Data Analytics Lab

Lab program 1:

Write a python program to perform Linear search

```
def linear_Search(list,n,x):

    # Searching list1 sequentially
    for i in range(0, n):
        if (array[i] == x):
            return i
    return -1


array = [1 ,3, 5, 4, 7, 9]


n = len(array)
x=int(input("enter the item"))
res = linear_Search(list,n,x)
if(res == -1):
    print("Element not found")
else:
    print("Element found at index: ", res)
```

OUTPUT:

```
enter the item 9
Element found at index: 5
```

Lab program 2:

Write a python program to insert an element into a sorted list

```
    def insert(list, n):

        index = len(list)
        # Searching for the position
        for i in range(len(list)):
            if list[i] > n:
                index = i
                break
```

```
        # Inserting n in the list
        if index == len(list):
            list = list[:index] + [n]
        else:
            list = list[:index] + [n] + list[index:]
        return list

# Driver function
list = [1, 2, 4]
n = 3

print(insert(list, n))
```

output:

```
[1, 2, 3, 4]
```

Lab program 3:

Write a python program using object oriented programming to demonstrate encapsulation, overloading and inheritance

```
##parent class
class Operations:
#constructor
        def __init__(self,a,b):
                self.a=a
                self.b=b
        def add(self):
                sum=self.a+self.b
                print ("sum of a and b is : ", sum)
#Child class
class Myclass(Operations):
#constructor
        def _init__(self,a,b):
#invoking parent class constructor
                super(self,Operations).__init__(a,b)
                self.a=a
                self.b=b
        def sub(self):
                sub=self.a-self.b
                print("Subtraction of C and D is : ", sub)
ob1=Myclass(20,30)
ob2=Myclass("ONE","TWO")
ob1.add()#to sum up integers
ob2.add()#to sum up String
ob1.sub()
```

OUTPUT:
sum of a and b is : 70
sum of a and b is : ONETWO
Subtraction of C and D is : -10

Lab program 4:

Write a python program to demonstrate

i)import datasets ii)cleaning the data iii)data frame manipulation using Numpy

```
import pandas as pd
import numpy as np
#READING THE FILE INTO A DATAFRAME
df = pd.read_csv('C:/Users/Vybhav/Desktop/Python Materials/BL-Flickr-Images-Book.csv')
#VIEWING FIRST 5 ROWS OF THE DATASET
df.head()
#CLEANING DATASET - REMOVING A FEW UNWANTED COLUMNS
to_drop = ['Edition Statement',
           'Corporate Author',
           'Corporate Contributors',
           'Former owner',
           'Engraver',
          'Contributors',
          'Issuance type','Shelfmarks']
df.drop(to_drop, inplace=True, axis=1)
df.head()
#REPLACING THE INDEX
df = df.set_index('Identifier')
df.head()
#DISPLAYING DATE OF PUBLICATION FIELD
df.loc[1905:, 'Date of Publication'].head(10)
#FORMATTING FIELD DATE OF PUBLICATION
#Remove the extra dates in square brackets
#Convert date ranges to their “start date”
#Completely remove the dates we are not certain about and replace them with NumPy’s NaN
```

```
#Convert the string nan to NumPy's NaN value
xtr = df['Date of Publication'].str.extract(r'^(\d{4})', expand=False)
xtr.head()
#CHANGING DATA TYPE OF DATE OF PUBLICATION
df['Date of Publication'] = pd.to_numeric(xtr)
df['Date of Publication'].dtype
df.head()
#DATA CLEANING USING NUMPY
df['Place of Publication'].head(10)
#DATE OF PUBLICATION - Newcastle-upon-Tyne
df.loc[4157862]
# DATE OF PUBLICATION - Newcastle upon Tyne
df.loc[4159587]
df.head()
#DATA CLEANING USING NUMPY
df['Place of Publication'].head(10)
#CLEANING COLUMN Place of Publication USING NUMPY WHERE
pub = df['Place of Publication']
london = pub.str.contains('London')
oxford = pub.str.contains('Oxford')
df['Place of Publication'] = np.where(london, 'London',
                                      np.where(oxford, 'Oxford',
                                               pub.str.replace('-', ' ')))
#DATE OF PUBLICATION - Newcastle-upon-Tyne
df.loc[4157862]
df.head()
```

OUTPUT

Out[1]:

Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
206	London	1879.0	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
216	London	1868.0	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
218	London	1869.0	Bradbury, Evans & Co.	Love the Avenger. By the author of "All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
472	London	1851.0	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
480	London	1857.0	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

Lab program 5:

Implement a Python program to demonstrate the following using Numpy

5A. Array manipulation,searching,sorting and splitting

```
import numpy as np
import numpy as np2

a = np.array([[1, 4, 2],
              [3, 4, 6],
             [0, -1, 5]])
#array before sorting
print ("Array elements before sorting:\n")
print(a)

# sorted array

print ("Array elements in sorted order:\n")
print(np.sort(a, axis = None))
i = np.where(a == 6)
print("i = {}".format(i))

# specify sort algorithm

print ("Column wise sort by applying merge-sort:\n")
print(np.sort(a, axis = 0, kind = 'mergesort'))

#  looking for value 30 in arr and storing its index in i

#Splitting of Arrays
b = np.array([[3, 4, 5],
              [6, 7, 8],
              [9, 10, 11]])
print(b)
print(b[0:3,1])
```

OUTPUT:

```
Array elements before sorting:

[[ 1  4  2]
 [ 3  4  6]
```

```
 [ 0 -1  5]]
Array elements in sorted order:

[-1  0  1  2  3  4  4  5  6]
i = (array([1]), array([2]))
Column wise sort by applying merge-sort:

[[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]
[[ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
[ 4  7 10]
```

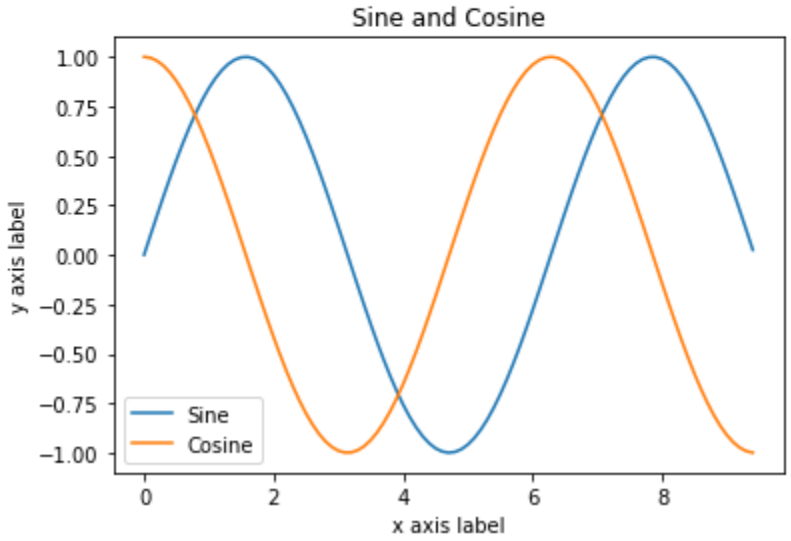
5B. Broadcasting and plotting numpy arrays

```
import numpy as np

import matplotlib.pyplot as plt

#To Demonstrate Numpy Broadcasting

A = np.array([[11, 22, 33], [10, 20, 30]])

print(A)

b = 4

print(b)

C = A + b

print(C)

# Plotting using Broadcasting

# Computes x and y coordinates for

# points on sine and cosine curves

x = np.arange(0, 3 * np.pi, 0.1)

y_sin = np.sin(x)

y_cos = np.cos(x)

# Plot the points using matplotlib

plt.plot(x, y_sin)

plt.plot(x, y_cos)
```

```
plt.xlabel('x axis label')

plt.ylabel('y axis label')

plt.title('Sine and Cosine')

plt.legend(['Sine', 'Cosine'])

plt.show()
```

OUTPUT:

```
[[11 22 33]
 [10 20 30]]
4
[[15 26 37]
 [14 24 34]]
```



Lab program 6:

Implement python program to demonstrate

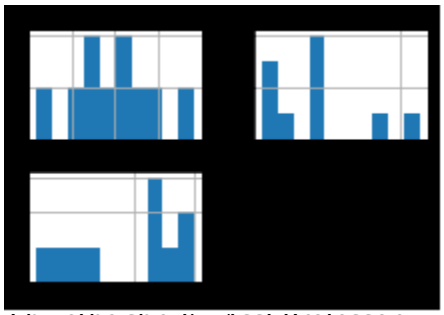
Data Visualization with various types of graphs using Numpy

```
import pandas as pd

import matplotlib.pyplot as plt

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] ]

# dataframe created with

# the above data array

df = pd.DataFrame(data, columns = ['EMPID', 'Gender', 'Age', 'Sales','BMI', 'Income'] )

# create histogram for numeric data

df.hist()

# show plot

plt.show()
```

OUTPUT:

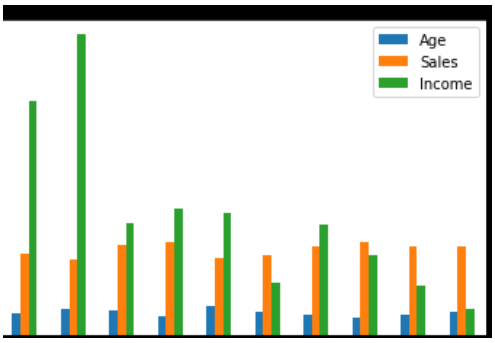
(Histogram)



```
# Dataframe of previous code is used here
# Plot the bar chart for numeric values
# a comparison will be shown between
# all 3 age, income, sales
df.plot.bar()
# plot between 2 attributes
plt.bar(df['Age'], df['Sales'])
plt.xlabel("Age")
plt.ylabel("Sales")
plt.show()
```

OUTPUT:

(Column Chart)



Lab program 7:

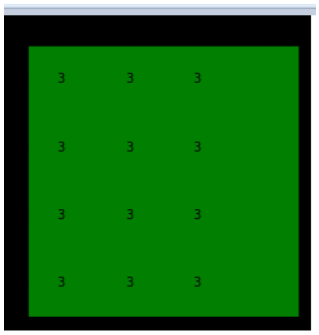
Write a python program to take m x n integer matrix and print its attributes in matplolib

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

```
fig, ax = plt.subplots()

m = int(input('number of rows, m = '))

n = int(input('should be greater than m value , number of columns, n = '))

#fetching max value to make sure the matrix is plotted as max * max

matrix = np.random.randint(m, n, size=(n, n))

#matshow - Plot the values of a 2D matrix or array as color-coded image

ax.matshow(matrix, cmap='ocean')

for i in range(m):

  for j in range(n):

    c = matrix[i, j]

    ax.text(i, j, str(c), va='center', ha='center')

plt.show()
```

output:

```
number of rows, m = 3
should be greater than m value , number of columns, n = 4
```



Lab program 8:

Write a python program to demonstrate LinearRegression model

```
#SIMPLE LINEAR REGRESSION MODEL

import numpy as np

from sklearn.linear_model import LinearRegression
```

```
#Creating dummy dataset

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))

y = np.array([5, 20, 14, 32, 22, 38])


#creating instance to represent the regression model

model = LinearRegression().fit(x, y)


#to get results of the model

r_sq = model.score(x, y)

print('coefficient of determination:', r_sq)

print('intercept:', model.intercept_)

print('slope:', model.coef_)

y_pred = model.predict(x)

print('predicted response:', y_pred)


#APPLYING MODEL TO NEW DATA

x_new = np.arange(5).reshape((-1, 1))

print(x_new)

y_new = model.predict(x_new)

print(y_new)
```

OUTPUT:

```
('coefficient of determination:', 0.7158756137479542)
('intercept:', 5.633333333333329)
('slope:', array([0.54]))
('predicted response:', array([ 8.33333333, 13.73333333, 19.13333333, 24.53333333,
29.93333333,
```

```
       35.33333333]))
[[0]
 [1]
 [2]
 [3]
 [4]]
[5.63333333 6.17333333 6.71333333 7.25333333 7.79333333]
```

Lab program 9

Write a Python program to demonstrate the generation of logistic regression models uning python

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from matplotlib import pyplot as plt
import seaborn as sns
#IMPORTING DATA
df = pd.read_csv('C:/Users/Vybhav/Desktop/creditcard_v2.csv')
#CLEANING DATA
#TO REMOVE DUPLICATE ROWS
df.drop_duplicates(inplace=True)
#REMOVING TIME COLUMN AS IT IS NOT USEFUL
df.drop('Time', axis=1, inplace=True)
#DIVIDING DATA IN TARGET COLUMN AND FEATURE COLUMN
X = df.iloc[:,df.columns != 'Class']
y = df.Class
#DIVIDING DATA INTO TRAINING AND TEST SETS
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.20, random_state=5, stratify=y)
#SCALING THE DATA
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
```

```
#BUILDING THE MODEL
model = LogisticRegression()
#TRAINING THE MODEL
model.fit(X_train_scaled, y_train)
#EVALUATING THE MODEL
train_acc = model.score(X_train_scaled, y_train)
print("The Accuracy for Training Set is {}".format(train_acc*100))
#EVALUATING ON TEST SET
X_test_scaled = scaler.transform(X_test)
y_pred=model.predict(X_test_scaled)
test_acc = accuracy_score(y_test, y_pred)
print("The Accuracy for Test Set is {}".format(test_acc*100))
#GENERATING CLASSIFICATION REPORT
print(classification_report(y_test, y_pred))
```

OUTPUT:

```
The Accuracy for Training Set is 99.7743973186
The Accuracy for Test Set is 99.9641191245
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     16712
           1       0.64      0.90      0.75        10

   micro avg       1.00      1.00      1.00     16722
   macro avg       0.82      0.95      0.87     16722
weighted avg       1.00      1.00      1.00     16722
```

Lab program 10

write a program to demonstrate Time series analysis with pandas

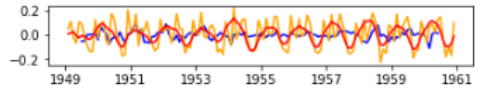
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima_model import ARIMA
#IMPORTING DATA
df = pd.read_csv("C:/Users/Vybhav/Desktop/python lab programs/AirPassengers.csv")
#DISPLAYING FIRST 5 ROWS
print(df.head())
#CONVERTING MONTH INTO DATETIME OBJECT
df['Month'] = pd.to_datetime(df['Month'], format='%Y-%m')
print(df.head())
#CONVERTING MONTH TO AN INDEX
df.index = df['Month']
del df['Month']
#Estimating & Eliminating Trend by taking log transform
ts_log = np.log(df)
#Eliminating Trend and Seasonality using Differencing
ts_log_diff = ts_log - ts_log.shift()
#Decomposing
decomposition = seasonal_decompose(ts_log)
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid
plt.subplot(414);
plt.plot(residual, label='Residuals', color='blue');
model = ARIMA(ts_log, order=(2, 1, 2))
results_ARIMA = model.fit(disp=-1)
plt.plot(ts_log_diff, color='orange');
plt.plot(results_ARIMA.fittedvalues, color='red');
```

OUTPUT:

```
Month  #Passengers
0  1949-01          112
1  1949-02          118
2  1949-03          132
```

```
3  1949-04          129
4  1949-05          121
       Month  #Passengers
0 1949-01-01          112
1 1949-02-01          118
2 1949-03-01          132
3 1949-04-01          129
4 1949-05-01          121
```

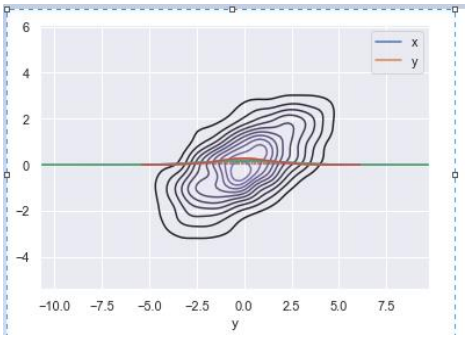


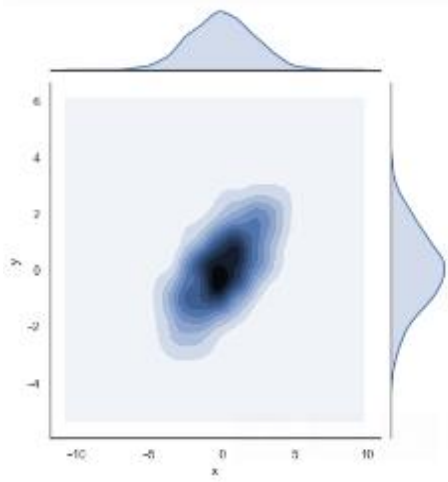
Lab program 11

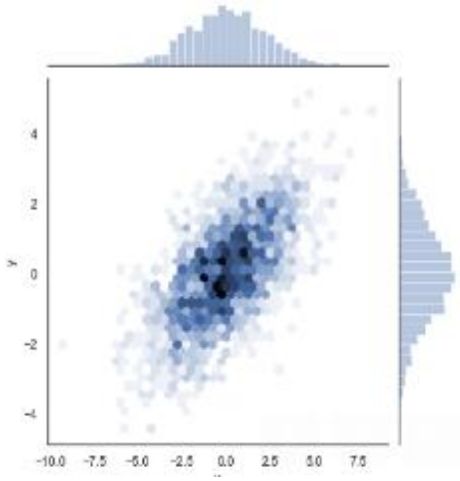
Write a python program to demonstrate Data visualization using Seaborn

```
import seaborn as sns
import numpy as np

sns.set()

data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]], size=2000)

data = pd.DataFrame(data, columns=['x', 'y'])

#KERNEL DENSITY ESTIMATION

for col in 'xy':

    sns.kdeplot(data[col], shade=True)

#COMBINING KDE AND HISTOGRAMS USING distplot

sns.distplot(data['x'])

sns.distplot(data['y'])

#USING kdeplot TO FETCH TWO-DIMENSIONAL VISUALIZATION

sns.kdeplot(data)

#USING jointplot

with sns.axes_style('white'):

    sns.jointplot("x", "y", data, kind='kde')

with sns.axes_style('white'):

    sns.jointplot("x", "y", data, kind='hex')

with sns.axes_style('white'):

    sns.jointplot("x", "y", data, kind='hex')
```

OUTPUT:


x
y
6
4
2
0
−2
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
y


y
x


y
4
2
0
−2
−4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5