**FINAL PRACTICAL FILE**

**Name:** Shivang Shukla

**Semester:** 6th Semester

**Course:** BSc. Hons. Computer Science

**Roll No.:** 21058570054

**Que1** Perform elementary mathematical operations in R like addition, multiplication, division and exponentiation.

**Ans**

sum <-1+2

diff <-1-2

prod <-1\*2

quot <-1/2

exp <-2^6

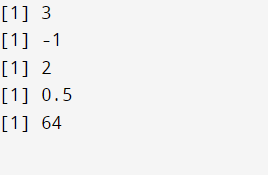
print(sum)

print(diff)

print(prod)

print(quot)

print(exp)

****

**Que2** Perform elementary logical operations in R like OR, AND, checking for equality, NOT and XOR.

**Ans**

and <-TRUE&&FALSE

or <-TRUE||FALSE

not <-!TRUE

ne <-1!=2

xor <-xor(FALSE,TRUE)

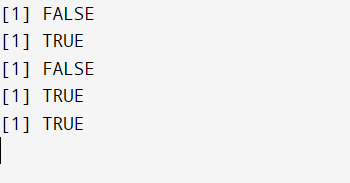
print(and)

print(or)

print(not)

print(ne)

print(xor)



**Que3** Create, initialize and display simple variables and simple strings and use simple formatting for variables.

**Ans**

integer\_var = 42

float\_var = 3.14

boolean\_var = True

string\_var = "Hello, world!"

formatted\_string = "The value of integer\_var is: {}, and the value of float\_var is: {:.2f}".format(integer\_var, float\_var)

print("Simple Variables:")

print("Integer Variable:", integer\_var)

print("Float Variable:", float\_var)

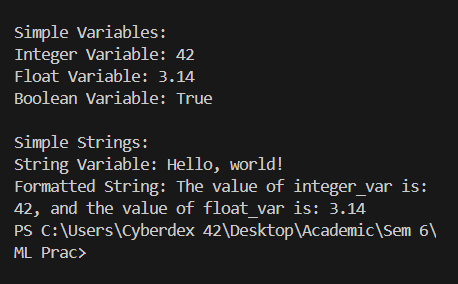
print("Boolean Variable:", boolean\_var)

print()

print("Simple Strings:")

print("String Variable:", string\_var)

print("Formatted String:", formatted\_string)



**Que4** Create single dimension/ multidimensional arrays, and arrays with specific values like array of all ones, all zeros, array with random values within a range, or a diagonal matrix.

**Ans**

import numpy as np

single\_dim\_array = np.array([1, 2, 3, 4, 5])

multi\_dim\_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

ones\_array = np.ones((3, 3))

zeros\_array = np.zeros((2, 4))

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

diagonal\_matrix = np.diag([1, 2, 3, 4, 5])

print("Single-dimensional Array:")

print(single\_dim\_array)

print()

print("Multi-dimensional Array:")

print(multi\_dim\_array)

print()

print("Array of All Ones:")

print(ones\_array)

print()

print("Array of All Zeros:")

print(zeros\_array)

print()

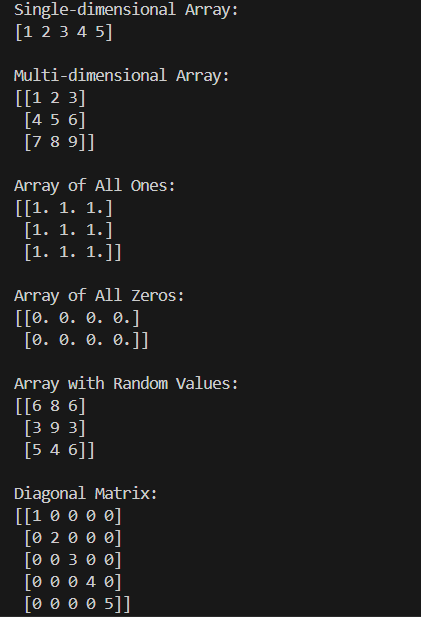
print("Array with Random Values:")

print(random\_array)

print()

print("Diagonal Matrix:")

print(diagonal\_matrix)



**Que5** Use commands to compute the size of a matrix, size of a particular row/column, load data from a text file, store matrix data to a text file, finding out variables and their features in the current scope.

**Ans**

import numpy as np

def compute\_matrix\_size(matrix):

return matrix.shape

def row\_size(matrix):

return matrix.shape[1]

def load\_matrix\_from\_file(filename):

matrix = np.loadtxt(filename)

return matrix

def store\_matrix\_to\_file(matrix, filename):

np.savetxt(filename, matrix)

def variables\_in\_scope():

variables = globals().copy()

return variables.keys()

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

matrix\_size = compute\_matrix\_size(matrix)

print("Size of matrix:", matrix\_size)

row\_index = 1

row\_size = row\_size(matrix)

print("Size of row", row\_index, ":", row\_size)

filename = "matrix\_data.txt"

loaded\_matrix = load\_matrix\_from\_file(filename)

print("Loaded matrix from file:")

print(loaded\_matrix)

store\_matrix\_to\_file(matrix, filename)

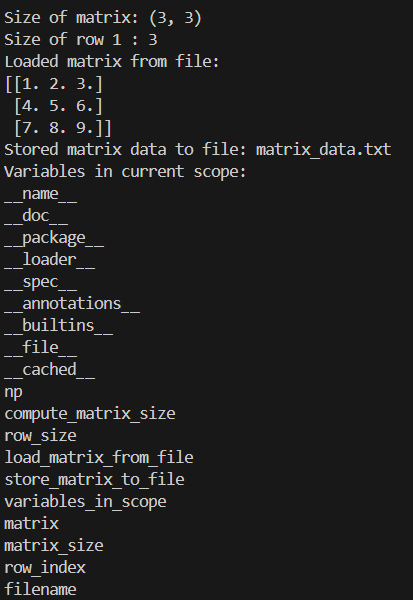
print("Stored matrix data to file:", filename)

print("Variables in current scope:")

variables = variables\_in\_scope()

for variable in variables:

print(variable)



**Que6** Perform basic operations on matrices(like addition, subtraction, multiplication) and display specific rows or columns of the matrix.

**Ans**

def matadd(X,Y):

m,o = len(X),len(Y[0])

n1,n2 = len(X[0]),len(Y)

if(m!=n2 or n1!=o):

raise Exception('Incompatible matrices')

else:

prod = [[0]\*n1 for i in range(m)]

for i in range(m):

for j in range(n1):

prod[i][j] = X[i][j]+Y[i][j]

return prod

def matsub(X,Y):

m,o = len(X),len(Y[0])

n1,n2 = len(X[0]),len(Y)

if(m!=n2 or n1!=o):

raise Exception('Incompatible matrices')

else:

prod = [[0]\*n1 for i in range(m)]

for i in range(m):

for j in range(n1):

prod[i][j] = X[i][j]-Y[i][j]

return prod

def matmul(X,Y):

m,o = len(X),len(Y[0])

n1,n2 = len(X[0]),len(Y)

if(n1!=n2):

raise Exception('Incompatible matrices')

else:

prod = [[0]\*o for i in range(m)]

for i in range(m):

for k in range(o):

for j1 in range(n1):

prod[i][k]+=(X[i][j1]\*Y[j1][k])

return prod

def printMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for i in range(m):

for k in range(n):

print(f'{X[i][k]}',end="\t")

print('\n')

def printCol(X,j):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for i in range(m):

print(f'{X[i][j]}')

def printRow(X,i):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for j in range(n):

print(f'{X[i][j]}',end=" ")

print("\n")

ex = [[1,2],[3,4]]

print(f'Our given matrix(M) is: ')

printMat(ex)

print(f'M + M is: ')

printMat(matadd(ex,ex))

print(f'M - M is: ')

printMat(matsub(ex,ex))

print(f'M \* M is: ')

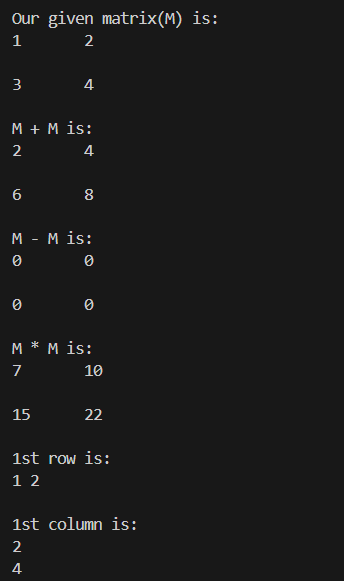
printMat(matmul(ex,ex))

print(f'1st row is: ')

printRow(ex,0)

print(f'1st column is: ')

printCol(ex,1)



**Que7** Perform other matrix operations like converting matrix data to absolute values, taking negative of matrix values, adding/removing rows/columns from a matrix, finding the maximum or minimum values in a a matrix or in a row/ column, and finding the sum of some/all elements in a matrix.

**Ans**

from copy import deepcopy

def printMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for i in range(m):

for k in range(n):

print(f'{X[i][k]}',end="\t")

print('\n')

def absMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

prod = deepcopy(X)

for i in range(m):

for k in range(n):

prod[i][k] = abs(X[i][k])

return prod

def negMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

prod = deepcopy(X)

for i in range(m):

for k in range(n):

prod[i][k] = -(X[i][k])

return prod

def sumMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

sum = 0

for i in range(m):

for k in range(n):

sum+=(X[i][k])

return sum

ex = [[-1,2],[3,-4]]

print("The original matrix is: ")

printMat(ex)

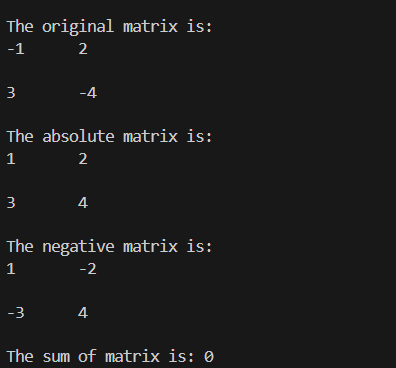
print("The absolute matrix is: ")

printMat(absMat(ex))

print("The negative matrix is: ")

printMat(negMat(ex))

print(f"The sum of matrix is: {sumMat(ex)}")



**Que8** Create various types of plots/charts like histograms, plot based on sine/cosine function based on data from a matrix. Further label different axes in a plot and data in a plot.

**Ans**

from copy import deepcopy

from matplotlib.pyplot import hist,plot,show,legend

from math import sin,cos,pi

def printMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for i in range(m):

for k in range(n):

print(f'{X[i][k]}',end="\t")

print('\n')

ex = [[1,2,3],[2,3,4]]

print("The original matrix is: ")

printMat(ex)

hist(ex[0])

show()

plot([pi\*i/50 for i in range(1,50,1)],[sin(pi\*i/50) for i in range(1,50,1)])

plot([pi\*i/50 for i in range(1,50,1)],[cos(pi\*i/50) for i in range(1,50,1)])

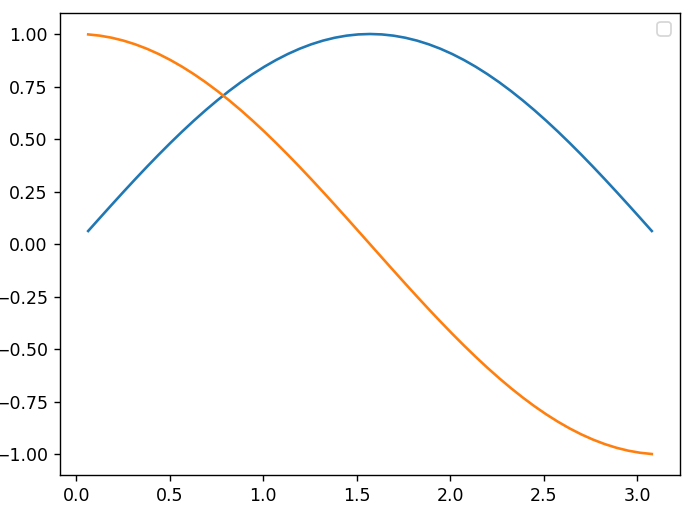
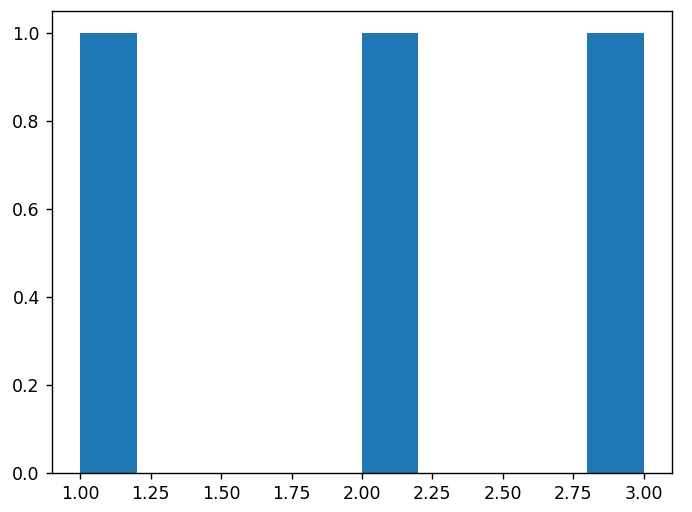
legend()

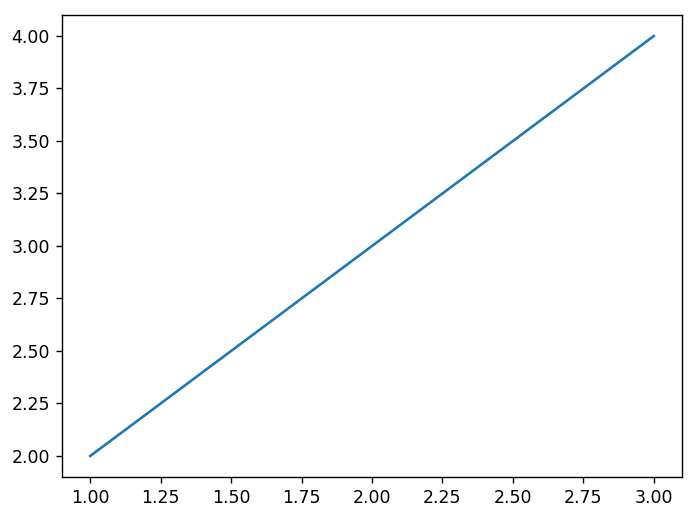
show()

plot(ex[0],ex[1])

show()

ex = [[-1,2],[3,-4]]





**Que9** Generate different subplots from a given plot and color plot data.

**Ans**

from matplotlib.pyplot import hist,plot,show,legend,subplots

fig,ax = subplots(2,2)

def printMat(X):

m,n = len(X),len(X[0])

if(m<=0 or n<=0):

raise Exception('Misshapen matrix')

else:

for i in range(m):

for k in range(n):

print(f'{X[i][k]}',end="\t")

print('\n')

ex = [[1,2,3],[2,3,4]]

print("The original matrix is: ")

printMat(ex)

fig.subplots\_adjust(hspace=0.5)

ax[0,0].set\_facecolor('#272727')

ax[0,0].set\_title('X vs Y')

ax[0,0].plot(ex[0],ex[1])

ax[0,1].set\_facecolor('#01377D')

ax[0,1].set\_title('Y vs X')

ax[0,1].plot(ex[1],ex[0],c='yellow')

ax[1,0].set\_facecolor('#489EF1')

ax[1,0].set\_title('Square')

ax[1,0].plot(range(1,10),[i\*i for i in range(1,10)],c='green')

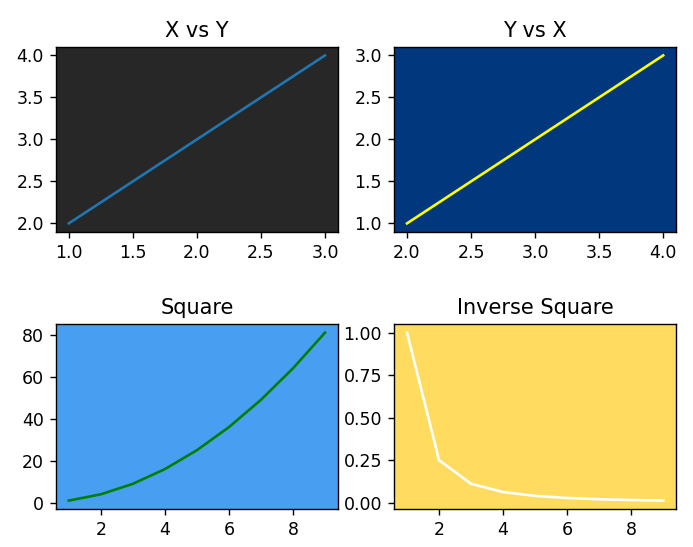
ax[1,1].set\_facecolor('#ffdc5f')

ax[1,1].set\_title('Inverse Square')

ax[1,1].plot(range(1,10),[1/(i\*i) for i in range(1,10)],c='white')

show()

ex = [[-1,2],[3,-4]]



**Que10** Use conditional statements and different types of loops based on simple examples.

**Ans**

print("For loop with array:")

array = [1, 2, 3, 4, 5]

for element in array:

print(element)

print()

print("While loop with array:")

array = [1, 2, 3, 4, 5]

i = 0

while i < len(array):

print(array[i])

i += 1

print()

print("Do-while loop equivalent with array:")

array = [1, 2, 3, 4, 5]

i = 0

while True:

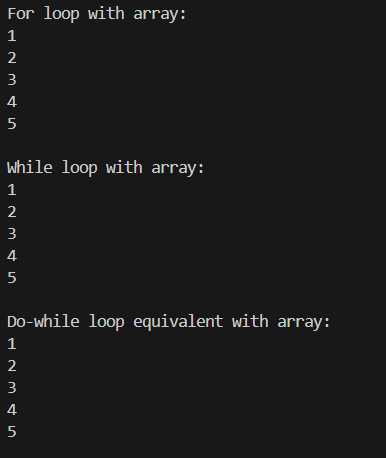
print(array[i])

i += 1

if i >= len(array):

break

print()



**Que11** Perform vectorized implementation of simple matrix operations like finding the transpose of a matrix, adding, subtracting or multiplying two matrices.

**Ans**

import numpy as np

def matrix\_transpose(matrix):

return np.transpose(matrix)

def matrix\_addition(matrix1, matrix2):

return np.add(matrix1, matrix2)

def matrix\_subtraction(matrix1, matrix2):

return np.subtract(matrix1, matrix2)

def matrix\_multiplication(matrix1, matrix2):

return np.dot(matrix1, matrix2)

matrix\_a = np.array([[1, 2, 3],

[4, 5, 6]])

matrix\_b = np.array([[7, 8, 9],

[10, 11, 12]])

print("Matrix Transpose:")

print(matrix\_transpose(matrix\_a))

print()

print("Matrix Addition:")

print(matrix\_addition(matrix\_a, matrix\_b))

print()

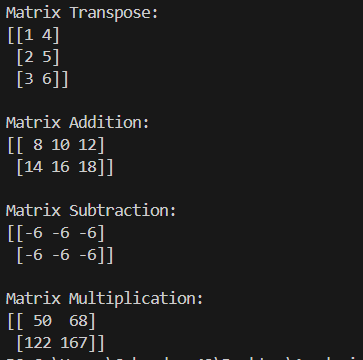
print("Matrix Subtraction:")

print(matrix\_subtraction(matrix\_a, matrix\_b))

print()

print("Matrix Multiplication:")

print(matrix\_multiplication(matrix\_a, matrix\_b.T))



**Que12** Implement Linear Regression problem. For example, based on a dataset of prices and area/size of the houses predict the estimated price of a given house.

**Ans**

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import pandas as pd

from warnings import filterwarnings

data = pd.read\_csv('housing.csv')

x,y = data['area'],data['price']

lr = LinearRegression()

lr.fit(x.to\_frame(),y.to\_frame())

area = int(input("Enter the area of the plot: "))

ypred = lr.predict(pd.DataFrame([area]))

print(f'The predicted house price is Rs.{ypred[0][0]}')



**Que13** Based on multiple features perform Linear Regression. For example, based on a number of additional features like number of bedrooms, servant room, number of balconies, number of houses of years a house has been built- predict the price of a house.

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score

import pandas as pd

from warnings import filterwarnings

data = pd.read\_csv('housing.csv')

x,y = data.iloc[:,1:],data.iloc[:,0]

le = LabelEncoder()

for col in x.columns:

if (x[col].dtype=='object'):

x[col] = le.fit\_transform(x[col])

lr = LinearRegression()

lr.fit(x,y)

pred = lr.predict(pd.DataFrame([[5000,4, 2, 3, 1, 0, 0, 0, 1, 2, 1, 0]]))

print(f"Predicted price of house is Rs.{pred[0]}")



**Que14** Implement a classification problem. For example, based on different features of students data, classify whether a student is suitable for a particular activity. Based on the available dataset, a student can also implement another classification problem like checking whether an email is spam or not.

**Ans**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LogisticRegression

from warnings import filterwarnings

filterwarnings('ignore')

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

le = LabelEncoder()

le.fit(data['variety'])

map = dict([(i,le.inverse\_transform([i])[0]) for i in range(len(data['variety'].unique()))])

data['variety'] = le.transform(data['variety'])

x,y = data.iloc[:,:-1],data.iloc[:,-1]

logreg = LogisticRegression()

logreg.fit(x,y)

features = []

for i in x.columns:

feat = float(input(f"Enter the {i}:"))

features.append(feat)

print(f'Given flower was a {map[logreg.predict(pd.DataFrame([features]))[0]]}')

****

**Que15** Use some function for regularization of the dataset based on problem 14.

**Ans**

from sklearn.datasets import load\_iris

from sklearn.preprocessing import StandardScaler

import pandas as pd

from warnings import filterwarnings

filterwarnings('ignore')

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

x,y = data.iloc[:,:-1],data.iloc[:,-1]

scaler = StandardScaler()

xstan = scaler.fit\_transform(x)

xstan = pd.DataFrame(xstan)

xstan.columns = x.columns



**Que16** Use some function for neural networks(like Stochastic Gradient Descent or backpropagation) algorithm to predict the value of a variable based on the dataset of problem14.

**Ans**

sum <-1+2

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

from sklearn.neural\_network import MLPClassifier

from warnings import filterwarnings

filterwarnings('ignore')

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

le = LabelEncoder()

le.fit(data['variety'])

map = dict([(i,le.inverse\_transform([i])[0]) for i in range(len(data['variety'].unique()))])

data['variety'] = le.transform(data['variety'])

x,y = data.iloc[:,:-1],data.iloc[:,-1]

mlp = MLPClassifier(alpha=0.05,activation='logistic')

mlp.fit(x,y)

features = []

for i in x.columns:

feat = float(input(f"Enter the {i}:"))

features.append(feat)

print(f'Given flower was a {map[mlp.predict(pd.DataFrame([features]))[0]]}')

