**Data Science Lab-Record**

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

print("20 Gopika Vijayan ")

n1=int(input("enter lower limit:"))

n2=int(input("enter upper limit:"))

for i in range(n1 , n2):

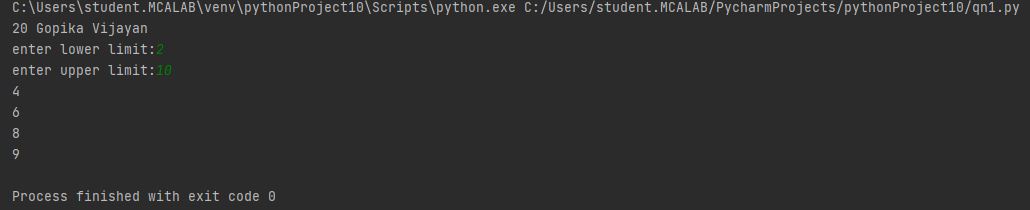
for j in range(2 , i):

if(i % j) == 0:

print(i)

break

Output:



2.Program to print the first N Fibonacci numbers.

print("20 Gopika Vijayan ")

nterms = int(input("How many terms? "))

n1, n2 = 0, 1

count = 0

if nterms <= 0:

print("Please enter a positive integer")

elif nterms == 1:

print("Fibonacci sequence upto",nterms,":")

print(n1)

else:

print("Fibonacci sequence:")

while count < nterms:

print(n1)

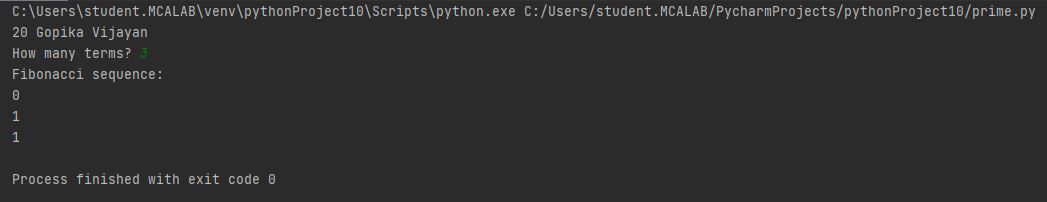
nth = n1 + n2

n1 = n2

n2 = nth

count += 1

Output



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

isosceles, equilateral or scalene.

**print("20 Gopika Vijayan ")**

**n1=int(input("enter the side of triangle:"))**

**n2=int(input("enter the side of triangle:"))**

**n3=int(input("enter the side of triangle:"))**

**if n1==n2 and n2==n3:**

**print("triangle is equilateral")**

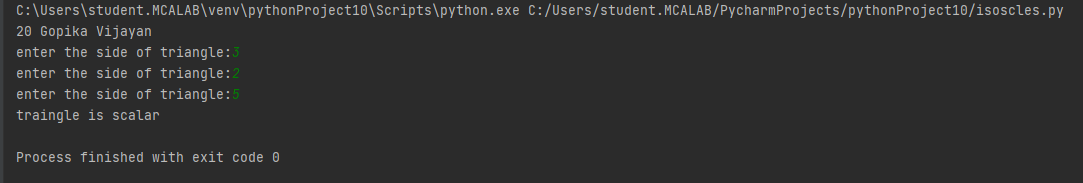
**elif n1==n2 or n2==n3 or n1==n3:**

**print("triangle is isoscless")**

**else:**

**print("traingle is scalar")**

Output:



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

print("20 Gopika Vijayan ")

def are\_coprime(a, b):

hcf = 1

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

if a % i == 0 and b % i == 0:

hcf = i

return hcf == 1

first = int(input('Enter first number: '))

second = int(input('Enter second number: '))

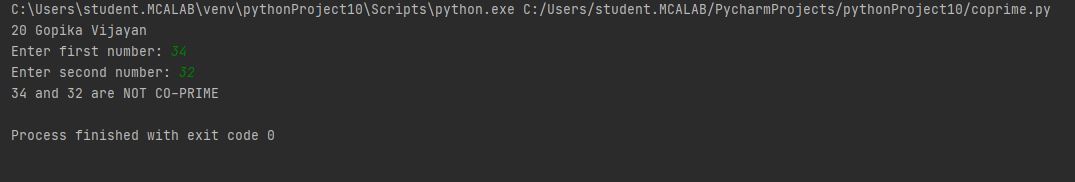
if are\_coprime(first, second):

print('%d and %d are CO-PRIME' % (first, second))

else:

print('%d and %d are NOT CO-PRIME' % (first, second))

Output:



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

print("20 Gopika Vijayan ")

from math import sqrt

print("Quadratic function : (a \* x^2) + b\*x + c")

a = float(input("a: "))

b = float(input("b: "))

c = float(input("c: "))

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

if r > 0:

num\_roots = 2

x1 = (((-b) + sqrt(r))/(2\*a))

x2 = (((-b) - sqrt(r))/(2\*a))

print("There are 2 roots: %f and %f" % (x1, x2))

elif r == 0:

num\_roots = 1

x = (-b) / 2\*a

print("There is one root: ", x)

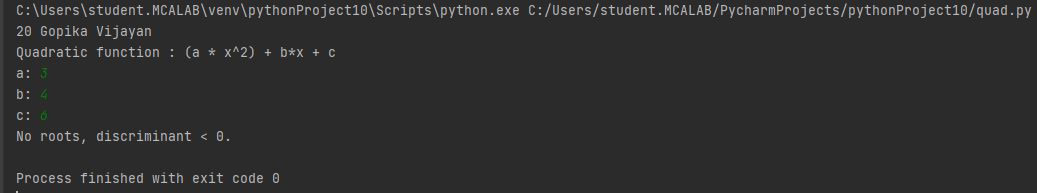
else:

num\_roots = 0

print("No roots, discriminant < 0.")

exit()

Output:



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

=number)

print("20 Gopika Vijayan ")

n=int(input("enter the no:"))

sum=0

for i in range(1,n):

if (n % i == 0):

sum=sum+i

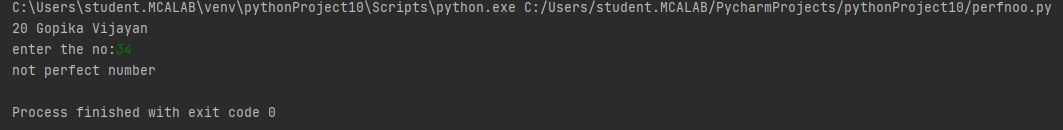
if (sum == n):

print("perfect number")

else:

print("not perfect number")

Output:



7) Program to display armstrong numbers upto 1000

print("20 Gopika Vijayan ")

for i in range(1,1000):

temp=i

sum=0

while i > 0:

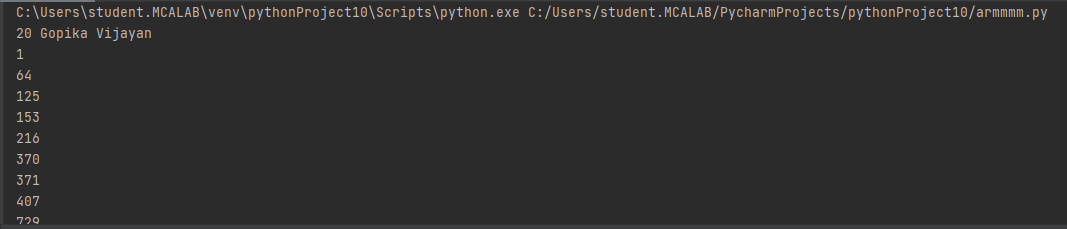
r = i % 10

sum = sum+( r \* r \* r )

i = i // 10

if sum == temp:

print(temp)



8. Store and display the days of a week as a List, Tuple, Dictionary, Set. Also

demonstrate different ways to store values in each of them. Display its type also.

print("20 Gopika Vijayan ")

list1=['sunday','monday','tuesday','wednesday','thursday','friday','saturday']

tuple1=('sunday','monday','tuesday','wednesday','thursday','friday','saturday')

set1={'sunday','monday','tuesday','wednesday','thursday','friday','saturday'}

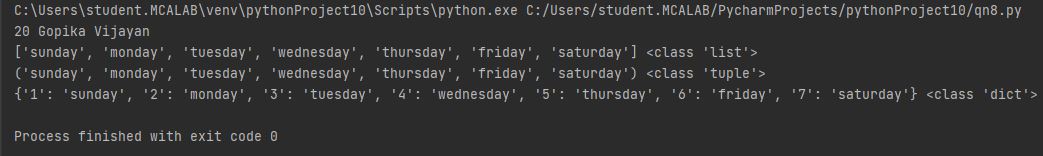
dict1={'1':'sunday','2':'monday','3':'tuesday','4':'wednesday','5':'thursday','6':'friday','7':'saturday'}

print(list1,type(list1))

print(tuple1,type(tuple1))

print(dict1,type(dict1))

Output:



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

print("20 Gopika Vijayan ")

n1=int(input("enter the list 1 size:"))

list1=[]

print("enter the values of list 1:")

for i in range(0,n1):

x=int(input())

list1.append(x)

n2=int(input("enter the list 2 size:"))

list2=[]

print("enter the values of list 2")

for i in range(0,n2):

y=int(input())

list2.append(y)

print(list1)

print(list2)

list3=[]

if n1==n2:

for i in range(0,n1):

element=list1[i]+list2[i]

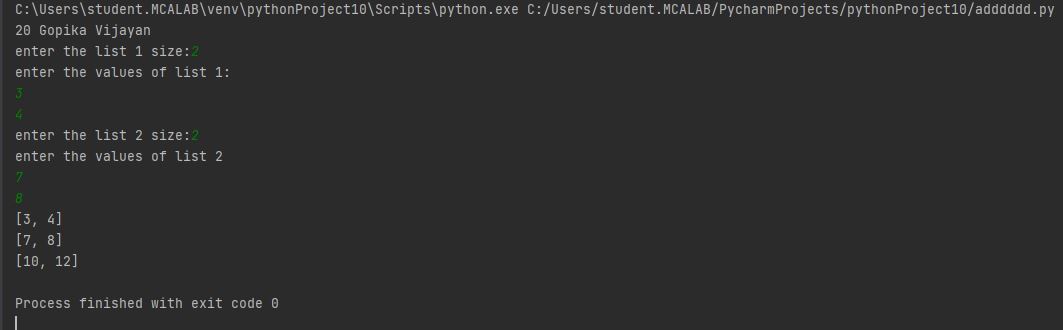
list3.append(element)

print(list3)

else:

print("No of elements are not equal")

Output:



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

print("20 Gopika Vijayan ")

n1=int(input("enter the no of rows of 1st matrix:"))

n2=int(input("enter the no of coloumns of 1st matrix:"))

print("Enter the elements")

matrix=[]

for i in range(0,n1):

a=[]

for j in range(0,n2):

a.append(int(input()))

matrix.append(a)

for i in range(0,n1):

for j in range(0,n2):

print(matrix[i][j], end = " ")

print()

n1=int(input("Enter the number of rows for 2nd matix"))

n2=int(input("Enter the number of columns for 2nd matix"))

print("Enter the elements")

matrix2=[]

for i in range(0,n1):

b=[]

for j in range(0,n2):

b.append(int(input()))

matrix2.append(b)

for i in range(0,n1):

for j in range(0,n2):

print(matrix2[i][j], end = " ")

print()

print("Matrix sum is:")

result = [[0, 0, 0], [0, 0, 0], [0, 0, 0]]

for i in range(0, n1):

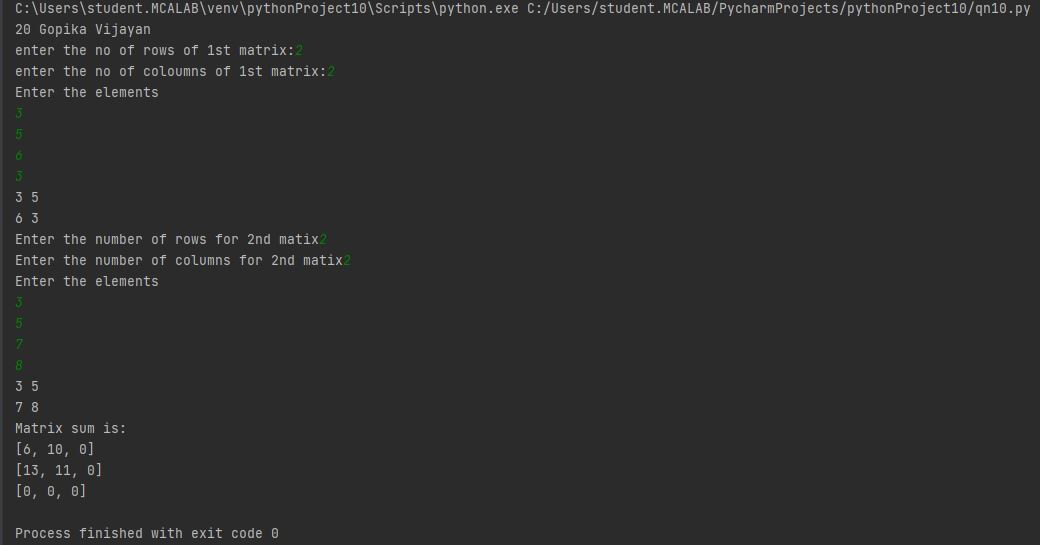
for j in range(0, n2):

result[i][j] = matrix[i][j] + matrix2[i][j]

for r in result:

print(r)

Output:



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

print("20 Gopika Vijayan")

def bubFunc(a, val):

for i in range(val -1):

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

if(a[j] > a[j + 1]):

temp = a[j]

a[j] = a[j + 1]

a[j + 1] = temp

a = []

val = int(input("Please Enter the Total Elements : "))

for i in range(val):

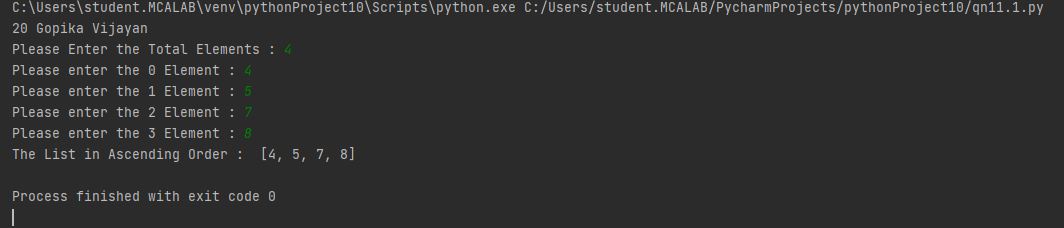
value = int(input("Please enter the %d Element : " %i))

a.append(value)

bubFunc(a, val)

print("The List in Ascending Order : ", a)

Output:



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

print("20 Gopika Vijayan ")

str=input("Enter a string:")

print(str)

vowels='aeiou'

print("count of vowels in the string:")

count={}.fromkeys(vowels,0)

for i in str:

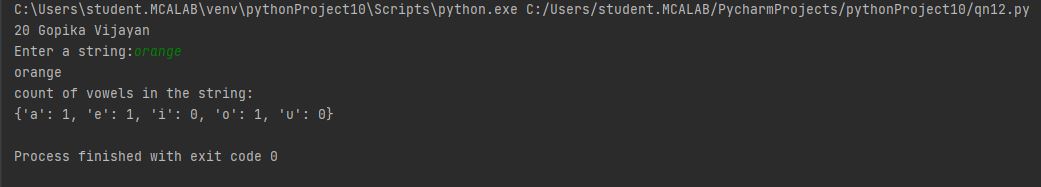
for j in count:

if(i==j):

count[j]=count[j]+1

print(count)

Output:



13. Write a Python program that accept a positive number and subtract from this

number the sum of its digits and so on. Continues this operation until the number is

positive(eg: 256-&gt;2+5+6=13

256-13=243

243-9=232……..

print("20 Gopika Vijayan ")

num=int(input("enter number:"))

def digitsum(num):

sum=0

while num>0:

rem=num%10;

sum=sum+rem;

num=num//10

return sum

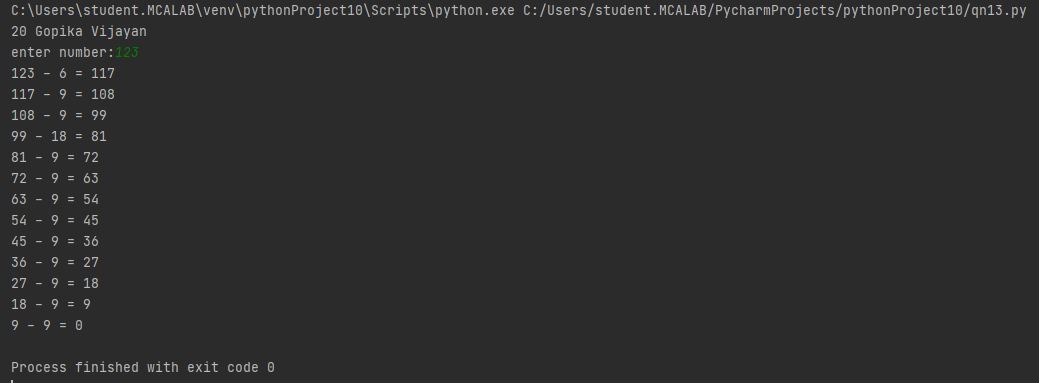
while(num>0):

sum=digitsum(num)

print("{} - {} = {}".format(num,sum,num-sum))

num=num-sum

Output:



14. Write a Python program that accepts a 10 digit mobile number, and find the digits

which are absent in a given mobile number

print("20 Gopika Vijayan")

num = int(input("Enter a 10 digit mobile number : "))

nums = []

for i in range(0, 10):

n = num % 10

nums.append(n)

num = num // 10

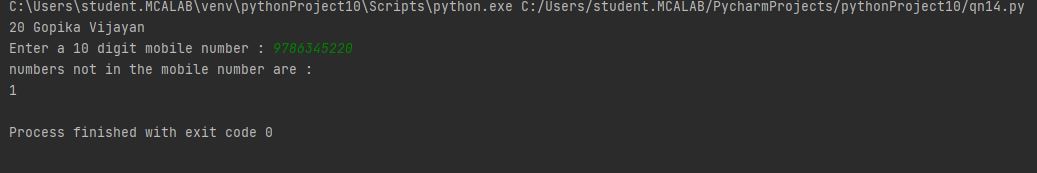
print("numbers not in the mobile number are : ")

for i in range(0, 10):

if i not in nums:

print(i)

Output:



**CYCLE - 2**

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

print("20 Gopika Vijayan")

import numpy as np

ar = np.array([

[

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

],

[

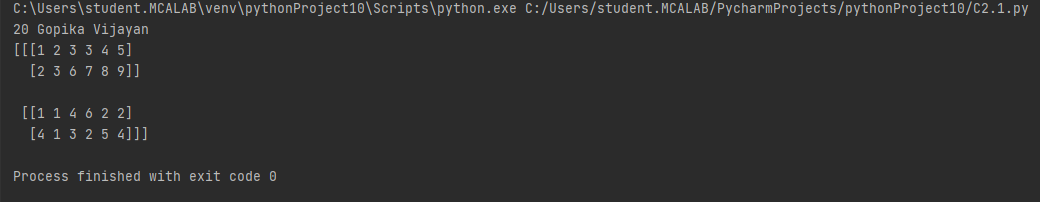
[1,1,4,6,2,2],[4,1,3,2,5,4]

]

])

print(ar)

Output:



2. Create a 2 dimensional array (2X3) with elements belonging to complex data

type and print it. Also display

a. the no: of rows and columns

b. dimension of an array

c. reshape the same array to 3X2

print("20 Gopika Vijayan")

import numpy as np

arr = np.array([

[1+4j,2+5j,3+6j],

[4+6j,9+1j,5+2j],

],

dtype=complex)

print(arr)

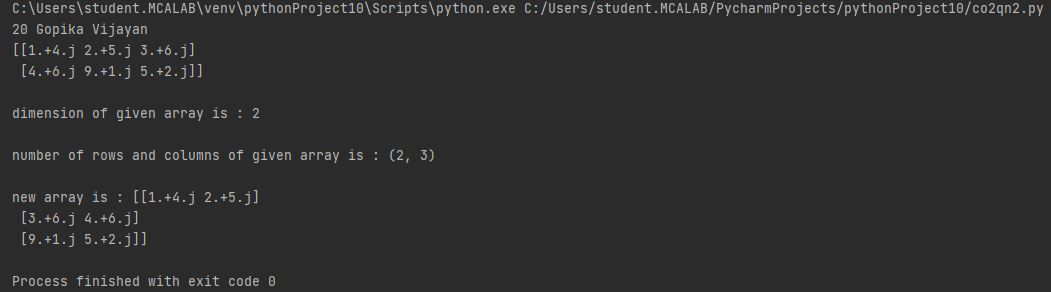
print("\ndimension of given array is :",arr.ndim)

print("\nnumber of rows and columns of given array is :",arr.shape)

newarr = arr.reshape(3,2)

print("\nnew array is :",newarr)

Output:



3. Familiarize with the functions to create

a) an uninitialized array

b) array with all elements as 1,

c) all elements as 0

print("20 Gopika Vijayan")

import numpy as np

arr=np.empty([2,2],dtype="int")

print("an uninitialized array\n",arr)

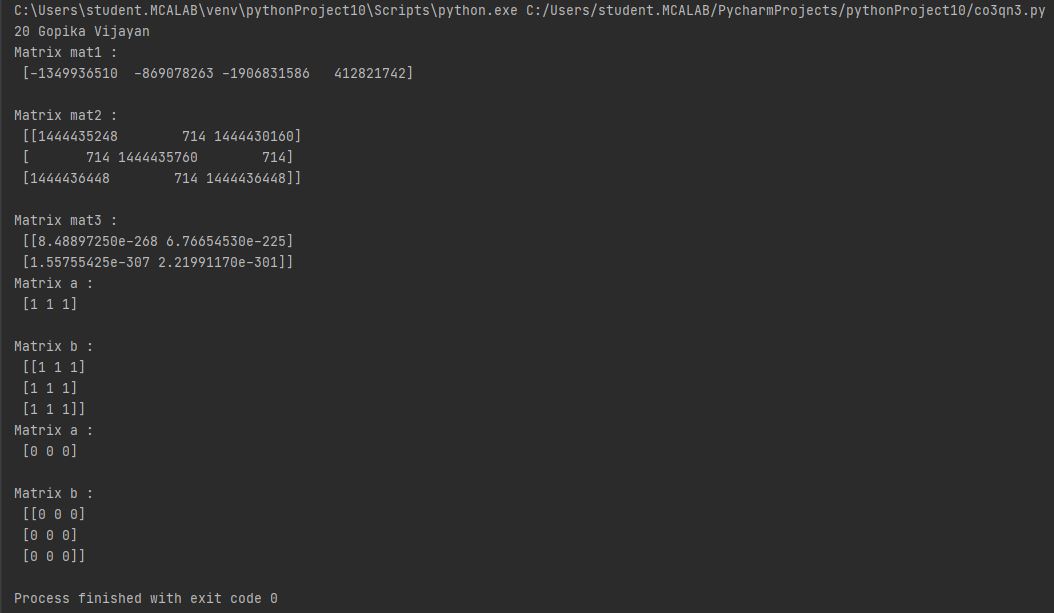
arr=np.ones([2,2],dtype="int")

print("\narray with all elements as 1\n",arr)

arr=np.zeros([2,2],dtype="int")

print("\narray with all elements as 0\n",arr)

Output:



4. Create an one dimensional array using the arrange function containing 10 elements.

Display

a. First 4 elements

b. Last 6 elements

c. Elements from index 2 to 7

print("20 Gopika Vijayan")

import numpy as np

arr=np.arange(start=1,stop=11,step=1,dtype="int")

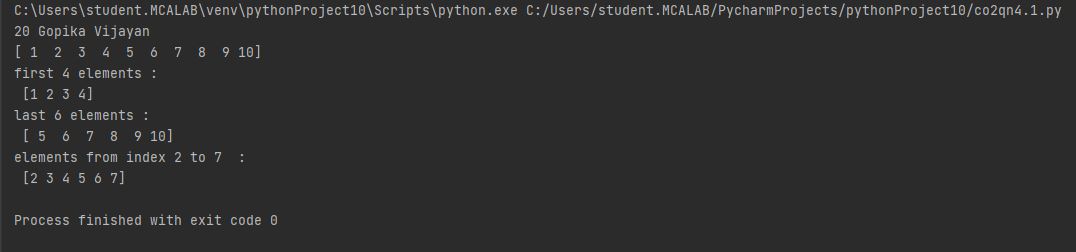
print(arr)

print("first 4 elements :\n",arr[:4])

print("last 6 elements :\n",arr[-6:])

print("elements from index 2 to 7 :\n",arr[1:7])

Output:



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

a. Elements from index 2 to 8 with step 2(also demonstrate the same

using slice function)

b. Last 3 elements of the array using negative index

c. Alternate elements of the array

d. Display the last 3 alternate elements

print("20 Gopika Vijayan")

import numpy as np

ar=np.arange(start=0,stop=30,step=2)

print(ar)

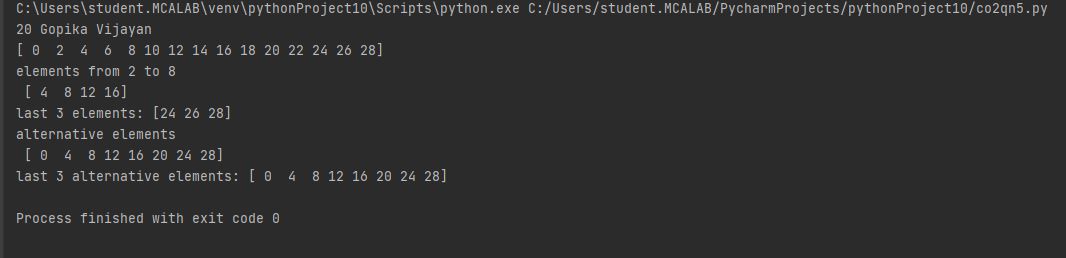
print("elements from 2 to 8\n",ar[2:9:2])

print("last 3 elements:",ar[-3:])

print("alternative elements\n",ar[0:30:2])

print("last 3 alternative elements:",ar[:30:2])

Output:



6. Create a 2 Dimensional array with 4 rows and 4 columns.

a. Display all elements excluding the first row

b. Display all elements excluding the last column

c. Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row

d. Display the elements of 2 nd and 3 rd column

e. Display 2 nd and 3 rd element of 1 st row

f. Display the elements from indices 4 to 10 in descending order(use

–values)

print("20 Gopika Vijayan")

import numpy as np

ar=np.array([[1,2,3,4],

[4,6,7,3],

[8,9,0,1],

[5,6,3,2]

])

print("Display all elements excluding the first row\n",ar[1:4])

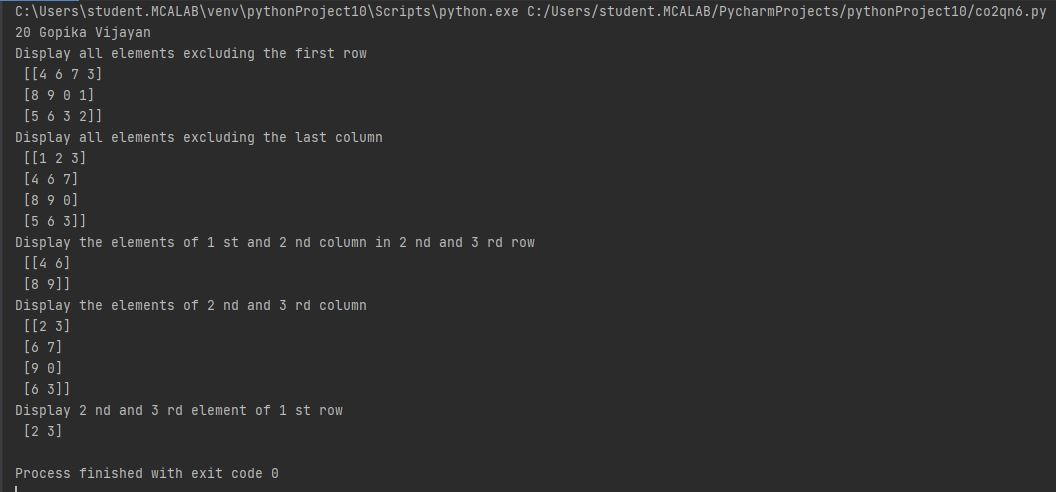
print("Display all elements excluding the last column\n",ar[:,0:3])

print("Display the elements of 1 st and 2 nd column in 2 nd and 3 rd row\n",ar[1:3,0:2])

print("Display the elements of 2 nd and 3 rd column\n",ar[:,1:3])

print("Display 2 nd and 3 rd element of 1 st row\n",ar[0,1:3])

Output:



7. Create two 2D arrays using array object and

a. Add the 2 matrices and print it

b. Subtract 2 matrices

c. Multiply the individual elements of matrix

d. Divide the elements of the matrices

e. Perform matrix multiplication

f. Display transpose of the matrix

g. Sum of diagonal elements of a matrix

print("20 Gopika Vijayan")

import numpy as np

m1=np.array([[1,2,4],

[5,4,3],

[2,3,4]

])

m2=np.array([[2,3,4],

[4,5,4],

[2,3,5]

])

m3=np.add(m1,m2);

print("sum of matrices is:",m3)

m3=np.subtract(m1,m2);

print("Difference between 2 matrices:",m3)

m3=np.multiply(m1,m2);

print("Multiply the individual elements of matrix",m3)

m3=np.divide(m1,m2);

print("Divide the elements of the matrices",m3)

m3=np.matmul(m1,m2);

print("Perform matrix multiplication",m3)

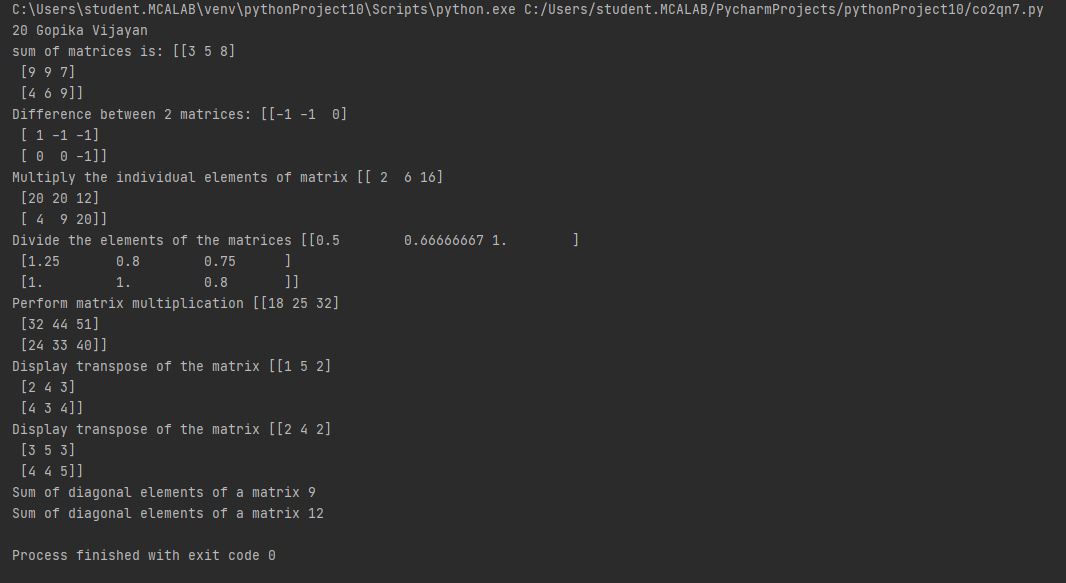
print("Display transpose of the matrix",np.transpose(m1));

print("Display transpose of the matrix",np.transpose(m2));

print("Sum of diagonal elements of a matrix",np.trace(m1));

print("Sum of diagonal elements of a matrix",np.trace(m2));

Output:



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

print("20 Gopika Vijayan")

import numpy as np

arr1=np.array([1,2,3,4,5,6])

print("\narray 1:",arr1)

print("\narray 1 after insertion:",np.insert(arr1,3,9))

arr2=np.array([

[1,2,3],

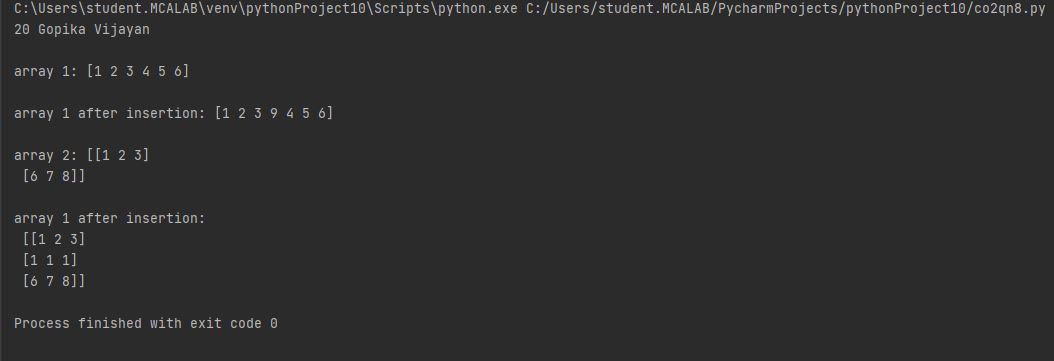
[6,7,8]

])

print("\narray 2:",arr2)

print("\narray 1 after insertion:\n",np.insert(arr2, 1, np.array((1, 1, 1)), 0))

Output:



9. Demonstrate the use of diag() function in 1D and 2D array.

print("20 Gopika Vijayan")

import numpy as np

ar1=np.array([1,2,3])

ar2=np.array([

[2,3,4],

[4,5,6],

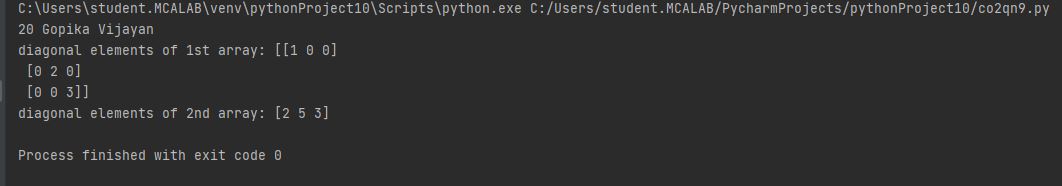
[2,1,3]

])

print("diagonal elements of 1st array:",np.diag(ar1))

print("diagonal elements of 2nd array:",np.diag(ar2))

Output:



10. Demonstrate the use of append() function in 1D and 2D

array.

print("20 Gopika Vijayan")

import numpy as np

ar1=np.array([1,2,3])

print("1st array is:",ar1)

print("1d array after append:",np.append(ar1,[4,5,6]))

ar2=np.array([

[5,6,7],

[1,2,7],

[3,4,8]

])

ar3=np.array([

[5,5,7],

[9,2,7],

[3,6,8]

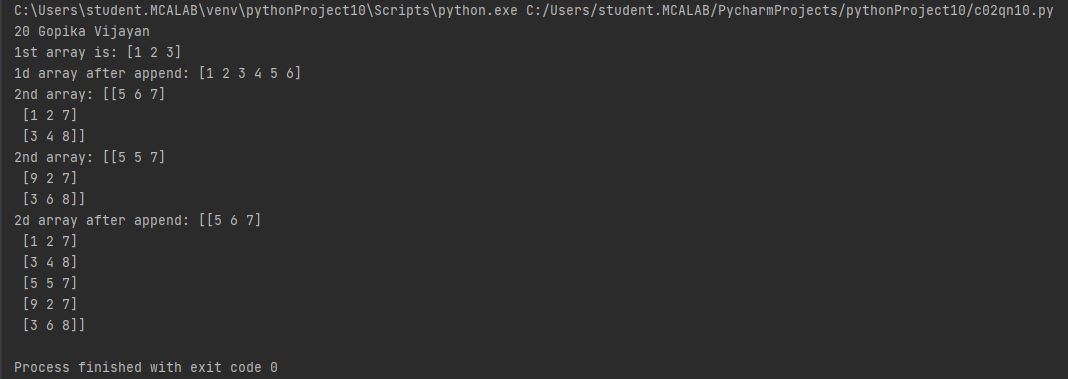
])

print("2nd array:",ar2)

print("2nd array:",ar3)

print("2d array after append:",np.append(ar2,ar3,axis=0))

Output:



11. Demonstrate the use of sum() function in 1D and 2D array.

print("20 Gopika Vijayan")

import numpy as np

ar1=np.array([1,2,3])

ar2=np.array([

[5,6,7],

[1,2,7],

])

print("1st array is:",ar1)

print("2nd array:",ar2)

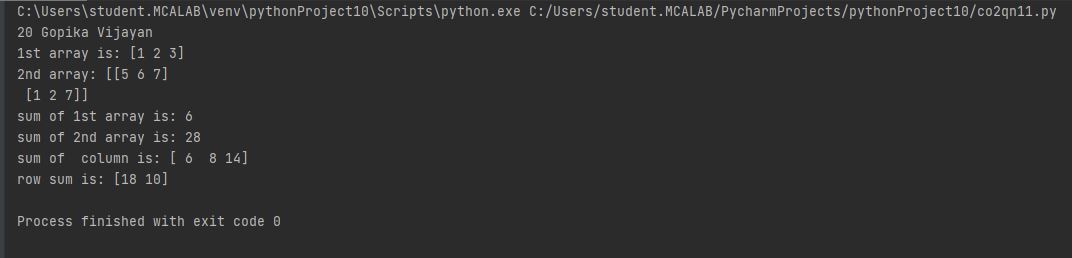
print("sum of 1st array is:",np.sum(ar1))

print("sum of 2nd array is:",np.sum(ar2))

print("sum of column is:",np.sum(ar2,axis=0))

print("row sum is:",np.sum(ar2,axis=1))

Output:



**CYCLE - 2**

1. 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

print("20 Gopika Vijayan")

import numpy as np

n=np.random.randint(15,size=(2,2))

print(n)

print("Inverse Of Matrix is:",np.linalg.inv(n))

print("rank of matrix",np.linalg.matrix\_rank(n))

print("determinant of matrix",np.linalg.det(n))

d=n.flatten(order='c')

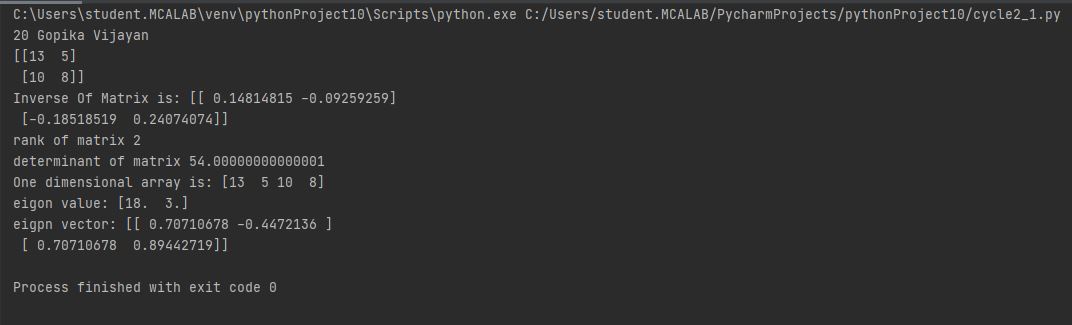
print("One dimensional array is:",d)

u,v=np.linalg.eig(n)

print("eigon value:",u)

print("eigon vector:",v)

Output:



2. 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.

iv) Create a matrix Y with same dimension as X and perform the operation X 2 +2Y

print("20 Gopika Vijayan")

import numpy as np

m=np.array([

[2,3],

[4,5]

]

)

print("Matrix is:",m)

print("cube using ,multiply:",np.multiply(m,np.multiply(m,m)))

print("power using multiply:",np.power(m,3))

print("cube using \*\* :",m\*\*3)

print("cube using \*:",m\*m\*m)

print('identity matrix is:\n',np.identity(2,dtype=int))

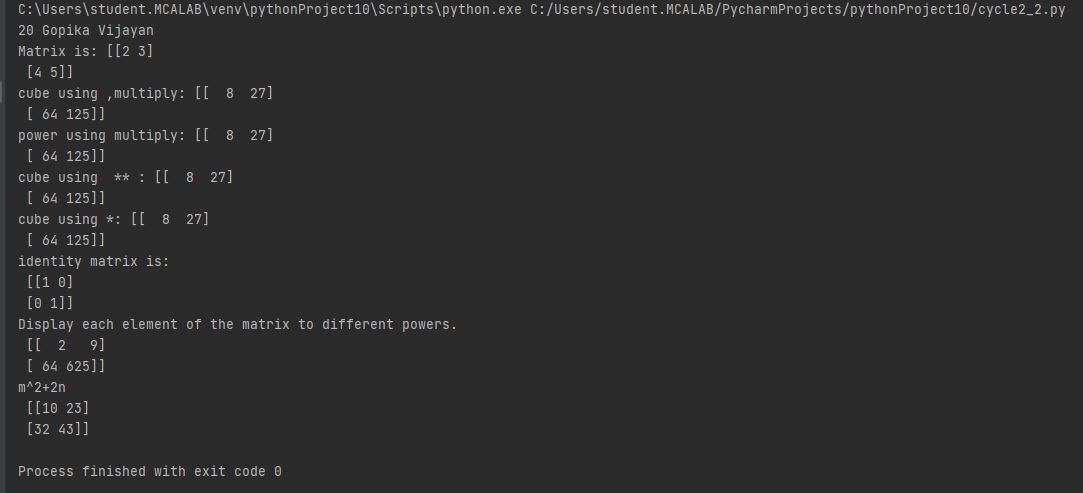
print("Display each element of the matrix to different powers.\n",np.power(m,[[1,2],[3,4]]))

n=np.array([[3,7],

[8,9]])

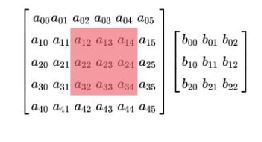
print("m^2+2n\n",(m\*\*2)+(2\*n))

Output:



3.Multiply a matrix with a submatrix of another matrix and replace the same in larger

matrix.



print("20 Gopika Vijayan")

import numpy as np

m1=np.random.randint(0,10,size=(5,6))

m2=np.random.randint(0,10,size=(3,3))

print("Matrix of order 5x6 is:",format(m1))

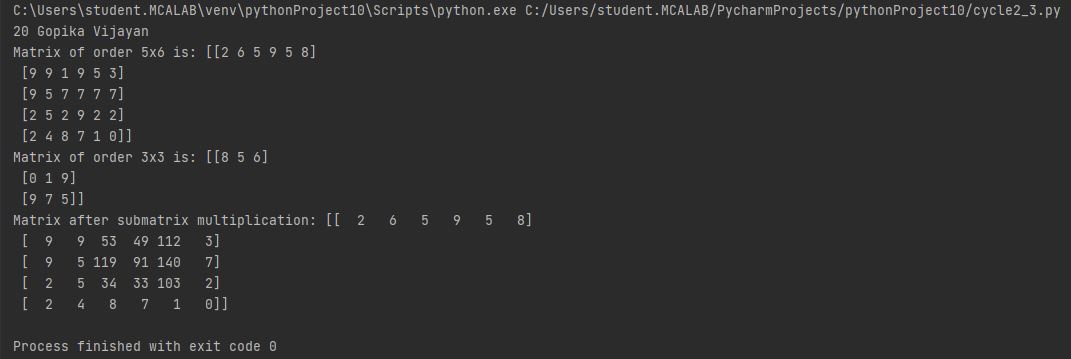
print("Matrix of order 3x3 is:",format(m2))

m3=m1[1:4,2:5]@m2

m1[1:4,2:5]=m3

print("Matrix after submatrix multiplication:",format(m1))

Output:



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

the 3 matrices.

print("20 Gopika Vijayan")

import numpy as np

m1=np.random.randint(0,10,size=(2,2))

m2=np.random.randint(0,10,size=(2,3))

m3=np.random.randint(0,10,size=(3,3))

print("first ,matrix is:",format(m1))

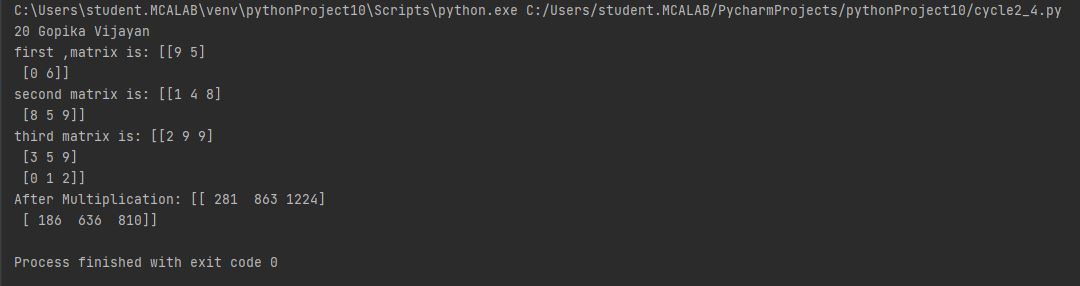
print("second matrix is:",format(m2))

print("third matrix is:",format(m3))

m4=np.matmul(np.matmul(m1,m2),m3)

print("After Multiplication:",m4)

Output:



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

print("20 Gopika Vijayan")

import numpy as np

#m1=np.random.randint(0,10,size=(3,3))

m1=np.matrix([

[1,2,3],

[2,-5,7],

[3,5,7]

])

print("Matrix is:",format(m1))

m2=m1.transpose()

if m1.all() == m2.all():

print("Matrix is symmetric")

else:

print("Matrix is not symmetric")

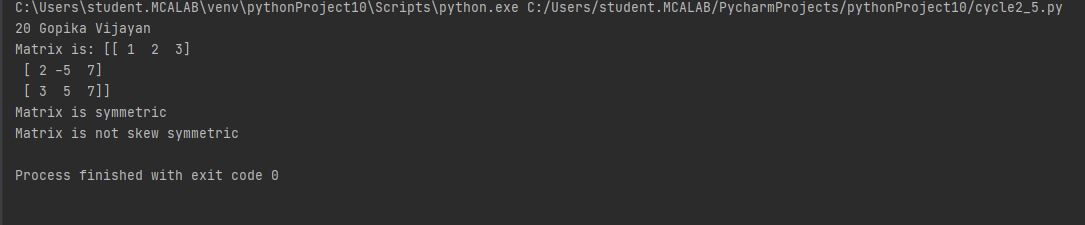
if np.allclose(-m1,m2)==True:

print("Matrix is skew symmetric")

else:

print("Matrix is not skew symmetric")

Output:



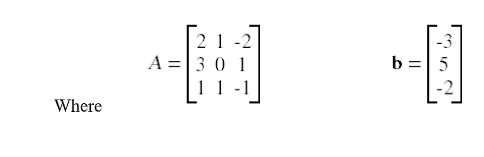
6. Write a program to find out the value of X using solve(), given A and b as below.

Solving systems of equations with numpy

One of the more common problems in linear algebra is solving a matrix-vector equation.

Here is an example. We seek the vector x that solves the equation

A X = b

****

And X=A -1 b.

Numpy provides a function called solve for solving such equations.

print("20 Gopika Vijayan")

import numpy as np

m1=np.array([

[2,1,3],

[4,5,6],

[7,5,1]

])

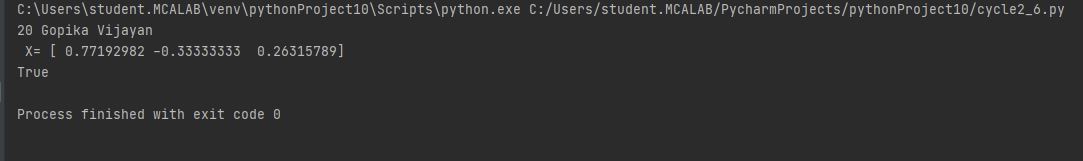
m2=np.array([2,3,4])

x = np.linalg.solve(m1,m2)

print(" X=",x)

print(np.allclose(np.dot(m1, x), m2))

Output:



7. Write a program to perform the SVD of a given matrix. Also reconstruct the given matrix

from the 3 matrices obtained after performing SVD.

Singular value Decomposition

Matrix decomposition, also known as matrix factorization, involves describing a given

matrix using its constituent elements.

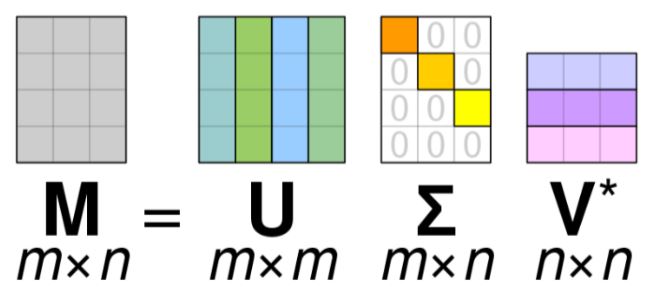
The Singular-Value Decomposition, or SVD for short, is a matrix decomposition method for

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.

M= U ∑V^T



M-is original matrix we want to decompose

U-is left singular matrix (columns are left singular vectors). U columns contain

eigenvectors of matrix MMᵗ

Σ-is a diagonal matrix containing singular (eigen) values.

V-is right singular matrix (columns are right singular vectors). V columns contain

eigenvectors of matrix MᵗM

Numpy provides a function for performing svd, which decomposes the given matrix into 3

Matrices.

print("20 Gopika Vijayan")

import numpy as np

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

print(A)

# Singular-value decomposition

U, s, VT = np.linalg.svd(A)

#print(u)

#print(s,vt)

# create m x n Sigma matrix

Sigma = np.zeros((A.shape[0], A.shape[1]))

# populate Sigma with n x n diagonal matrix

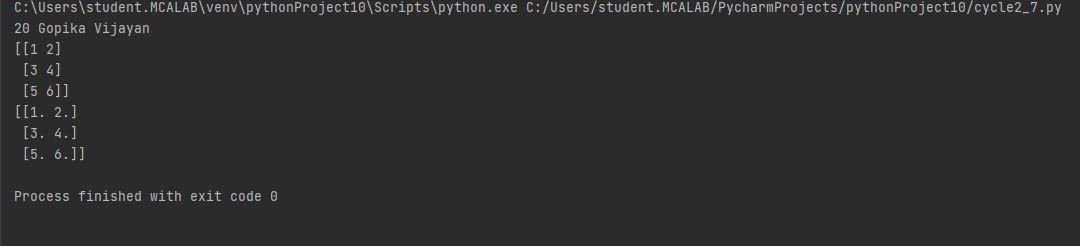
Sigma[:A.shape[1], :A.shape[1]] = np.diag(s)

# reconstruct matrix

B = U.dot(Sigma.dot(VT))

print(B)

Output:



Cycle - 3

Mathplotlib

Demonstrate creating various types of charts and plots using functions in mathplotlib library

1. 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 dashdot and Line-color should be red

point using \* symbol with green color and size 20

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

import numpy as np

x =np.array([2001,2002,2003,2004,2005,2006,2007])

y =np.array([24000,22500,19700,17500,14500,10000,5800])

plt.plot(x,y,linestyle='dashdot', marker = '\*', ms = 20,color = 'red',mfc = "green")

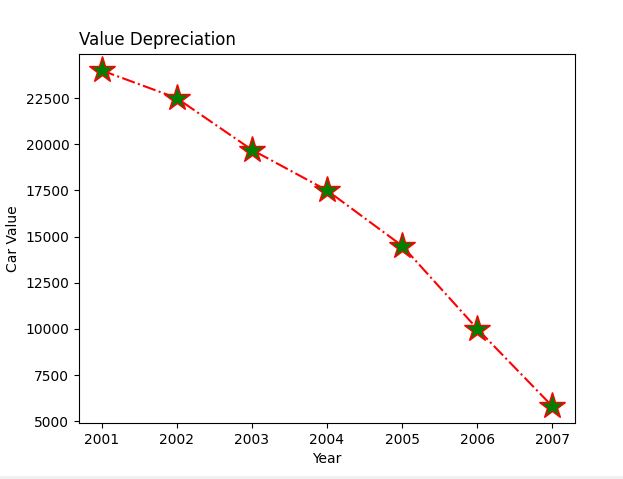
plt.xlabel("Year")

plt.ylabel("Car Value")

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

plt.show()

Output:



Subplot() provides multiple plots in one figure.

1. 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

1. 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

1. 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

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

import numpy as np

x=np.array(['Mon','Tues','Wed','Thurs','Fri'])

y=np.array([300,450,150,400,650])

plt.subplot(1,2,1)

plt.plot(x,y,linestyle = "dotted",color = "cyan",marker = "H",mfc = "magenta",mec ="black")

plt.xlabel("Days of week")

plt.ylabel("Sale of Drinks")

plt.title(label="Sales Data1",loc="right")

plt.grid(color= "blue",linestyle = "dotted")

y=np.array([400,500,350,300,500])

plt.subplot(1,2,2)

plt.plot(x,y,linestyle = "dashed", color = "yellow",marker = "D",mfc = "green",mec = "red")

plt.xlabel("Days of Week")

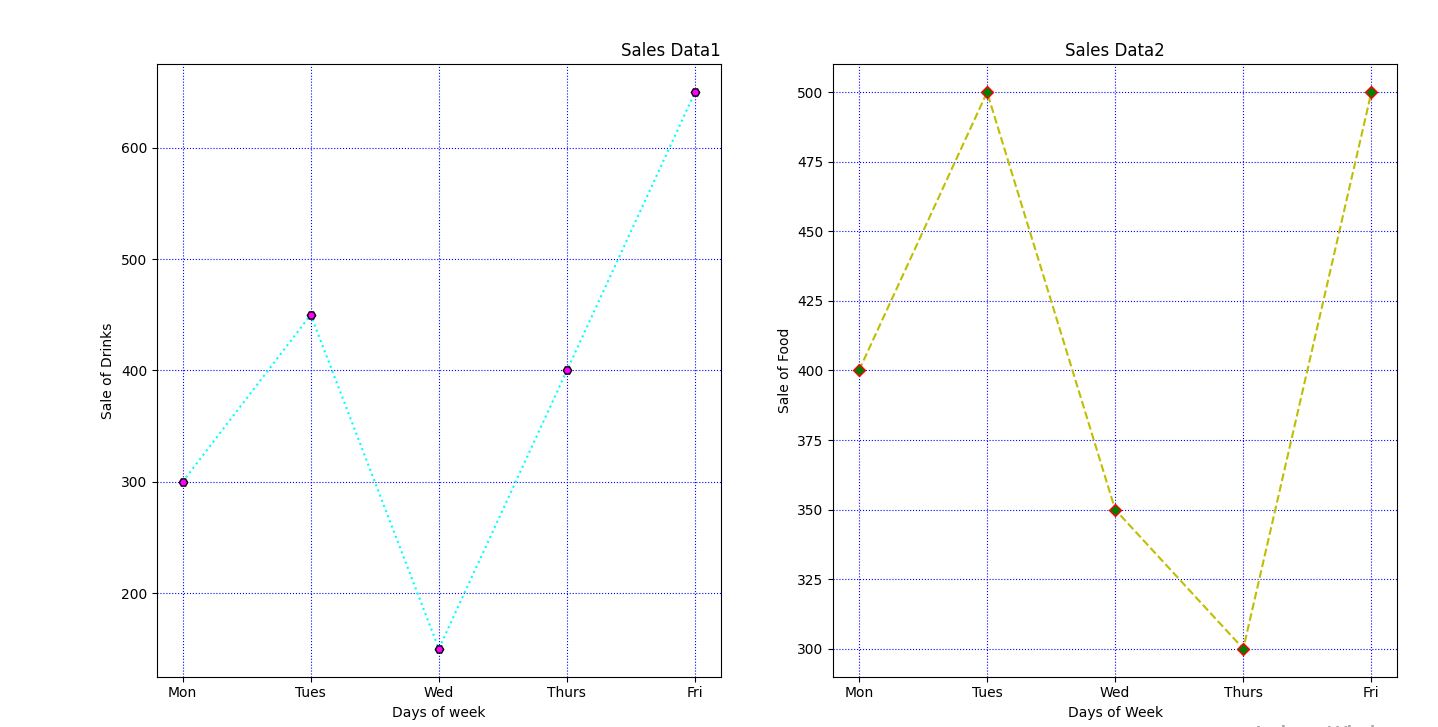
plt.ylabel("Sale of Food")

plt.title(label="Sales Data2",loc="center")

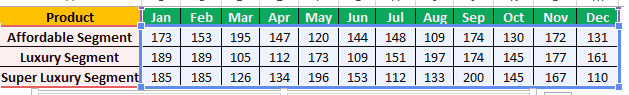
plt.grid(color= "blue",linestyle = "dotted")

plt.show()

Output:



1. 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
* Color for Super luxury segment-blue

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

import numpy as np

x=np.array(['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec'])

y=np.array([173,153,195,147,120,144,148,109,174,130,172,131])

plt.scatter(x,y,color = "hotpink")

y=np.array([189,189,105,112,173,109,151,197,174,145,177,161])

plt.scatter(x,y,color= "yellow")

y=np.array([185,185,126,134,196,153,112,133,200,145,167,110])

plt.scatter(x,y,color="blue")

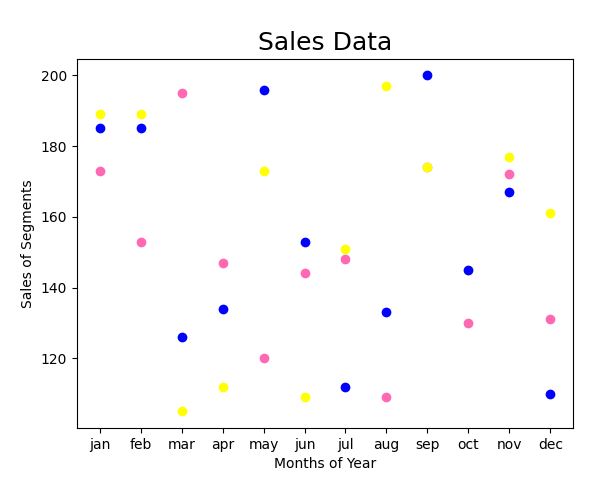
plt.xlabel("Months of Year",fontsize=18)

plt.ylabel("Sales of Segments")

plt.title(label="Sales Data")

plt.show()

Output:



4. 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

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

import numpy as np

x=np.array(['Jan','feb','Mar','Apr','May','jun','jul','Aug','Sep','Oct','Nov','Dec'])

y=np.array([173,153,195,147,120,144,148,109,174,130,172,131])

plt.plot(x,y, label = "line 1", linestyle="-")

y=np.array([189,189,105,112,173,109,151,197,174,145,177,161])

plt.plot(x,y,label= "line 2",linestyle=":")

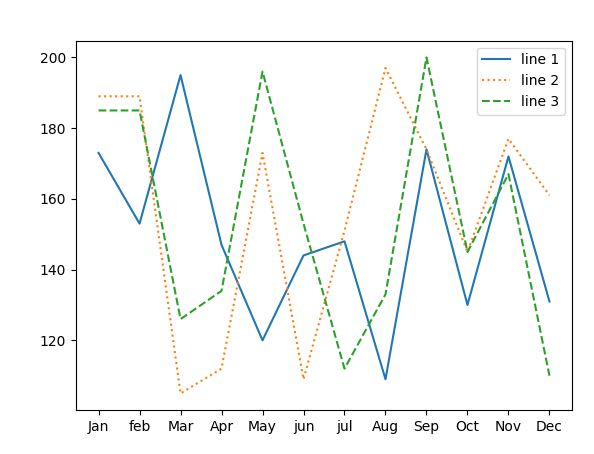
y=np.array([185,185,126,134,196,153,112,133,200,145,167,110])

plt.plot(x,y,label="line 3",linestyle="--")

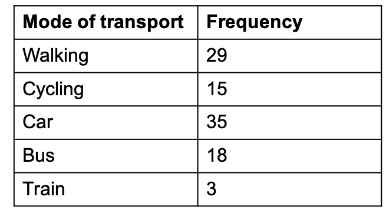
plt.legend()

plt.show()

Output:



5. 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

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

import numpy as np

x=np.array(['Walking','Cycling','Car','Bus','Train'])

y=np.array([29,15,35,18,3])

plt.xlabel("Mode Of transport")

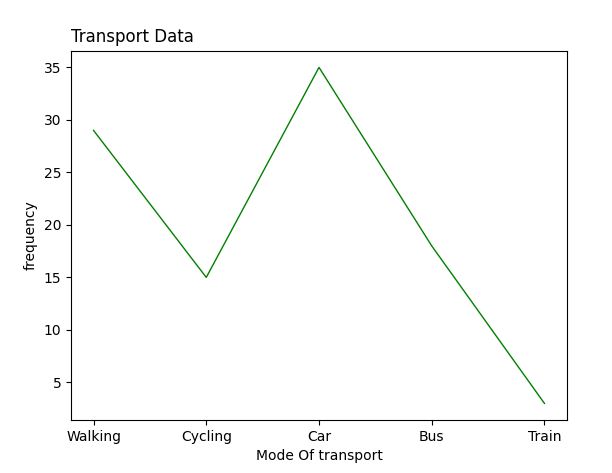
plt.ylabel("frequency")

plt.title(label = "Transport Data",loc="left")

plt.plot(x,y,linewidth=1,color="g")

plt.show()

Output:



6. 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

print("20 Gopika Vijayan")

import matplotlib.pyplot as plt

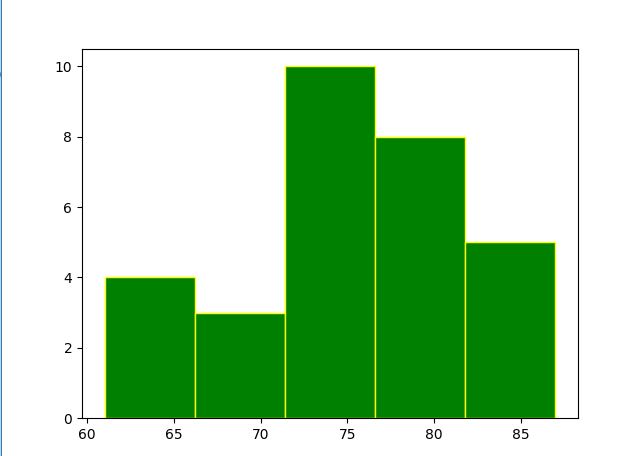
import numpy as np

x=np.array([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=5,edgecolor="yellow",color="green")

plt.show()

Output:



**CYCLE - 4**

DATA HANDLING USING ‘Pandas’ and DATA VISUALIZATION USING ‘Seaborn’

Using the pandas function read\_csv(), read the given ‘iris’ data set.

1. Use appropriate functions in pandas to display

(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

print("20 Gopika Vijayan")

import pandas as pd

df = pd.read\_csv('iris.csv')

print(df.to\_string())

shape=df.shape

#q1

print("Shape of the data set",shape)

print("\n")

print("First 5 and last five rows of data set(head)",df.head())

print("\n")

print("First 5 and last five rows of data set(tail)",df.tail())

print("\n")

print("Size of dataset",df.size)

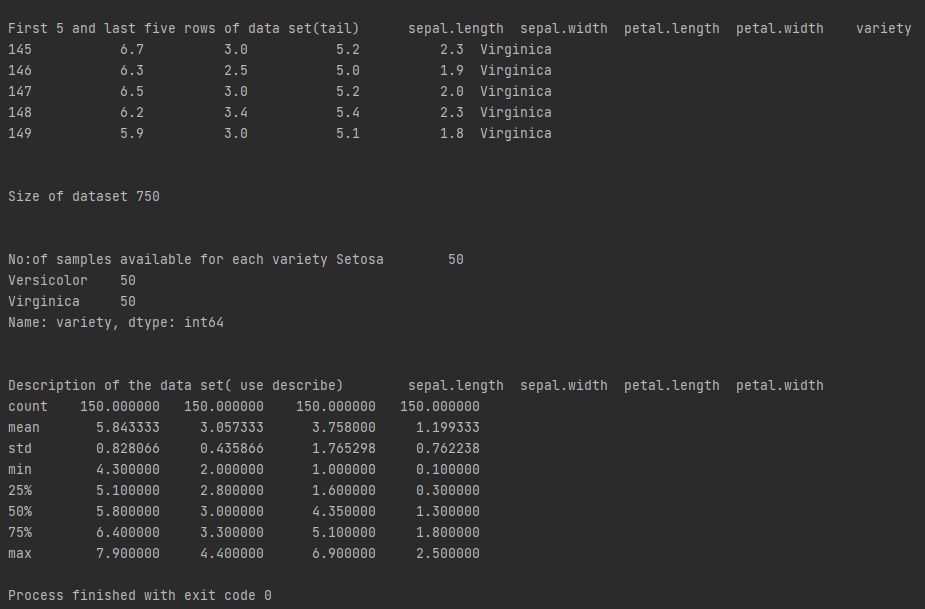
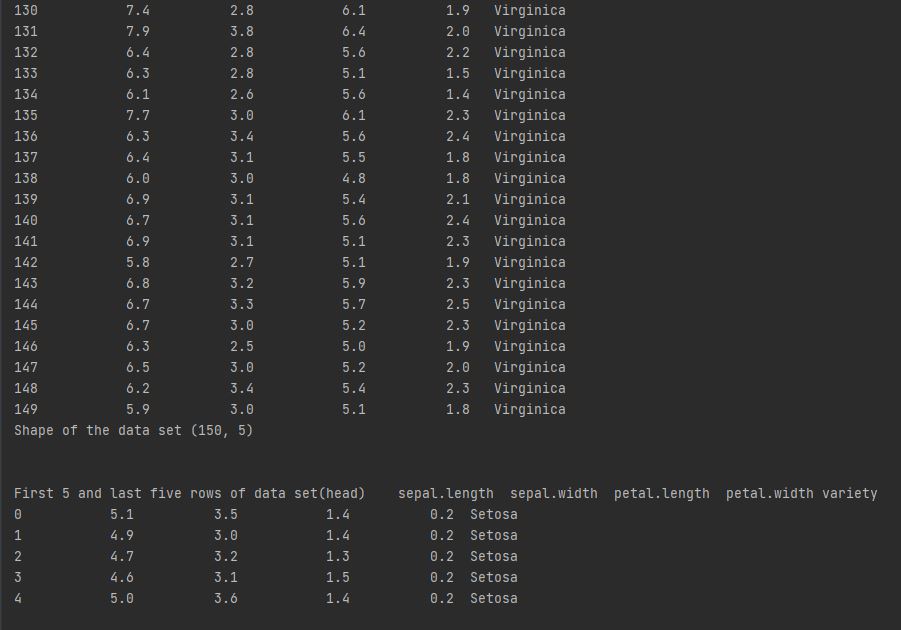
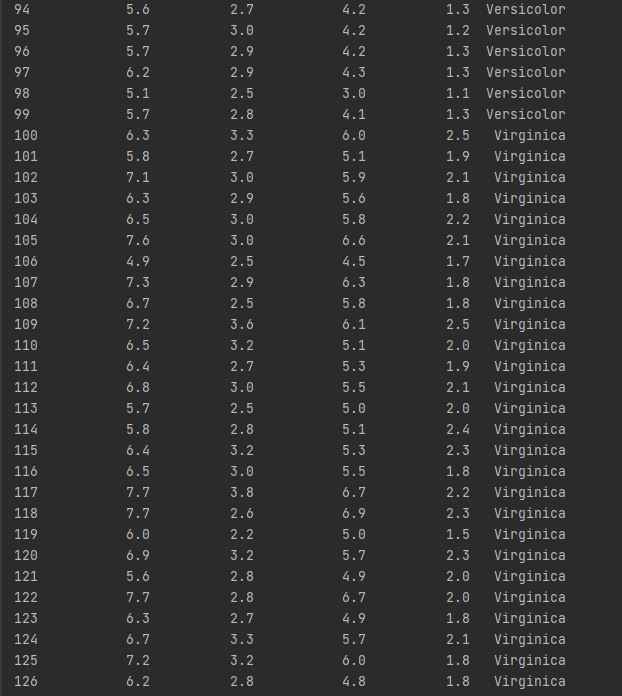
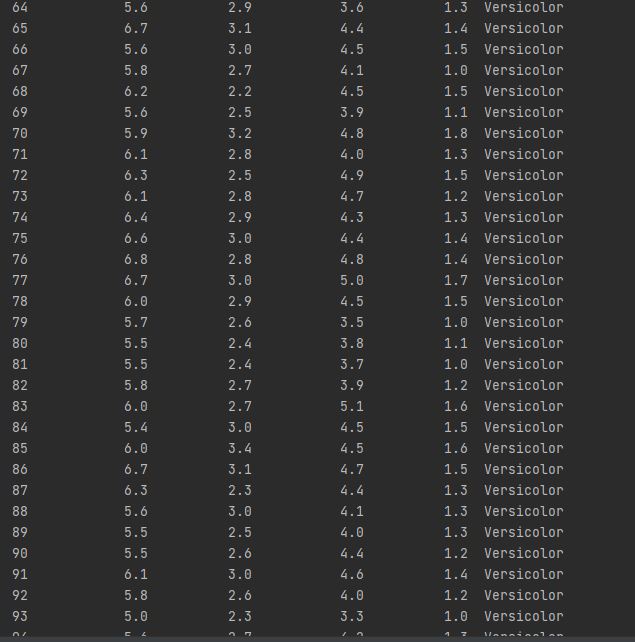
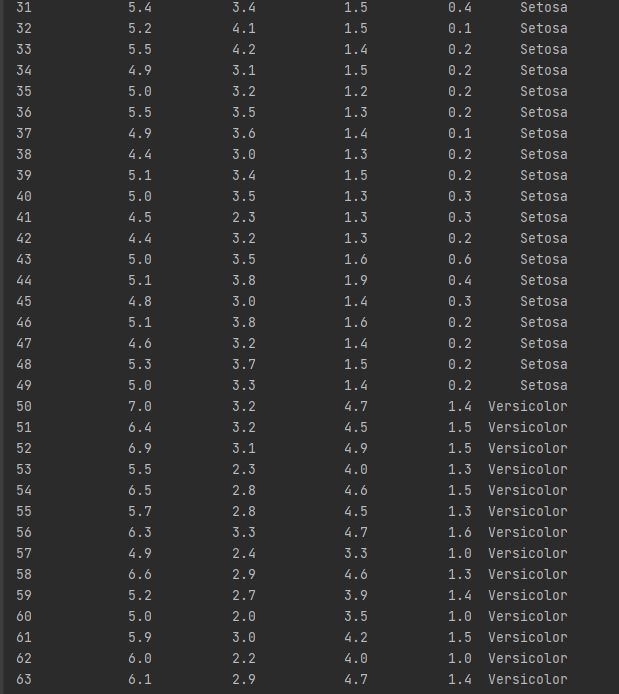
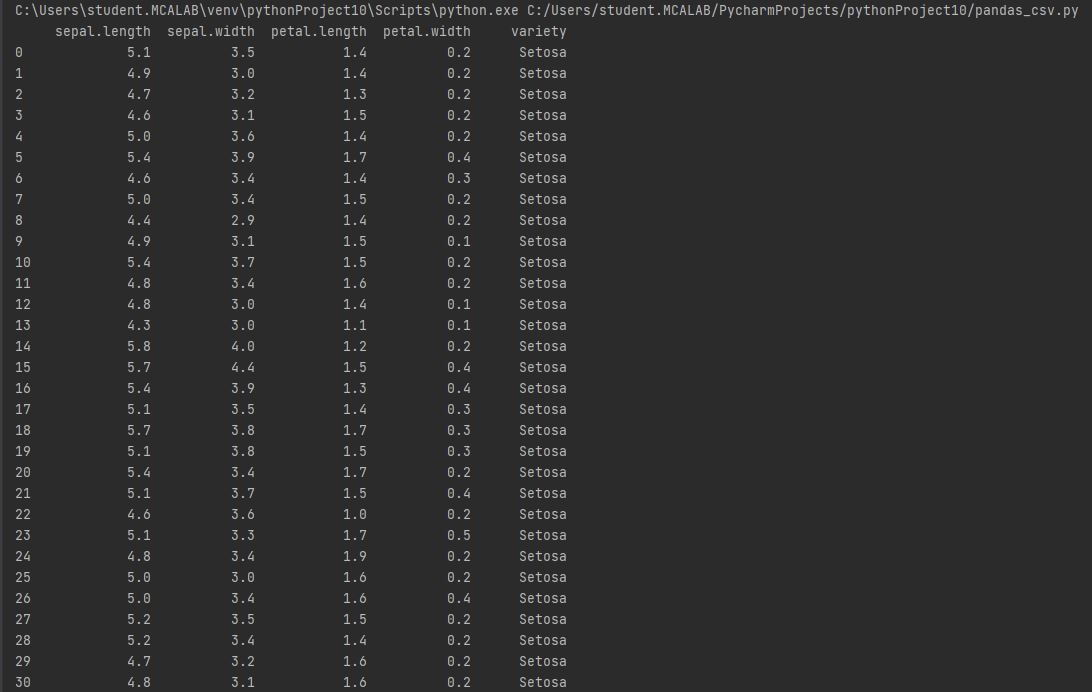
print("\n")

print("No:of samples available for each variety",df["variety"].value\_counts())

print("\n")

print("Description of the data set( use describe)",df.describe())

OUTPUT:



2. Use pairplot() function to display pairwise relationships between attributes. Try

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

print("20 Gopika Vijayan")

import pandas as pd

import seaborn as sns

import matplotlib .pyplot as plt

data=pd.read\_csv("iris.csv")

iris = sns.load\_dataset("iris")

plot=sns.pairplot(iris)

sns.pairplot(iris, hue="species", kind="hist")

plt.show()

sns.pairplot(iris, kind="kde")

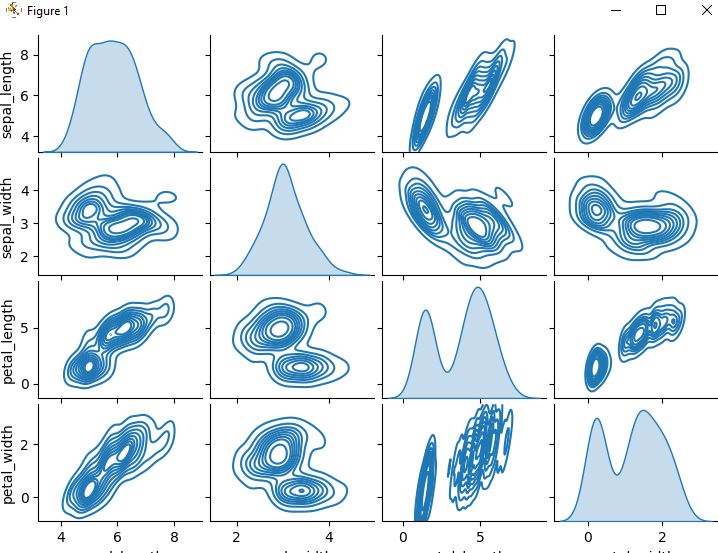
plt.show()

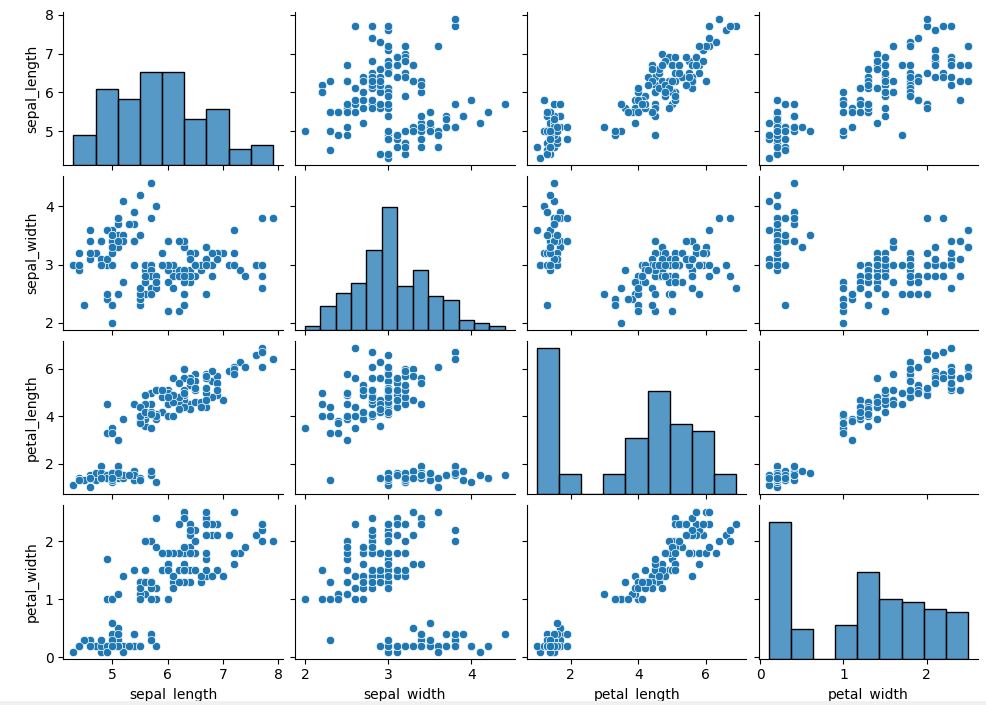
sns.pairplot(iris, kind="reg", hue="species")

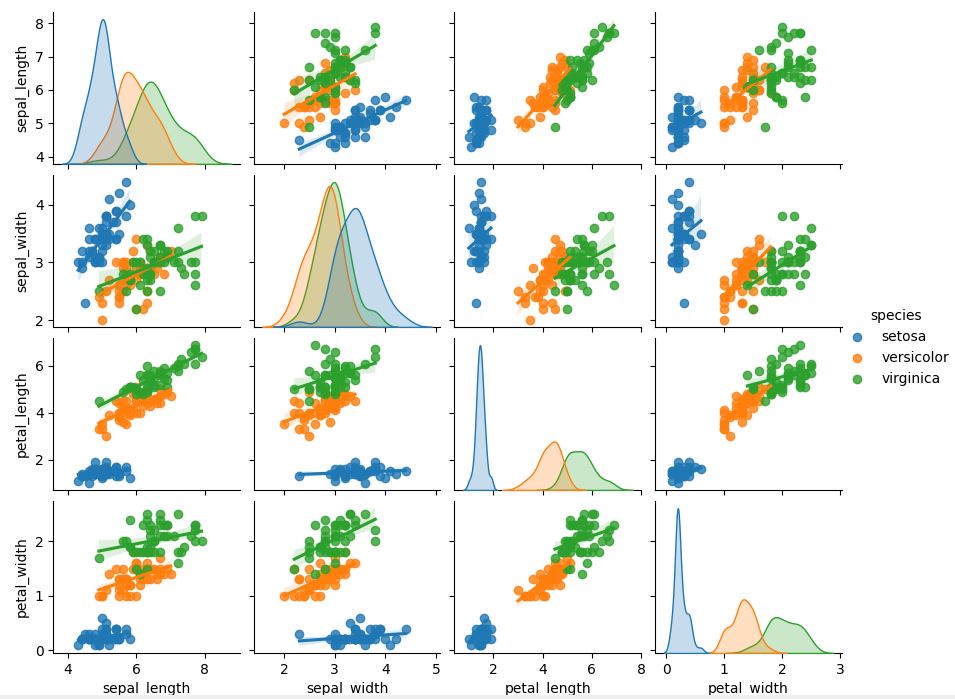
plt.show()

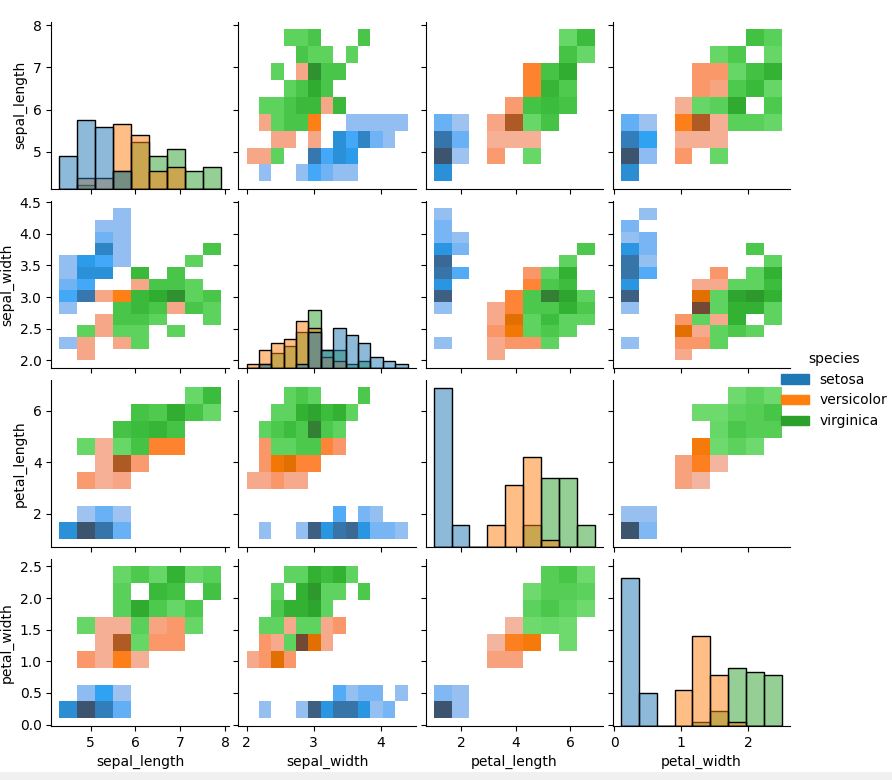
sns.pairplot(iris,kind="scatter")

plt.show()









3. using the iris data set,get familiarize with functions:

1)displot()

2) histplot()

3) relplot()

**print("20 Gopika Vijayan")**

**import pandas as pd**

**import seaborn as sns**

**import matplotlib .pyplot as plt**

**data=pd.read\_csv("iris.csv")**

**iris = sns.load\_dataset("iris")**

**plot=sns.pairplot(iris)**

**sns.displot(iris)**

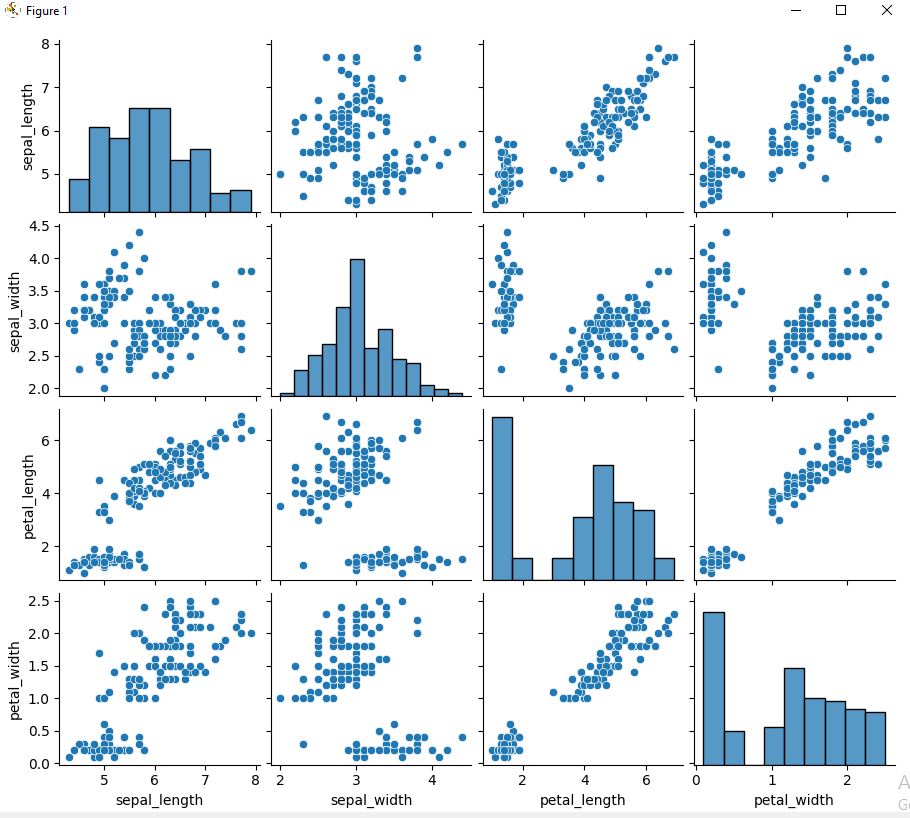
**plt.show()**

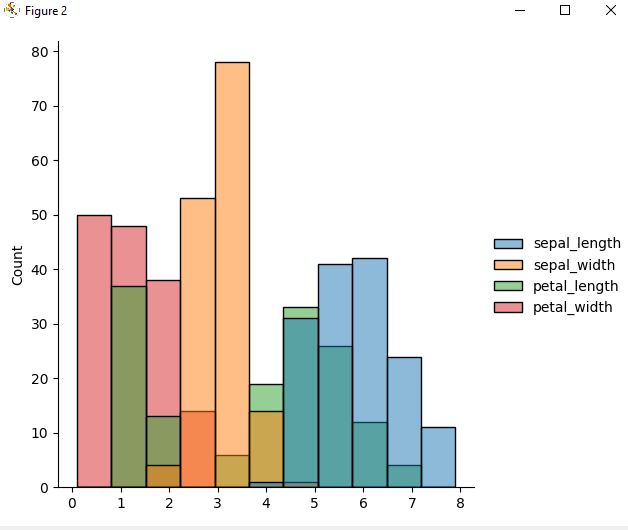
**sns.histplot(iris)**

**plt.show()**

**sns.relplot(iris)**

Output:

****

****

**NEW QUESTIONS**

1. 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 create 6 clusters of customers based on their spending pattern.Visualize the same in a scatter plot with each cluster in a different color scheme.

print("20 Gopika Vijayan")

import pandas as pd

from matplotlib import pyplot as plt

from sklearn.cluster import KMeans

customers = pd.read\_csv('customer\_data.csv')

customers.head()

points = customers.iloc[:, 3:5].values

x = points[:, 0]

y = points[:, 1]

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

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

plt.ylabel('Spending Score')

plt.show()

kmeans = KMeans(n\_clusters=6, random\_state=0)

kmeans.fit(points)

predicted\_cluster\_indexes = kmeans.predict(points)

plt.scatter(x, y, c=predicted\_cluster\_indexes, s=50, alpha=0.7, cmap='viridis')

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c='red', s=100)

plt.show()

Output:

