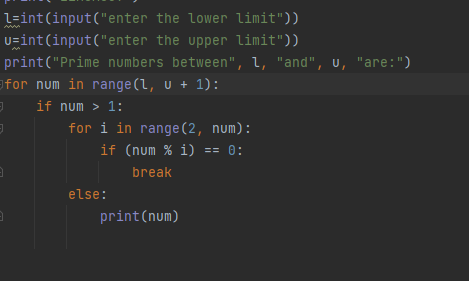
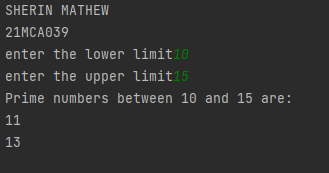
**DATA SCEINCE & MACHINE LEARNING:**

**LAB CYCLE 1**

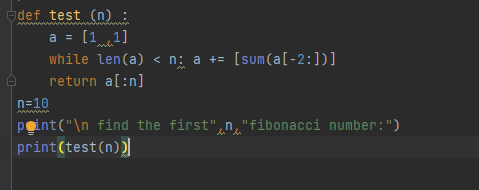
1. **Program to Print all non-Prime Numbers in an Interval**



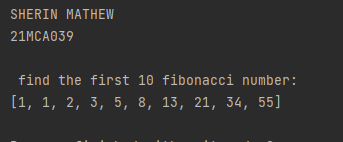
**OUTPUT**



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

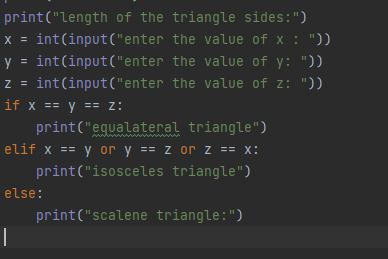


**OUTPUT**

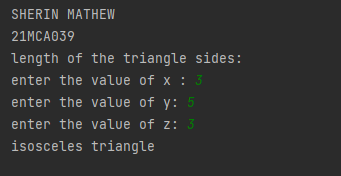


**3.** **Given sides of a triangle, write a program to check whether given triangle is an**

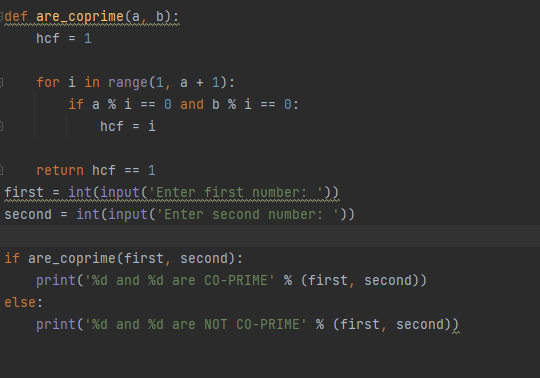
**isosceles, equilateral or scalene.**



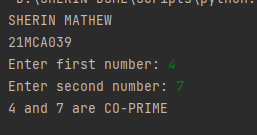
**OUTPUT**



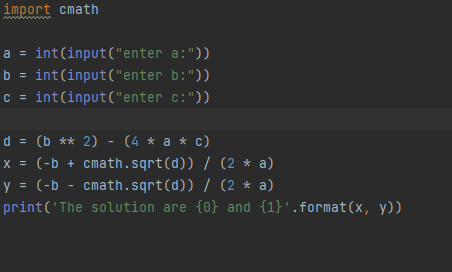
**4.Program to check whether given pair of number is coprime**



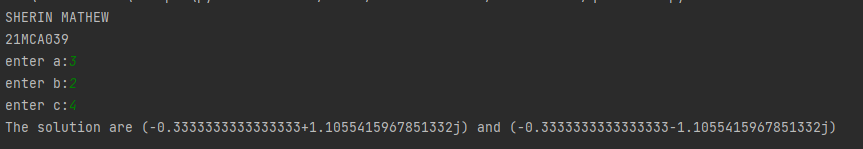
**OUTPUT**



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

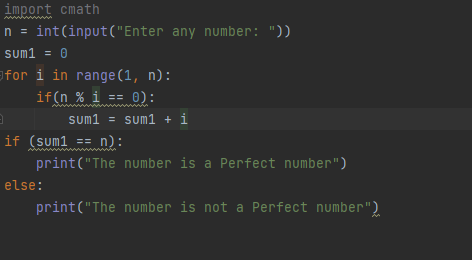


**OUTPUT**

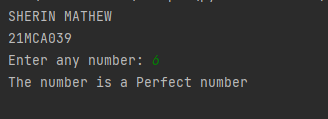


**6.** **Program to check whether a given number is perfect number or not(sum of factors**

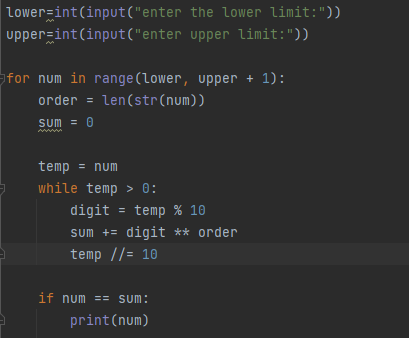
**=number)**



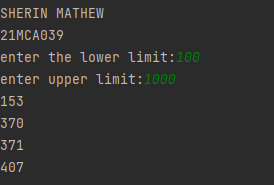
**OUTPUT**



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

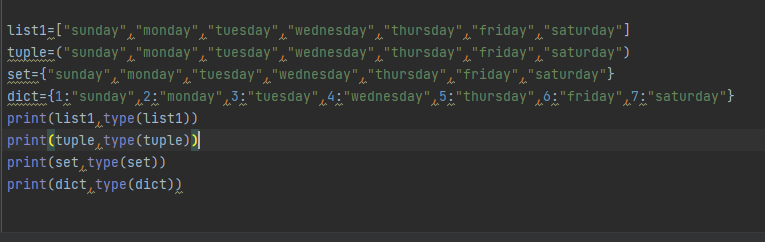


**OUTPUT:**

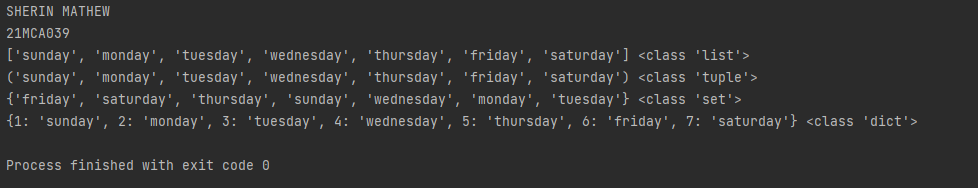


**8.** **Store and display the days of a week as a List, Tuple, Dictionary, Set. Also**

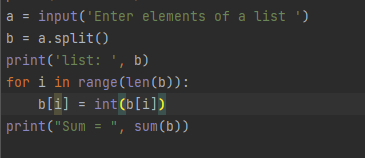
**demonstrate different ways to store values in each of them. Display its type also.**



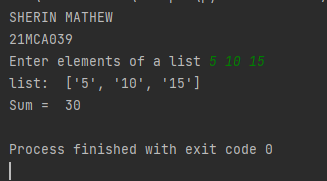
**OUTPUT:**



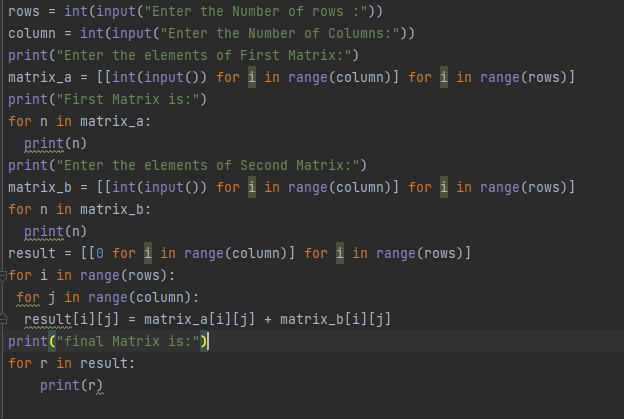
**9.** **Write a program to add elements of given 2 lists**



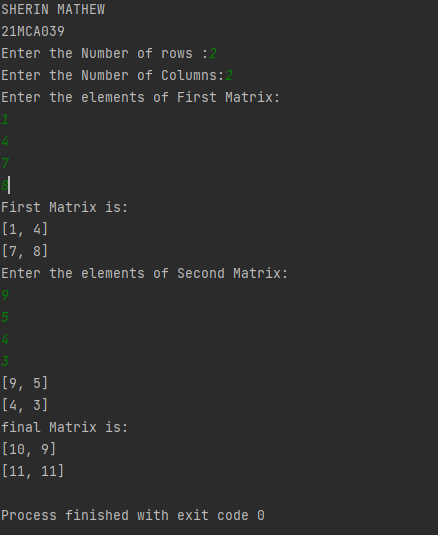
**OUTPUT**



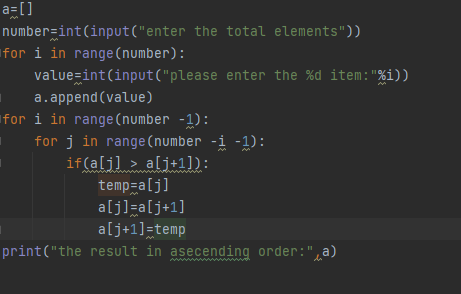
**10.** **Write a program to find the sum of 2 matrices using nested List.**



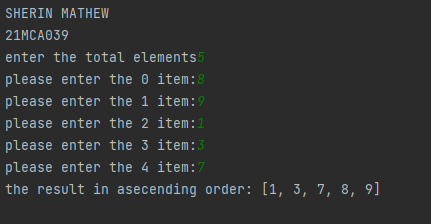
**OUTPUT:**



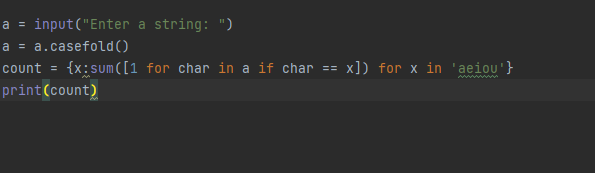
**11.** **Write a program to perform bubble sort on a given set of elements.**



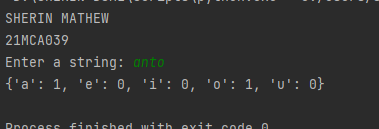
**OUTPUT**



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



**OUTPUT**



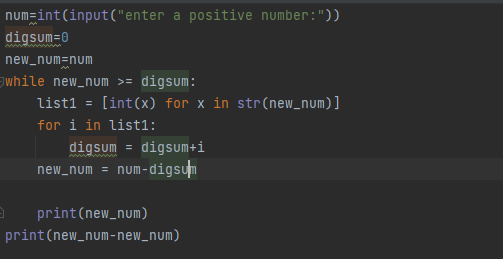
**13.** **Write a Python program that accept a positive number and subtract from this**

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

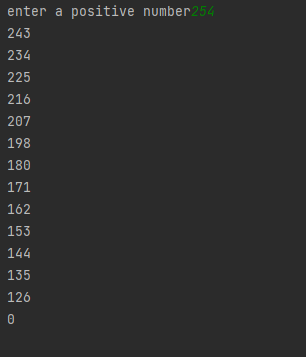
**positive(eg: 256-&gt;2+5+6=13**

**256-13=243**

**243-9=232……..**

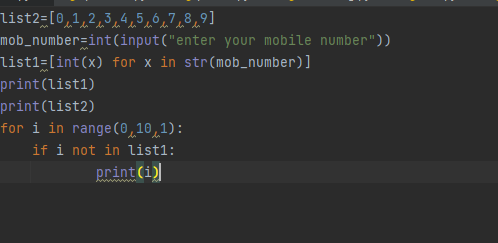


**OUTPUT**

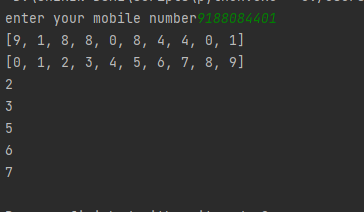


**14. Write a Python program that accepts a 10 digit mobile number, and find the digits**

**which are absent in a given mobile number**

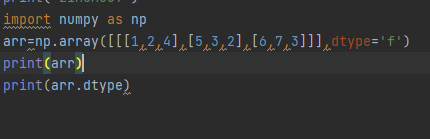


**OUTPUT**

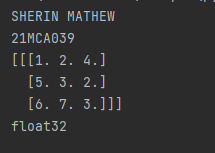


**LAB CYCLE 2**

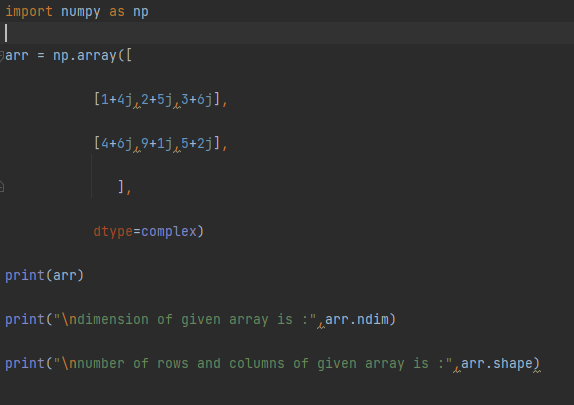
1. **Create a three dimensional array specifying float data type and print it.**



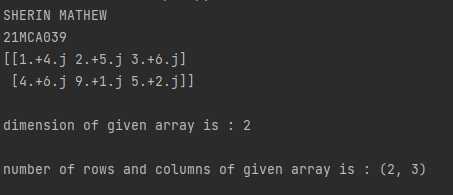
**OUTPUT**



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

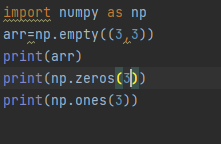


**OUTPUT**

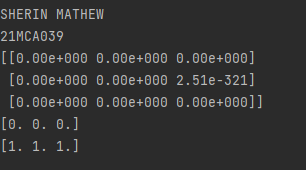


**3.** **Familiarize with the functions to create**

1. **an uninitialized array**
2. **array with all elements as 1,**
3. **all elements as 0**



**OUTPUT**



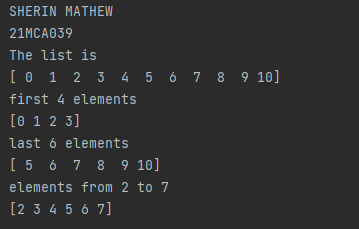
1. **Create an one dimensional array using arrange function containing 10 elements.**

**Display**

1. **First 4 elements**
2. **Last 6 elements**
3. **Elements from index 2 to 7**



**OUTPUT**



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

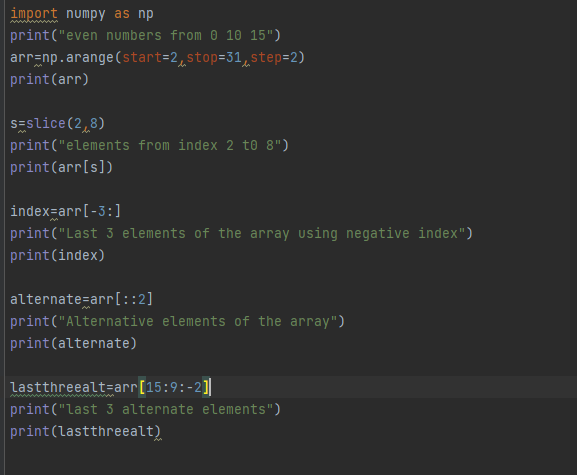
**a. Elements from index 2 to 8 with step 2(also demonstrate the same**

**using slice function)**

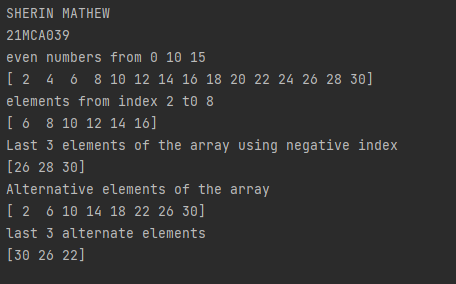
**b. Last 3 elements of the array using negative index**

**c. Alternate elements of the array**

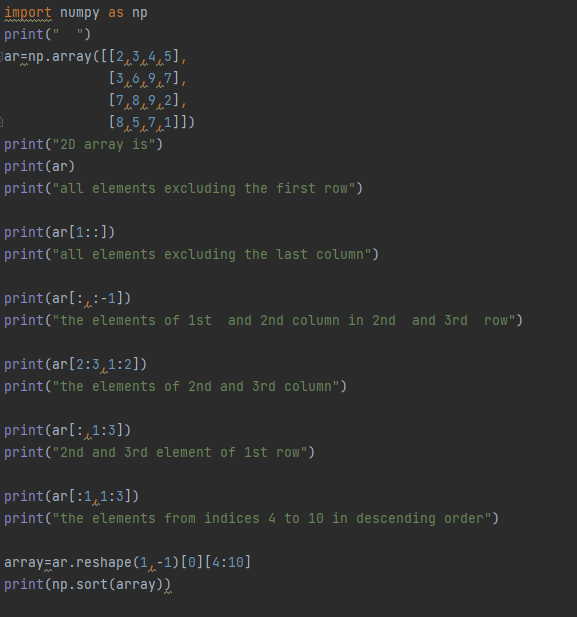
**d. Display the last 3 alternate elements**



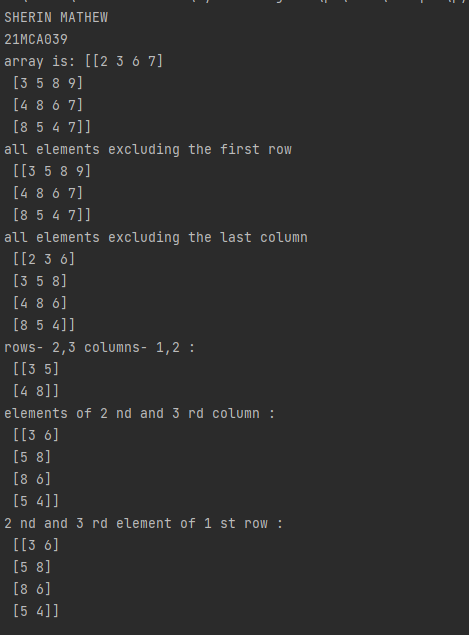
**OUTPUT**



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



**OUTPUT**



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

**a. Add the 2 matrices and print it**

**b. Subtract 2 matrices**

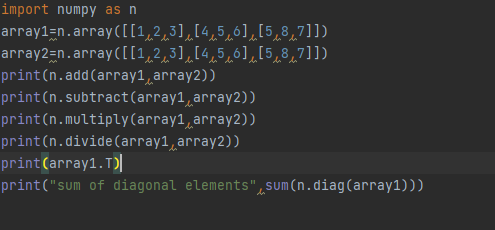
**c. Multiply the individual elements of matrix**

**d. Divide the elements of the matrices**

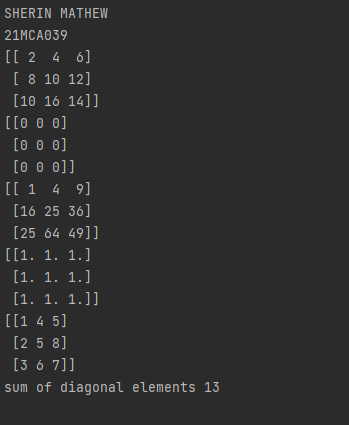
**e. Perform matrix multiplication**

**f. Display transpose of the matrix**

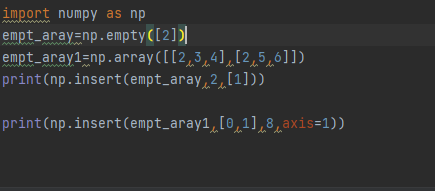
**g. Sum of diagonal elements of a matrix**



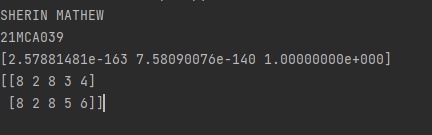
**OUTPUT**



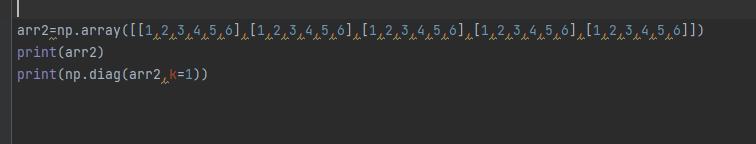
**8. Demonstrate the use of insert() function in 1D and 2D array**



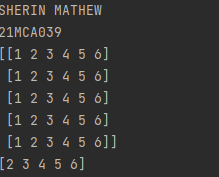
**OUTPUT**



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

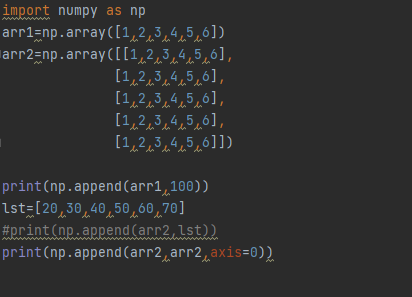


**OUTPUT**

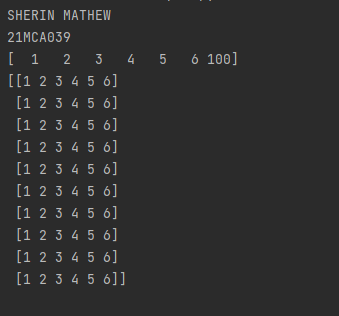


**10. Demonstarte the use of append() function in 1D and 2D**

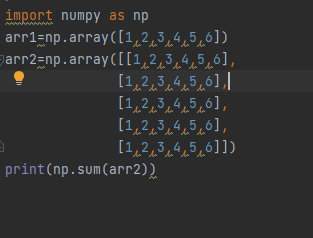
**array.**



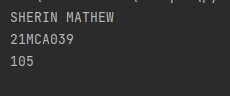
**OUTPUT**



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



**OUTPUT**



**Part B**

**1. Create a square matrix with random integer values(use randint()) and use**

**appropriate functions to find:**

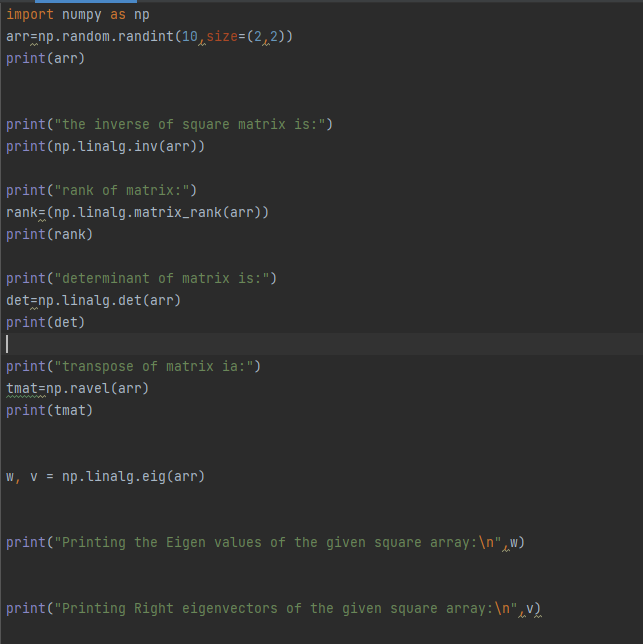
**i) inverse**

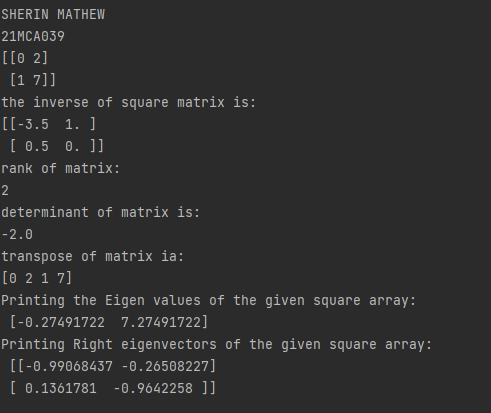
**ii) rank of matrix**

**iii) Determinant**

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

**v) eigen values and vectors**





**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(),\*\*)**

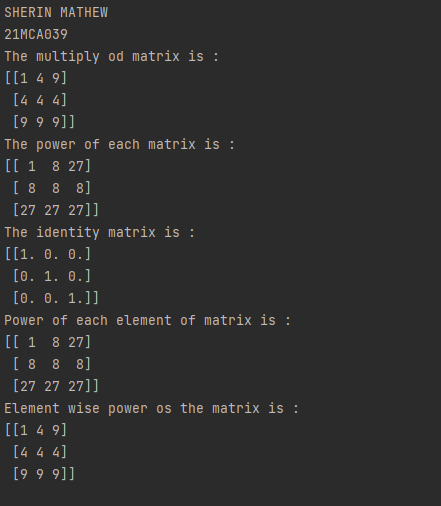
**ii) Display identity matrix of the given square matrix.**

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

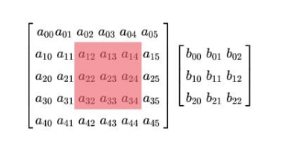
**iv) Create a matrix Y with same dimension as X and perform the operation X 2 +2Y**

import numpy as np  
  
A = np.array([ [1, 2, 3], [2, 2, 2], [3, 3, 3] ])  
#B = np.array([ [3, 2, 1], [1, 2, 3], [1, 2, 3] ])  
  
arrA = np.multiply(A,A)  
print("The multiply od matrix is :")  
print(arrA)  
  
arrB = np.power(A, 3)  
print("The power of each matrix is :")  
print(arrB)  
  
arrC = np.identity(3)  
print("The identity matrix is :")  
print(arrC)  
  
arrD = np.power(A,3)  
print("Power of each element of matrix is : ")  
print(arrD)  
  
arrE=np.power(A,2)  
print("Element wise power os the matrix is :")  
print(arrE)

**OUTPUT**

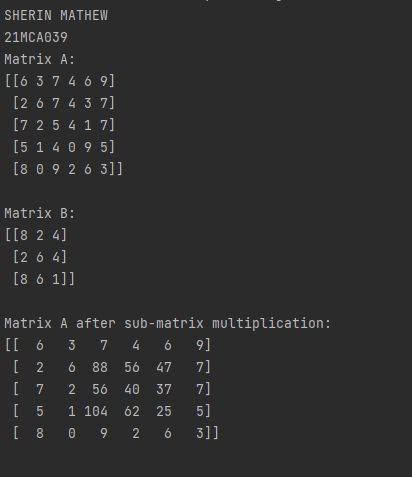


**3.** **Multiply a matrix with a submatrix of another matrix and replace the same in larger matrix.**



import numpy as np  
np.random.seed(42)  
A = np.random.randint(0, 10, size=(5, 6))  
B = np.random.randint(0, 10, size=(3, 3))  
print("Matrix A:\n{}\n".format(A))  
print("Matrix B:\n{}\n".format(B))  
C = A[1:4, 2:5] @ B  
A[1:4, 2:5] = C  
print("Matrix A after sub-matrix multiplication:\n{}\n".format(A))

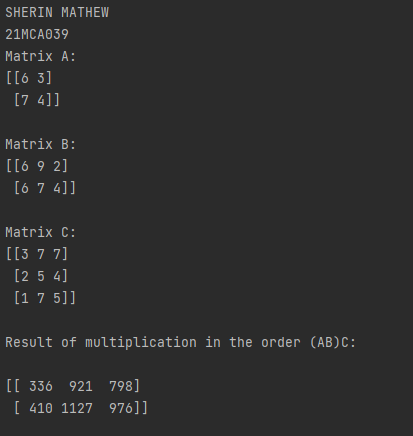
**OUTPUT**



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

import numpy as np  
np.random.seed(42)  
A = np.random.randint(0, 10, size=(2, 2))  
B = np.random.randint(0, 10, size=(2, 3))  
C = np.random.randint(0, 10, size=(3, 3))  
print("Matrix A:\n{}\n".format(A))  
print("Matrix B:\n{}\n".format(B))  
print("Matrix C:\n{}\n".format(C))  
D = np.matmul(np.matmul(A, B), C)  
print("Result of multiplication in the order (AB)C:\n\n{}\n".format(D)

**OUTPUT**



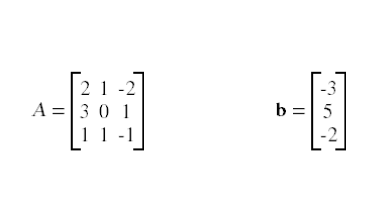
1. **Write a program to check whether given matrix is symmetric or Skew Symmetric.**

**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**

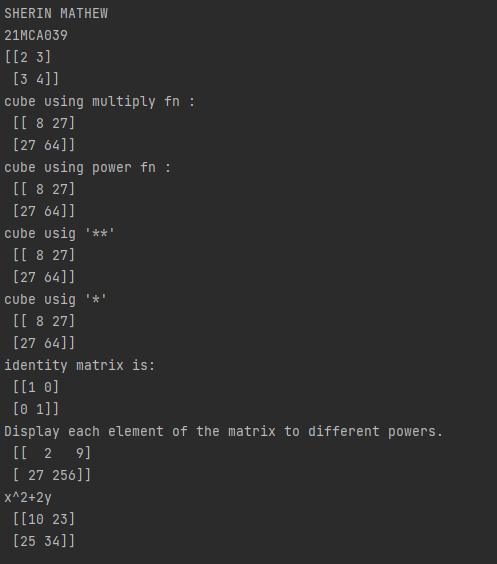


**And X=A -1 b.**

**Numpy provides a function called solve for solving such eauations.**

import numpy as np  
x=np.array([[2,3],  
 [3,4]])  
print(x)  
print("cube using multiply fn :\n",np.multiply(x,np.multiply(x,x)))  
print("cube using power fn :\n",np.power(x,3))  
print("cube usig '\*\*'\n",x\*\*3)  
print("cube usig '\*'\n",x\*x\*x)  
print('identity matrix is:\n',np.identity(2,dtype=int))  
print("Display each element of the matrix to different powers.\n",np.power(x,[[1,2],[3,4]]))  
y=np.array([[3,7],  
 [8,9]])  
print("x^2+2y\n",(x\*\*2)+(2\*y))

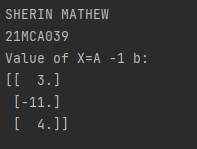
**OUTPUT**



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

import numpy as np  
A = np.array([[2, 1, -2],  
 [3, 0, 1],  
 [1, 1, -1]])  
  
b = np.array([[-3],  
 [5],  
 [-2]])  
a=np.linalg.inv(A)  
x= np.linalg.solve(a, b)  
print("Value of X=A -1 b: ")  
print(x)

**OUTPUT**

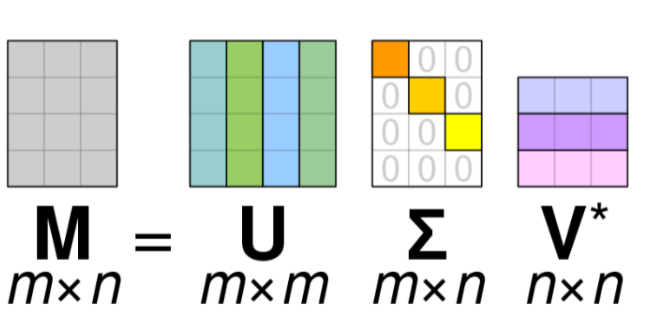


**7.Write a program to perform the SVD of a given matrix. Also reconstruct the given matrix from the 3 matrices obtained after performing SVD.**

**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.**

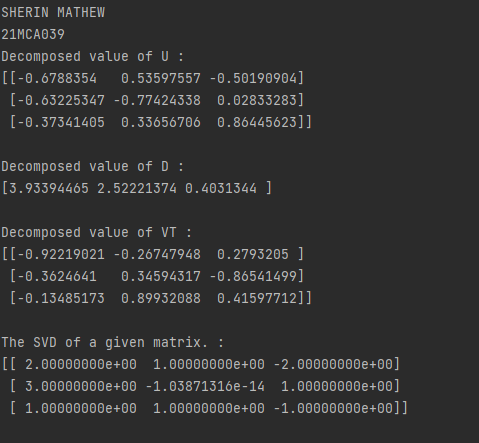
** V-is right singular matrix (columns are right singular vectors). V columns contain**

**eigenvectors of matrix MᵗM**

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

import numpy as np  
A = np.array([[2, 1, -2],  
 [3, 0, 1],  
 [1, 1, -1]])  
  
U, D, VT = np.linalg.svd(A)  
print("Decomposed value of U :")  
print(U)  
print()  
print("Decomposed value of D :")  
print(D)  
print()  
print("Decomposed value of VT :")  
print(VT)  
print()  
  
A\_remake = (U @ np.diag(D) @ VT)  
print("The SVD of a given matrix. :")  
print(A\_remake)

**OUTPUT**



**COURSE OUTCOME-3**

**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&#39;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**

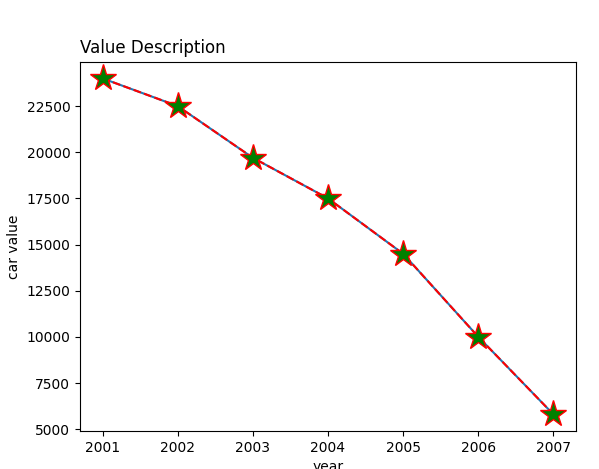
**Y –axis - Car Value**

** title –Value Depreciation (left Aligned)**

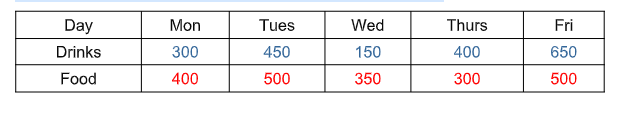
** Line Style dashdot and Line-color should be red**

** point using \* symbol with green color and size 20**

print("SHERIN MATHEW")  
print("21mca039")  
from matplotlib import 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)  
plt.xlabel("year")  
plt.ylabel("car value")  
plt.title("Value Description",loc='left')  
plt.plot(x, y, linestyle='dashed',color='r',marker='\*',markersize='20',markerfacecolor='green')  
plt.show()



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



**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**

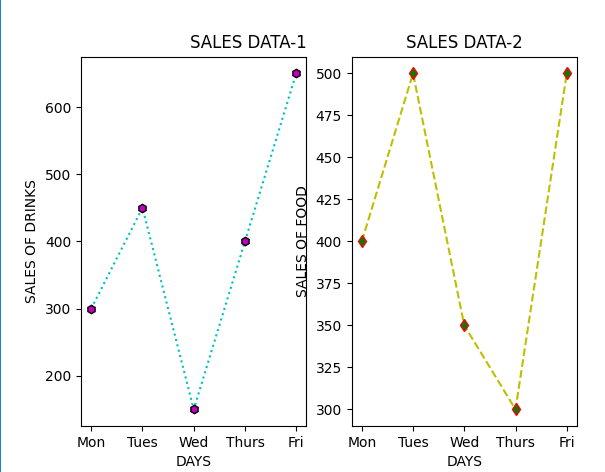
** Y label-Sale of Food**

** Title-Sales Data2 ( center aligned)**

** Line –dashed with yellow color**

** Points- diamond shape with color green and outline red**

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,color='c',linestyle='dotted',marker='h',mfc='m',mec='k')  
plt.xlabel('DAYS')  
plt.ylabel("SALES OF DRINKS")  
plt.title("SALES DATA-1",loc='right')  
x=np.array(['Mon','Tues','Wed','Thurs','Fri'])  
y=np.array([400,500,350,300,500])  
plt.subplot(1,2,2)  
plt.plot(x,y,color='y',linestyle='dashed',marker='d',mfc='g',mec='r')  
plt.xlabel('DAYS')  
plt.ylabel("SALES OF FOOD")  
plt.title("SALES DATA-2",loc='center')  
plt.show()



**3. Create scatter plot for the below data:(use Scatter function)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Product** | **Jan** | **Feb** | **Mar** | **Apr** | **May** | **Jun** | **Jul** | **Aug** | **Sep** | **Oct** | **Nov** | **Dec** |
| **Affordable Segment** | **173** | **153** | **195** | **147** | **120** | **144** | **148** | **109** | **174** | **130** | **172** | **131** |
| **Luxury Segment** | **189** | **189** | **105** | **112** | **173** | **109** | **151** | **197** | **174** | **145** | **177** | **161** |
| **Super Luxury Segment** | **185** | **185** | **126** | **134** | **196** | **153** | **112** | **133** | **200** | **145** | **167** | **110** |

**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**

import matplotlib.pyplot as plt  
x=['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']  
y=[173,153,195,147,120,144,148,109,174,130,172,131]  
plt.scatter(x,y,color='r')  
plt.xlabel('months of year',fontsize=18)  
plt.ylabel('sales of segments')  
plt.title(' sales data')  
  
x=['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']  
y=[189,189,105,112,173,109,151,197,174,145,177,161]  
plt.scatter(x,y,color='c')  
  
x=['jan','feb','mar','apr','may','jun','jul','aug','sep','oct','nov','dec']  
y=[185,185,126,134,196,153,112,133,200,145,167,110]  
plt.scatter(x,y,color='k')  
  
plt.show()

**OUTPUT**

