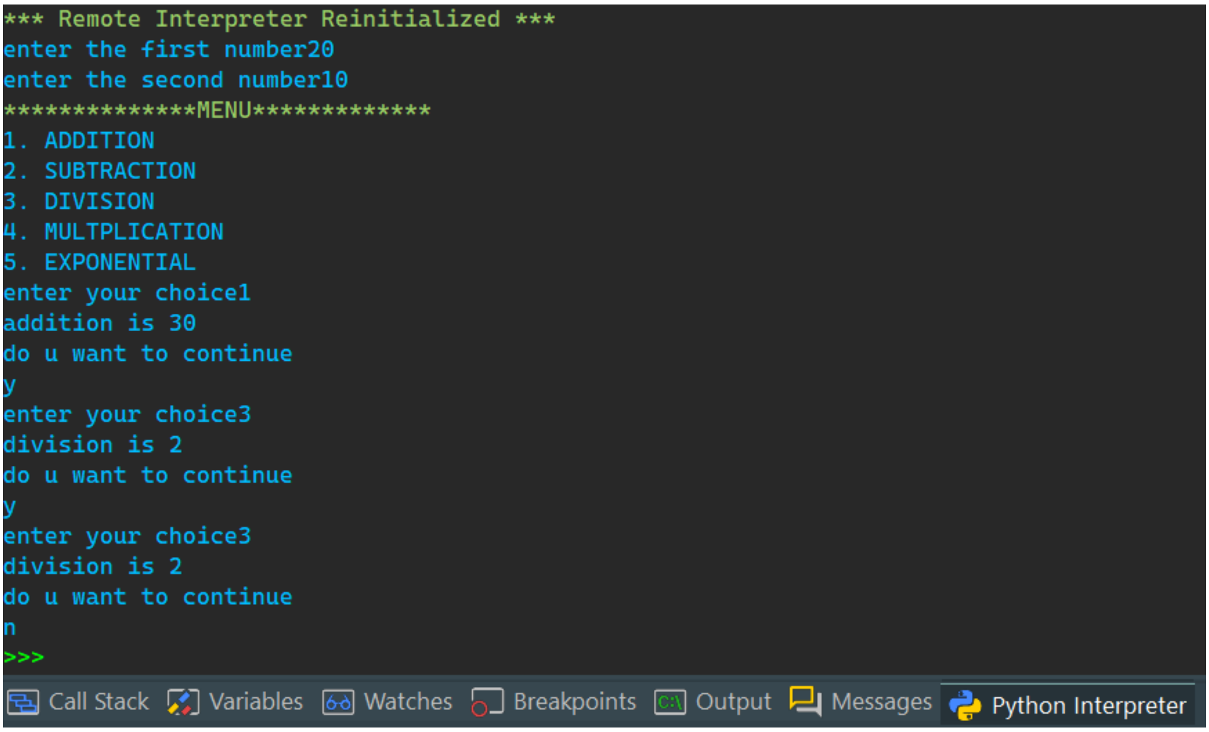
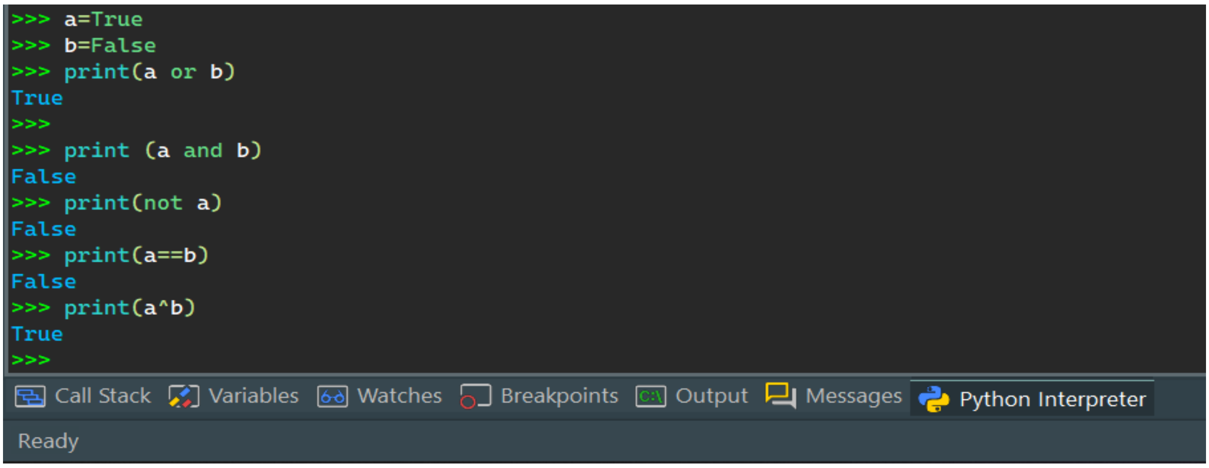
Q1. Perform elementary mathematical operations in octave/MATLAB like addition, multiplication, division and exponentiation.

```
a=int(input("enter the first number"))
b=int(input("enter the second number"))
ch="y"
print("**************MENU*************")
print("1. ADDITION")
print("2. SUBTRACTION")
print("3. MULTIPLICATION")
print("4. DIVISION")
print("5. EXPONENTIAL")


while ch=="y":
   var=int(input("enter your choice"))
   if var==1:
      print("addition is %d"%(a+b))
   elif var==2:
      print("subraction is %d"%(a-b))
   elif var==3:
      print("division is %d"%(a/b))
   elif var==4:
      print("mul is %d"%(a*b))
   else:
      print("exponential is %d"%pow(a,b))
   print("do u want to continue")
   ch=input()
```



```
*** Remote Interpreter Reinitialized ***
enter the first number20
enter the second number10
*************MENU*************
1. ADDITION
2. SUBTRACTION
3. DIVISION
4. MULTPLICATION
5. EXPONENTIAL
enter your choice1
addition is 30
do u want to continue
y
enter your choice3
division is 2
do u want to continue
y
enter your choice3
division is 2
do u want to continue
n
>>>
```

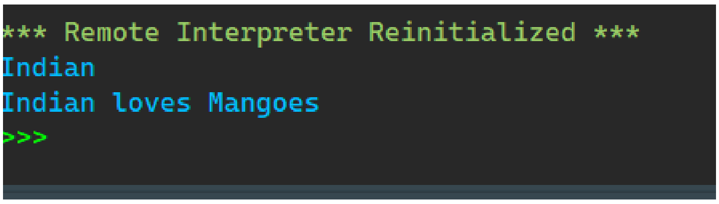
Q2. Perform elementary logical operations in octave/MATLAB likeOR,AND,checking for equality,NOT,XOR).



```
>>> a=True
>>> b=False
>>> print(a or b)
True
>>>
>>> print (a and b)
False
>>> print(not a)
False
>>> print(a==b)
False
>>> print(a^b)
True
>>>
```

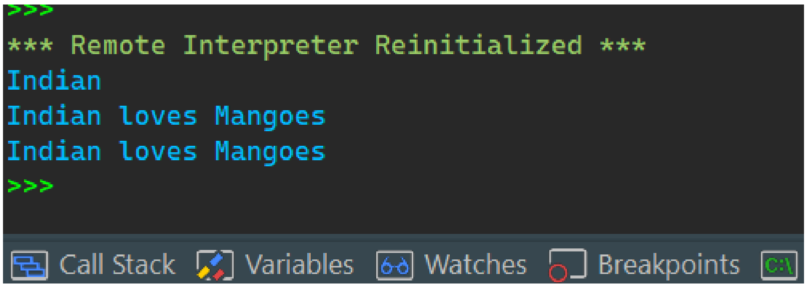
Q3. Create, initialize and display simple variable and simple string and use simple formatting for variable.

```
a="Indian"
b="Mangoes"
print(a)
print(a+" loves "+b)
```

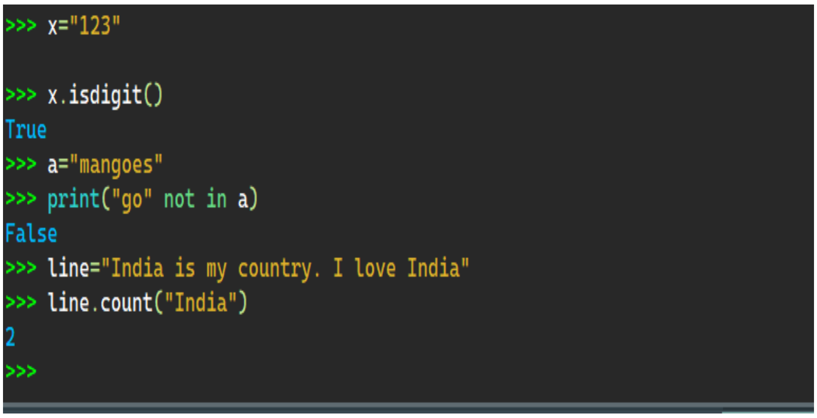


```
*** Remote Interpreter Reinitialized ***
Indian
Indian loves Mangoes
>>>
```

```
print("{0} loves {1}".format(a,b))
```

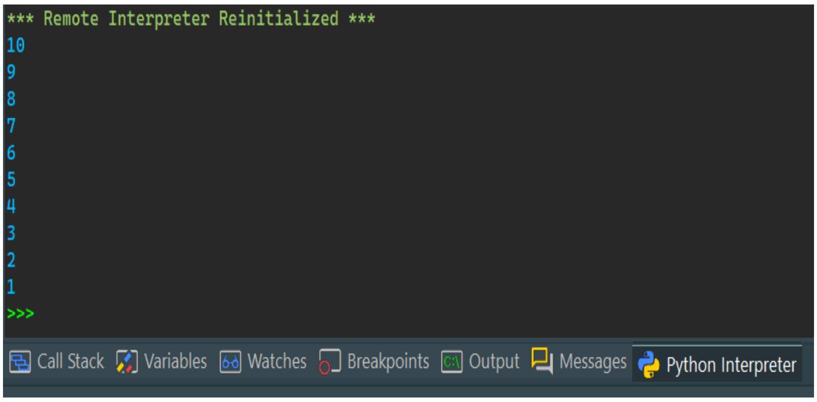


```
>>>
*** Remote Interpreter Reinitialized ***
Indian
Indian loves Mangoes
Indian loves Mangoes
>>>
```



```
>>> x="123"

>>> x.isdigit()
True
>>> a="mangoes"
>>> print("go" not in a)
False
>>> line="India is my country. I love India"
>>> line.count("India")
2
>>>
```

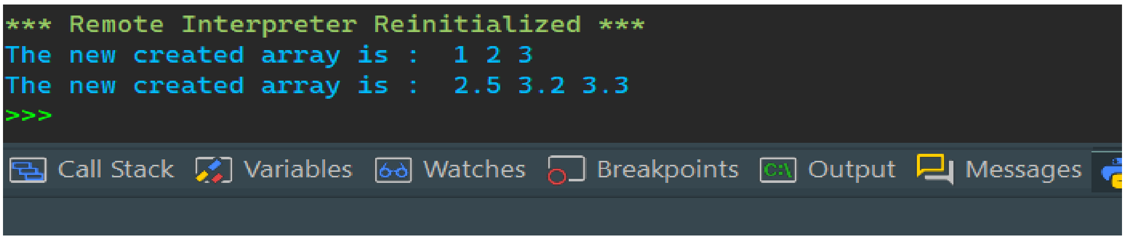
```
for i in range(10,0,-1):
    print(i)
```



```
*** Remote Interpreter Reinitialized ***
10
9
8
7
6
5
4
3
2
1
>>>
```

Q4. Create define single dimension/multi dimension arrays ,and arrays for specific values likeArrays of all ones,allzeros,arrays with random values within range or diagonal matrix.

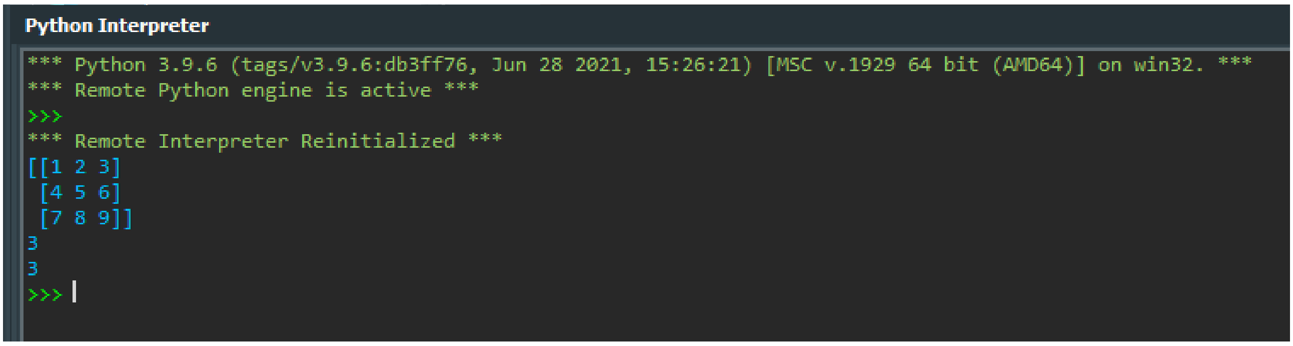
```
import array as arr

a = arr.array('i', [1, 2, 3])

print ("The new created array is : ", end =" ")
for i in range (0, 3):
        print (a[i], end =" ")
print()

b = arr.array('d', [2.5, 3.2, 3.3])

print ("The new created array is : ", end =" ")
for i in range (0, 3):
        print (b[i], end =" ")
```



```
*** Remote Interpreter Reinitialized ***
The new created array is :  1 2 3
The new created array is :  2.5 3.2 3.3
>>>
```

Q5. Use command to compute the size of a matrix, size/length 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.

```
from numpy import*
arr = array([[1,2,3],[4,5,6],[7,8,9]])
print(arr)

size(arr)

rows = len(arr)
print(rows)

column = len(arr[0])
print(column)
```



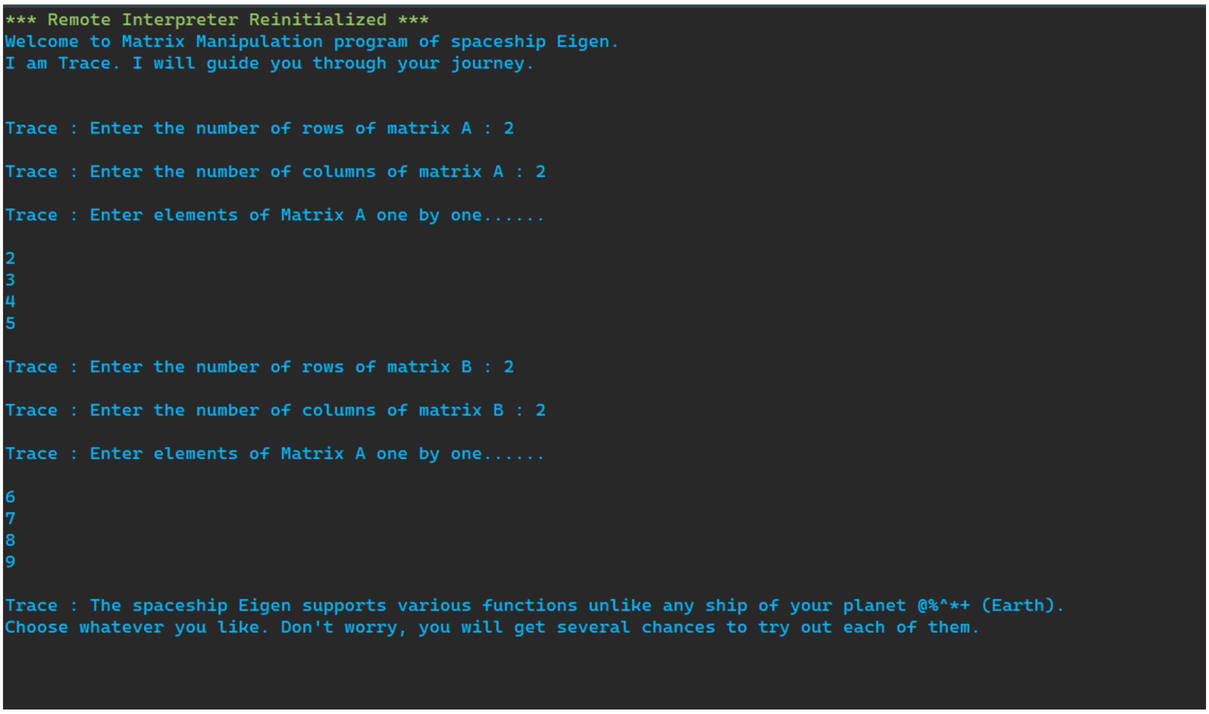
```
*** Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
*** Remote Interpreter Reinitialized ***
[[1 2 3]
 [4 5 6]
 [7 8 9]]
3
3
>>> 
```

Q6. Perform basic operations on matrices (like addition, subtraction, multiplication) and display specific rows or columns of the matrix.

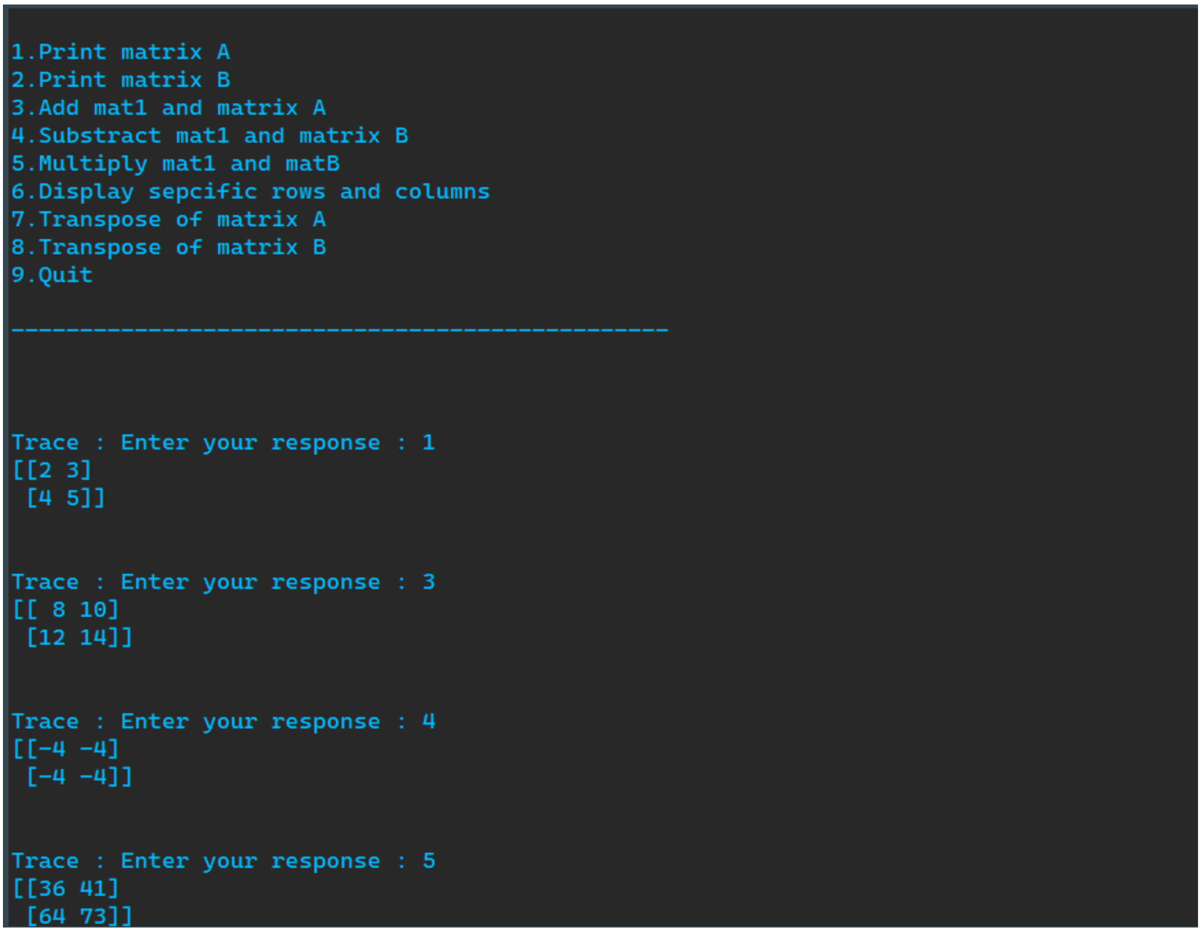
```
from numpy import*
import sys

lst1 = []
lst2 = []

print("Welcome to Matrix Manipulation program of spaceship Eigen.\nI am Tra
ce. I will guide you through your journey.\n")


r1 = int(input("\nTrace : Enter the number of rows of matrix A : "))
c1 = int(input("\nTrace : Enter the number of columns of matrix A : "))
print("\nTrace : Enter elements of Matrix A one by one......\n")
```

```
for i in range(0, r1*c1):
   lst1.append(int(input()))
array1 = array(lst1)
mat1 = array1.reshape(r1,c1)

r2 = int(input("\nTrace : Enter the number of rows of matrix B : "))
c2 = int(input("\nTrace : Enter the number of columns of matrix B : "))
print("\nTrace : Enter elements of Matrix A one by one......\n")

for i in range(0, r2*c2):
   lst2.append(int(input()))
array2 = array(lst2)
mat2 = array2.reshape(r2,c2)
print("\nTrace : The spaceship Eigen supports various functions unlike any ship
 of your planet @%^*+ (Earth).\nChoose whatever you like. Don't worry, you w
ill get several chances to try out each of them.\n\n")
print("\n-------------------menu----------------------\n")
print("1.Print matrix A")
print("2.Print matrix B")
print("3.Add mat1 and matrix A")
print("4.Substract mat1 and matrix B")
print("5.Multiply mat1 and matB")
print("6.Display sepcific rows and columns")
print("7.Transpose of matrix A")
print("8.Transpose of matrix B")
print("9.Quit")
print("\n-----------------------------------------------\n")

while True:

   response = int(input("\n\nTrace : Enter your response : "))
   if response == 1:
      print(mat1)
   elif response == 2:
      print(mat2)
   elif response == 3:
      print(mat1+mat2)
   elif response == 4:
      print(mat1-mat2)
```

```
elif response == 5:
    print(matmul(mat1,mat2))
elif response == 6:
    r = int(input("\nTrace : Enter row of matrx A : "))
    r = r-1
    print(mat1[r,:])
    print("")
    r = int(input("\nTrace : Enter col of matrix A : "))
    print(mat1[:,r])
    print("")
    r = int(input("\nTrace : Enter row of matrix B : "))
    print(mat2[r,:])
    print("")
    r = int(input("\nTrace : Enter col of matrix B : "))
    print(mat2[:,r])
    print("")
elif response == 7:
    print(transpose(mat1))
elif response == 8:
    print(transpose(mat2))
elif response == 9:
    print("\n\n'\\_(*_*)_/'\n\nTrace : Be seeying you.....\n")
    print("\n\n\n\n\nPress any key to exit the program : ")
    input()
    sys.exit()
```



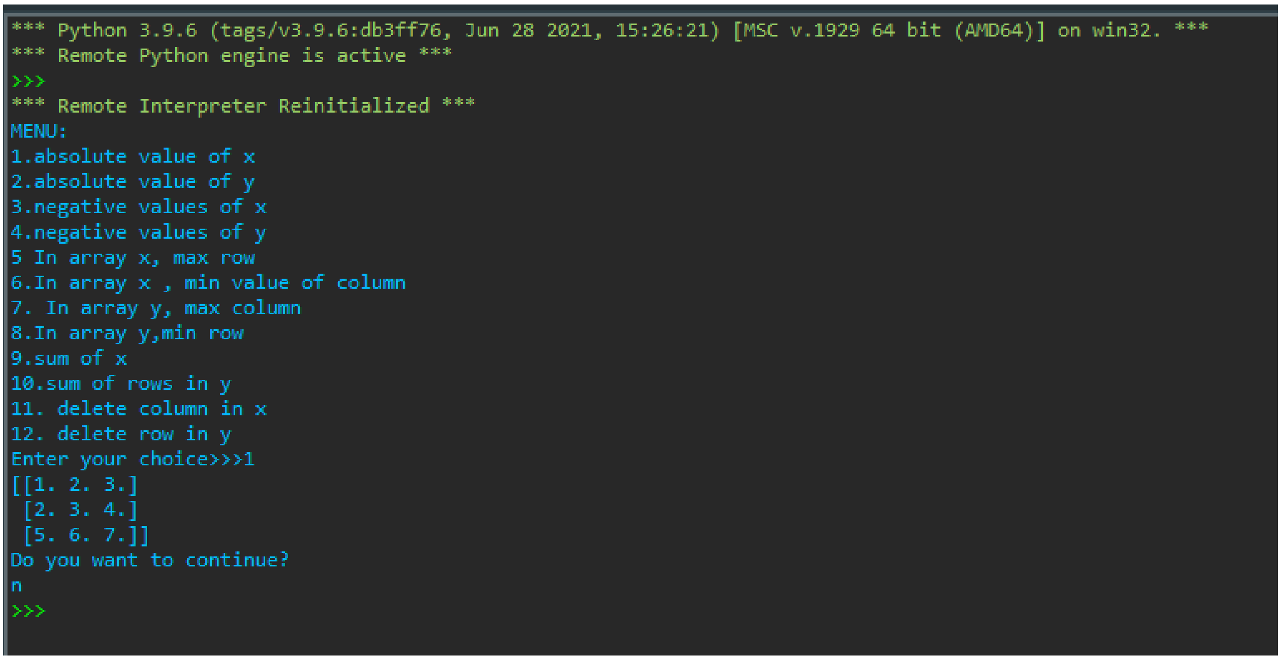
```
*** Remote Interpreter Reinitialized ***
Welcome to Matrix Manipulation program of spaceship Eigen.
I am Trace. I will guide you through your journey.


Trace : Enter the number of rows of matrix A : 2

Trace : Enter the number of columns of matrix A : 2

Trace : Enter elements of Matrix A one by one......

2
3
4
5

Trace : Enter the number of rows of matrix B : 2

Trace : Enter the number of columns of matrix B : 2

Trace : Enter elements of Matrix A one by one......

6
7
8
9

Trace : The spaceship Eigen supports various functions unlike any ship of your planet @%^*+ (Earth).
Choose whatever you like. Don't worry, you will get several chances to try out each of them.
```



```
1.Print matrix A
2.Print matrix B
3.Add mat1 and matrix A
4.Substract mat1 and matrix B
5.Multiply mat1 and matB
6.Display sepcific rows and columns
7.Transpose of matrix A
8.Transpose of matrix B
9.Quit

------------------------------------------------



Trace : Enter your response : 1
[[2 3]
 [4 5]]


Trace : Enter your response : 3
[[ 8 10]
 [12 14]]


Trace : Enter your response : 4
[[-4 -4]
 [-4 -4]]


Trace : Enter your response : 5
[[36 41]
 [64 73]]
```

Q7. Perform other matrix operation like converting matrix data to absolute value, taking the negative of matrix values, adding removing/column from the matrix. Find the maximum and minimum values in a matrix or in a row/column and finding the sum of some/all elements in the matrix.

```
import numpy as np
x=np.array([[1,2,3],[2,3,4.0],[5,6,7]])
y=np.array([[8,9,10],[11,12,13],[14,15,16]])
ch="y"
print("MENU:")
print("1.absolute value of x \n2.absolute value of y\n3.negative values of
x\n4.negative values of y\n5 In array x, max row\n6.In array x , min value of
column\n7. In array y, max column\n8.In array y,min row\n9.sum of
x\n10.sum of rows in y\n11. delete column in x\n12. delete row in y")
while ch=="y":
  var=int(input("Enter your choice>>>"))
  if var==1:
   print(np.abs(x))
  elif var==2:
   print(np.abs(y))
  elif var==3:
    np.negative(x)
  elif var==4:
    np.negative(y)
  elif var==5:
    x.max(axis=0)
  elif var==6:
    x.min(axis=1)
  elif var==7:
    y.max(axis=1)
  elif var==8:
    y.min(axis=0)
  elif var==9:
   x.sum()
  elif var==10:
    y.sum(axis=0)
  elif var==11:
    a=np.delete(x,0,1)
    print(a)
  elif var==12:
        a=np.delete(y,1,0)
        print(a)

    print("Do you want to continue?")
    ch=input()
```

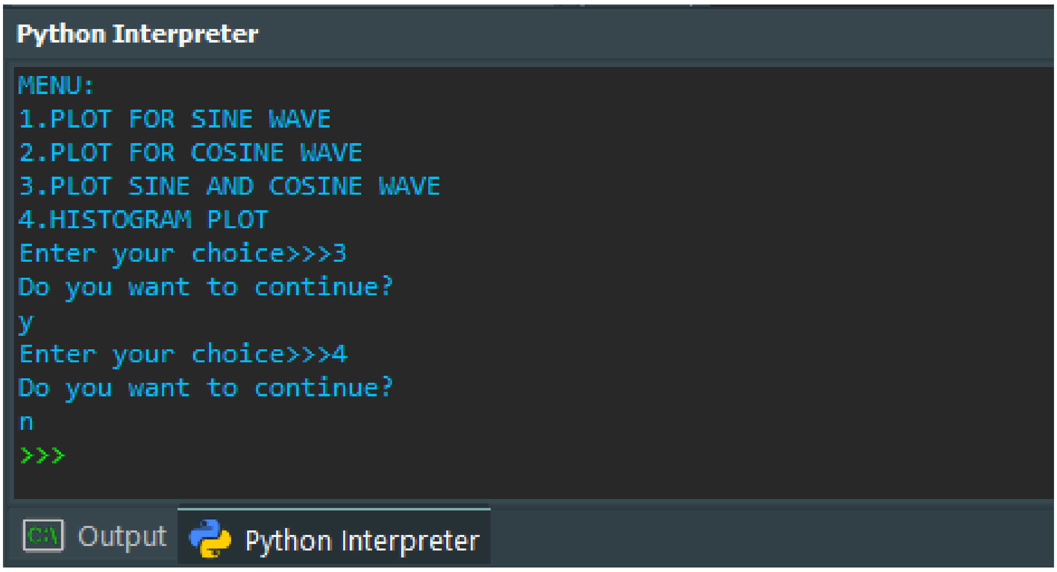


```
*** Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
*** Remote Interpreter Reinitialized ***
MENU:
1.absolute value of x
2.absolute value of y
3.negative values of x
4.negative values of y
5 In array x, max row
6.In array x , min value of column
7. In array y, max column
8.In array y,min row
9.sum of x
10.sum of rows in y
11. delete column in x
12. delete row in y
Enter your choice>>>1
[[1. 2. 3.]
 [2. 3. 4.]
 [5. 6. 7.]]
Do you want to continue?
n
>>>
```

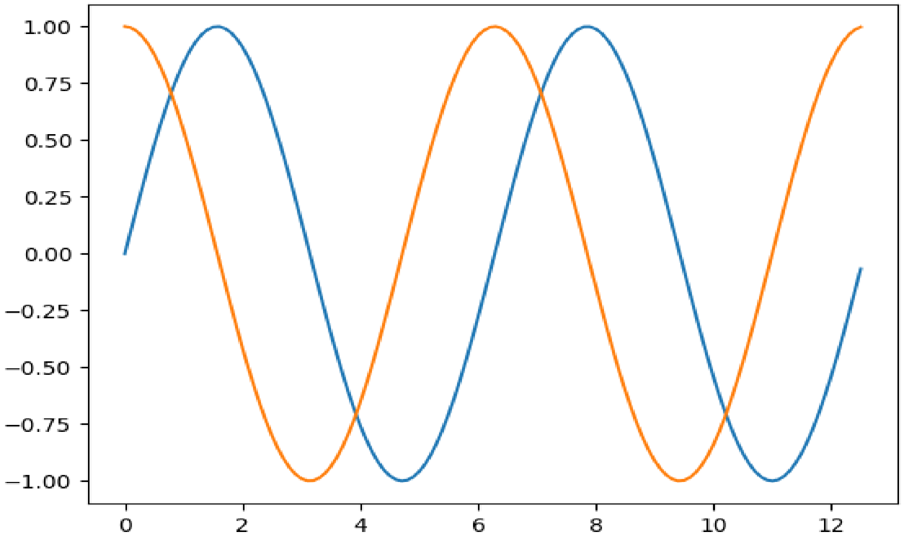
Q8. Create various type of plots/charts like histograms, plot based on sine/cosine function based on data from a matrix. Further label different axes in a plot and data in a plot.

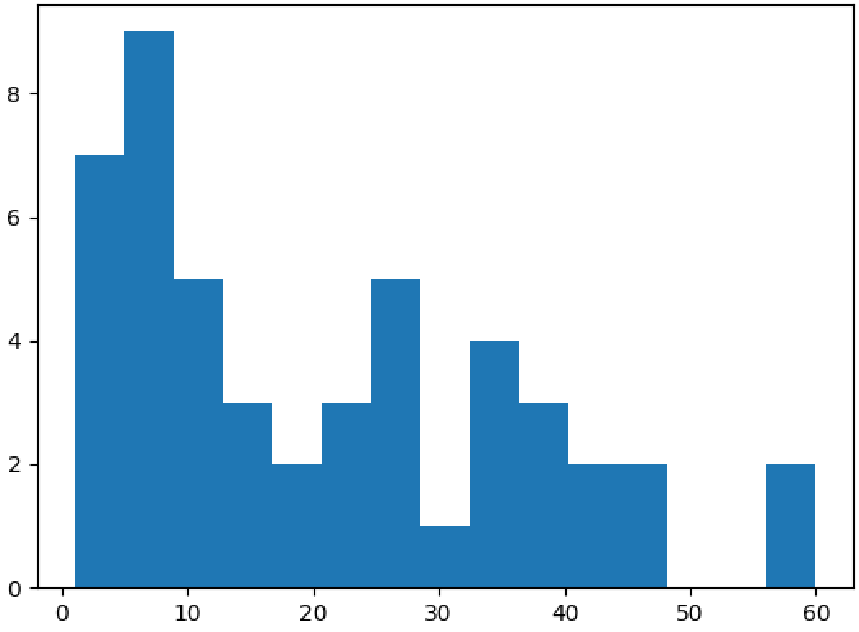
```
import matplotlib.pyplot as plt
import numpy as np

ch="y"
print("MENU:")
print("1.PLOT FOR SINE WAVE \n2.PLOT FOR COSINE WAVE\n3.PLOT SINE AND COSINE WAVE\n4.HISTOGRAM PLOT")
while ch=="y":
  var=int(input("Enter your choice>>>"))
  if var==1:
      x=np.array([[0,1,2],[3,4,5],[6,7,8]])
      print("PLOT FOR SINE")
      plt.plot(x,np.sin(x))
```

```
        plt.xlabel('X')
        plt.ylabel('SIN X')
        plt.grid()
        plt.show()
    elif var==2:
        x=np.array([[0,1,2],[3,4,5],[6,7,8]])
        print("PLOT FOR COSINE")
        plt.plot(x,np.cos(x))
        plt.xlabel('X')
        plt.ylabel('COS X')
        plt.grid()
        plt.show()
    elif var==3:
        x=np.arange(0,4*np.pi,0.1)
        y=np.sin(x)
        z=np.cos(x)
        plt.plot(x,y,x,z)
        plt.show()
    elif var==4:

x=[1,1,1,1,2,2,4,5,5,6,7,7,7,7,8,8,9,10,11,12,12,13,15,16,17,20,22,23,23,25,26,
27,28,28,29,33,33,34,36,39,40,40,41,43,46,46,59,60]
        plt.hist(x,bins=15)
        plt.show()

    print("Do you want to continue?")
    ch=input()
```



```
MENU:
1.PLOT FOR SINE WAVE
2.PLOT FOR COSINE WAVE
3.PLOT SINE AND COSINE WAVE
4.HISTOGRAM PLOT
Enter your choice>>>3
Do you want to continue?
y
Enter your choice>>>4
Do you want to continue?
n
>>>
```

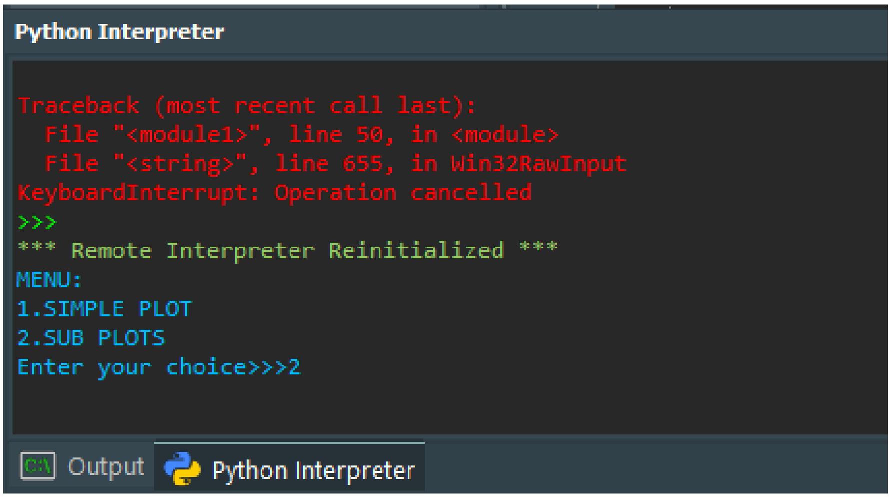




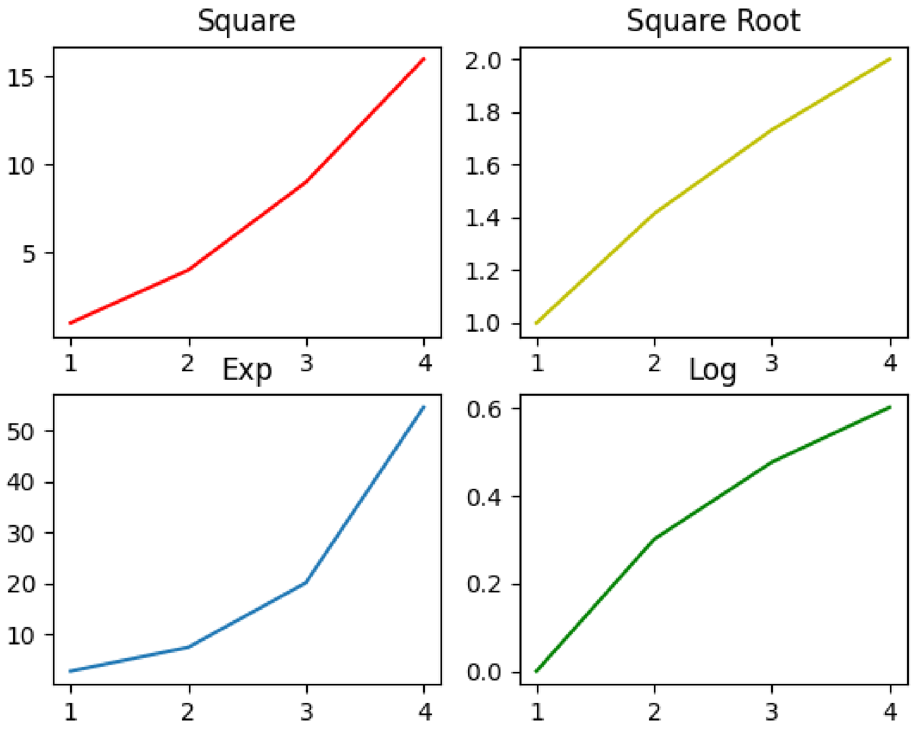
Q9. Generate different subplots from a given plot and color plot data.

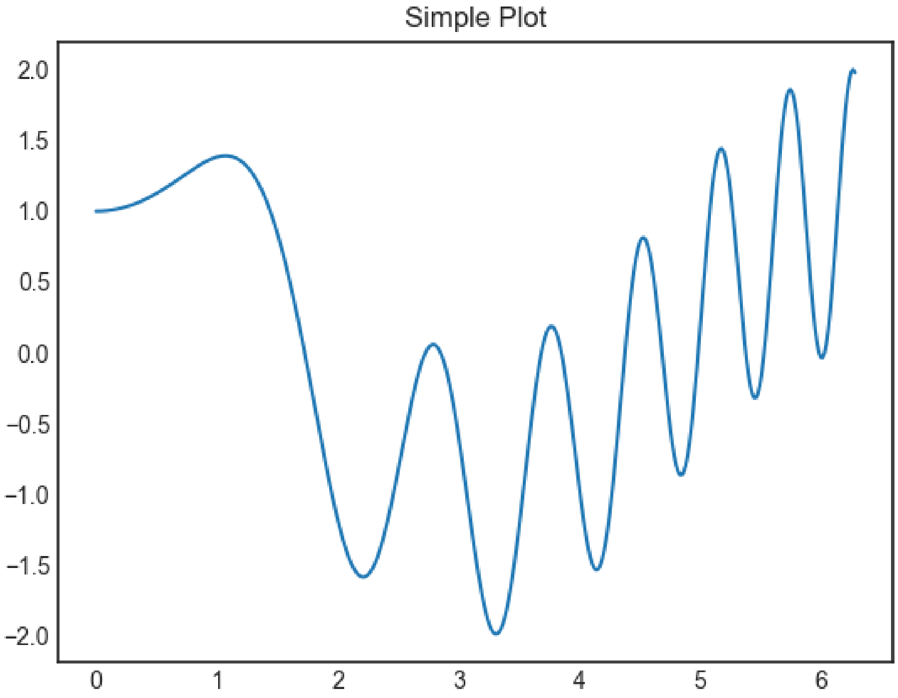
```
import matplotlib.pyplot as plt
import numpy as np

ch="y"
print("MENU:")
print("1.SIMPLE PLOT \n2.SUB PLOTS")
while ch=="y":
  var=int(input("Enter your choice>>>"))
  if var==1:
      plt.style.use('seaborn-white')
      fig,ax=plt.subplots()

      x=np.linspace(0,2*np.pi,400)
      y=np.sin(x**2)+np.cos(x)
      ax.plot(x,y)
      ax.set_title('Simple Plot')
      plt.show()
  elif var==2:
      fig,ax=plt.subplots(2,2)
```

```
    x=np.arange(1,5)
    ax[0][0].plot(x,x*x,'r')
    ax[0][0].set_title('Square')

    ax[0][1].plot(x,np.sqrt(x),'y')
    ax[0][1].set_title('Square Root')

    ax[1][0].plot(x,np.exp(x))
    ax[1][0].set_title('Exp')

    ax[1][1].plot(x,np.log10(x),'g')
    ax[1][1].set_title('Log')
    plt.show()

print("Do you want to continue?")
ch=input()
```



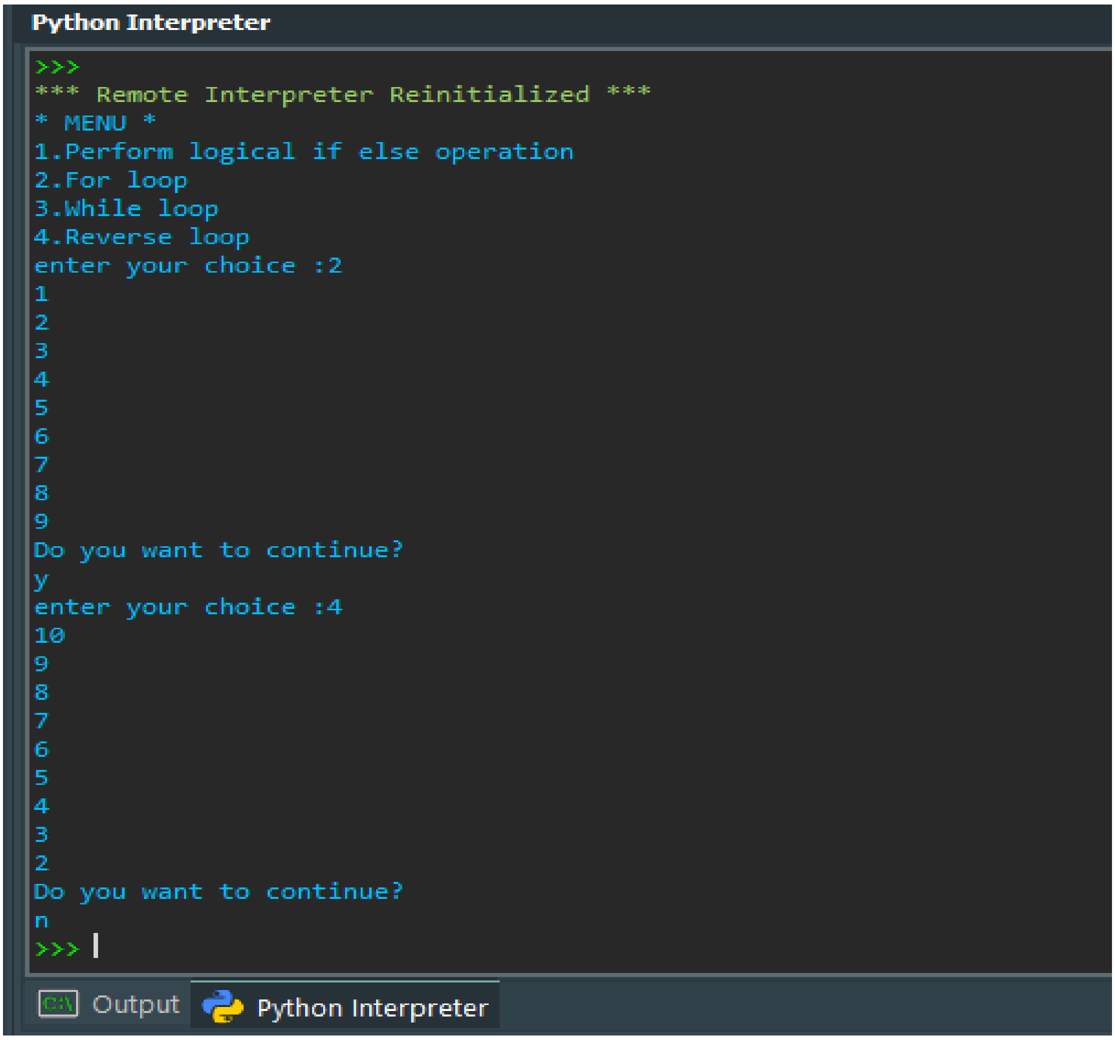
```
Traceback (most recent call last):
  File "<module1>", line 50, in <module>
  File "<string>", line 655, in Win32RawInput
KeyboardInterrupt: Operation cancelled
>>>
*** Remote Interpreter Reinitialized ***
MENU:
1.SIMPLE PLOT
2.SUB PLOTS
Enter your choice>>>2
```


Square
15
10
5
1
2
3
4
Square Root
2.0
1.8
1.6
1.4
1.2
1.0
1
2
3
4
Exp
50
40
30
20
10
1
2
3
4
Log
0.6
0.4
0.2
0.0
1
2
3
4


Simple Plot
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
6

Q10. Use conditional statements and different type of loops based on simple example/s,

```
ch="y"
print("* MENU *")
print("1.Perform logical if else operation\n2.For loop\n3.While
loop\n4.Reverse loop")

while ch=="y":
    var=int(input("enter your choice :"))
    if var==1:
        x=int(input("enter a number to check whether is greater than 100"))
        if(x>100):
            print("true")
        else:
            print("false")

    elif var==2:
        for i in range(1,10):
            print(i)

    elif var==3:
        n=10
        i=0
        while i<=n:
            print(i)

    elif var==4:
        for i in range(10,1,-1):
            print(i)

    print("Do you want to continue?")
    ch=input()
```

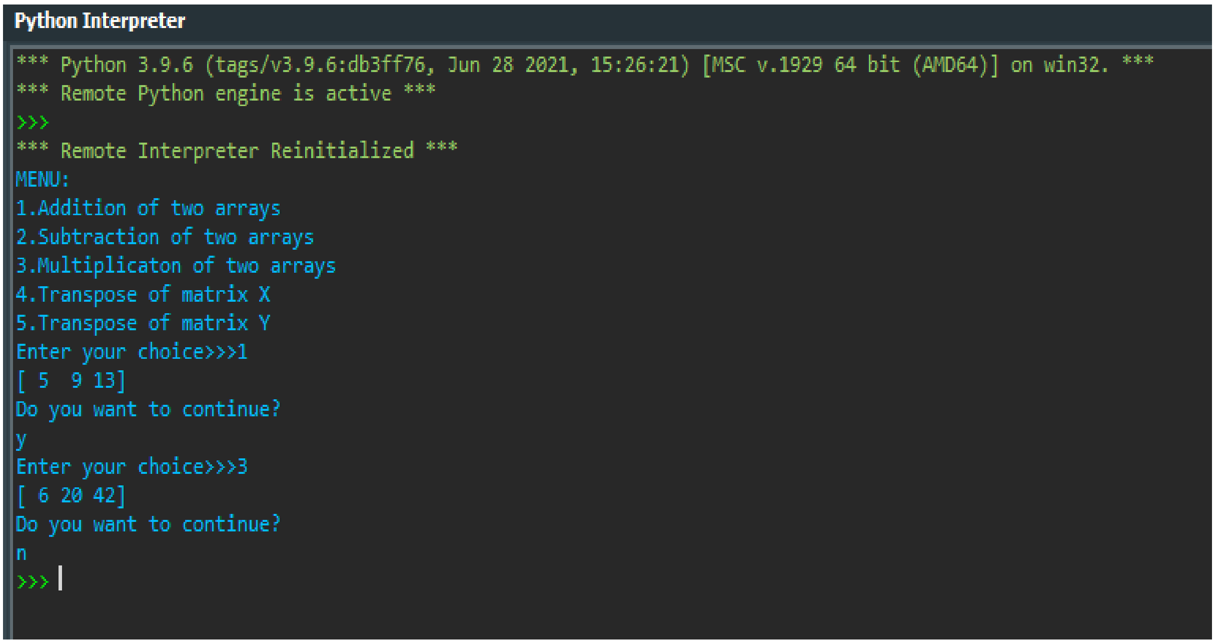


```
>>>
*** Remote Interpreter Reinitialized ***
* MENU *
1.Perform logical if else operation
2.For loop
3.While loop
4.Reverse loop
enter your choice :2
1
2
3
4
5
6
7
8
9
Do you want to continue?
y
enter your choice :4
10
9
8
7
6
5
4
3
2
Do you want to continue?
n
>>> 
```

Q11. Perform vectorized implementation of simple matrix operation like finding the transpose of a matrix, adding,subtracting or multiplying two matrices.

```
import numpy as np
x=np.array([2,4,6])
y=np.array([3,5,7])

ch="y"
print("MENU:")
print("1.Addition of two arrays \n2.Subtraction of two arrays\n3.Multiplicaton of two arrays\n4.Transpose of matrix X\n5.Transpose of matrix Y")
while ch=="y":
  var=int(input("Enter your choice>>>"))
  if var==1:
    R=x+y
```

```
          print(R)
        elif var==2:
            R=x-y
            print(R)
        elif var==3:
            R=x*y
            print(R)
        elif var==4:
            transpose(x)
        elif var==5:
            transpose(y)


        print("Do you want to continue?")
        ch=input()
```

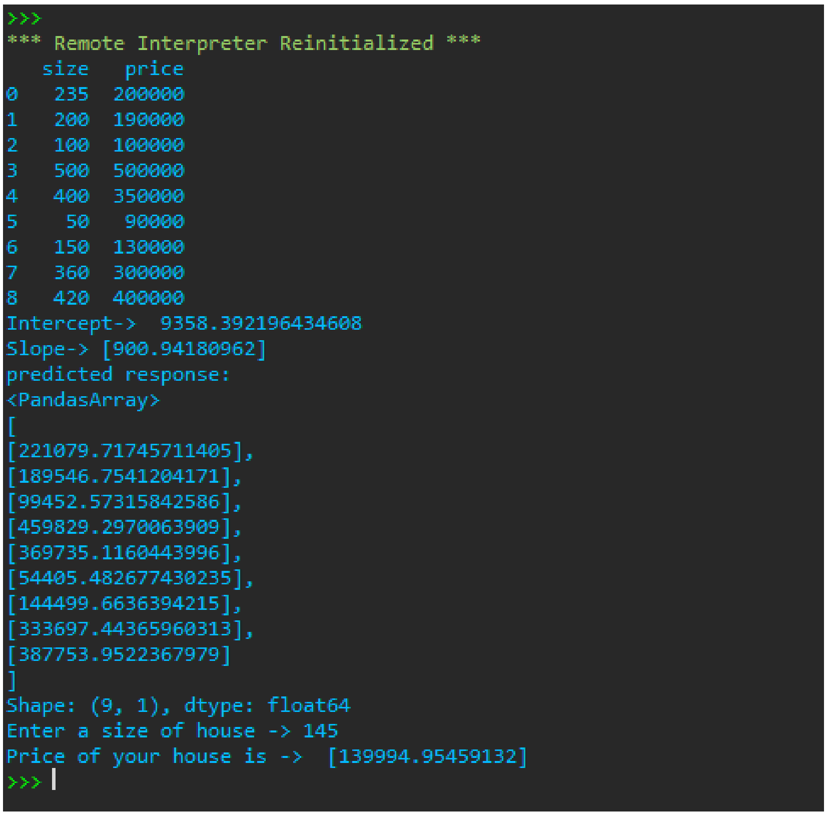


```
*** Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32. ***
*** Remote Python engine is active ***
>>>
*** Remote Interpreter Reinitialized ***
MENU:
1.Addition of two arrays
2.Subtraction of two arrays
3.Multiplicaton of two arrays
4.Transpose of matrix X
5.Transpose of matrix Y
Enter your choice>>>1
[ 5  9 13]
Do you want to continue?
y
Enter your choice>>>3
[ 6 20 42]
Do you want to continue?
n
>>> |
```

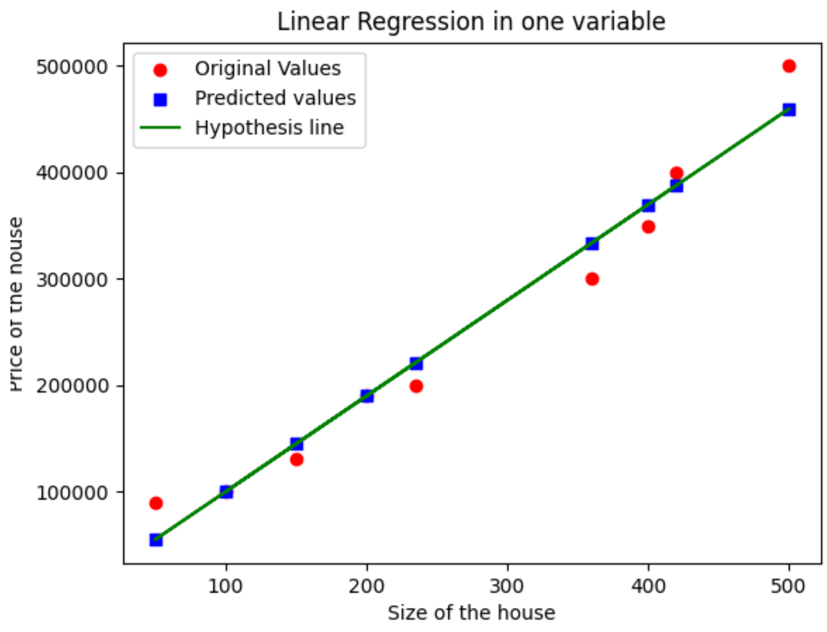
Q12. Implement Linear Regression problem. For example, based on a dataset comprising of existing set of prices and area of the houses, predict the estimated price of a given house.

```
#importing necessary libraries

from numpy import *
```

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
import pandas as pd

# importing the dataset
df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/ML
Datasets/House_price_data(q12).csv")
print(df)

x = df['size']
y = df['price']

x = x.array.reshape(len(x),1)

# Creating a hypothesis model
model = LinearRegression()
model.fit(x, y)

print('Intercept-> ', model.intercept_)
print('Slope->', model.coef_)

# Predicting y values
def prid_y_func(x):
  return  model.coef_ * x + model.intercept_

y_pred = prid_y_func(x)
print('predicted response:', y_pred, sep='\n')

# Plotting a graph
plt.scatter(x, y, color = 'red')
plt.scatter(x, y_pred, color = 'blue',marker='s')
plt.plot(x, y_pred, color = 'green')
plt.xlabel('Size of the house')
plt.ylabel('Price of the house')
plt.title(" Linear Regression in one variable ")
plt.legend(['Original Values','Predicted values','Hypothesis line'])
plt.show()

# Testing a new data
x_new = int(input("Enter a size of house -> "))
new_y = prid_y_func(x_new)
print("Price of your house is -> ", new_y)
```



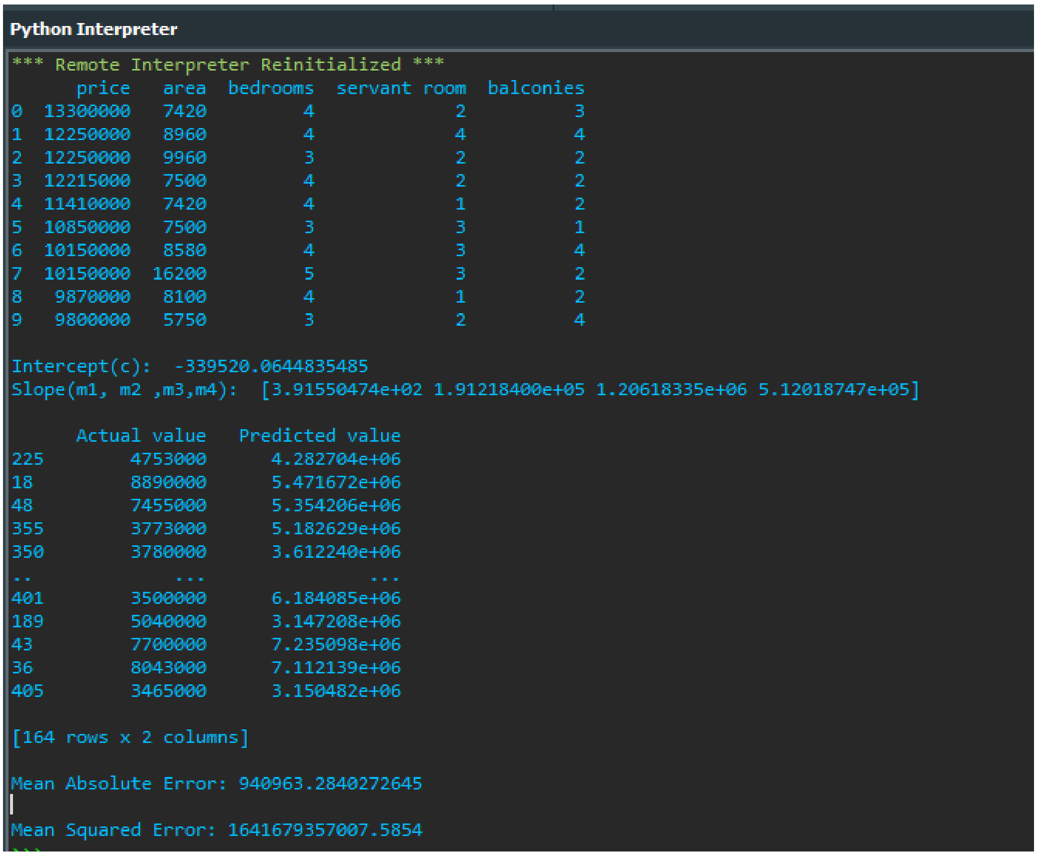
```
>>>
*** Remote Interpreter Reinitialized ***
   size   price
0   235  200000
1   200  190000
2   100  100000
3   500  500000
4   400  350000
5    50   90000
6   150  130000
7   360  300000
8   420  400000
Intercept->  9358.392196434608
Slope-> [900.94180962]
predicted response:
<PandasArray>
[
[221079.71745711405],
[189546.7541204171],
[99452.57315842586],
[459829.2970063909],
[369735.1160443996],
[54405.482677430235],
[144499.6636394215],
[333697.44365960313],
[387753.9522367979]
]
Shape: (9, 1), dtype: float64
Enter a size of house -> 145
Price of your house is ->  [139994.95459132]
>>> 
```



Q13. Based on multiple features/variables 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.

```
#importing necessary libraries
import pandas as pd

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error


#importing the dataset

df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/ML
Datasets/Housing_data(q13).csv")

print(df.head(10))



#assigning independent variables to x and dependent variable to y

x = df[['area', 'bedrooms', 'servant room','balconies']]

y = df['price']


#splitting the dataframe into training and testing data

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state =
101)


model = LinearRegression()

model.fit(x_train, y_train)


print('\nIntercept(c): ', model.intercept_)

print('Slope(m1, m2 ,m3,m4): ', model.coef_)
```

```
#prediction of test data

y_predict = model.predict(x_test)


model_diff = pd.DataFrame({'Actual value ': y_test, 'Predicted value ': y_predict})

print('\n', model_diff)


meanAbErr = mean_absolute_error(y_test, y_predict)

meanSqErr = mean_squared_error(y_predict, y_test)


print('\nMean Absolute Error:', meanAbErr)

print('\nMean Squared Error:', meanSqErr)
```

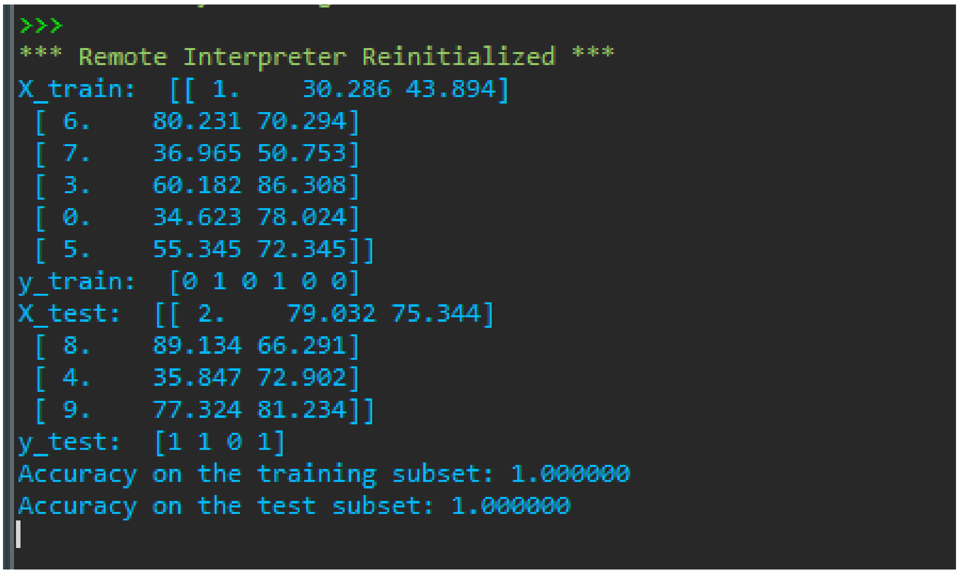


```
*** Remote Interpreter Reinitialized ***
      price   area  bedrooms  servant room  balconies
0  13300000   7420         4             2          3
1  12250000   8960         4             4          4
2  12250000   9960         3             2          2
3  12215000   7500         4             2          2
4  11410000   7420         4             1          2
5  10850000   7500         3             3          1
6  10150000   8580         4             3          4
7  10150000  16200         5             3          2
8   9870000   8100         4             1          2
9   9800000   5750         3             2          4

Intercept(c):  -339520.0644835485
Slope(m1, m2 ,m3,m4):  [3.91550474e+02 1.91218400e+05 1.20618335e+06 5.12018747e+05]

      Actual value   Predicted value 
225        4753000      4.282704e+06
18         8890000      5.471672e+06
48         7455000      5.354206e+06
355        3773000      5.182629e+06
350        3780000      3.612240e+06
..             ...               ...
401        3500000      6.184085e+06
189        5040000      3.147208e+06
43         7700000      7.235098e+06
36         8043000      7.112139e+06
405        3465000      3.150482e+06

[164 rows x 2 columns]

Mean Absolute Error: 940963.2840272645

Mean Squared Error: 1641679357007.5854
```

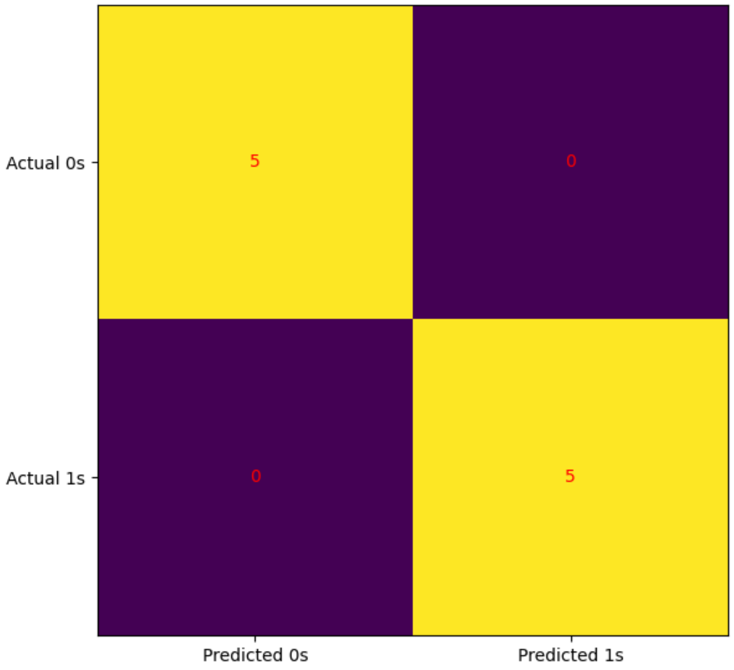
Q14. Implement a classification/ logistic regression problem. For example based on different features of students data, classify, whether a student is suitable for a particular activity. Based on the available dataset, a student can also implement another classification problem like checking whether an email is spam or not.

```
#importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/ML Datasets/Students_data(q14).csv")
df

#assigning independent variables to x and dependent variable to y
X = df.iloc[:, :-1].values
X
y = df.iloc[:, -1].values
y

#splitting the dataframe into training and testing data
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 0)
print ("X_train: ", X_train)
print ("y_train: ", y_train)
print("X_test: ", X_test)
print ("y_test: ", y_test)

model = LogisticRegression()
model.fit(X_train, y_train)
model = LogisticRegression().fit(X_train,y_train)

#prediction of test data
y_pred = model.predict(X_test)
print("Accuracy on the training subset: {:3f}".format(model.score(X_train,y_train)))
print("Accuracy on the test subset: {:3f}".format(model.score(X_test,y_test)))

model.classes_
model.predict_proba(X)

#printing confusion matrix
confusion_matrix(y, model.predict(X))
cm = confusion_matrix(y, model.predict(X))

fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(cm)
ax.grid(False)
ax.xaxis.set(ticks=(0, 1), ticklabels=('Predicted 0s', 'Predicted 1s'))
ax.yaxis.set(ticks=(0, 1), ticklabels=('Actual 0s', 'Actual 1s'))

ax.set_ylim(1.5, -0.5)

for i in range(2):

   for j in range(2):

       ax.text(j, i, cm[i, j], ha='center', va='center', color='red')

plt.show()
```



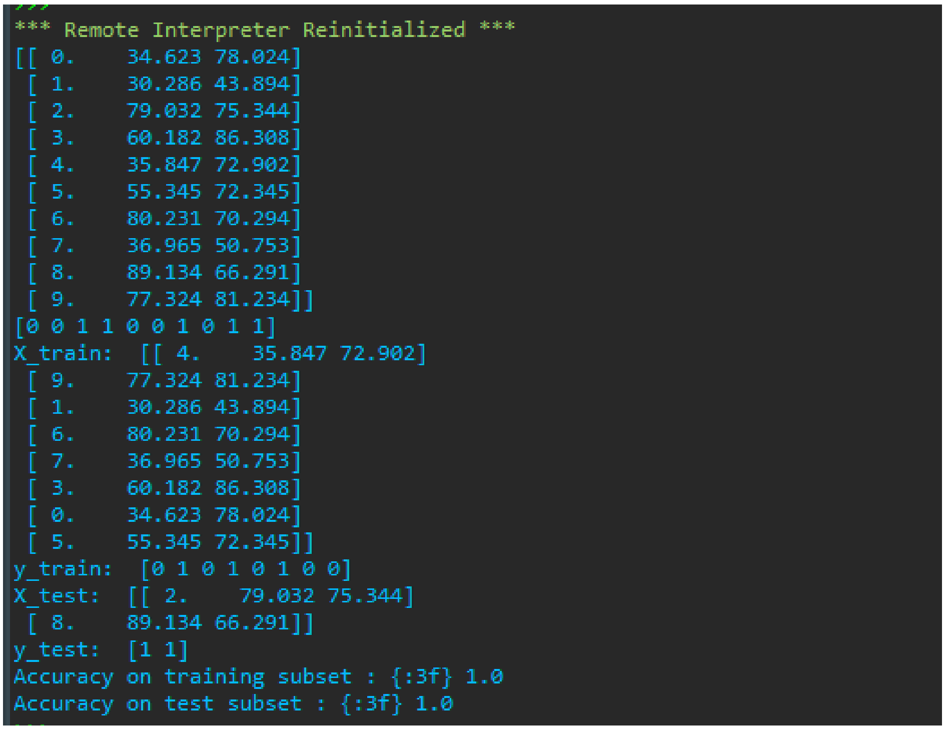
```
>>>
*** Remote Interpreter Reinitialized ***
X_train:  [[ 1.     30.286 43.894]
 [ 6.     80.231 70.294]
 [ 7.     36.965 50.753]
 [ 3.     60.182 86.308]
 [ 0.     34.623 78.024]
 [ 5.     55.345 72.345]]
y_train:  [0 1 0 1 0 0]
X_test:  [[ 2.     79.032 75.344]
 [ 8.     89.134 66.291]
 [ 4.     35.847 72.902]
 [ 9.     77.324 81.234]]
y_test:  [1 1 0 1]
Accuracy on the training subset: 1.000000
Accuracy on the test subset: 1.000000
```



Q15. Use some function for regularization of dataset based on problem 14.

```
#importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split



#importing the dataset
df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/ML 
Datasets/Students_data(q14).csv")
df


#assigning independent features to x and dependent features of y
```

```
X = df.iloc[:, :-1].values
print(X)

y = df.iloc[:, -1].values
print(y)

#splitting the dataframe into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
print ("X_train: ", X_train)
print ("y_train: ", y_train)
print("X_test: ", X_test)
print ("y_test: ", y_test)

#assigning different values to hyperparameter in order to initiate regularization
model = LogisticRegression(C = 100)
model.fit(X_train,y_train)
model.coef_

model = LogisticRegression(C = 1)
model.fit(X_train,y_train)
model.coef_

y_predicted = model.predict(X_test)

#printing accuracy score of model on training and testing data
print('Accuracy on training subset : {:3f}',format(model.score(X_train,y_train)))
print('Accuracy on test subset : {:3f}',format(model.score(X_test,y_test)))
```



```
*** Remote Interpreter Reinitialized ***
[[ 0.     34.623 78.024]
 [ 1.     30.286 43.894]
 [ 2.     79.032 75.344]
 [ 3.     60.182 86.308]
 [ 4.     35.847 72.902]
 [ 5.     55.345 72.345]
 [ 6.     80.231 70.294]
 [ 7.     36.965 50.753]
 [ 8.     89.134 66.291]
 [ 9.     77.324 81.234]]
[0 0 1 1 0 0 1 0 1 1]
X_train:  [[ 4.     35.847 72.902]
 [ 9.     77.324 81.234]
 [ 1.     30.286 43.894]
 [ 6.     80.231 70.294]
 [ 7.     36.965 50.753]
 [ 3.     60.182 86.308]
 [ 0.     34.623 78.024]
 [ 5.     55.345 72.345]]
y_train:  [0 1 0 1 0 1 0 0]
X_test:  [[ 2.     79.032 75.344]
 [ 8.     89.134 66.291]]
y_test:  [1 1]
Accuracy on training subset : {:3f} 1.0
Accuracy on test subset : {:3f} 1.0
```

Q16. Use some function for neural network, like Stochastic Gradient Descent or backpropagation – algorithm to predict the value based on the dataset of problem 14.

#importing necessary libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn.linear_model import LogisticRegression

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.neural_network import MLPClassifier

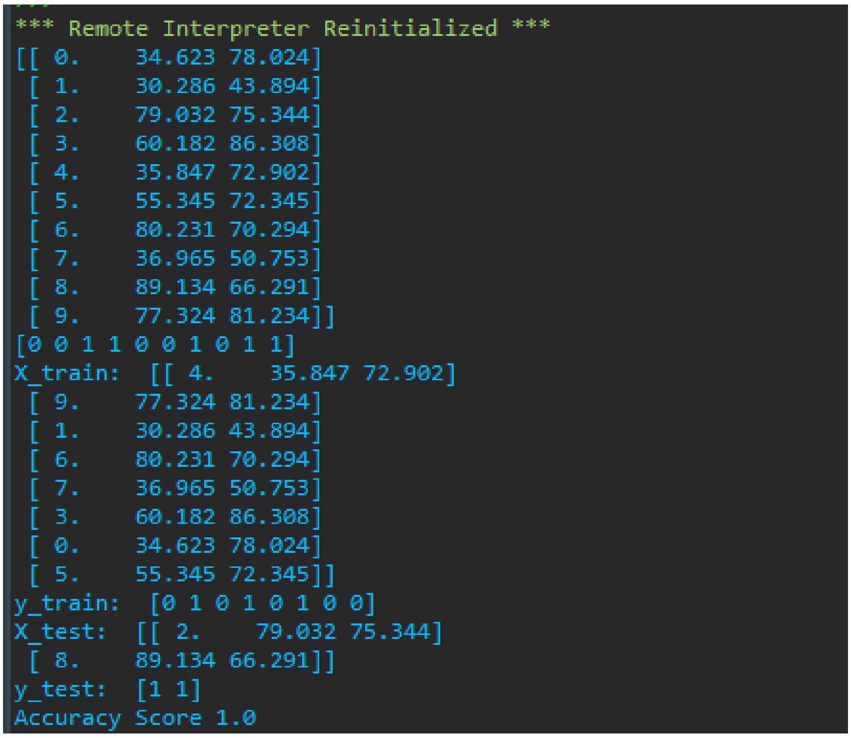
from sklearn.metrics import classification_report, confusion_matrix

from sklearn.metrics import accuracy_score

#importing the dataset

```
df = pd.read_csv("C:/Users/DELL/OneDrive/Desktop/ML Datasets/Students_data(q14).csv")
df


#assigning independent features to x and dependent features of y
X = df.iloc[:, :-1].values
print(X)


y = df.iloc[:, -1].values
print(y)


#splitting the dataframe into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
print ("X_train: ", X_train)
print ("y_train: ", y_train)
print("X_test: ", X_test)
print ("y_test: ", y_test)


#fit only to the training data
scaler = StandardScaler()
scaler.fit(X_train)


#scaling the data
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


#there are 3 layers with same number of neurons
```

```
mlp = MLPClassifier(hidden_layer_sizes = (30,30,30))

mlp.fit(X_train,y_train)


#printing accuracy score of the model

predictions = mlp.predict(X_test)

print('Accuracy Score',accuracy_score(y_test,predictions))
```



```
*** Remote Interpreter Reinitialized ***
[[ 0.     34.623 78.024]
 [ 1.     30.286 43.894]
 [ 2.     79.032 75.344]
 [ 3.     60.182 86.308]
 [ 4.     35.847 72.902]
 [ 5.     55.345 72.345]
 [ 6.     80.231 70.294]
 [ 7.     36.965 50.753]
 [ 8.     89.134 66.291]
 [ 9.     77.324 81.234]]
[0 0 1 1 0 0 1 0 1 1]
X_train:  [[ 4.     35.847 72.902]
 [ 9.     77.324 81.234]
 [ 1.     30.286 43.894]
 [ 6.     80.231 70.294]
 [ 7.     36.965 50.753]
 [ 3.     60.182 86.308]
 [ 0.     34.623 78.024]
 [ 5.     55.345 72.345]]
y_train:  [0 1 0 1 0 1 0 0]
X_test:  [[ 2.     79.032 75.344]
 [ 8.     89.134 66.291]]
y_test:  [1 1]
Accuracy Score 1.0
```