Data Driven Computing and Networking (DDCN-2019)

Programming with Python

Solution for Lab Session – 01

1. Open the Spyder (IDE for Python) installed on your system and create a new Python project named **DDCN-2019**. Create three folders under the project DDCN-2019 – "**code**", "**data**" and "**figures**". "code" directory will be used to store Python code scripts written in DDCN-2019, "data" directory will be used to store the datasets which is to be used in DDCN-2019 and "figures" directory will be used to store the plots and figures drawn in DDCN-2019. All the further exercises to be carried out in the workshop – DDCN-2019 are to be done in this project only.

Solution: In the menu bar of the Spyder IDE, select Project -> New Project.

For creating folders, right click the project name DDCN-2019 in the project explorer window and then select New -> Folder. Create all the three folders one by one.

2. Create two lists **X** and **Y**. **X** have numbers {12, 23, 34, 45, 56, 67, 78, 89, 90} and **Y** has a sequence of equal-spaced 9 numbers from 54 to 6. With reference to list **X** and **Y**, perform the following tasks:

Solution: Import package numpy for creating array

```
import numpy as np
X=np.array([12, 23, 34, 45, 56, 67, 78, 89, 90])
Y=np.array(range(54, 0, -6))
```

a. Display **X**, **Y** and their length.

```
print ("X is", X)
print ("Y is", Y)
print ("Length of X is", len(X))
print ("Length of Y is", len(Y))
```

b. Add 5 to all members of **X** and store it in a list **X_add_5**. Display **X_add_5**.

```
X_add_5 = X + 5
print ("X_add_5 is",  X_add_5)
```

c. Multiply 5 to all members of **Y** and store it in a list **Y_times_5**. Display **Y_times_5**.

```
Y_times_5 = Y * 5
print ("Y_times_5 is",  Y_times_5)
```

d. Add all the individual elements of **X** and **Y** and store it in a list **Sum_XY**. Display **Sum_XY**.

```
Sum_XY=X+Y
print ("Sum_XY is", Sum_XY)
```

e. Multiply all the individual elements of **X** and **Y** and store it in a list **Prod_XY**. Display **Prod_XY**.

```
Prod_XY=X*Y
print ("Prod_XY is", Prod_XY)
```

f. List all the objects created along with their details.

```
whos
```

g. Delete the list **Prod_XY**.

```
del Prod_XY
```

h. Display square root of all the elements of **Sum_XY**.

```
Sum_XY**(0.5)
```

i. Create a list **Z** as sequence of numbers 0 to 8. Compute list **Power** such that each element of list **Y** is raised to the power of corresponding element at same position in list **Z**. Display list **Power**.

```
Z=np.array(range(9))
Power = Y ** Z
print ("Power is", Power)
```

j. Generate a vector **Rand_Var** of random normal variables which has 50 elements, whose mean is 123 and standard deviation is 20. Set the seed of the above operation as 111 to generate the same random vector again.

```
np.random.seed(111)
L =np.random.normal(123.0, 20.0, 50)
print (L) # It will display truncated list .. L[0], L[1], .. L[49] can be printed
```

k. Compute the correlation between two lists - **X** and **Y**.

```
print (np.corrcoef(X,Y))
```

l. Compute mean, variance and standard deviation of vector **Power**.

```
print ("mean of Power is ", np.mean(Power))
print ("Variance of Power is ", np.var(Power))
print ("Standard Deviation of Power is ", np.std(Power))
```

m. Plot a line graph of **X** Vs **Y_times_5**, with label of x-axis as XX and label of y-axis as YY, and color of graph line as "Red".

```
import matplotlib.pyplot as plt
plt.plot(X, Y_times_5,'r')
plt.xlabel('XX')
plt.ylabel('YY')
plt.show()
```

n. Add a dashed line of **X_add_5** Vs **Y** in "Blue" color to the graph drawn in (m). Save the drawn plot as "asgn1_n.pdf" at path – DDCN-2019/figures".

```
#From Spyder IDE(top right) browse working directory to DDCN-2019\figures
plt.plot(X, Y_times_5, 'r', X_add_5, Y, 'b--')
plt.xlabel('XX')
plt.ylabel('YY')
plt.savefig(' asgn1_n.pdf')
plt.show()
```

o. Add points for **X_add_5** Vs **Sum_XY** in "Green" color to the graph drawn in (n). Save the drawn plot as "asgn1_o.jpeg" at path – DDCN-2019/figures".

```
plt.plot(X, Y_times_5, 'r', X_add_5, Y, 'b--', X_add_5, Sum_XY, 'go')
plt.xlabel('XX')
plt.ylabel('YY')
plt.savefig(' asgn1_o.jpeg')
plt.show()
```

3. Write a Python Script – "**prime_number.py**" at path – "DDCN-2019/code" to enter any number from user and display whether it's prime or not. The script should terminate if user does not opt for further continuation. The output should be as follows:

Enter an integer: 37

The entered number 37 is prime

Do you want to continue (y/n) : y

Enter an integer: 35

The entered number 35 is composite

Do you want to continue (y/n) : n

Solution : # From Spyder IDE(top right) browse working directory to DDCN-2019\code

```
ch = 'y'
while (ch == 'y'):
    num_s = input("Enter a number : ")
    num = int(num_s)
    flag = 0
    i = 2
    while (i < num**(0.5)):
        if (num % i == 0):
            flag = 1
            break
        i = i + 1
    if (flag == 1):
        print( "The entered number ", num, " is Composite")
    else:
        print ("The entered number ", num, " is Prime")
    ch = input("Do you want to continue(y/n) : ")
```

4. Download the dataset "Auto.csv", available at ftp and save it in folder – "DDCN-2019/data". Perform the following tasks on the "Auto.csv":

Solution: #From Spyder IDE(top right) browse working directory to DDCN-2019\data

```
# Import package pandas for importing .csv file
```

```
import pandas as pd
```

a. Import the data set – "Auto.csv" into Python, under the name "myData" and view it.

```
myData = pd.read_csv('C:/Users/pc//Desktop/DDCN 2019/basic/Dataset/Auto.csv') # change path as required
for i in range(0, len(myData.index)):
    print (myData.iloc[i])
```

b. Display the dimensions and summary of "myData".

```
print( myData.shape)
print( myData.describe())
```

c. Display row number 33, 127, 331, 337 and 355 of myData and verify that they have missing values(?).

```
for i in (32, 126,330,336,354):
    print( myData.iloc[i])
```

d. Replace the values in "myData" containing "?" with an "NaN" and view row number 33, 127, 331, 337 and 355.

```
myData=pd.read_csv('C:/Users/pc/Desktop/DDCN-2019/Basic/Dataset/Auto.csv',
na_values=["?"]) # change path as required
for i in (32, 126,330,336,354):
    print (myData.iloc[i])
```

e. Remove all the rows of "myData" which have missing values, store result into a new dataframe – "newData" and write "newData" into a file "newAuto.csv".

```
newData = myData.dropna()
newData.to_csv('newAuto.csv', index=True)
```

f. Display dimension and the summary of "newData".

```
print (newData.shape)
print (newData.describe())
```

5. Consider the given dataset:

	EMPID	Gender	Age	Sales	BMI	Income
0	E001	M	34	123	Normal	350
1	E002	F	40	114	Overweight	450
2	E003	F	37	135	Obesity	169
3	E004	M	30	139	Underweight	189
4	E005	F	44	117	Underweight	183
5	E006	M	36	121	Normal	80
6	E007	M	32	133	Obesity	166
7	E008	F	26	140	Normal	120
8	E009	M	32	133	Normal	75
9	E010	M	36	133	Underweight	40

a. Create a dataframe with the above data array.
b. Plot a histogram for the above data to represent age, sales and income.
c. Now compare the above mentioned attributes (age, sale and income) using a column chart.
d. Draw a box plot for for each of the three mentioned attribute: age, sale and income.
e. Also represent the above information using pie chart and scatter plot.

Solution:

```
import pandas as pd
import matplotlib.pyplot as plt


# create 2D array of table given above
data = [['E001', 'M', 34, 123, 'Normal', 350],
        ['E002', 'F', 40, 114, 'Overweight', 450],
        ['E003', 'F', 37, 135, 'Obesity', 169],
        ['E004', 'M', 30, 139, 'Underweight', 189],
        ['E005', 'F', 44, 117, 'Underweight', 183],
        ['E006', 'M', 36, 121, 'Normal', 80],
```

['E007', 'M', 32, 133, 'Obesity', 166],

['E008', 'F', 26, 140, 'Normal', 120],

['E009', 'M', 32, 133, 'Normal', 75],

['E010', 'M', 36, 133, 'Underweight', 40]]

a.
```
# dataframe created with
# the above data array
df = pd.DataFrame(data, columns = ['EMPID', 'Gender',
                  'Age', 'Sales',
                  'BMI', 'Income'] )
```

b.
```
# create histogram for numeric data
df.hist()

# show plot
plt.show()
```

c.
```
# create column chart
df.plot.bar()
# plot between 2 attributes
plt.bar(df['Age'], df['Sales'])
plt.xlabel("Age")
plt.ylabel("Sales")
plt.show()
```

d.
```
#plot box plot for each of the 3 attribute individually.
plt.boxplot(df['Age'])
plt.show()
plt.boxplot(df['Sales'])
plt.show()
plt.boxplot(df['Income'])
plt.show()
```

e.
```
# Plot pie chart and scatter plot for each of the attribute.
```

```
plt.pie(df['Age'], labels = {"A", "B", "C",
                             "D", "E", "F",
                             "G", "H", "I", "J"},

autopct ='% 1.1f %%', shadow = True)
plt.show()

plt.pie(df['Income'], labels = {"A", "B", "C",
                              "D", "E", "F",
                              "G", "H", "I", "J"},

autopct ='% 1.1f %%', shadow = True)
plt.show()

plt.pie(df['Sales'], labels = {"A", "B", "C",
                             "D", "E", "F",
                             "G", "H", "I", "J"},
autopct ='% 1.1f %%', shadow = True)
plt.show()
# scatter plot between income and age
plt.scatter(df['Income'], df['Age'])
.show()

# scatter plot between income and sales
plt.scatter(df['Income'], df['Sales'])
plt.show()

# scatter plot between sales and age
plt.scatter(df['Sales'], df['Age'])
plt.show()
```