DATA SCIENCE WITH PYTHON

STUDENT NAME : REDDI DURGA PRASAD

ID NUMBER : N191128

ClASS : CSE-03

DEPARTMENT OF COMPUTER SCIENCE
AND ENGINEERING RGUKT-NUZ-AP
E2-SEMESTER-II, AY-2022-23

TABLE OF CONTENTS

LAB-01

AIM : a) Python Basics: Your first program, Types Expressions and Variables

String Operations

Code:

```
print("hello world")
color="green"
print(type(color))
a=3
print(a,type(a))
b=-3.5
print(b,type(b))
c=2+3j
print(type(c))
d,e,f=2,3,-4
print(f)
print(e)
print(d)
h=j=k="RAJA"
print(h,j,k)
id1='How are you?'
print(id1[1:7])
x=0b11
print(type(x))
val=None
print(val)
#python string
id1="Mariya babu"
print(id1[1])
```

```
#negative indexing
print(id1[-3])
#id1[3]=q
#multiline strings
string="""mariya babu is the roommate of durgaprasad”
Hari is friend of mariya"""
print(string)
#python string operation
id2=" is the roommate of Durgaprasad"
print(id1+id2)
id3="babu"
id4="babu"
print(id3==id4)
id3="babu"
id4="babu1"
print(id3==id4)
#iterton
gr='welcome'
for letter in gr:
 print(letter)
gr='welcome'
for letter in gr:
 print(gr)
print(len(gr))
#membership
print("a" in gr)
print("a" not in gr)
print(gr.upper())
print(gr.lower())
print(gr.startswith("h"))
id='name'
name='DURGA PRASAD'
print(f'my {id} is {name}')
#escape sequence
ex="he said,\"what's is there?\""
print(ex)
```

output:

```
hello world
<class 'str'>
3 <class 'int'>
-3.5 <class 'float'>
<class 'complex'>
-4
3
2
RAJA RAJA RAJA
ow are
<class 'int'>
None
a
a
mariya babu is the roommate of durgaprasa
Hari is friend of mariya
Mariya babu is the roommate of Durgaprasad
True
False
w
e
l
c
o
m
e
welcome
welcome
welcome
welcome
welcome
welcome
welcome
7
False
True
WELCOME
welcome
False
my name is DURGA PRASAD
he said,"what's is there?"
```

LAB-02

AIM : Python Data Structures: Lists and Tuples Sets,and Dictionaries

CODE

```
""" list,tuple,dic,set"""
a=[2,'a','aba','aaa']
print(a)
num=(1,5,3)
print(num)
b={'a':3,'ba':456,'a':4}
print(b)
c={1,4,3,2,5,}
print(c)
d={2,'a','aba','aaa'}
print(d)
lan=["telugu","tamil","kannada"]
print(lan[2])
print(type(lan))
e={2,2,2,3}
print(e)
a=True
print(a)
b=False
print(b)
#list
a=[4,6,7]
print(a)
print(a[0])
print(a[-3])
print(a[0:2])
#append
a.append(2)
print(a)
#extend
b=[8,9,7]
a.extend(b)
print(a)
a[0]=0
print(a)
#del
del b[1]
print(b)
a.remove(0)
a.sort()
print(a)
a.reverse()
print(a)
a.pop(2)
print(a)
#checking
print(1 in a)
print(len(a))
#list comprehension
c=[]
for x in range(1,6):
 c.append(x*x)
print(c)
```

```
#tuple
print("tuples")
a=(3,4,5)
print(a)
b="hello",
print(type(b))
c=("hello")
print(type(c))
#tuple accessing
print(a[-1])
print(a[1])
print(a[0:2])
#tuple methods
d=(6,5,7,7,7,8,4,9,0)
print(d.count(7))
print(d.index(6))
#iteration
for x in d:
 print(x)
print(7 in d)
#sets
a={3,5,6,7,8,9,4,5,6}
b={10,20,30,40}
print("set")
print(a)
print(type(a))
a.add(10)
print(a)
```

```
#min
print(min(a))
#max
print(max(a))
#len
print(len(a))
#all
print(all(a))
#any
print(any(a))
#enumerate
print(enumerate(a))
#sum
print(sum(a))

#sorted
print(sorted(a))
#union
print(a|b)
print(a.union(b))
#intersection
print(a&b)
print(a.intersection(b))
#symmetric difference
print(a^b)
#equal
print(a==b)
#dictonary
```

```
dic={1:"a",2:"b",3:"c",4:"d",5:"e"}
print(dic)
print(type(dic))
#adding
dic[6]="f"
print(dic)
#changing
dic[3]="C"
print(dic)
#accessing
print(dic[3])
#remove
del dic[6]
print(dic)
# sorted
sorted(c)
print(dic)
#membership
print(1 in dic)
print(4 not in dic)
```

OUTPUT:

```
output:
[2, 'a', 'aba', 'aaa']
(1, 5, 3)
{'a': 4, 'ba': 456}
{1, 2, 3, 4, 5}
{2, 'aba', 'a', 'aaa'}
Kannada
<class 'list'>
{2, 3}
True
False
[4, 6, 7]
4
4
[4, 6]
[4, 6, 7, 2]
[4, 6, 7, 2, 8, 9, 7]
[0, 6, 7, 2, 8, 9, 7]
[8, 7]
[2, 6, 7, 7, 8, 9]
[9, 8, 7, 7, 6, 2]
[9, 8, 7, 6, 2]
False
5
[1, 4, 9, 16, 25]
tuples
(3, 4, 5)
<class 'tuple'>
<class 'str'>
5
4
(3, 4)
6
5
7
7
7
8
4
9
0
True
set
{3, 4, 5, 6, 7, 8, 9}
<class 'set'>
```

```
{3, 4, 5, 6, 7, 8, 9, 10}
3
10
8
True
True
<enumerate object at 0x000001803DC96E80>
52
[3, 4, 5, 6, 7, 8, 9, 10]
{3, 4, 5, 6, 7, 8, 9, 10, 40, 20, 30}
{3, 4, 5, 6, 7, 8, 9, 10, 40, 20, 30}
{10}
{10}
{3, 4, 5, 6, 7, 40, 8, 9, 20, 30}
False
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
<class 'dict'>
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f'}
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e', 6: 'f'}
C
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e'}
{1: 'a', 2: 'b', 3: 'C', 4: 'd', 5: 'e'}
True
False
```

LAB-03

AIM:Python Programming Fundamentals: Conditions and Branching Loops, Functions,Objects and Classes

CODE:

if-else

```
number=int(input("Enter a Number:"))
if number>10:
print('Number is greater than 10')
else:
print('Number is less than 10')
```

Output:

```
Enter a number:11
Number is greater than 10
```

If-elif-else

```
num=int(input('Enter a Number:'))
if num>0:
print('Positive Number')
elif num<0:
print('Negative Number')
else:
print('Positive Number')
print('This statement is always executed')
```

Output:

```
Enter a number:10
Positive Number
```

nested-if

```
num=int(input('Enter a Number:'))
if(num>=0):
if num==0:
print('Number is 0')
else:
print('Number is positive')
else:
print('Number is Negative')
```

output:

```
Enter a Number:15
Number is positive
```

short-hand-if

```
a=10;
b=20;
if a<b: print('This is if')
```

Output:

```
This is if
```

shorthand-if-else

```
a=30;
b=20;
print('This is if') if a<b else print('this is else')'''
```

Output:

```
This is else
```

for-loop

```
lang=['swift','c','python','c++']
for x in lang:
print(x)
range function
a=range(6)
for x in a:
print(x)
a=range(1,6)
for x in a:
print(x)
a=range(2,22,2)
for x in a:
print(x)
for i in range(1,1001):
for j in range (1,11):
print(i*j,end=" ")
print()
```

for loops with else

```
digits=[0,1,2]
for i in digits:
print(i)
else:
print("No items left.")
```

while loop

```
i=1
n=5
while i<=n:
```

```
print(i)
i=i+1
```

Python oops Concept

python inheritence

```
class Animal:
def speak(self):
print("Animal Speaking")
class Dog(Animal):
def bark(self):
print("dog barking")
class DogChild(Dog):
def eat(self):
print("Eating bread...")
d=DogChild()
d.speak()
d.bark()
d.eat()
```

Output:

```
Dog Barking
Animal Speaking
Eating Bread
```

Method overriding

```
'class Animal:
def speak(self):
print("Speaking")
class Dog(Animal):
def speak(self):
print("Not Speaking")
class Cat(Dog):
def speak(self):
print("Is this a cat")
d=Cat()
d.speak()
```

Output:

```
Speaking
Not Speaking
Is this a cat
```

Data Abstraction

```
class Employee:
__count=0;
def __init__(self):
```

```
Employee.__count=Employee.__count+1
def display(self):
print("The number of Employees",Employee.__count)
emp=Employee()
try:
print(emp.__count)
finally:
emp.display()
```

Output:

```
Number of Employees:3
```

Abstract Method

```
from abc import ABC, abstractmethod
class Car(ABC):
def mileage(self):
pass
class Tesla(Car):
def mileage(self):
print("The mileage is 30kmph")
class Suzuki(Car):
def mileage(self):
print("The mileage is 25kmph ")
class Duster(Car):
def mileage(self):
print("The mileage is 24kmph ")
class Renault(Car):
def mileage(self):
print("The mileage is 27kmph ")
# Driver code
t= Tesla ()
t.mileage()
r = Renault()
r.mileage()
s = Suzuki()
s.mileage()
d = Duster()
d.mileage()
```

Output

```
The mileage is 30kmph
The mileage is 25kmph
The mileage is 24kmph
The mileage is 27kmph
```

LAB-04

AIM: Working with Data in Python: Reading files with open, Writing files with open,Loading data with Pandas, Working with and Saving data with Pandas

CODE:

```
import pandas as pd
import numpy as np
print(pd.__version__)
b=[1,2,3,4]
c=pd.Series(b)
print(c)
b=['s','d']
c=pd.Series(b[-1])
print(c)
d=np.array(['a','b','c','d'])
s=pd.Series(d)
r=pd.DataFrame(d)
print(s)
print(r)
print(len(s))
s=pd.Series(d,index=[101,103,103,104])
j=pd.Series(d,index=["x","y","z","w"])
print(s)
print(j)
dataset={'icecreams':['vanila','strawberry','badam','pista'],
 'rating':[4.5,3.8,4.2,4.6]
}
ds=pd.DataFrame(dataset)
print(ds)
```

```
ds=pd.Series(dataset)

print(ds)
```

Output:

```
2.0.1
0 1
1 2
2 3
3 4
dtype: int64
0 d
dtype: object
0 a
1 b
2 c
3 d
dtype: object
 0
0 a
1 b
2 c
3 d
4
101 a
103 b
103 c
104 d
dtype: object
x a
y b
z c
w d
dtype: object
 icecreams rating
0 vanila 4.5
1 strawberry 3.8
2 badam 4.2
3 pista 4.6
icecreams [vanila, strawberry, badam, pista]
rating [4.5, 3.8, 4.2, 4.6]
dtype: object
```

Attribute of series

```
import pandas as pd
import numpy as np
ds=np.array(['a','b','c','d'])
d=pd.Series(ds)
print(d)
```

N190302 19

DASARI GANGADHAR Data Science with Python Lab

```
d=pd.Series(ds ,index=[101,102,103,"e"])
print(d)
print(d[103])
ds1={'d1':100,'d2':200,'d3':300}
d=pd.Series(ds1)
print(d)
j=pd.Series(ds1,index=['d1','d2'])
print(j)
print(j.name)
print(j.values)
print(j.size)
print(d.shape)
print(d.ndim)
print(d.nbytes)
print(d.memory_usage)
print(j.empty)
j.name='raj'
print(j.name)
```

output:

```
0 a
1 b
2 c
3 d
dtype: object

101 a
102 b
103 c
e d
dtype: object

c
d1 100
d2 200
d3 300
dtype: int64

d1 100
d2 200
dtype: int64

None
[100 200]
2
(3,)
1
24

<bound method Series.memory_usage of d1 100
d2 200

d3 300
dtype: int64>
False
Raj
```

Multiplication of series :

```
import pandas as pd
import numpy as np
ds1=np.array([1,1,2,3,4])
d1=pd.Series(ds1)
ds2=np.array([2,2,3,4,5])
d2=pd.Series(ds2)
a=d1.add(d2)
print(a)
b=d1.sub(d2)
print(b)
c=d1.mul(d2)
print(c)
d=d1.multiply(4)
print(d)
e=d1.div(d2)
print(e)
f=d2.mod(d1)
print(f)
g=d2.pow(3)
print(g)
h=d2.le(d1)
print(h)
i=d2.gt(d1)
print(i)
j=d2.equals(d1)
print(j)
```

output:

```
0 3
1 3
2 5
3 7
4 9

dtype: int32
0 -1
1 -1
2 -1
3 -1
4 -1

dtype: int32
0 2
1 2
2 6
3 12
4 20

dtype: int32
0 4
1 4
2 8
3 12
4 16

dtype: int32
0 0.500000
1 0.500000
2 0.666667
3 0.750000
4 0.800000

dtype: float64
0 0
1 0
2 1
3 1
4 1
dtype: int32
0 8
1 8
2 27
3 64
```

```
4 125
dtype: int32
0 False
1 False
2 False
3 False
4 False

dtype: bool
0 True
1 True
2 True
3 True
4 True

dtype: bool
False
```

LAB - 05

Working with Numpy Arrays:Numpy 1d Arrays,Numpy 2D Arrays

Code:

```
import numpy as np
#arrange
arr=np.arange(20)
print(arr)
#shape
arr.shape
print("Shape of array:",arr)
print(arr[4])
#asisgning a value
arr[7]=777
print(arr)
#reshape the existing array
arr=np.arange(20).reshape(4,5)
print("Rearranging the array:",arr)
print(arr.shape)
print(arr[1][2])
array=np.arange(27).reshape(3,3,3)
print(array)
#zero function
print(np.zeros((2,4)))
#ones function
print(np.ones((2,4)))
#empty function
print(np.empty((2,2)))
```

```
#full function
print(np.full((4,3),7))
#eye function
print(np.eye(3,3))
#linespace
print(np.linspace(0, 100, num=5))
#conversion from list to array
list=[4,5,6]
print(list)
array=np.array(list)
print(array)
print(type(array))
#random funcion
print(np.random.random((2,2)))
print(np.shape(array))
print(np.size(array))
print(np.dtype(float))
array1=np.array([1,2,3,4,5])
print(array1[1:5])
print(array1[:])
print(array1[3:])#copying the array
myarray=np.copy(array1)
print(myarray)
myarray[2]=8
print(myarray,array1)
#view function
arrayv=array1.view()
print(arrayv)
```

```
arrayv[2]=9
print(arrayv,array1)
yammu=np.array(['A','B'])
ary=np.array([11,22,33,44])
print(ary)
print(np.delete(ary,2))
#atack function
a=np.array([1,2,3,4])
b=np.array([5,6,7,8])
c=np.stack((a,b),axis=1)
print(c)
#concatenate
x=np.array([[1,2],[3,4]])
y=np.array([[12,30]])
r=np.array([[33,44]])
z=np.concatenate((x,y),axis=0)
print(z)
print(np.vstack((x,y)))
print(np.hstack((y,r)))
print(np.dstack((y,r)))
split=np.array([11,22,33,44,55,66])
newarr=np.array_split(split,3)
print(newarr)
#where function
t=np.arange(12)
s=np.where(a<6,a,5*a)
print(s)
fun=np.array([1,11,2,22,3,33])
```

```
print(np.max(fun))
print(np.min(fun))
print(np.mean(fun))
print(np.median(fun))
print(np.var(fun))
print(np.std(fun))
```

Output:

```
[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19] Shape of array: [ 0 1
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19] 4
[ 0 1 2 3 4 5 6 777 8 9 10 11 12 13 14 15 16 17 18 19]
Rearranging the array: [[ 0 1 2 3 4]
[ 5 6 7 8 9]
[10 11 12 13 14]
[15 16 17 18 19]]
(4, 5)
7
[[[ 0 1 2]
[ 3 4 5]
[ 6 7 8]]
[[ 9 10 11]
[12 13 14]
[15 16 17]]
[[18 19 20]
[21 22 23]
[24 25 26]]]
[[0. 0. 0. 0.]
[0. 0. 0. 0.]]
[[1. 1. 1. 1.]
[1. 1. 1. 1.]]
[[2.12199579e-314 4.67296746e-307]
[1.11658836e-320 1.04614393e-311]]
[[7 7 7]
[7 7 7]
[7 7 7]
[7 7 7]]
[[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]]
```

```
[ 0. 25. 50. 75. 100.]
[4, 5, 6]
[4 5 6]
<class 'numpy.ndarray'>
[[0.04181302 0.70924674]
[0.96165792 0.17146781]]
(3,)
3
float64
[2 3 4 5]
[1 2 3 4 5]
[4 5]
[1 2 3 4 5]
[1 2 8 4 5] [1 2 3 4 5]
[1 2 3 4 5]
[1 2 9 4 5] [1 2 9 4 5]
[11 22 33 44]
[11 22 44]
[[1 5]
[2 6]
[3 7]
[4 8]]
[[ 1 2]
[ 3 4]
[12 30]]
[[ 1 2]
[ 3 4]
[12 30]]
[[12 30 33 44]]
[[[12 33]
[30 44]]]
[array([11, 22]), array([33, 44]), array([55, 66])]
[1 2 3 4]
33
1
12.0
7.0
140.66666666666666
11.86029791643813
```

LAB-06

Aim: Importing Datasets: Learning Objectives, Understanding the Domain, Understandingthe Dataset, Python package for data science, Importing and Exporting Data in Python, BasicInsights from Datasets
Cleaning and Preparing the Data: Identify and Handle Missing Values, Data Formatting, Data Normalization Sets, Binning, Indicator variables

Code:

Importing datasets and preparing the data

```
import pandas as pd

df=pd.read_csv(r'C:\Users\DURGAPRASAD\Desktop\DSP\data1.csv')

d=pd.DataFrame(df)

print(d)

d=df.loc[4]

print(d)

d=df.loc[2:3]

print(d)

print(df.loc[1,"Name"])

print(df.loc[0:4,["Name","marks"]])

print(df.loc[4:8,"Name":"marks"])

"""ILOC"""

print(df.iloc[3])

print(df.iloc[3:8])

print(df.iloc[3:8,1])

print(df.iloc[5:9,1:3])

print(df.iloc[[2,4,6,7]]
```

OUTPUT

```
Unnamed: 0 Name id marks
0 1 Dasari R1254 14
1 2 Gangadhar R1255 14
2 3 Sree R1256 13
3 4 Raj R1257 12
4 5 Ram R1258 15
```

```
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
8 9 Siri NaN 12

9 10 Lava R1263 10
Unnamed: 0 5
Name Ram
id R1258
marks 15
Name: 4, dtype: object
 Unnamed: 0 Name id marks
2 3 Sree R1256 13
3 4 Raj R1257 12
Gangadhar
 Name marks

0 Dasari 14
1 Gangadhar 14
2 Sree 13
3 Raj 12
4 Ram 15
 Name id marks
4 Ram R1258 15
5 Roja R1259 13
6 Rahul R1260 14
7 Ramya R1261 11
8 Siri NaN 12
Unnamed: 0 4
Name Raj
id R1257
marks 12
Name: 3, dtype: object

 Unnamed: 0 Name id marks
3 4 Raj R1257 12
4 5 Ram R1258 15
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
3 Raj
4 Ram
5 Roja
6 Rahul
```

```
7 Ramya

Name: Name, dtype: object
 Name id
5 Roja R1259
6 Rahul R1260
7 Ramya R1261
8 Siri NaN

 Unnamed: 0 Name id marks
2 3 Sree R1256 1
```

Data cleaning

dropna()

```
import pandas as pd

import numpy as np

df=pd.read_csv(r'C:\Users\DASARI GANGADHAR\Desktop\DSP\data1.csv')

print(df)

d=df.dropna()

print(d)

print(df)

print(df.loc[:,["marks","Name"]].dropna())

d=df.dropna(inplace=True)

print(d)

print(df)
```

output:

```
Unnamed: 0 Name id marks
0 1 Dasari R1254 14
1 2 Gangadhar R1255 14
2 3 Sree R1256 13
3 4 Raj R1257 12
4 5 Ram R1258 15
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
8 9 Siri NaN 12
```

```
9 10 Lava R1263 10
 Unnamed: 0 Name id marks

0 1 Dasari R1254 14
1 2 Gangadhar R1255 14
2 3 Sree R1256 13
3 4 Raj R1257 12
4 5 Ram R1258 15
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
9 10 Lava R1263 10
 Unnamed: 0 Name id marks
0 1 Dasari R1254 14
1 2 Gangadhar R1255 14

2 3 Sree R1256 13
3 4 Raj R1257 12
4 5 Ram R1258 15
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
8 9 Siri NaN 12
9 10 Lava R1263 10
 marks Name

0 14 Dasari
1 14 Gangadhar
2 13 Sree
3 12 Raj
4 15 Ram
5 13 Roja
6 14 Rahul
7 11 Ramya
8 12 Siri
9 10 Lava
None
 Unnamed: 0 Name id marks
0 1 Dasari R1254 14
1 2 Gangadhar R1255 14
2 3 Sree R1256 13
3 4 Raj R1257 12
4 5 Ram R1258 15
5 6 Roja R1259 13
6 7 Rahul R1260 14
7 8 Ramya R1261 11
```

```
9 10 Lava R1263 10
```

fillna()

```
import pandas as pd

df=pd.read_excel(r"C:\Users\DURGA PRASAD\Desktop\DSP\data2.xlsx")

print(df)

d=df.fillna("missing")

print(d)

df.fillna("missing",inplace=True)

print(df)
```

output:

```
name gender age weight

0 John M 48.0 128.6

1 Peter NaN 58.0 158.3

2 Liz F NaN 115.5

3 Joe M 28.0 170.1

 name gender age weight

0 John M 48.0 128.6

1 Peter missing 58.0 158.3

2 Liz F missing 115.5

3 Joe M 28.0 170.1

 name gender age weight

0 John M 48.0 128.6

1 Peter missing 58.0 158.3

2 Liz F missing 115.5

3 Joe M 28.0 170.1
```

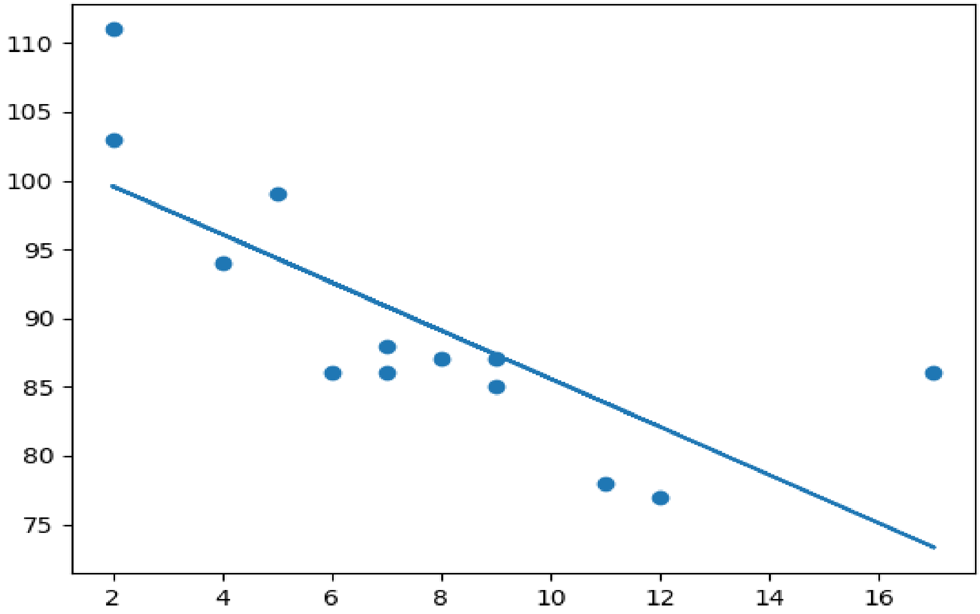
LAB-07

Aim: Model Development: Simple and Multiple Linear Regression, Model EvaluationUsingVisualization, Polynomial Regression and Pipelines, R-squared and MSE for In-Sample Evaluation, Prediction and Decision Making

SIMPLE LINEAR REGRESSION:

```
import matplotlib.pyplot as plt
from scipy import stats


x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]


slope, intercept, r, p, std_err = stats.linregress(x, y)


def myfunc(x):
  return slope * x + intercept


mymodel = list(map(myfunc, x))
plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

OUTPUT



Multiple Regression

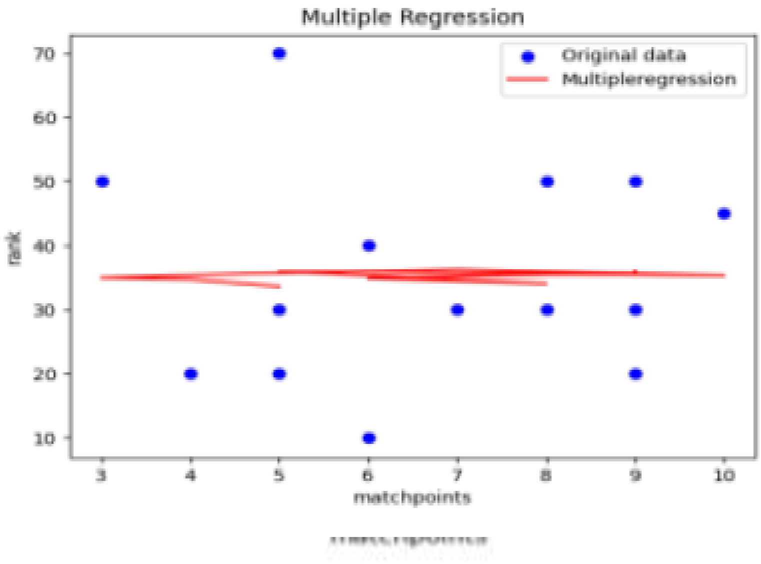
```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

#from sklearn.linear_model import LinearRegression as lr

# Load the dataset

data = pd.read_csv('pra.csv')

# Extract the independent variables (features)

X = data[['$cost','rating']].values

# Extract the dependent variable

y = data['discount'].values

# Create and fit the multiple regression model

from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

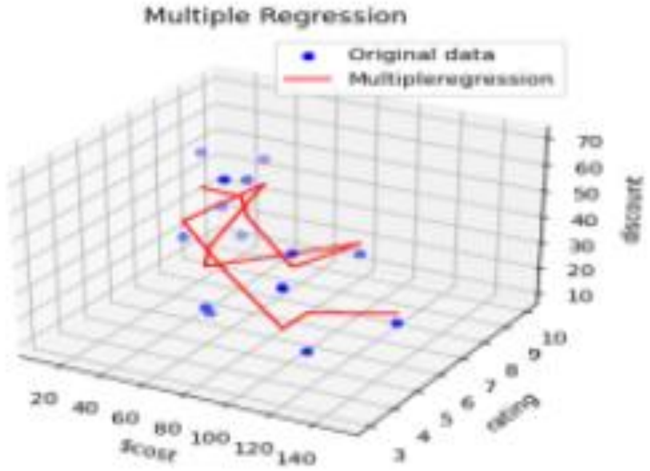
```
model.fit(X, y)
# Generate predicted values
predicted_y = model.predict(X)
print("Coefficients: ", model.coef_)
print("Intercept: ", model.intercept_)
print("Predicted values: ", predicted_y)
# Plot the original data points and the predicted values
plt.scatter( X[:, 1], y, c='blue', label='Original data') plt.plot(
X[:, 1], predicted_y, c='red', label='Multipleregression')
plt.xlabel('matchpoints')
plt.ylabel('rank')
plt.title('Multiple Regression')
plt.legend()
plt.show()
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X[:, 0], X[:, 1], y, c='blue', label='Original data')
ax.plot(X[:, 0], X[:, 1], predicted_y, c='red', label='Multipleregression')
ax.set_xlabel('$cost')
ax.set_ylabel('rating')
ax.set_zlabel('discount')
ax.set_title('Multiple Regression')
ax.legend()
plt.show()
```

Output:

```
Coefficients: [-0.02574741 -0.28450588]
Intercept: 38.95936897090988
Predicted values: [33.67472874 34.73165679 35.01616266 35.73452119 36.32414269
35.34188804
35.96496343 35.99199525 34.10858139 34.9350672 35.5246887 35.62639391
35.88386797 36.14134203]
```

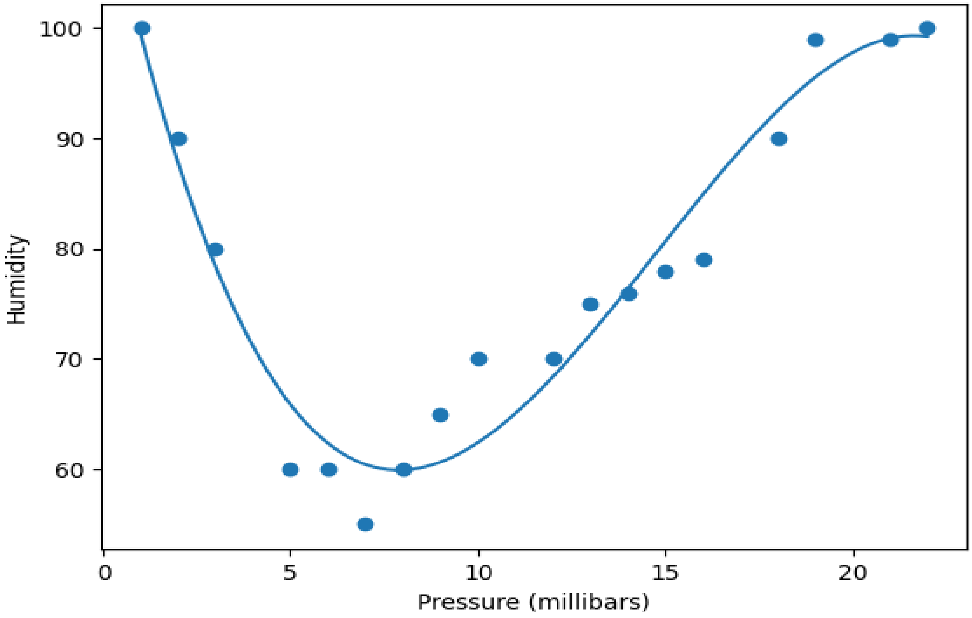
GRAPHS





Polynomial regression

```
# Import Pandas Library, used for data manipulation
# Import matplotlib, used to plot our data
# Import numpy for linear algebra operations
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
# Import our WeatherDataP.csv and store it in the variable rweather_data_p
weather_data_p = pd.read_csv("WeatherDataP.csv")
 # Display the data in the notebook
weather_data_p
# Set our input x to Pressure, use [[]] to convert to 2D array suitable for model input
X = weather_data_p[["Pressure (millibars)"]]
y = weather_data_p.Humidity
# Produce a scatter graph of Humidity against Pressure
plt.scatter(X, y, c = "black")
plt.plot(X,y)
plt.xlabel("Pressure (millibars)")
plt.ylabel("Humidity")
plt.show()
```



LAB-08

Aim: Model Evaluation: Model Evaluation, Over-fitting, Under-fitting and Model Selection,Ridge Regression, Grid Search, Model Refinement

Code: Ridge regression

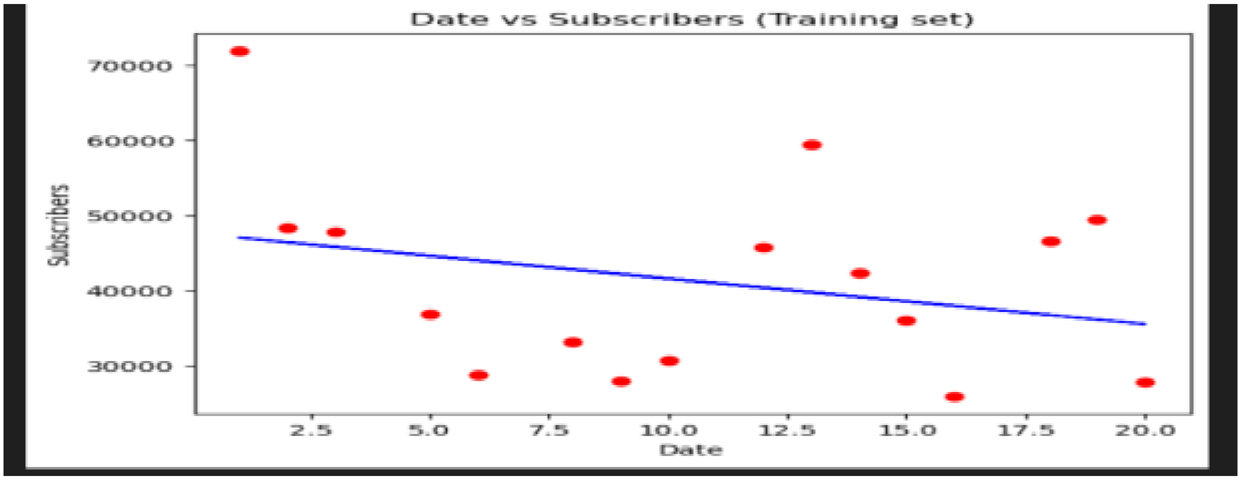
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn import metrics
import numpy as np
df=pd.read_csv("PewDiePie.csv")
#dividing the variables into dependent and independent
X=pd.DataFrame(df['Date'])
y=pd.DataFrame(df['Subscribers'])
#Split the data into train and test sets
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
#train the algorithm
ridge=Ridge(alpha=1.0)
ridge.fit(X_train,y_train)
#retriving the intercept
print(ridge.intercept_)
#retriving the slope
print(ridge.coef_)
#predecting the test results
y_pred = ridge.predict(X_test)
#evaluting the algorithm
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test,y_pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test,y_pred))
print('Root Mean Squared Error:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

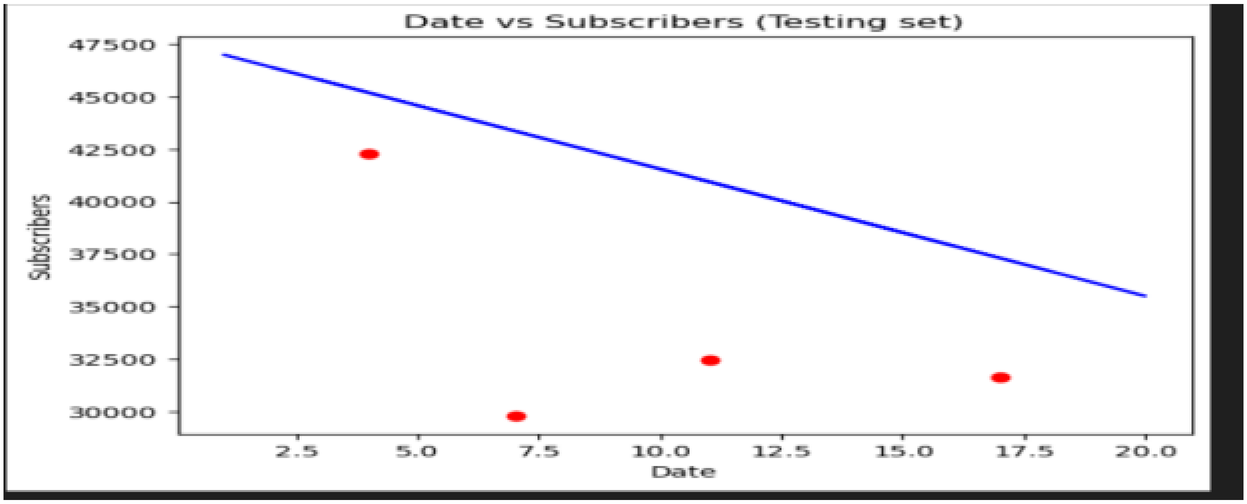
```
plt.scatter(X_train, y_train, color='red') # plotting the observation line
plt.plot(X_train, ridge.predict(X_train), color='blue') # plotting the regression line
plt.title("Date vs Subscribers (Training set)") # stating the title of the graph
plt.xlabel("Date") # adding the name of x-axis
plt.ylabel("Subscribers") # adding the name of y-axis
plt.show() # specifies end of graph
#plot for the test set
plt.scatter(X_test, y_test, color='red')
plt.plot(X_train, ridge.predict(X_train), color='blue') # plotting the regression line
plt.title("Date vs Subscribers (Testing set)")
plt.xlabel("Date")
plt.ylabel("Subscribers")
plt.show()
```

output:

```
[47611.65464541]
[[-605.65189665]]
Mean Absolute Error: 7670.798653106103
Mean Squared Error: 74374253.37775256
Root Mean Squared Error: 8624.0508682261
```

GRAPH





Overfitting and underfitting Problem:

Import numpy as np

Import matplotlib.pypplot as plt

from sklearn.pipeline import Pipeline from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear_model import LinearRegression #this allows us to create a random dataset

X = np.sort(np.random.rand(100)) #Lets create a true function

true_f = lambda X: np.cos(3.5 * np.pi * X)

y = true_f(X) + np.random.randn(100) * 0.1

degrees = [1,15]

plt.figure(figsize=(15, 10))

for i in range(len(degrees)):

ax = plt.subplot(1, len(degrees), i+1)

plt.setp(ax, xticks=(), yticks=()) polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False) linear_regression = LinearRegression()

pipeline=Pipeline([("polynomial_features",polynomial_features),("linear_regression", linear_regression)])

pipeline.fit(X[:, np.newaxis], y) #Testing

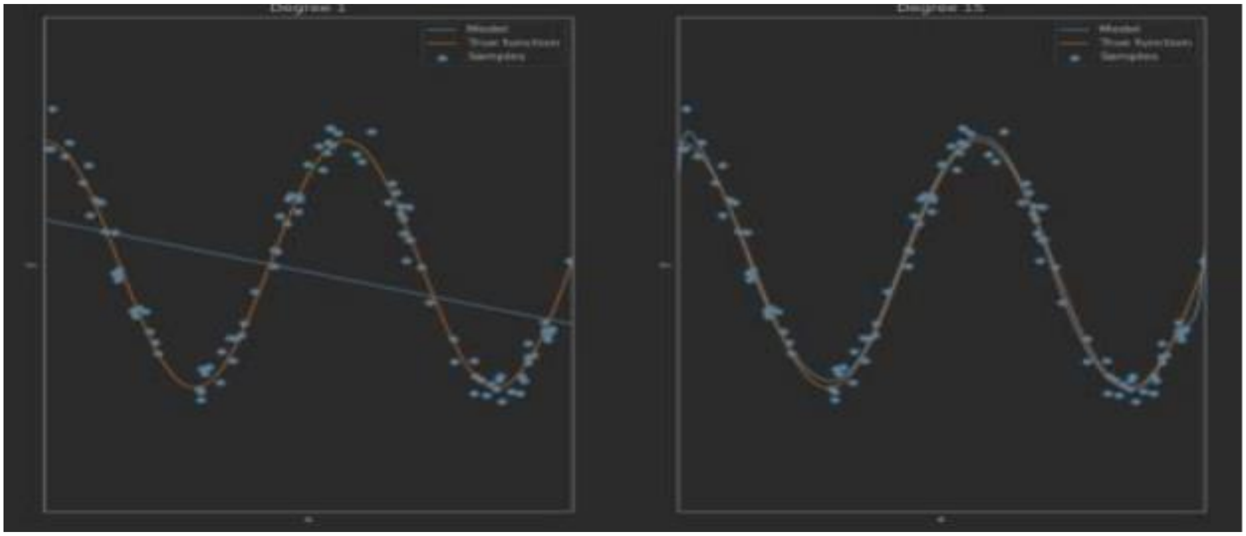
X_test = np.linspace(0, 1, 100)

hat = pipeline.predict(X_test[:, np.newaxis])

plt.plot(X_test, hat,label="Model")

```
plt.plot(X_test, true_f(X_test), label="True function") plt.scatter(X, y, label="Samples")
plt.xlabel("x") plt.ylabel("y")
plt.xlim((0, 1))
plt.ylim((-2, 2))
plt.legend(loc="best")
plt.title("Degree %d" % degrees[i])
plt.show()
```

OUTPUT:



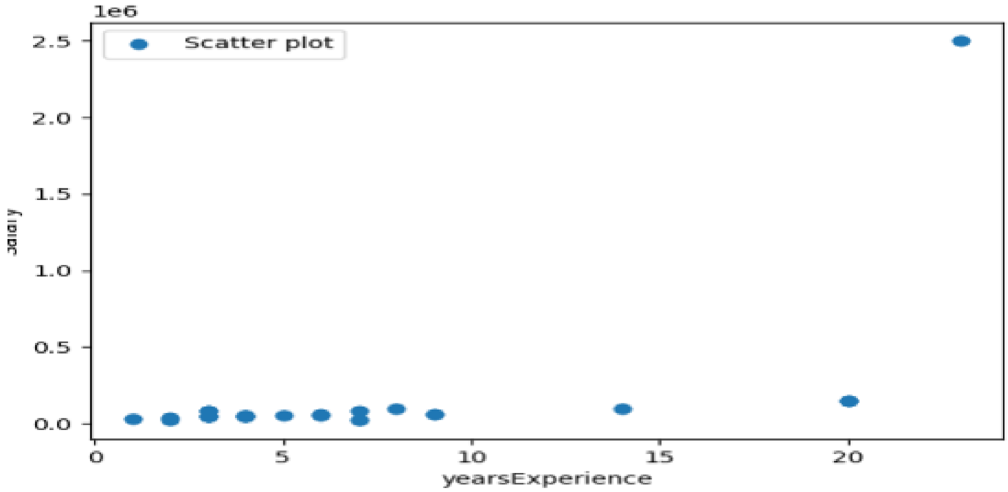
LAB-09

Aim:Introduction to Visualization Tools: Introduction to Data Visualization,Introduction to Matplotlib

CODE:

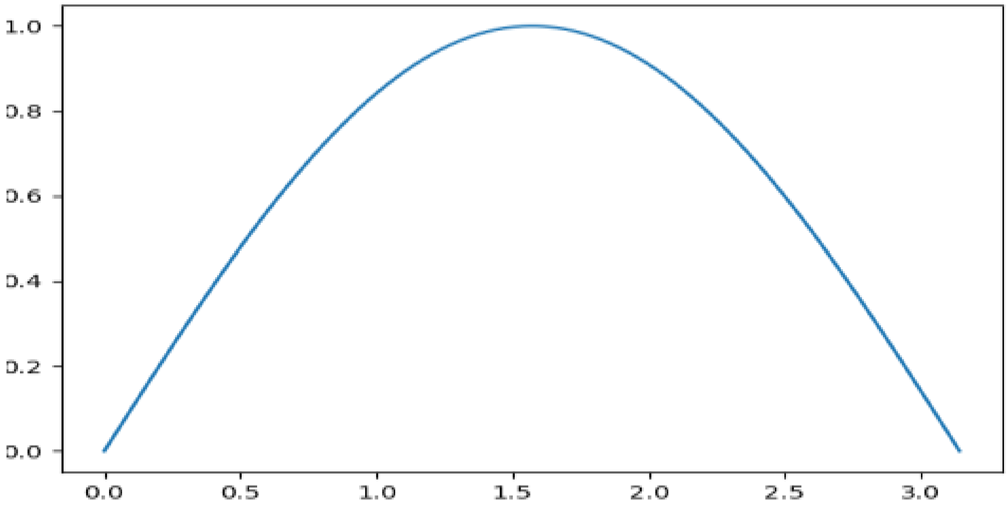
```
#scatterplot

d1=df.head(50)

x_scatter=d1['yearsExperience']

y_scatter=d1['salary']

plt.xlabel('yearsExperience')

plt.ylabel('Salary')

plt.scatter(x_scatter,y_scatter,label="Scatter plot")

plt.legend()

plt.show()
```

output:



```
import matplotlib.pyplot as plt

import numpy as np

x=np.linspace(0,1*np.pi,10000)

y=np.sin(x)

fig, ax=plt.subplots()

ax.plot(x,y)

plt.show()
```

GRAPH



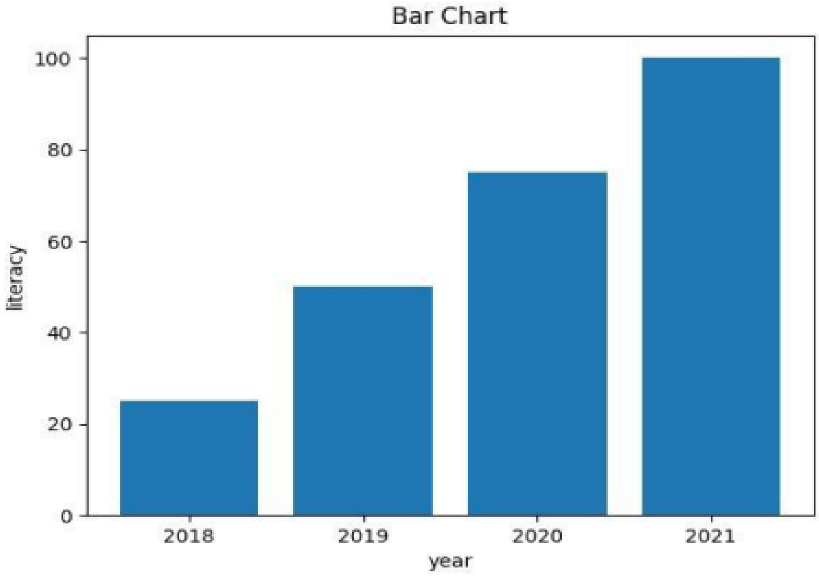
LAB-10

AIM:Basic Visualization Tools: Area Plots,Histograms,Bar Charts

BAR PLOT:-

```
import matplotlib.pyplot as plt
# Sample data
year= ['2018', '2019', '2020', '2021']
literacy = [25, 50, 75, 100]
# Create a bar chart
plt.bar(year, literacy)
# Add labels and title
plt.xlabel('year')
plt.ylabel('literacy')
plt.title('Bar Chart')
# Display the chart
plt.show()
```

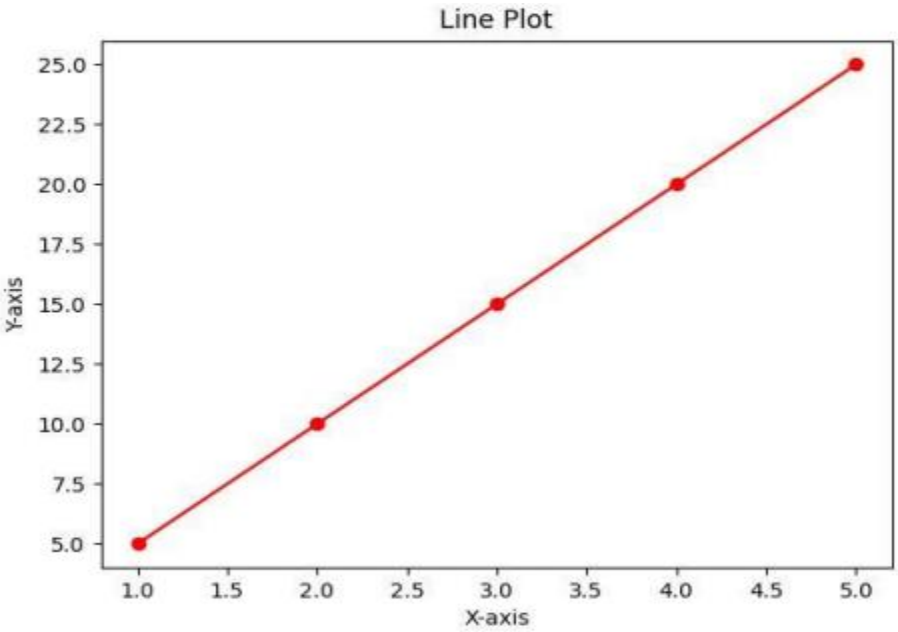
Output:-



LINE PLOT:-

```
#lineplot
x = [1, 2, 3, 4, 5]
y = [5, 10, 15, 20, 25]
# Create a figure and axis
fig, ax = plt.subplots()
# Plot the data
ax.plot(x, y, marker='o', linestyle='-', color='r')
# Customize the plot
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_title('Line Plot')
# Display the plot
plt.show()
```

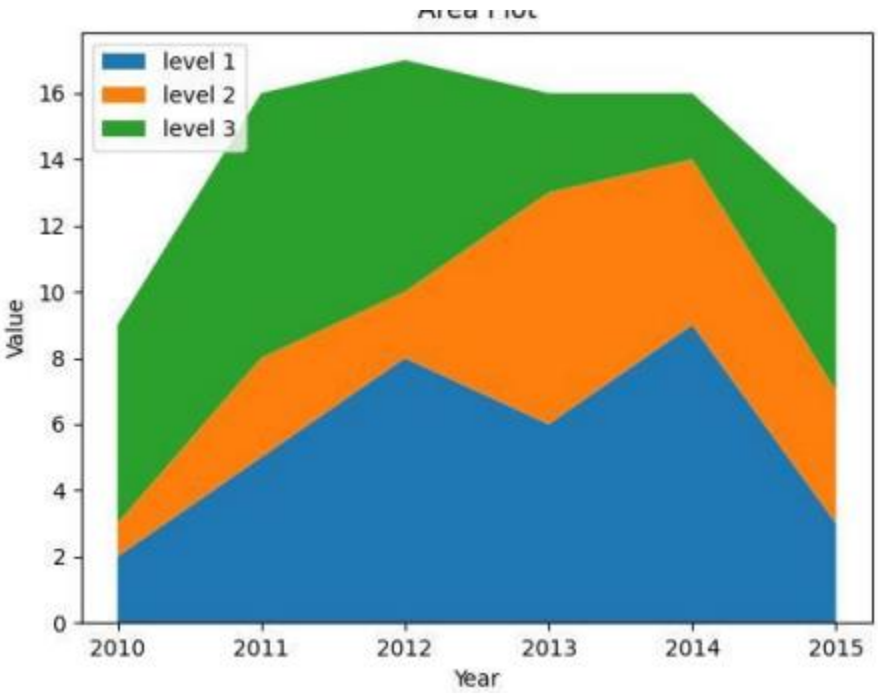
OUTPUT:-



AREA PLOT:-

```
#area plot
# Sample data
years = [2010, 2011, 2012, 2013, 2014, 2015]
level1 = [2, 5, 8, 6, 9, 3]
level2 = [1, 3, 2, 7, 5, 4]
level3 = [6, 8, 7, 3, 2, 5]
# Plotting
plt.stackplot(years, level1, level2, level3, labels=['level 1', 'level 2', 'level 3'])
plt.legend(loc='upper left')
plt.xlabel('Year')
plt.ylabel('Value')
plt.title('Area Plot')
plt.show()
```

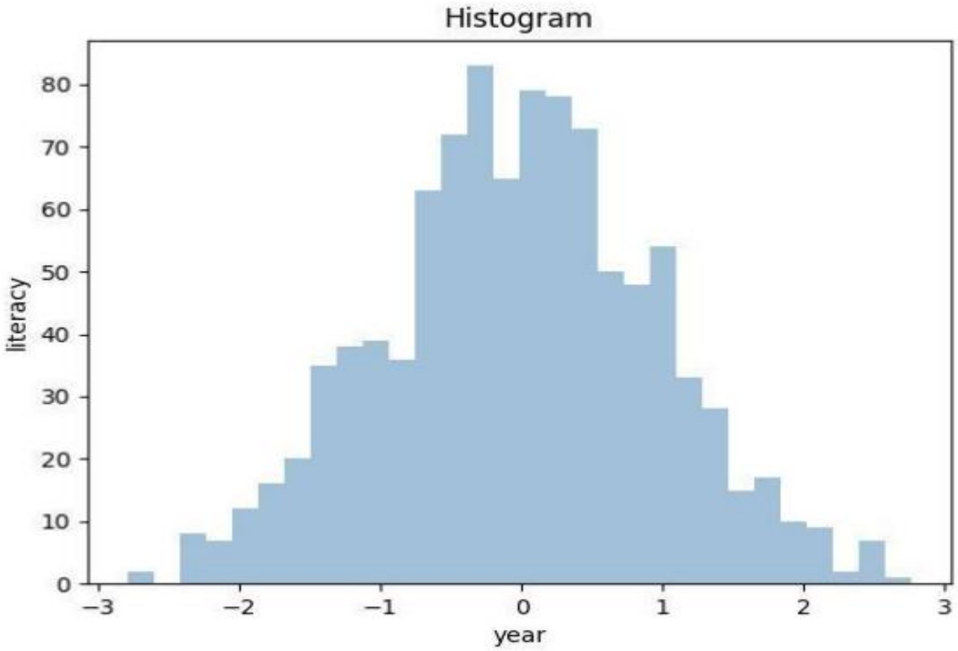
OUTPUT:-



#histogram

```
import numpy as np
data = np.random.randn(1000)
plt.hist(data, bins=30, alpha=0.5, color='steelblue')
plt.xlabel('year')
plt.ylabel('literacy')
plt.title('Histogram')
plt.show()
#pie chart
x=[2010,2012,2013,2014,2015]
y=[20,30,40,50,60]
plt.pie(y,labels=x,autopct='%1.1f%%')
plt.axis("equal")
plt.show()
```

OUTPUT:-



LAB-11

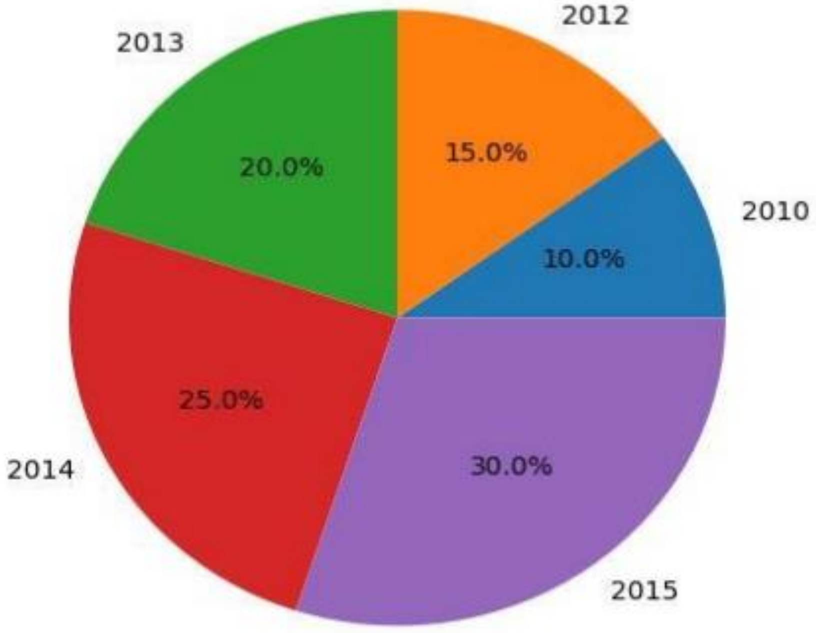
Aim: Specialized visualization tools pie charts ,boxplot

CODE:

#pie chart

```
x=[2010,2012,2013,2014,2015]

y=[20,30,40,50,60]

plt.pie(y,labels=x,autopct='%1.1f%%')

plt.axis("equal")

plt.show()
```

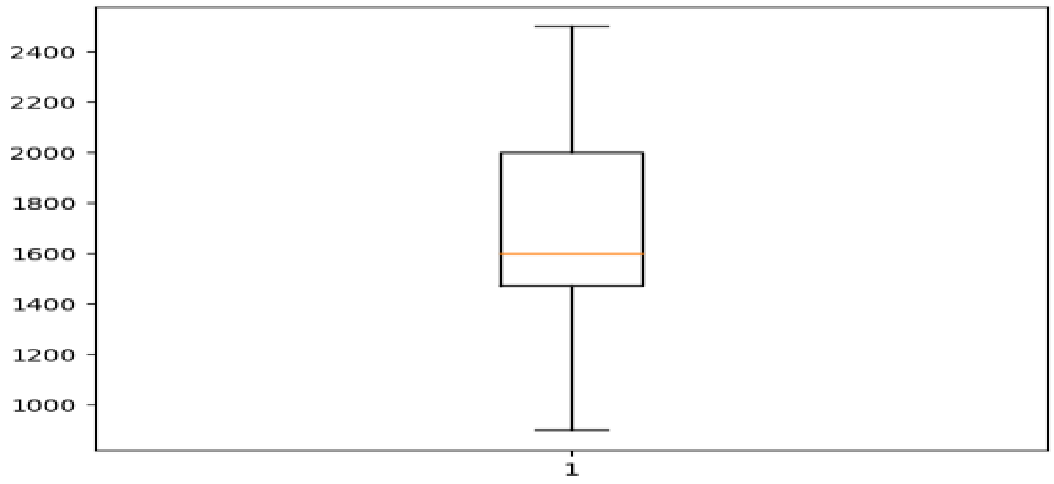
Output:



#box plot

```
import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

data = pd.read_csv("data.csv")

data.head()

x = data.Volume

plt.boxplot(x)

plt.show()
```

OUTPUT:



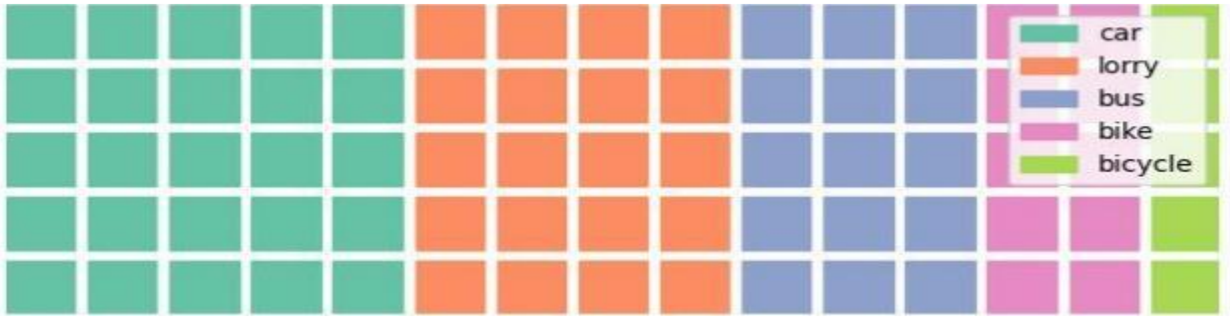
LAB-12

Aim: Advanced Visualization Tools: Waffle Charts, Word Clouds,Seaborn and Regression Plots

Waffle charts:

```
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle
data={'vehicles':['car', 'lorry','bus','bike','bicycle'],
'stock':[25,20,15,10,5]}
df=pd.DataFrame(data)
fig=plt.figure(FigureClass=Waffle,rows=5,values=df.stock,
labels=list(df.vehicles))
plt.show()
```

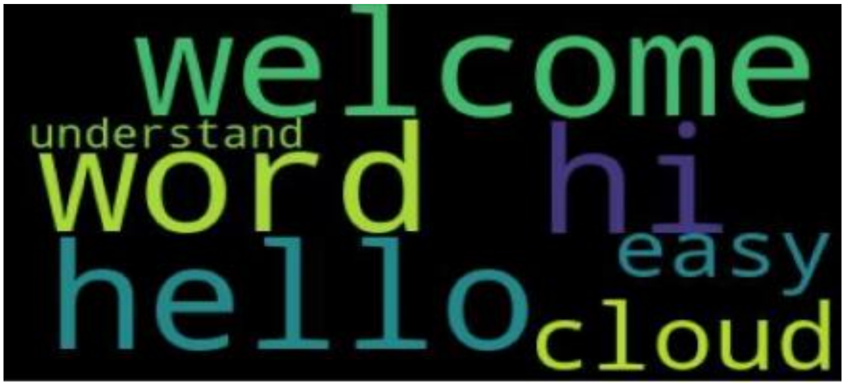
OUTPUT



Word clouds

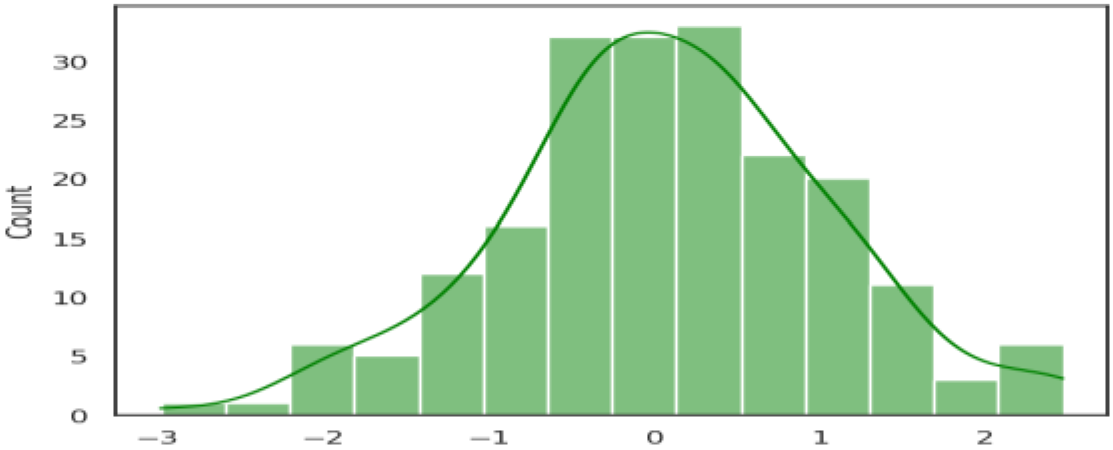
```
#word cloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
text="hi hello welcome to word cloud it is very easy to understand"
wc=WordCloud().generate(text)
plt.imshow(wc)
plt.axis("off")
plt.show()
```

Output:



#SEABORN

```
import numpy as np
import seaborn as sns
sns.set(style="white")
# Generate a random univariate dataset
rs = np.random.RandomState(10)
d = rs.normal(size=200)
# Plot a simple histogram and kde
sns.histplot(d, kde=True, color="green"
```



```
import folium
# Make an empty map
m = folium.Map(location=[20,0], tiles="OpenStreetMap", zoom_start=2)
# Import the pandas library
import pandas as pd
# Make a data frame with dots to show on the map
data = pd.DataFrame({
'lon':[-58, 20.5937, 145, 30.32, -4.03, -73.57, 36.82, -38.5],
'name':['Buenos Aires', 'norway', 'melbourne', 'St Petersbourg', 'Abidjan',
'Montreal', 'Nairobi', 'Salvador'],
'value':[10, 12, 40, 70, 23, 43, 100, 43]
}, dtype=str)
# add marker one by one on the map
for i in range(0,len(data)):
 folium.Marker(
 location=[data.iloc[i]['lat'], data.iloc[i]['lon']],
 popup=data.iloc[i]['name'],
 ).add_to(m)
```