt.savefig("./Outputs/4.png")

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.**

****

**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')

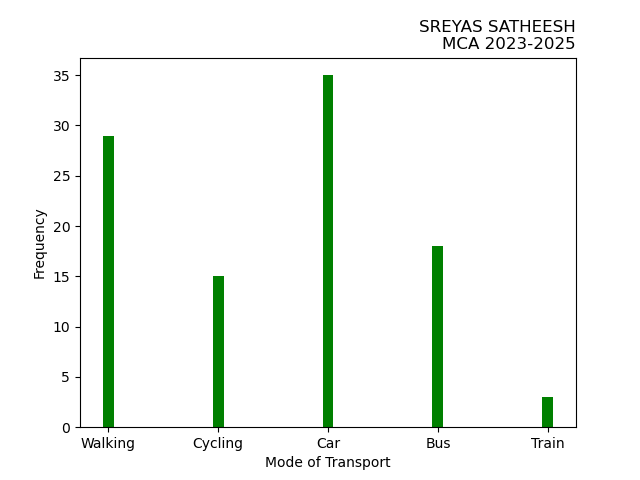
plt.title('SREYAS SATHEESH\nMCA 2023-2025', loc='right')

plt.bar(mode\_transport,feq, width=0.1, color='green')

plt.savefig("./Outputs/5.png")

plt.show()

**OUTPUT:**

****

**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')

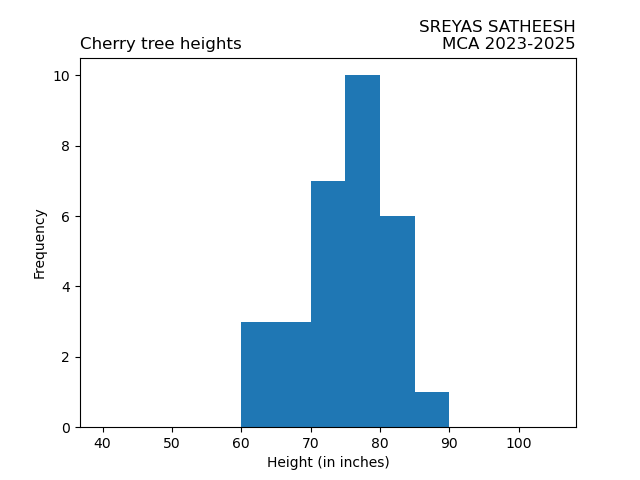
plt.xlabel('Height (in inches)')

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)

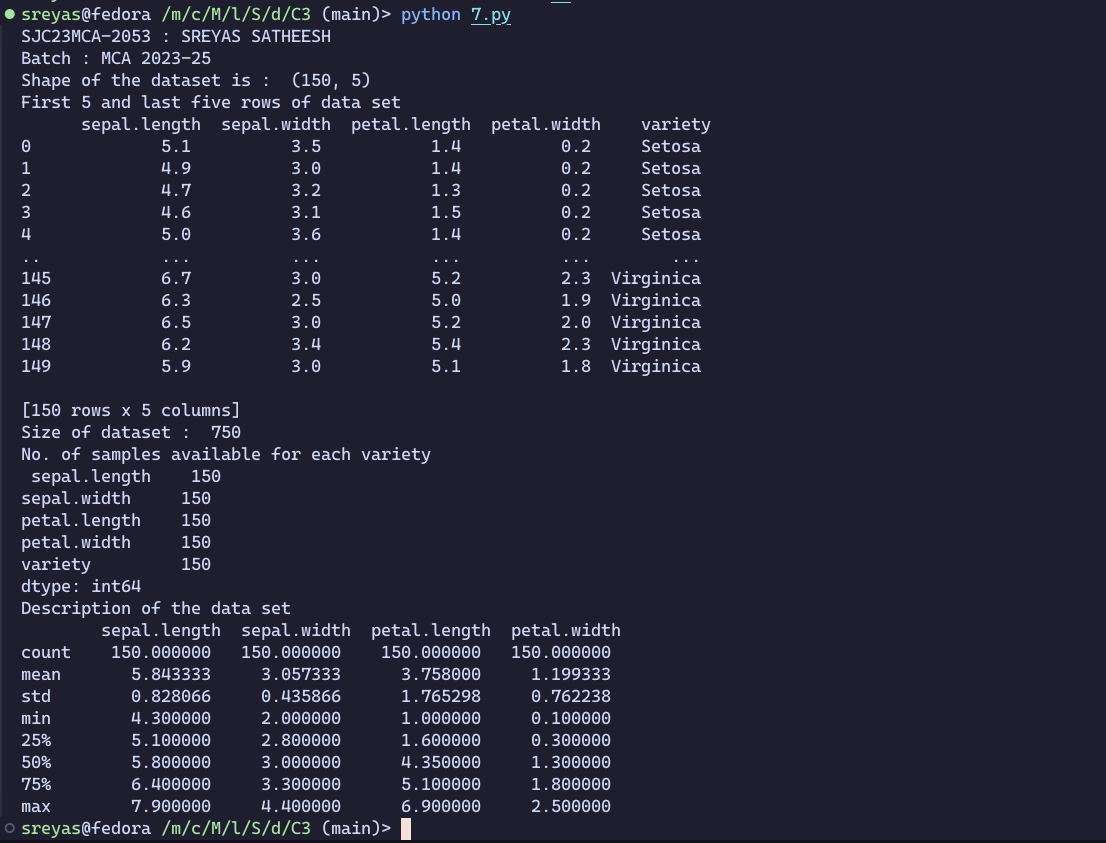
print("First 5 and last five rows of data set\n",df)

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:**



**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")

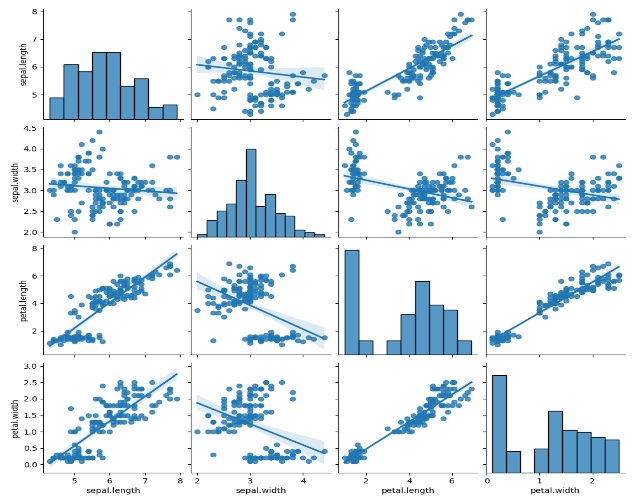
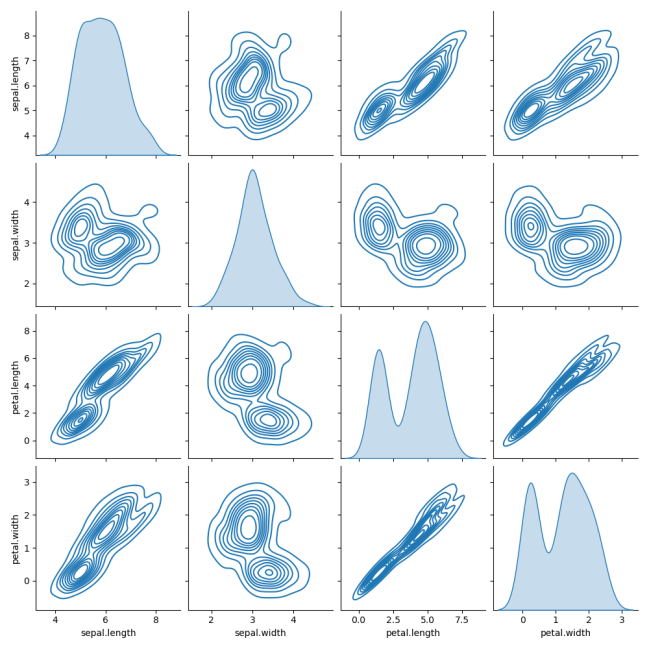
plt.savefig("./Outputs/8\_3.png")

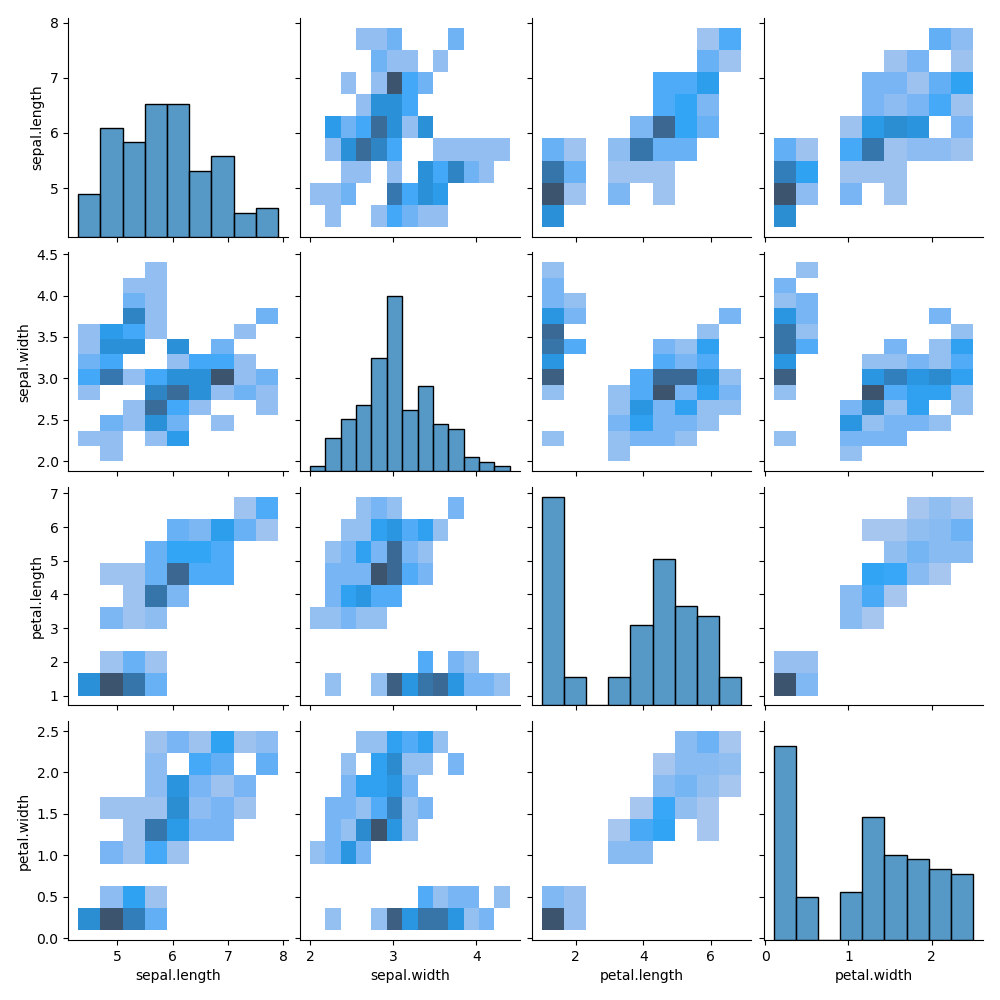
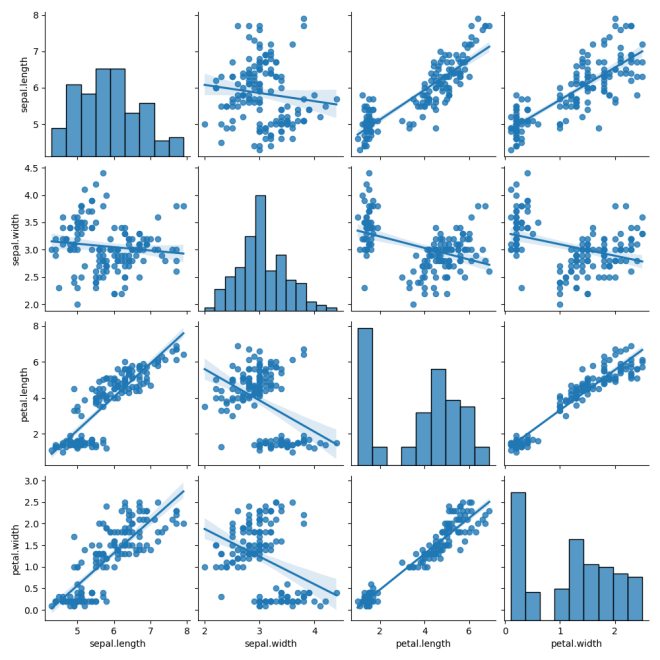
seaborn.pairplot(dataset, kind="reg")

plt.savefig("./Outputs/8\_4.png")

plt.show()

**OUTPUT:**

** **

**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()

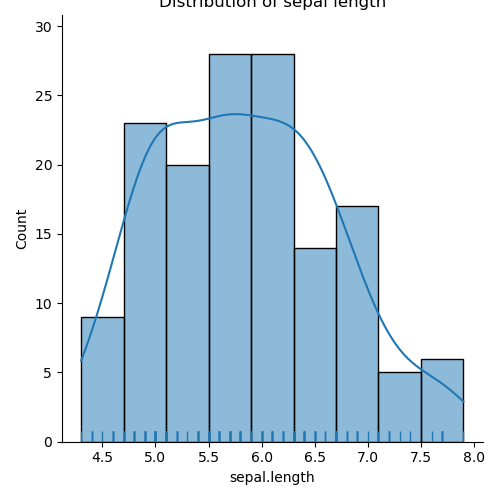
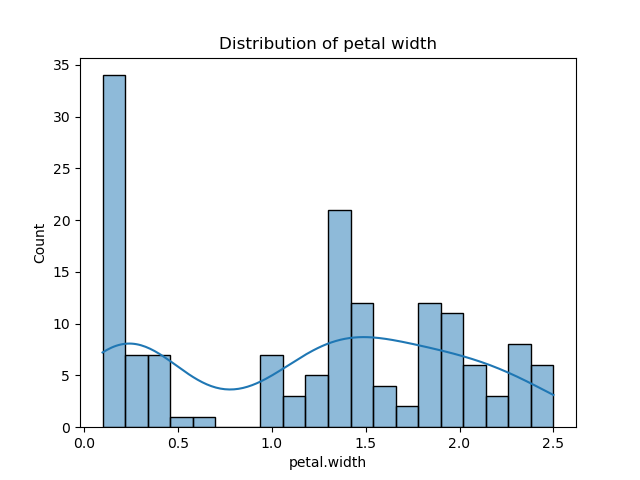
seaborn.relplot(x="sepal.length", y="sepal.width", data=iris\_dataset, hue="variety", style="variety")

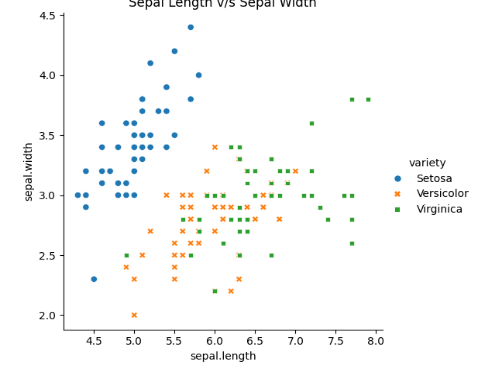
plt.title("sepal length vs sepal width")

plt.savefig("./Outputs/9\_3.png")

plt.show()

**OUTPUT:**

**** 



**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

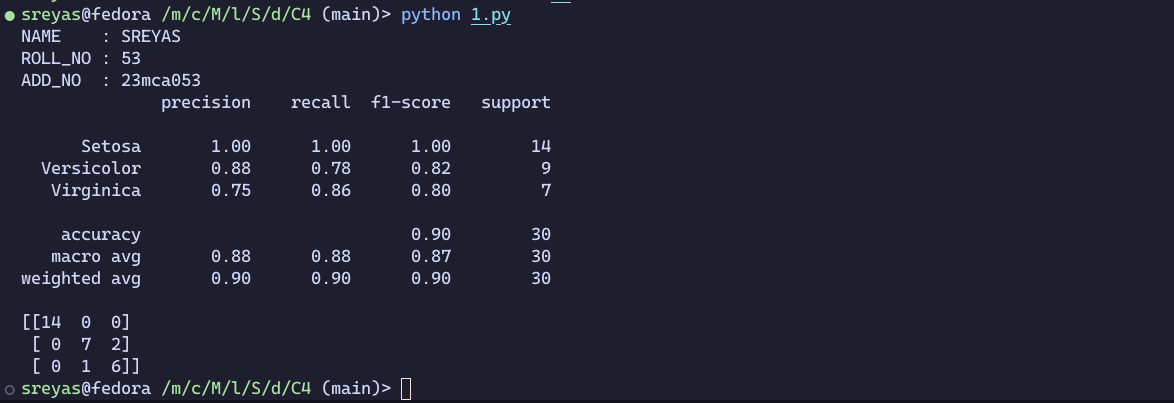
print(classification\_report(y\_test,y\_pred))

print(confusion\_matrix(y\_test,y\_pred))

**OUTPUT:**

K=5, TEST= 0.20, TRAIN= 0.80

K=3, TEST= 0.20, TRAIN= 0.80



**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)

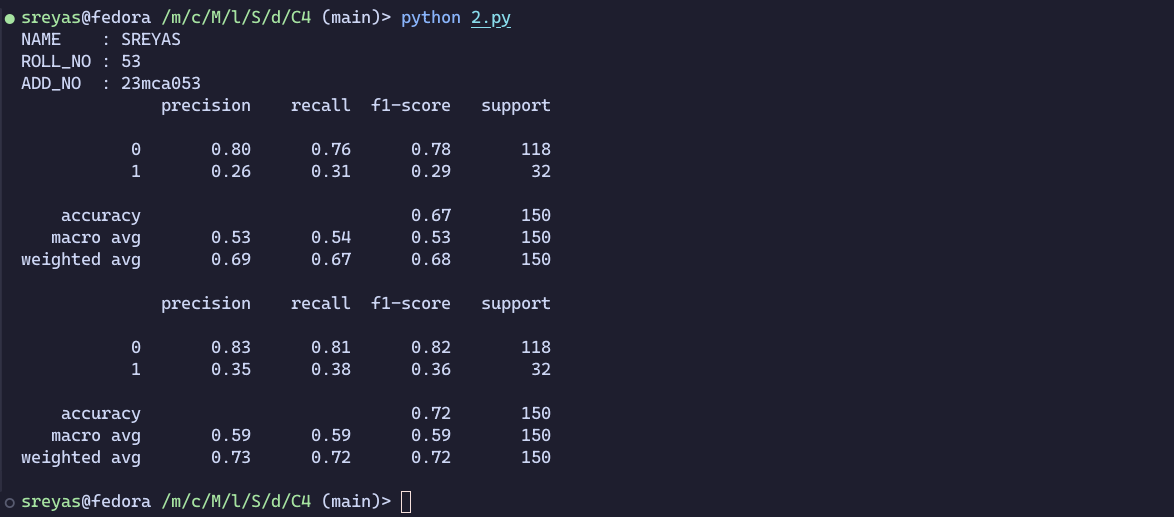
classifier.fit(x\_train,y\_train)

y\_pred=classifier.predict(x\_test)

from sklearn.metrics import classification\_report,confusion\_matrix

print(classification\_report(y\_test,y\_pred))

**OUTPUT:**



**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)

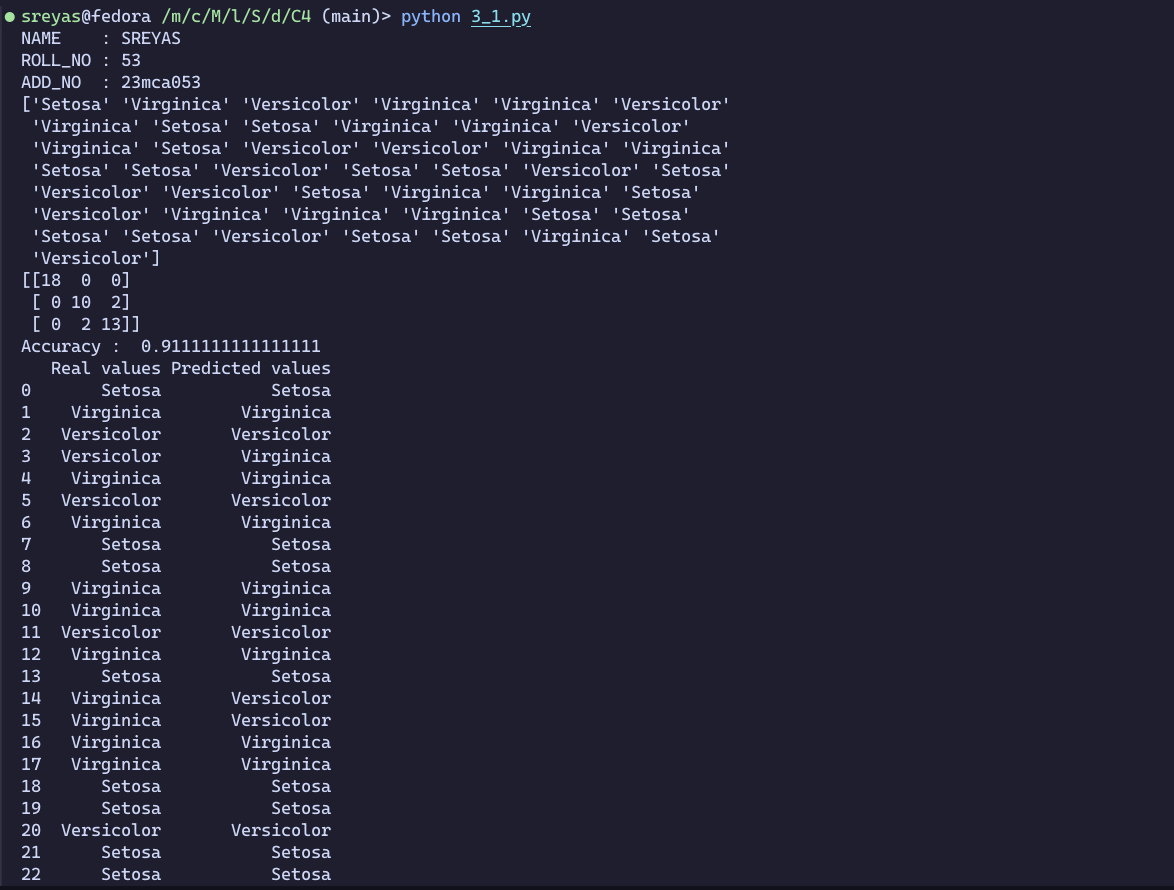
from sklearn.metrics import accuracy\_score

print("Accuracy : ",accuracy\_score(y\_test,y\_pred))

df=pd.DataFrame({'Real values':y\_test,'Predicted values':y\_pred})

print(df)

**OUTPUT:**



**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)

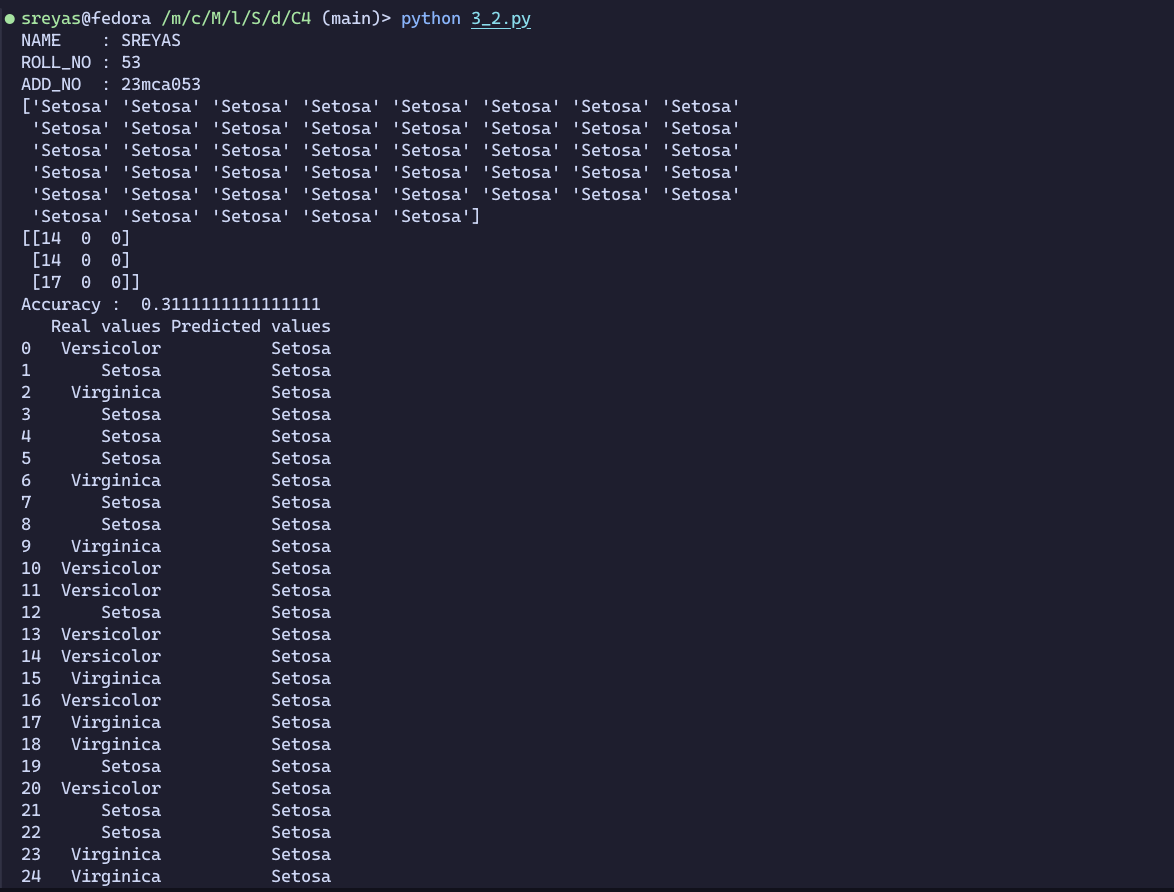
from sklearn.metrics import accuracy\_score

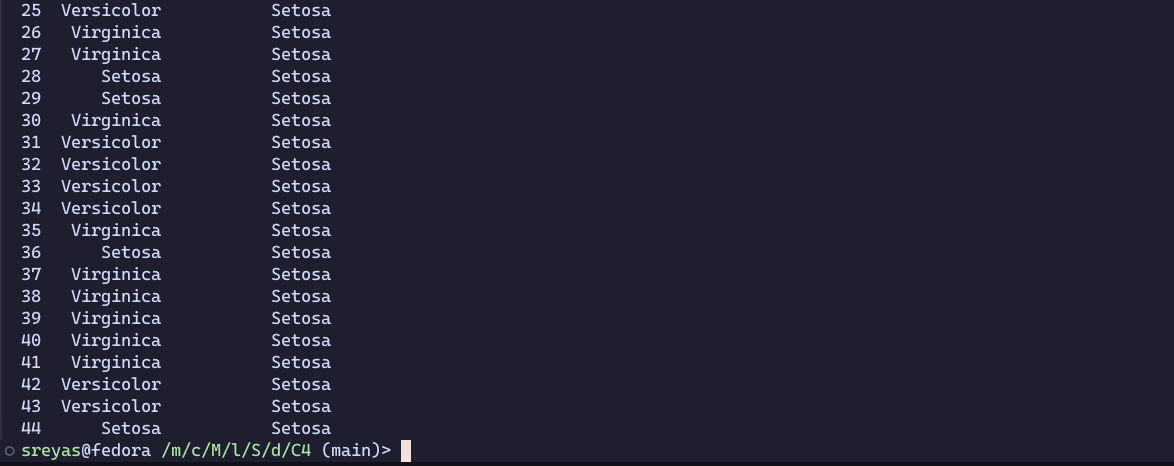
print("Accuracy : ",accuracy\_score(y\_test,y\_pred))

df=pd.DataFrame({'Real values':y\_test,'Predicted values':y\_pred})

print(df)

**OUTPUT:**





**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()

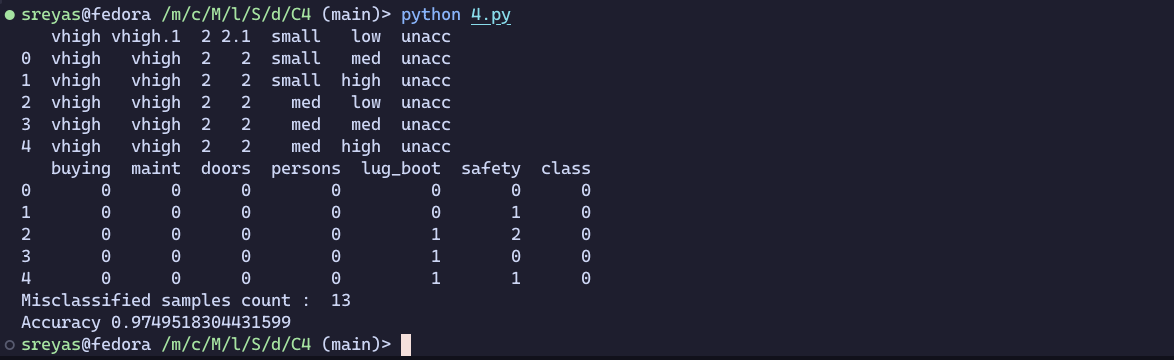
print('Misclassified samples count : ',count\_missclassified)

from sklearn.metrics import accuracy\_score

accuracy = accuracy\_score(y\_test,y\_pred)

print("Accuracy",accuracy)

**OUTPUT:**



**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)

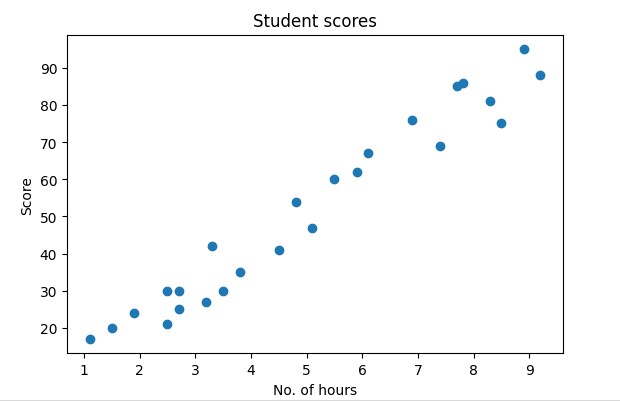
rmse = np.sqrt(mse)

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)

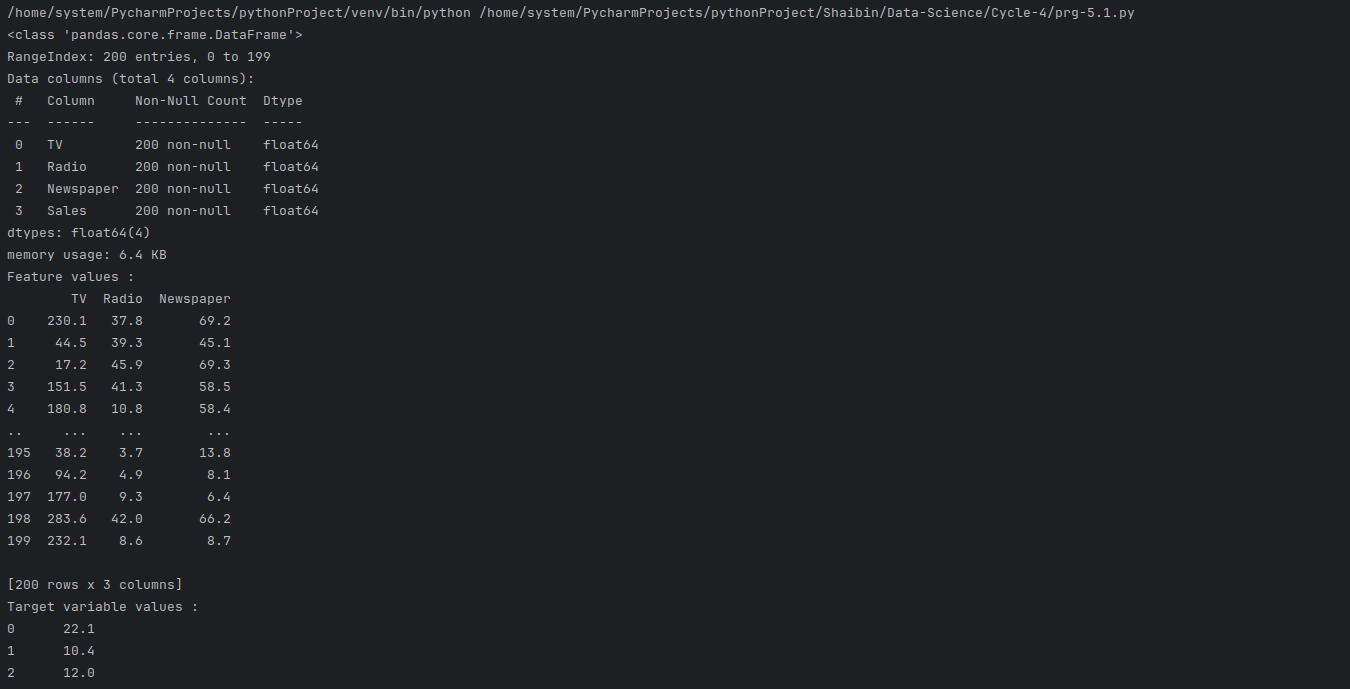
print("Number of mislabeled points from test data set : ",(y\_test != y\_pred).sum())

from sklearn import metrics print("Mean Absolute error :", metrics.mean\_absolute\_error(y\_test,y\_pred))

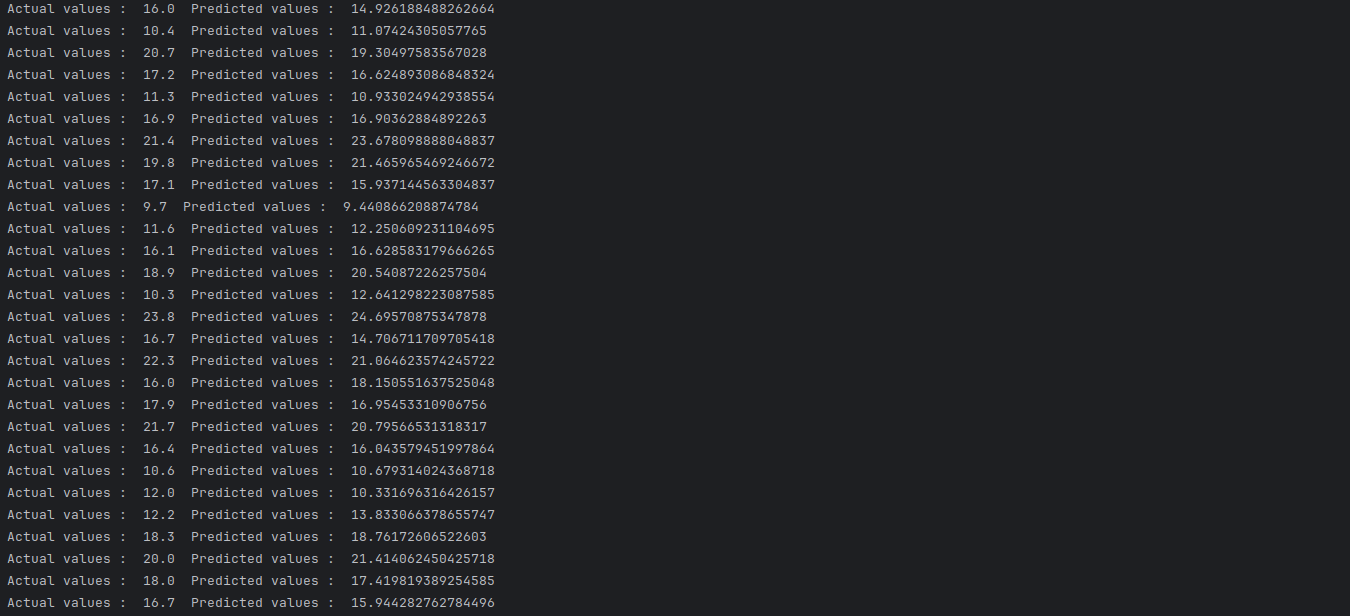
print("Mean Squared error :", metrics.mean\_squared\_error(y\_test,y\_pred))

print("Root Mean Squared error :", np.sqrt(metrics.mean\_squared\_error(y\_test,y\_pred)))

**OUTPUT :**



A black screen with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

**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:**

A graph of a model loss

Description automatically generated

**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 = ''

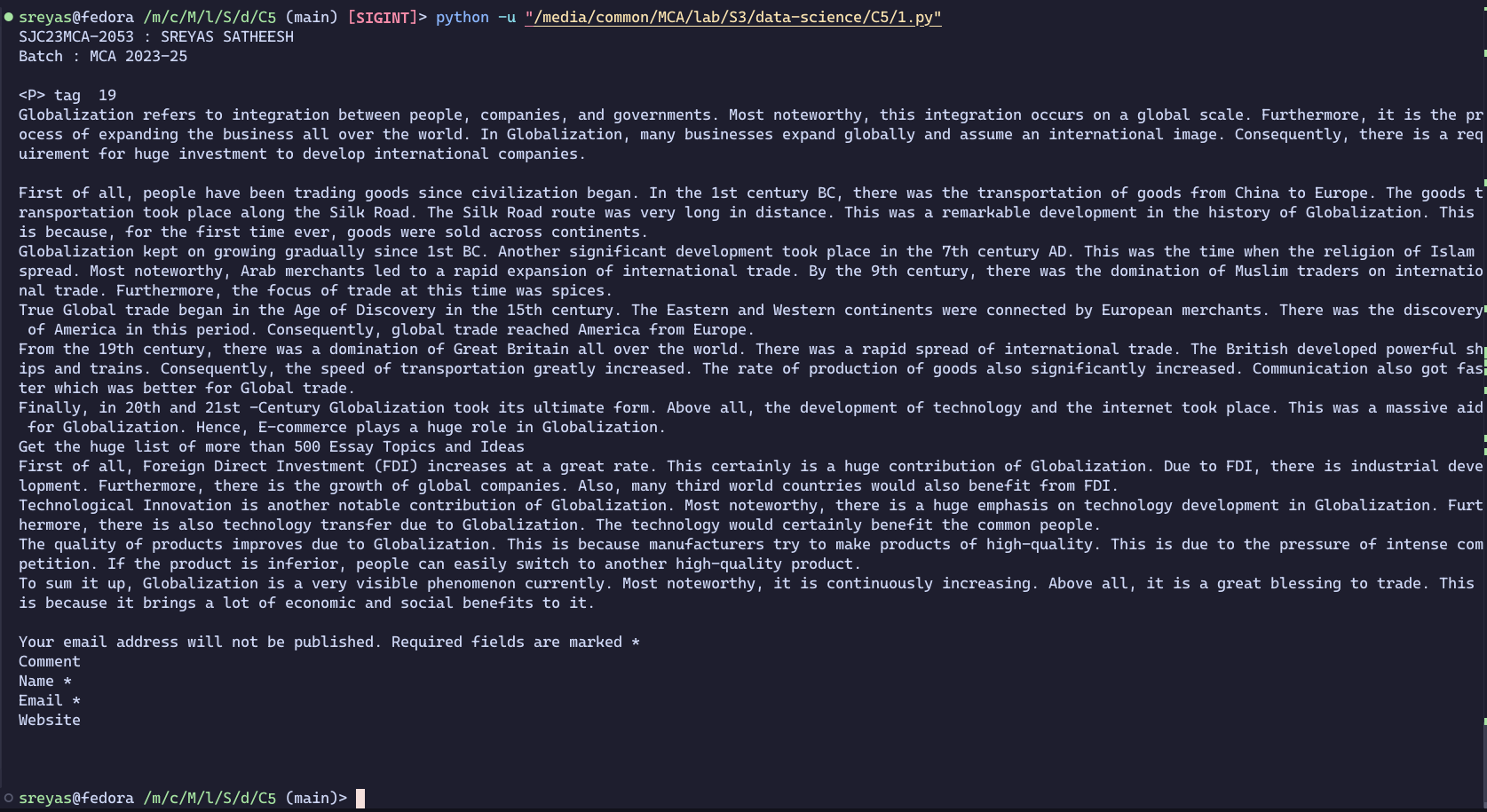
pr = len(soup.find\_all('p'))

print("<P> tag ", pr)

for data in soup.find\_all('p'):

print(data.get\_text())

**OUTPUT:**



**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")

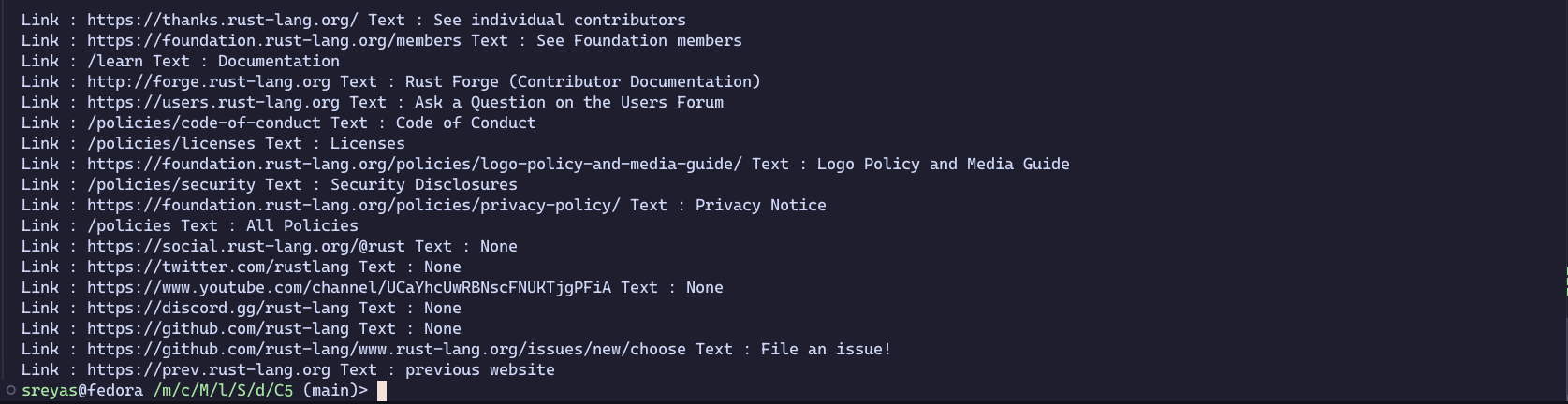
print("Total number of links : ",len(links))

for link in links:

if link.get("href") != "":

print("Link :",link.get("href"),"Text :",link.string)

**OUTPUT:**



**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):

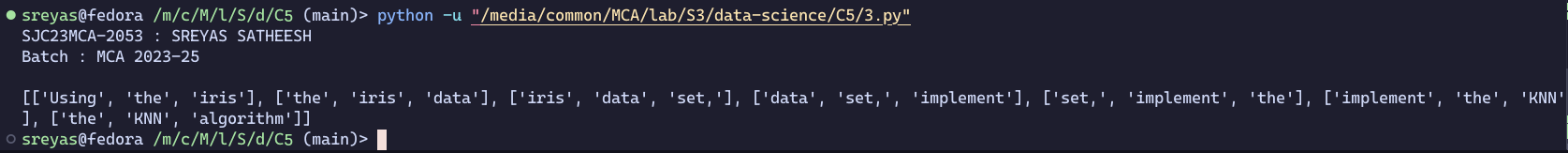
output.append(words[i:i+wordsToCombine])

return output

x = gen\_ngrams(text='Using the iris data set, implement the KNN algorithm',wordsToCombine=3)

print(x)

**OUTPUT:**



**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'

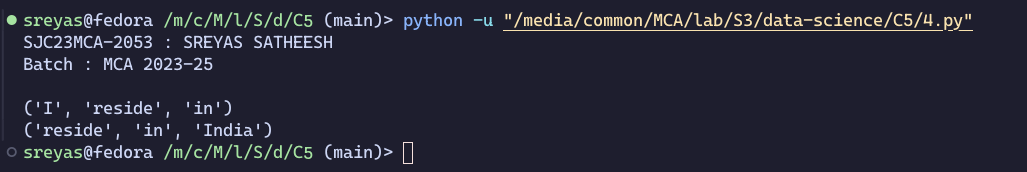
n = 3

trigrams = ngrams(sentence.split(),n)

for grams in trigrams:

print(grams)

**OUTPUT:**



**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("")

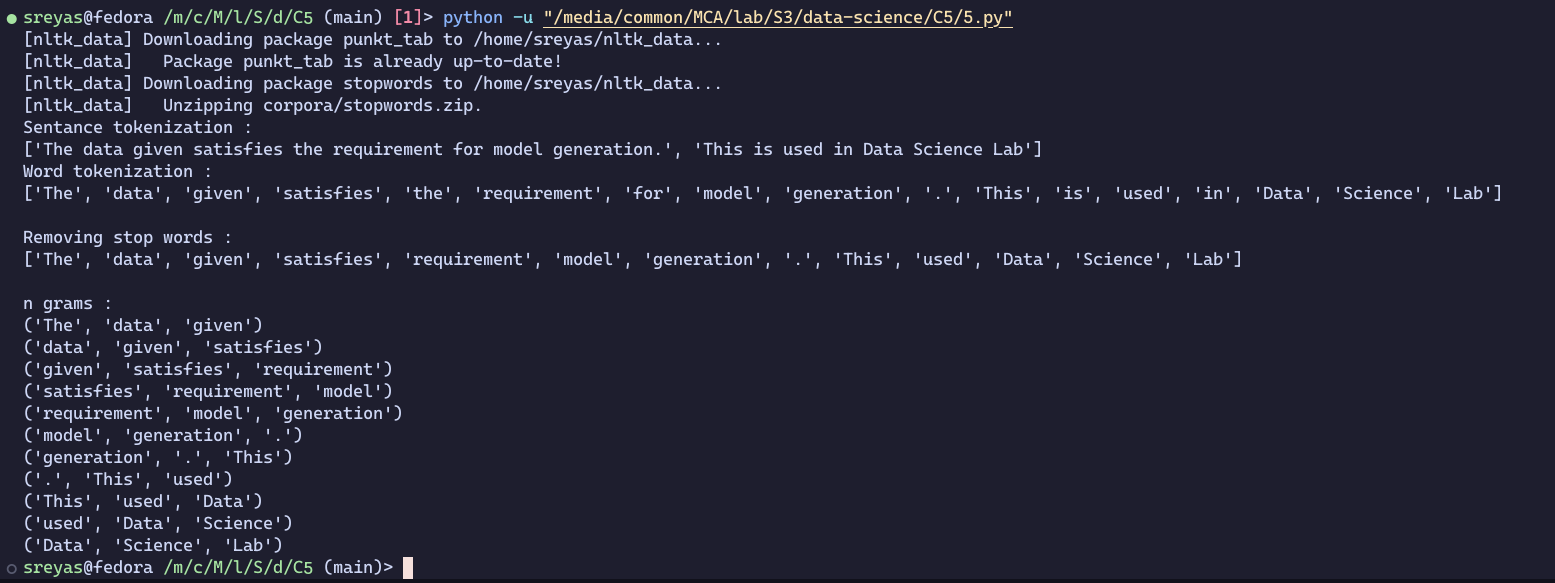
print("n grams : ")

unigrams = ngrams(text2,3)

for grams in unigrams:

print(grams)

**OUTPUT:**



**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')

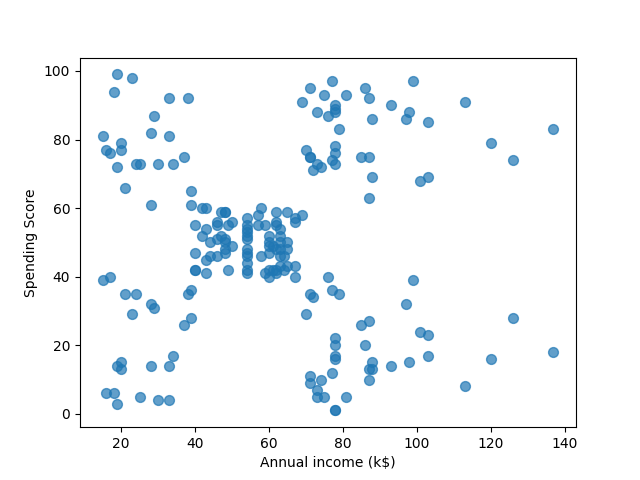
plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score')

plt.title('Cluster centers')

plt.show()

**OUTPUT:**

 A chart with many colored dots

Description automatically generated

A diagram of a cluster

Description automatically generated with medium confidence