COURSE LABORATORY MANUAL

A. LABORATORY OVERVIEW

Degree:	MCA	Programme:	MCA
Semester:	III	Academic Year:	2022-23
Laboratory Title:	**Data Analytics Lab**	Laboratory Code:	**20MCA36**
L-T-P-S:	0-0-4-0	Duration of SEE:	3 Hrs
Total Contact Hours:	40 Hrs	SEE Marks:	60
Credits:	02	CIE Marks:	40
Lab Manual Author:	**Dr.Vandana B.S**	Sign:	Dt :22/11/2022

B. DESCRIPTION

1. PREREQUISITES:

- Object oriented concepts and data analytics
- Programming Skill and Creative Thinking

2. BASE COURSE:

- Object Oriented Programming with Java (20MCA31)

3. COURSE OUTCOMES:

At the end of the course, the student will be able to;

CO1.Develop python program to perform search/sort on a given data set.
CO2.Demonstrate object oriented principles.
CO3. Demonstrate data visualization using Numpy for a given problem.
CO4. Demonstrate regression model for a given problem.
CO5.Deign and develop an application for the given problem .

4. RESOURSES REQUIRED:

- Hardware resources:
  - Personal Computer
  - Windows / Linux operating system
- Software resources:
  - Anaconda with jupyter notebook

Prepared by: Dr.Vandana B.S Director

5. RELEVANCE OF THE COURSE:

- IoT(20MCA37)

6. GENERAL INSTRUCTIONS:

- To execute the programs follow the steps given below.
  - Create python program with .py extension
  - Run python Program

7. CONTENTS:

Expt No.	Title of the Experiments	RBT	CO

1	Write a Python program to perform linear search.	L3	CO1
2	Write a Python program to insert an element into a sorted list.	L3	CO1
3	Write a python program using object oriented programming to demonstrate encapsulation, overloading and inheritance	L3	CO2
4	Implement a python program to demonstrate 1) Importing Datasets 2) Cleaning the Data 3) Data frame manipulation using Numpy	L3	CO3
5	Implement a python program to demonstrate the following using NumPy a) Array manipulation, Searching, Sorting and splitting. b) broadcasting and Plotting NumPy arrays	L3	CO3
6	Implement a python program to demonstrate Data visualization with various Types of Graphs using Numpy	L3	CO3
7	Write a Python program that creates a mxn integer arrayand Prints its attributes using matplotlib	L3	CO3
8	Write a Python program to demonstrate the generation of linear regression models.	L3	CO4
9	9.Write a Python program to demonstrate the generation of logistic regression models using Python.	L3	CO4
10	Write a Python program to demonstrate Timeseries analysis with Pandas.	L3	CO3
11	Write a Python program to demonstrate Data Visualization using Seaborn.	L3	CO3
12	Part-B -MiniProject		

8. REFERENCE:

1. Allen B. Downey, “Think Python: How to Think Like a Computer Scientist‘‘, 2nd edition,Updated for Python 3, Shroff/O‘Reilly Publishers, 2016 (http://greenteapress.com/wp/thinkpython/)
2. Guido van Rossum and Fred L. Drake Jr, —An Introduction to Python – Revised and updated for Python 3.2, Network Theory Ltd., 2011.
3. Jake Vander plas, “Python Data Science Handbook: Essential tools for working with data”, O‘Reilly Publishers, I Edition. References: 1. Mark Lutz, “Programming Python”, O'Reilly Media, 4th edition, 2010.
4. Tim Hall and J-P Stacey, “Python 3 for Absolute Beginners”, Apress, 1st edition, 2009.
5. Magnus Lie Hetland, “Beginning Python: From Novice to Professional”, Apress, Second Edition, 2005.
6. Shai Vaingast, “Beginning Python Visualization Crafting Visual Transformation Scripts”, Apress, 2nd edition, 2014.
7. 6. Wes Mc Kinney, “Python for Data Analysis”, O'Reilly Media, 2012

-

C. EVALUATION SCHEME

For CBCS 2020 scheme:

1. Laboratory Components: 20 Marks
   (Record writing, Laboratory performance and Viva-voce)
2. Laboratory IA tests: 20 Marks
   (Minimum 2 IAs are mandatory. For the final IA test marks, average of the 2 IA test

marks shall be considered and converted to maximum of 20)

3. Continuous Internal Evaluation (CIE) = 20 + 20 = 40 Marks
4. *SEE* : 60* Marks
   (*The SEE will be conducted for 100 marks and proportionally reduced to 60 marks)

-

D1. ARTICULATION MATRIX

Mapping of CO to PO												
POs												
COs	1	2	3	4	5	6	7	8	9	10	11	12
CO1.Develop python program to perform search/sort on a given data set.	2	-	-	-	-	-	-	-	-	-		-
CO2.Demonstrate object oriented principles.	2	-	-	-	-	-	-	-	-	-		-
CO3. Demonstrate data visualization using Numpy for a given problem.	2	-	-	-	-	-	-	-	-	-		-
CO4. Demonstrate regression model for a given problem.	2	-	-	-	-	-	-	-	-	-		-
CO5.	2	-	-	-	3	-	-	-	3	-	3	-

Note: Mappings in the Tables D1 (above) and D2 (below) are done by entering in the corresponding cell the Correllation Levels in terms of numbers. For Slight (Low): 1, Moderate (Medium): 2, Substantial (High): 3 and for no correllation: " - ".

D2. ARTICULATION MATRIX CO v/s PSO

Mapping of CO to PSO		
	PSOs	
COs	**PSO1**	**PSO2**
CO1.Develop python program to perform search/sort on a given data set.	-	2
CO2.Demonstrate object oriented principles.	-	2
CO3. Demonstrate data visualization using Numpy for a given problem.	-	2
CO4. Demonstrate regression model for a given problem.	-	2
CO5.Deign and develop an application for the given problem .	-	2

-

E. EXPERIMENTS

1. **EXPERIMENT NO: 1**
2. **TITLE:** Write a Python program to perform linear search
3. **LEARNING OBJECTIVES:** Is to learn basics of python programming.
4. **AIM:** Is to apply python constructs to execute Linear Search Algorithms.
5. **THEORY / HYPOTHESIS:**

```
linear_search (list, value)
  for each item in the list
    if match item == value
      return the item's location
```

```
    end if
  end for
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
numlist = []
n = int(input("Enter the list size "))
print("\n")
for i in range(0, n):
   print("Enter number at index", i, )
   item = int(input())
   numlist.append(item)
print("Array of numbers ", numlist)
key = int(input("Enter the key element to search"))
loc=search(numlist,key)
if loc==-1:
   print("Item Not Found")
else:
     print("Item found in loc",loc)

def search(numlist,key):
   for j in range(len(numlist)):
            if(numlist[j] == key):
                  return j
   return -1
```

7.OUTPUT: Enter the list size 10

```
Enter number at index 0
10
Enter number at index 1
23
Enter number at index 2
-23
Enter number at index 3
34
Enter number at index 4
45
Enter number at index 5
67
Enter number at index 6
89
Enter number at index 7
78
Enter number at index 8
98
Enter number at index 9
6
Array of numbers  [10, 23, -23, 34, 45, 67, 89, 78, 98, 6]
Enter the key element to search-23
Item found in loc 2
          Enter the list size 2
Enter number at index 0
```

```
10
Enter number at index 1
90
Array of numbers  [10, 90]
Enter the key element to search100
Item Not Found
```

8. **LEARNING OUTCOMES:** Able to apply basic python programming constructs.

9. **APPLICATION AREAS:** Easy to implement

10. **REMARKS: -**

1. **EXPERIMENT NO: 2**

2. **TITLE:** Write a Python program to insert an element into a sorted list

3. **LEARNING OBJECTIVES:** Student able to learn python List data structure and related functions

4. **AIM:** Is to apply list methods.

5. **THEORY / HYPOTHESIS:**

LIST AND ITS METHODS:

1.append():Used for appending and adding elements to the end of the List.
2.copy():It returns a shallow copy of a list
3.clear():This method is used for removing all items from the list.
4.count():This methods count the elements
5.extend():Adds each element of the iterable to the end of the List
6.index():Returns the lowest index where the element appears.
7.insert():Inserts a given element at a given index in a list.
8.pop():Removes and returns the last value from the List or the given index value.
9.remove():Removes a given object from the List.
10.reverse():Reverses objects of the List in place.
11.sort():Sort a List in ascending, descending, or user-defined order
12.max():Calculates maximum of all the elements of List

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
li=[]
flag=1
while(flag):
    ele=int(input("Enter an element"))
    j=len(li)-1
    while(j>=0):
        if(ele>li[j]):
            break;
        j-=1
    if(len(li)==0 or j==len(li)-1):
        li.append(ele)
    else:
        li.insert(j+1,ele)
    print(li)
    flag=int(input("Do you want to insert more elements(0/1)"))
```

7.Output Enter an element3
[3]

```
Do you want to insert more elements(0/1)1
Enter an element1
[1, 3]
Do you want to insert more elements(0/1)1
Enter an element5
[1, 3, 5]
Do you want to insert more elements(0/1)1
Enter an element-1
[-1, 1, 3, 5]
Do you want to insert more elements(0/1)1
Enter an element2
[-1, 1, 2, 3, 5]
Do you want to insert more elements(0/1)0
```

8. **LEARNING OUTCOMES: Able** to apply list data structure to solve given problem

9. **APPLICATION AREAS: List** is efficient data structure

10. **REMARKS: -**

1. EXPERIMENT NO: 3

2. **TITLE:** Write a python program using object oriented programming to demonstrate encapsulation, overloading and inheritance

3. **LEARNING OBJECTIVES:** Objective is to study object oriented concepts using python

4. **AIM:** To learn oops concepts using python libraries

5. **THEORY / HYPOTHESIS:**

Need to study class definition, Instantiate an Object in Python,__init__ method operator overloading and inheritance concepts.

6. **PROCEDURE / PROGRAMME / ACTIVITY:**

#demonstrate encapsulation, overloading and inheritance

```
class Time:
    def print_time(time):
        print('%.2d:%.2d:%.2d' % (time.hour, time.minute, time.second))
    def __init__(self, hour=0, minute=0, second=0):
        self.hour = hour
        self.minute = minute
        self.second = second
    def __str__(self):
        return '%.2d:%.2d:%.2d' % (self.hour, self.minute, self.second)
    def __add__(self, other):
        seconds = self.time_to_int() + other.time_to_int()
        return int_to_time(seconds)
    def time_to_int(time):
        minutes = time.hour * 60 + time.minute
        seconds = minutes * 60 + time.second
        return seconds
    def int_to_time(seconds):
        time = Time()
        minutes, time.second = divmod(seconds, 60)
        time.hour, time.minute = divmod(minutes, 60)
        return time
    start = Time(9, 45)
    print_time(start)
    duration = Time(1, 35)
    print_time(duration)
    print(start + duration)
```

#Demonstrate inheritance

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = [0 for i in range(no_of_sides)]

    def inputSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]

    def dispSides(self):
        for i in range(self.n):
            print("Side",i+1,"is",self.sides[i])
class Triangle(Polygon):
```

```
    def __init__(self):
        Polygon.__init__(self,3)
    def findArea(self):
        a, b, c = self.sides
        # calculate the semi-perimeter
        s = (a + b + c) / 2
        area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
        print('The area of the triangle is %0.2f' %area)
class rectangle(Polygon):
    def __init__(self):
        Polygon.__init__(self,2)
    def findArea(self):
        l,b= self.sides
        # calculate the semi-perimeter
        area = l*b
        print('The area of the triangle is %0.2f' %area)
t = Triangle()
t.inputSides()
t.dispSides()
t.findArea()
r = rectangle()
r.inputSides()
r.dispSides()
r.findArea()
```

```
09:45:00 01:35:00 11:20:00
============================================================
Enter side 1 : 2
Enter side 2 : 2
Enter side 3 : 2
Side 1 is 2.0
Side 2 is 2.0
Side 3 is 2.0
The area of the triangle is 1.73
 Enter side 1 : 2
Enter side 2 : 2
Side 1 is 2.0
 Side 2 is 2.0
 The area of the triangle is 4.00
```

8. LEARNING OUTCOMES: Students are able to handle object oriented concepts using python.

9. APPLICATION AREAS: Solve problems efficiently using objected oriented concepts.

10. REMARKS: -

COURSE LABORATORY MANUAL

1. EXPERIMENT NO: 4

2. **TITLE:** Implement a python program to demonstrate 1) Importing Datasets 2) Cleaning the Data 3) Data frame manipulation using Numpy.

3. LEARNING OBJECTIVES: Students are able to learn handle Numpy and Pandas.

4. **AIM:** Is to learn Numpy and Pandas Libraries

5. **THEORY / HYPOTHESIS:**

NumPy is a library for Python that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. Pandas is a high-level data manipulation tool that is built on the NumPy package.

6. **PROCEDURE / PROGRAMME / ACTIVITY:**

1) Importing Datasets

```
import pandas as pd
import io
import numpy as np
from google.colab import files
uploaded = files.upload()
# Read the data from the dataset into your pandas dataframe.
data = pd.read_csv(io.BytesIO(uploaded['file.csv']))
data.head(11)
```

```
Choose files  file.csv
• file.csv(text/csv) - 256 bytes, last modified: 20/11/2022 - 100% done
Saving file.csv to file (5).csv
```

	Sl_No	Name	qualification	Mobile	Address
0	1	Varun	B.E	9.448589e+09	Puttur
1	2	Deepa	Mtech	9.448889e+09	Mangalore
2	3	Ajay	MCA	9.448239e+09	Bangalore
3	4	Nandhini	MBA	NaN	NaN
4	5	Ajay	MCA	9.448239e+09	Bangalore
5	6	Ajay	MCA	9.448239e+09	Bangalore
6	7	sandeepa	MBA	NaN	Puttur
7	8	NaN	NaN	NaN	NaN

2) Cleaning Datasets

```
# remove duplicate rows
data.drop_duplicates(subset=["Name"], keep="last", inplace=True)
data.head(10)
```

	Sl_No	Name	qualification	Mobile	Address
0	1	Varun	B.E	9.448589e+09	Puttur
1	2	Deepa	Mtech	9.448889e+09	Mangalore
3	4	Nandhini	MBA	NaN	NaN
5	6	Ajay	MCA	9.448239e+09	Bangalore
6	7	sandeepa	MBA	NaN	Puttur
7	8	NaN	NaN	NaN	NaN

```
data.isnull().sum()
```

```
Sl_No          0
Name           1
qualification  1
Mobile         3
Address        2
dtype: int64
```

```
#Missing Data:
#Size of original dataset
print(data.shape)
 #Dropping the missing rows.
df_dropped = data.dropna()
data.head()
```

```
(6, 5)
```

	Sl_No	Name	qualification	Mobile	Address
0	1	Varun	B.E	9.448589e+09	Puttur
1	2	Deepa	Mtech	9.448889e+09	Mangalore
3	4	Nandhini	MBA	NaN	NaN
5	6	Ajay	MCA	9.448239e+09	Bangalore
6	7	sandeepa	MBA	NaN	Puttur

```
#Fill with provided value
data['Address']=data['Address'].fillna('Sullia')
data.head()
```

	Sl_No	Name	qualification	Mobile	Address
0	1	Varun	B.E	9.448589e+09	Puttur
1	2	Deepa	Mtech	9.448889e+09	Mangalore
3	4	Nandhini	MBA	NaN	Sullia
5	6	Ajay	MCA	9.448239e+09	Bangalore
6	7	sandeepa	MBA	NaN	Puttur

```
#Fill with provided value
mean_val=data['Mobile'].mean()
data['Mobile']=data['Mobile'].fillna(mean_val)
data
```

	Sl_No	Name	qualification	Mobile	Address
0	1	Varun	B.E	9.448589e+09	Puttur
1	2	Deepa	Mtech	9.448889e+09	Mangalore
3	4	Nandhini	MBA	9.448572e+09	Sullia
5	6	Ajay	MCA	9.448239e+09	Bangalore
6	7	sandeepa	MBA	9.448572e+09	Puttur
7	8	NaN	NaN	9.448572e+09	Sullia

3)Data frame manipulation using Numpy

```
# importiong the modules
import pandas as pd
import numpy as np
# creating the Numpy array
array = np.array([[1, 1, 1], [2, 3, 8], [3, 5, 27],])
# creating a list of index names
index_values = ['Row1', 'Row2', 'Row3']
# creating a list of column names
column_values = ['Col1', 'Col2', 'Col3']
```

```
# creating the dataframe
df = pd.DataFrame(data = array,
      index = index_values,
      columns = column_values)
# displaying the dataframe
print(df)
# calculating the Trace of a matrix
trace = np.trace(df)
 print("\nTrace of given 3X3 matrix:")
print(trace)
```

```
         first  second  third
first        1       1      1
second       2       3      8
third        3       5     27

Trace of given 3X3 matrix:
31
```

8. LEARNING OUTCOMES: Student are able to handle Dataframe using numpy and pandas.

9. APPLICATION AREAS: Data Pre-processing.

10. REMARKS: -

1. EXPERIMENT NO: 5

2. TITLE: Implement a python program to demonstrate the following using NumPy a) Array manipulation, Searching, Sorting and splitting. b) broadcasting and Plotting NumPy arrays

3. LEARNING OBJECTIVES: Objective is to learn broadcasting and Plotting NumPy arrays

4. AIM: Is to learn Numpy API's

5. THEORY / HYPOTHESIS:

✔ The term **broadcasting** refers to how numpy treats arrays with different Dimension during arithmetic operations which lead to certain constraints, the smaller array is broadcast across the larger array so that they have compatible shapes. Broadcasting provides a means of vectorizing array operations so that looping occurs in C instead of Python as we know that Numpy implemented in C. It does this without making needless copies of data and which leads to efficient algorithm implementations. There are cases where broadcasting is a bad idea because it leads to inefficient use of memory that slow down the computation.

Broadcasting: Rules:

- Broadcasting two arrays together follow these rules:
  If the arrays don't have the same rank then prepend the shape of the lower rank array with 1s until both shapes have the same length.
- The two arrays are compatible in a dimension if they have the same size in the dimension or if one of the arrays has size 1 in that dimension.
- The arrays can be broadcast together iff they are compatible with all dimensions.
- After broadcasting, each array behaves as if it had shape equal to the element-wise maximum of shapes of the two input arrays.
- In any dimension where one array had size 1 and the other array had size greater than 1, the first array behaves as if it were copied along that dimension.

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
#Array manipulation, Searching, Sorting and splitting.
import numpy as np
lt = []
n = int(input("Enter number of elements : "))
for i in range(0, n):
  ele = int(input())
  lt.append(ele)
print(lt)
arr=np.array(lt)
arr = np.sort(arr)
print("Sorted Array = {}".format(arr))
ele = int(input("Enter  element to search : "))
i = np.where(arr == ele)
print("index = {}".format(i))
s = int(input("Enter  spliting value : "))
newarr = np.array_split(arr, s)
print(newarr)
```

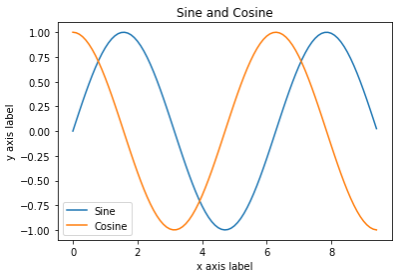
```
Enter number of elements : 5
 2 3 14 45 23
[2, 3, 14, 45, 23]
Sorted Array = [ 2 3 14 23 45]
```

```
Enter element to search : 23
index = (array([3]),)
Enter spliting value : 2
[array([ 2, 3, 14]), array([23, 45])]
```

```
#broadcasting and Plotting NumPy arrays
import numpy as np
from numpy import array, argmin, sqrt, sum
observation = array([111.0, 188.0])
codes = array([[102.0, 203.0],
               [132.0, 193.0],
               [45.0, 155.0],
               [57.0, 173.0]])
diff = codes - observation    # the broadcast happens here
dist = sqrt(sum(diff**2,axis=-1))
print(observation,observation.shape)
print(codes,codes.shape)
print(diff,diff.shape)
print(dist,dist.shape)
argmin(dist)
```

```
[111. 188.] (2,)
[[102. 203.]
 [132. 193.]
 [ 45. 155.]
 [ 57. 173.]] (4, 2)
[[ -9.  15.]
 [ 21.   5.]
 [-66. -33.]
 [-54. -15.]] (4, 2)
[17.49285568 21.58703314 73.79024326 56.04462508] (4,)
0
```

```
import numpy as np
import matplotlib.pyplot as plt
# 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()
```



8. **LEARNING OUTCOMES:** Student are able to handle interface class and its usage.

9. **APPLICATION AREAS:** Useful for Data Analysis

10. **REMARKS**: -

COURSE LABORATORY MANUAL

1. EXPERIMENT NO: 6

2. TITLE: Implement a python program to demonstrate Data visualization with various Types of Graphs using Numpy

3. LEARNING OBJECTIVES: Objective is to learn Data visualization with Python.

4. AIM: Is to learn Data visualization using Python libraries

5. THEORY / HYPOTHESIS:

Matplotlib tool for visualization in Python. Matplotlib is a multiplatform data visualization library built on NumPy arrays, and designed to work with the broader SciPy stack.

✔ Importing matplotlib

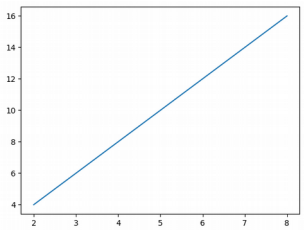
Just as we use the `np` shorthand for NumPy and the `pd` shorthand for Pandas, we will use some standard shorthands for Matplotlib imports:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

6. PROCEDURE / PROGRAMME / ACTIVITY:

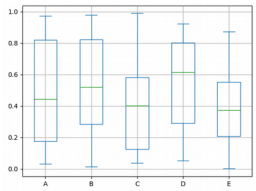
#1Line Plot

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([2,4,6,8]) # X-axis points
y = x*2 # Y-axis points
plt.plot(x, y) # Plot the chart
plt.show() # display
```



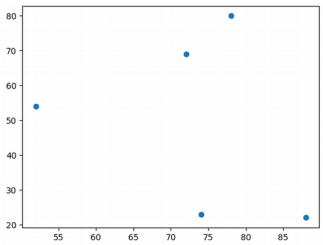
#Box Plot

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(20,5), columns=['A','B','C','D','E'])
df.plot.box(grid="True")
```



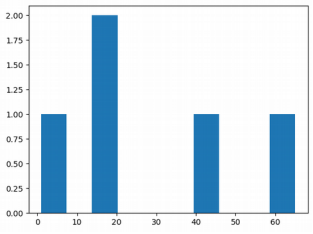
#Scatter Plot

```
from matplotlib import pyplot as plt
# x-axis values
x = [74, 88, 52, 72, 78]
# Y-axis values
y = [23, 22, 54, 69, 80]
# Function to plot scatter
plt.scatter(x, y)
# function to show the plot
plt.show()
```



#Histogram

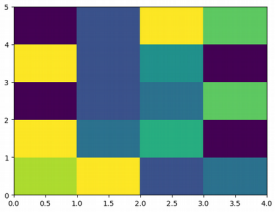
```
from matplotlib import pyplot as plt
# Y-axis values
y = [1,15,19,40,65]
# Function to plot histogram
plt.hist(y)
# Function to show the plot
plt.show()
```



#HeatMaps

```
from pandas import DataFrame
import matplotlib.pyplot as plt
```

```
data=[{8,9,3,4},{4,6,9,1},{7,4,1,3},{9,5,1,3},{7,1,3,9}]
Index= ['I1', 'I2','I3','I4','I5']
Cols = ['C1', 'C2', 'C3','C4']
df = DataFrame(data, index=Index, columns=Cols)
plt.pcolor(df)
plt.show()
```



8. LEARNING OUTCOMES:
Student are able to handle Data analysis with Visualization.

9. APPLICATION AREAS: Useful for Data Analysis

10. REMARKS: -

1. **EXPERIMENT NO: 7**

2. **TITLE:** Write a Python program that creates a mxn integer array and Prints its attributes using matplotlib

3. **LEARNING OBJECTIVES:**
Students are able to learn basics of matplotlib.

4. **AIM:** Is to apply python matplotlib to print array attributes.

5. **THEORY / HYPOTHESIS:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython,Qt, or GTK.

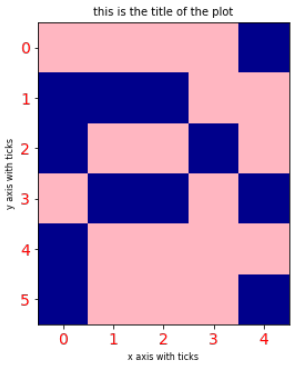
6. PROCEDURE / PROGRAMME / ACTIVITY:

```
# to start with, we will need matplotlib.pyplot
from matplotlib import pyplot
import random
data = [[random.randint(0,256) for x in range(0,5)], # row 1
    [random.randint(0,256) for x in range(0,5)], # row 2
    [random.randint(0,256) for x in range(0,5)], # row 3
    [random.randint(0,256) for x in range(0,5)], # row 4
    [random.randint(0,256) for x in range(0,5)], # row 5
    [random.randint(0,256) for x in range(0,5)]] # row 6
data
```

```
[[48, 169, 225, 30, 134],
 [95, 205, 60, 168, 253],
 [20, 79, 29, 112, 122],
 [118, 173, 19, 252, 238],
 [7, 84, 183, 158, 122],
 [245, 240, 226, 24, 182]]
```

```
from matplotlib import colors
pyplot.figure(figsize=(5,5))
pyplot.xlabel("x axis with ticks",size = 8)
pyplot.ylabel("y axis with ticks",size= 8)
pyplot.title("this is the title of the plot",size=10)
pyplot.xticks(size=14,color = "red")
pyplot.yticks(size=14,color = "red")
colormap = colors.ListedColormap(["lightpink","darkblue"])
pyplot.imshow(data,cmap=colormap)
```

7.OUTPUT:



8. **LEARNING OUTCOMES:**

- Able to apply basic of matplotlib.

9. **APPLICATION AREAS:**

- Data Visualization

10. **REMARKS: -**

1. **EXPERIMENT NO: 8**

2. **TITLE:**Write a Python program to demonstrate the generation of linear regression models**.**

3. **LEARNING OBJECTIVES:** Student able to learn linear regression models.

4. **AIM:** Is to apply python matplotlib to execute Linear Search Algorithms.

5. **THEORY / HYPOTHESIS:**

Simple linear regression is a regression model that estimates the relationship between one independent variable and one dependent variable using a straight line. Both variables should be quantitative.

6. PROCEDURE / PROGRAMME / ACTIVITY:

```
# Import packages and classes
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
%matplotlib inline
```

	Temperature	Revenue
0	24.566884	534.799028
1	26.005191	625.190122
2	27.790554	660.632289
3	20.595335	487.706960
4	11.503498	316.240194
..	...	...
495	22.274899	524.746364
496	32.893092	755.818399
497	12.588157	306.090719
498	22.362402	566.217304
499	28.957736	655.660388

```
# Read the IceCreamData.csv file
IceCream=pd.read_csv('IceCreamData.csv')
print(IceCream)
```

```
# Print first 5 data
IceCream.head()
```

```
# Print first 5 data
IceCream.head()
```

```
# Split 80% of the data to the training set while 20% of the data to test set

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

```
# Create a Linear Regression model and fit it
```

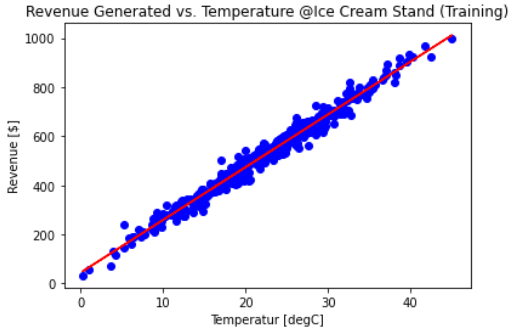
```
regressor =LinearRegression(fit_intercept=True)
regressor.fit(X_train,y_train)
```

```
# Getting Results
print('Linear Model Coeff (m) =' , regressor.coef_)
print('Linear Model Coeff (b) =' , regressor.intercept_)
```

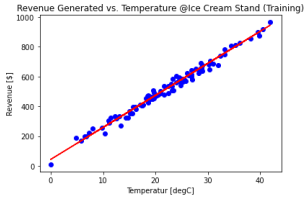
```
# Predicting the data
y_predict=regressor.predict(X_test)
print(y_predict)
```

```
[698.3385558  653.32331149 664.73027451 450.5192845  665.47469743
 441.36861407 584.06540609 623.82532723 667.48717467 468.72433832
 546.82733151 443.41191785 622.95162777 377.64639971 367.0607334
 945.67057977 893.79551974 694.45445099 546.05047608 420.58523672
 391.08500303 597.0141581  283.23582775 655.50055011 380.98796154
 412.31810124 371.05055651 510.23910289 479.70270426 456.68206658
 640.1157508  281.65224383 314.1894674  470.01363777 559.72453055
 539.75091165 307.72368191 508.65180339 571.43237276 732.25599161
 440.44010989 494.39422767 567.56536766 443.94181482 914.46632525
 603.19341879 541.83315574 199.94980451 694.04258508 351.09960842
 189.49123987 576.80689646 216.55393778 468.15141951 461.80905978
 448.43970076 494.89418532 801.3758273  331.24527072 540.42751209
 661.1953557  526.66690494 360.66507037 451.46656256 621.57729407
 254.83395119 290.1749214  525.5900171  656.68802152 663.1062835
 740.96627734 184.48524774 593.42653041 148.41501952 485.97744998
 611.03624804 664.50658946 473.51664017 785.34682628 422.11909846
 169.76879503 820.72328003 434.39990573 325.82688811 660.07484042
 586.46853445 415.89811147 651.95510136 865.22669518 265.88922879
 577.32110608  43.73357869 901.24435059 621.87554173 759.07316169
 465.78060018 758.74558525 711.30125473 394.45680968 559.53716333]
```

```
# Scatter plot on Training Data
plt.scatter(X_train,y_train,color='blue')
plt.plot(X_train,regressor.predict(X_train),color='red')
plt.ylabel('Revenue [$]')
plt.xlabel('Temperatur [degC]')
plt.title('Revenue Generated vs. Temperature @Ice Cream Stand (Training)')
```




```
# Scatter plot on Testing Data
plt.scatter(X_test,y_test,color='blue')
plt.plot(X_test,regressor.predict(X_test),color='red')
plt.ylabel('Revenue [$]')
plt.xlabel('Temperatur [degC]')
plt.title('Revenue Generated vs. Temperature @Ice Cream Stand (Training)')
```




```
# Prediction the revenve using Temperature Value directly
print('---------0---------')
Temp = -0
```

```
Revenue = regressor.predict([[Temp]])
print(Revenue)
print('--------35----------')
Temp = 35
Revenue = regressor.predict([[Temp]])
print(Revenue)
print('--------55----------')
Temp = 55
Revenue = regressor.predict([[Temp]])
print(Revenue)
```

7.OUTPUT:

```
---------0---------
[43.73357869]
--------35----------
[796.70225678]
--------55----------
[1226.97007282]
```

8. **LEARNING OUTCOMES:**

- Able to apply simple linear regression model.

9. **APPLICATION AREAS:** Predictive analysis of data.

10. **REMARKS: -**

1. **EXPERIMENT NO: 9**

2. **TITLE:** Write a Python program to demonstrate the generation of logistic regression models using Python.

3. **LEARNING OBJECTIVES:** Student able to learn logistic regression models.

4. **AIM:** Is to apply logistic regression model fro data prediction.

5. **THEORY / HYPOTHESIS:**

Simple logistic regression assumes that the relationship between the natural log of the odds ratio and the measurement variable is linear.

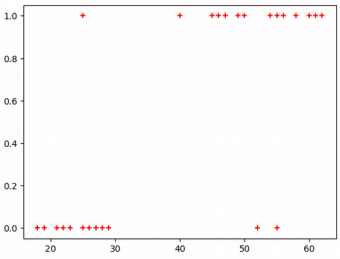
6. PROCEDURE / PROGRAMME / ACTIVITY:

```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
df = pd.read_csv("insurance_data.csv")
df.head()
plt.scatter(df.age,df.bought_insurance,marker='+',color='red')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df[['age']],df.bought_insurance,train_size=0.8)
X_test
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
        intercept_scaling=1, max_iter=100, multi_class='warn',
        n_jobs=None, penalty='l2', random_state=None, solver='warn',
        tol=0.0001, verbose=0, warm_start=False)
y_predicted = model.predict(X_test)
model.predict_proba(X_test)
```

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

```
model.score(X_test,y_test)
X_test
model.coef_
model.intercept_
```

7.OUTPUT:



	age
24	50
6	55
15	55
1	25
17	58
26	23

8. **LEARNING OUTCOMES:**
   - Able to apply logistic regression model for data prediction.

9. **APPLICATION AREAS:** Predictive analysis of data.
   -

10. **REMARKS: -**

1. **EXPERIMENT NO: 10**

2. **TITLE:** Write a Python program to demonstrate Timeseries analysis with Pandas.

3. **LEARNING OBJECTIVES:** Student able to learn basics of Pandas utilities.

4. **AIM:**Is to apply Pandas utilities to execute time series data analysis.

5. **THEORY / HYPOTHESIS:**

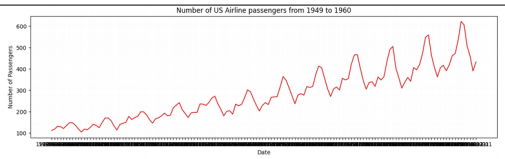
Complete Guide on Time Series Analysis in Python | Kaggle

6. PROCEDURE / PROGRAMME / ACTIVITY:

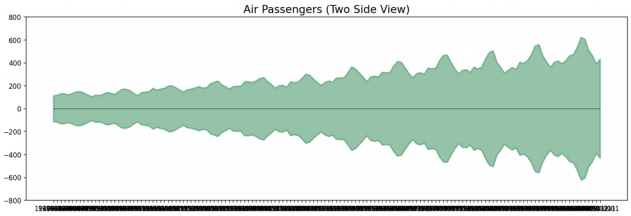
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib as mpl
import matplotlib.pyplot as plt  # data visualization
import seaborn as sns
url='https://raw.githubusercontent.com/selva86/datasets/master/AirPassengers.csv'
df = pd.read_csv(url)
df.head()
```

```
df.columns = ['Date','Number of Passengers']
df.head()
```

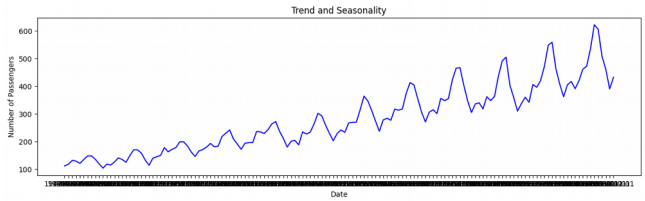
```
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Number of Passengers', dpi=100):
    plt.figure(figsize=(15,4), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
plot_df(df, x=df['Date'], y=df['Number of Passengers'], title='Number of US Airline passengers fro
m 1949 to 1960')
```



```
x = df['Date'].values
y1 = df['Number of Passengers'].values
# Plot
fig, ax = plt.subplots(1, 1, figsize=(16,5), dpi= 120)
plt.fill_between(x, y1=y1, y2=-y1, alpha=0.5, linewidth=2, color='seagreen')
plt.ylim(-800, 800)
plt.title('Air Passengers (Two Side View)', fontsize=16)
plt.hlines(y=0, xmin=np.min(df['Date']), xmax=np.max(df['Date']), linewidth=.5)
plt.show()
```




```
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Number of Passengers', dpi=100):
    plt.figure(figsize=(15,4), dpi=dpi)
    plt.plot(x, y, color='blue')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
plot_df(df, x=df['Date'], y=df['Number of Passengers'], title='Trend and Seasonality')
```



8. **LEARNING OUTCOMES:**Able to apply basic python programming constructs.

9. **APPLICATION AREAS: Time series Analysis**

10. **REMARKS: -**

1. **EXPERIMENT NO: 11**

2. **TITLE:** Write a Python program to demonstrate Data Visualization using Seaborn.

3. **LEARNING OBJECTIVES:** Student able to learn basics of Data Visualization using Seaborn.

4. **AIM:** To study basics of Data Visualization using Seaborn.

5. THEORY / hypothesis: Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

6. PROCEDURE / PROGRAMME / ACTIVITY:

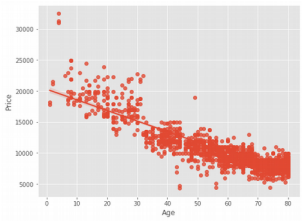
```
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt  # data visualization
url='https://raw.githubusercontent.com/gchoi/Dataset/master/ToyotaCorolla.csv'
cars_data = pd.read_csv(url)
cars_data.head()
```

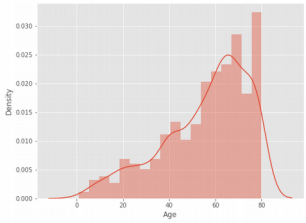
```
#Scatter Plot:
plt.style.use("ggplot")
plt.figure(figsize=(8,6))
sns.regplot(x = cars_data["Age"], y = cars_data["Price"])
plt.show()
```

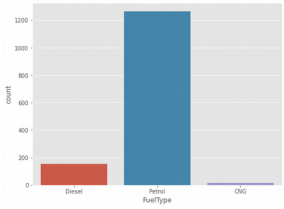
```
#Histogram
plt.figure(figsize=(8,6))
sns.distplot(cars_data['Age'])
plt.show()
```

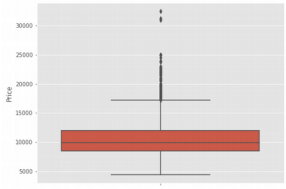
```
#Bar Plot:
plt.figure(figsize=(8,6))
sns.countplot(x="FuelType", data=cars_data)
plt.show()
```

```
#Box and Whiskers Plot:
plt.figure(figsize=(8,6))
sns.boxplot(y=cars_data["Price"])
plt.show()
```









8. **LEARNING OUTCOMES:**
   - Able to apply basic of data Visualization using Seaborn.

9. **APPLICATION AREAS:** Data Visualization

10. **REMARKS:** Student can use any data set.