Exercise 1

a) Installation and Environment setup of python.

b) Write a program to demonstrate the use of basic Data Types

c) Write a program to demonstrate the Operators and Expressions

d) Write a program to demonstrate the Functions and parameter passing Techniques.

1.a) Installation and environment setup of python

python installation in windows

Here are the steps to install Python on Windows machine.

1.Open a Web browser and go to https://www.python.org/downloads/.

2.Follow the link for the Windows installer python-3.7.2.msi file where 3.7.2 is the version you need to install.

3.To use this installer python-3.7.2.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

4.Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Setting PATH at Windows.

At the step of "INSTALL NOW"

-> click on ADD PYTHON TO PATH.

WINDOWS PATH IMAGE

(or)

-> Go to file location of python.

-> copy location path of python

-> Go to system properties -> open environmental variables.

-> Navigate to system variables -> open PATH. IMAGE

-> click on *NEW* tab and paste the PATH ADDRESS of python file location.

-> Click OK.

1.b) Write a program to demonstrate the use of basic Data Types

Basic datatypes are :

1. Numeric Datatype : Integer, Float, Complex number.
2. Sequence Datatype : List, Tuple, Range.
3. Boolean Datatype : bool
4. Text type : string(str)
5. Mapping type : dict
6. Set Datatype : set, frozenset.
7. Binary Types : bytes, bytearray, memoryview.

```
#To check datatype
x = 5
print(type(x))
```

```
<class 'int'>
```

```
#NUMERIC DATATYPE
a = 5
print(a, "is of type", type(a))

a = 2.0
print(a, "is of type", type(a))

a = 1+2j
print(a, "is of type", type(a), " complex :",isinstance(1+2j,complex))
```

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is of type <class 'complex'>  complex : True
```

```
#SEQUENCE DATATYPE
print("------------------------LIST----------------------------")
a = [1, 2.2, 'python', 5, 10, 20, 30]
print(a)

print("a[2] = ", a[2])

print("a[0:3] = ", a[0:3])

print("a[5:] = ", a[5:])

a[2] = 4
print(a)
del a
print("------------------------TUPLE----------------------------")
t = (5,'program', 1+3j)
```

```
# t[1] = 'program'
print("t[1] = ", t[1])
# t[0:3] = (5, 'program', (1+3j))
print("t[0:3] = ", t[0:3])

# Generates error
# Tuples are immutable
#t[0] = 10
print("------------------------RANGE----------------------------")
x = range(6)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
------------------------LIST---------------------------
[1, 2.2, 'python', 5, 10, 20, 30]
a[2] =  python
a[0:3] =  [1, 2.2, 'python']
a[5:] =  [20, 30]
[1, 2.2, 4, 5, 10, 20, 30]
------------------------TUPLE---------------------------
t[1] =  program
t[0:3] =  (5, 'program', (1+3j))
------------------------RANGE---------------------------
range(0, 6)
<class 'range'>
```

```
#BOOLEAN DATATYPE
x = True
y = False

#display x:
print(x)
print(y)

#display the data type of x:
print(type(x))
print(type(y))
```

```
True
False
<class 'bool'>
<class 'bool'>
```

```
#TEXT TYPE
x = "Hello World"

#display x:
print(x)

#display the data type of x:
```

```
print(type(x))
```

```
Hello World
<class 'str'>
```

```
#MAPPING TYPE
x = {"name" : "John", "age" : 36}

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
{'name': 'John', 'age': 36}
<class 'dict'>
```

```
#SET DATATYPE
print("-------------------------SET DATATYPE------------------------------")
x = {"apple", "banana", "cherry"}

#display x:
print(x)

#display the data type of x:
print(type(x))

print("---------------------FROZEN SET DATATYPE--------------------------")

x = frozenset({"apple", "banana", "cherry"})

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
-------------------------SET DATATYPE------------------------------
{'apple', 'banana', 'cherry'}
<class 'set'>
---------------------FROZEN SET DATATYPE--------------------------
frozenset({'apple', 'banana', 'cherry'})
<class 'frozenset'>
```

```
#BINARY DATATYPES
print("-------------------------BYTES-------------------------------")
x = b"Hello"

#display x:
print(x)

#display the data type of x:
```

```
print(type(x))


print("-------------------------BYTEARRAY-----------------------------")
x = bytearray(5)

#display x:
print(x)

#display the data type of x:
print(type(x))


print("-----------------------MEMORYVIEW----------------------------")
x = memoryview(bytes(5))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
     ---------------------------BYTES-------------------------------
     b'Hello'
     <class 'bytes'>
     -------------------------BYTEARRAY-----------------------------
     bytearray(b'\x00\x00\x00\x00\x00')
     <class 'bytearray'>
     -----------------------MEMORYVIEW----------------------------
     <memory at 0x7f54a4bfe2c0>
     <class 'memoryview'>
```

1.c) Write a program to demonstrate the Operators and Expressions

Python has 7 types of operators that you can use:

Arithmetic Operators

Comparison Operators

Assignment Operators

Logical Operators

Membership Operators

Identity Operators

Bitwise Operators

```
print("-------------------Arithmetic operators-----------------------")
a = 10
```

```
b = 2
print("Addition :",a+b)
print("subtraction :",a-b)
print("multiplication :",a*b)
print("division :",a/b)
print("exponentiation :",a**b)
print("Floor division :",a//b)
print("modulus :",a%b)
print("--------------------Comparison operators-----------------------")
i=int(5)
print(i == 8)
print(i<=5)
print(i!=6)
print("--------------------Logical operators-----------------------")
a = True
b = False

print(a and b)
print(a or b)
print("--------------------Assignment operators-----------------------")
a = 10
a = a + 5
print(a)

a = a * 2
print(a)

b = 2
b = 47 - b
print(b)

b = 2
b = 120 // b
print(b)
print("--------------------membership operator----------------------")
list1 = [3,4,3,2,4]
print(13 not in list1)
print(13 in list1)
print("--------------------identity operator----------------------")
print("identity of a:",id(a))
print("identity of b:",id(b))
print("--------------------Bitwise operator----------------------")
a = 0
b = 1
print("AND=",a & b)
print("OR =",a | b)
print("NOT",~a)
print("XOR = ",a^b)

a = 10
b = -10

# print bitwise right shift operator
print("a >> 1 =", a >> 1)
print("b >> 1 =", b >> 1)
```

```
a = 5
b = -10

# print bitwise left shift operator
print("a << 1 =", a << 1)
print("b << 1 =", b << 1)
```

```
--------------------Arithmetic operators------------------------
Addition : 12
subtraction : 8
multiplication : 20
division : 5.0
exponentiation : 100
Floor division : 5
modulus : 0
--------------------Comparison operators------------------------
False
True
True
--------------------Logical operators------------------------
False
True
--------------------Assignment operators------------------------
15
30
45
60
--------------------membership operator---------------------
True
False
--------------------identity operator---------------------
identity of a: 94124459629984
identity of b: 94124459630944
--------------------Bitwise operator---------------------
AND= 0
OR = 1
NOT -1
XOR =  1
a >> 1 = 5
b >> 1 = -5
a << 1 = 10
b << 1 = -20
```

1.d)Write a program to demonstrate the Functions and parameter passing Techniques.

A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result

```
print("\n------------FUNCTION WITH PARAMETER AND RETURNTYPE----------------")
def sum(x,y):
```

```
  return x+y

sum(100,50)
```

```
     ------------FUNCTION WITH PARAMETER AND RETURNTYPE----------------
     150
```

```
print("\n------------FUNCTION WITHOUT PARAMETER AND RETURNTYPE----------------")
def sum():
  a=int(input())
  b=int(input())
  return a+b

sum()

print("\n------------FUNCTION WITH PARAMETER AND NO RETURNTYPE----------------")
def sum(a,b):
  print("the sum of a,b is :",a+b)

sum(100,50)

print("\n-----------FUNCTION WITH NO PARAMETER AND NO RETURNTYPE--------------")
def sum():
  x=int(input())
  y=int(input())
  print("the sum of x,y is :",x+y)

sum()
```

```
     ------------FUNCTION WITHOUT PARAMETER AND RETURNTYPE----------------
     100
     50

     ------------FUNCTION WITH PARAMETER AND NO RETURNTYPE----------------
     the sum of a,b is : 150

     -----------FUNCTION WITH NO PARAMETER AND NO RETURNTYPE--------------
     100
     50
     the sum of x,y is : 150
```

Exercise 2

a) Write a Program to implement i. Packages ii. Modules iii. Built-in Functions

b) Write a Program to implement i. List ii. Tuple iii. Dictionaries

c) Programs on Strings, String Operations and Regular Expressions

2.a) Write a Program to implement i. Packages ii. Modules iii. Built-in Functions

```
#packages
import moviepy
help(moviepy)

from moviepy import audio
help(audio)
from moviepy import Clip

# Import everything needed to edit video clips
from moviepy.editor import *

# loading video
clip = VideoFileClip("/content/drive/MyDrive/20915A3501.mp4") #import video into colab

# clipping of the video
# getting video for only starting 10 seconds
clip = clip.subclip(0, 10)

# rotating video by 0 degree
clip = clip.rotate(0)

# Reduce the audio volume (volume x 0.5)
clip = clip.volumex(0.5)

# showing clip
clip.ipython_display(width = 720)
```

```
Help on package moviepy:

NAME
    moviepy

PACKAGE CONTENTS
    Clip
    audio (package)
    compat
    config
    config_defaults
    decorators
    editor
    tools
    version
    video (package)

VERSION
    0.2.3.5

FILE
    /usr/local/lib/python3.7/dist-packages/moviepy/__init__.py


Help on package moviepy.audio in moviepy:

NAME
    moviepy.audio

PACKAGE CONTENTS
    AudioClip
    fx (package)
    io (package)
    tools (package)

FILE
    /usr/local/lib/python3.7/dist-packages/moviepy/audio/__init__.py


100%|██████████| 221/221 [00:00<00:00, 878.83it/s]
100%|██████████| 250/251 [00:10<00:00, 24.26it/s]
```

```
#Modules
import math

help(math)
```

```
        log1p(x, /)
            Return the natural logarithm of 1+x (base e).

            The result is computed in a way which is accurate for x near zero.

        log2(x, /)
            Return the base 2 logarithm of x.

        modf(x, /)
            Return the fractional and integer parts of x.
```

```
            Both results carry the sign of x and are floats.

        pow(x, y, /)
            Return x**y (x to the power of y).

        radians(x, /)
            Convert angle x from degrees to radians.

        remainder(x, y, /)
            Difference between x and the closest integer multiple of y.

            Return x - n*y where n*y is the closest integer multiple of y.
            In the case where x is exactly halfway between two multiples of
            y, the nearest even value of n is used. The result is always exact.

        sin(x, /)
            Return the sine of x (measured in radians).

        sinh(x, /)
            Return the hyperbolic sine of x.

        sqrt(x, /)
            Return the square root of x.

        tan(x, /)
            Return the tangent of x (measured in radians).

        tanh(x, /)
            Return the hyperbolic tangent of x.

        trunc(x, /)
            Truncates the Real x to the nearest Integral toward 0.


            Uses the __trunc__ magic method.

    DATA
        e = 2.718281828459045
        inf = inf
        nan = nan
        pi = 3.141592653589793
        tau = 6.283185307179586

    FILE
        (built-in)
```

```
math.tan(-90)
```

```
1.995200412208242
```

```
math.sqrt(10)
```

```
3.1622776601683795
```

```
#Built-in functions
```

```
#Built in functions
print("Absolute Value:")
a=int(input("Enter a value:"))
print(abs(a))
print("\nBinary Value:")
b=int(input("Enter a value: "))
print(bin(b))
print("\nQuotient and remainder of 54 and 9 are:")
print(divmod(54,9))
print("\nFloat Value")
c=input("Enter a value: ")
print(float(c))
print("\nInt Value")
d=input("Enter a value: ")
print(int(d))
print("\nMaximum Value of 10,78,5")
print(max(10,78,5))
print("\nPower function")
e=int(input("Enter a value: "))
f=int(input("Enter a value: "))
print("The power of the given values is: ",pow(e,f))
print("\nSquarer Root")
import math
val=int(input("Enter a value: "))
print("The Square Root of the given number is : ",math.sqrt(val))
```

```
Absolute Value:
Enter a value:5
5

Binary Value:
Enter a value: 1001
0b1111101001

Quotient and remainder of 54 and 9 are:
(6, 0)

Float Value
Enter a value: 2.0
2.0

Int Value
Enter a value: 2
2

Maximum Value of 10,78,5
78

Power function
Enter a value: 3
Enter a value: 3
The power of the given values is:  27

Squarer Root
Enter a value: 2
The Square Root of the given number is :  1.4142135623730951
```

2.b) Write a Program to implement i. List ii. Tuple iii. Dictionaries

```
#list
print("--------------------------list---------------------------")
print("program 1:")
grocery=["deo","soap","pencil","utensils"]

print(type(grocery))

print("\n",grocery[0])

print("\n",grocery[0:3])

print("\n",grocery[::-1])

print("-------------------------------------------------")

print("\n program 2:")

a=(5,67,[1,3,[8,9]],"hello","test")
var=a[2][2][1]
print("\n",var)
```

```
--------------------------list---------------------------
program 1:
<class 'list'>

 deo

 ['deo', 'soap', 'pencil']

 ['utensils', 'pencil', 'soap', 'deo']
-------------------------------------------------

 program 2:

 9
```

```
#tuple
print("----------------------Tuple----------------------------")
tp=tuple()
tp=(1,2,3)
print("\n",type(tp))
print("\n",tp)
```

```
----------------------Tuple----------------------------

 <class 'tuple'>

 (1, 2, 3)
```

```
#update tuples
x = ("apple", "kiwi", "strawberry")
```

```
x = ("apple", "kiwi", "strawberry")
y = list(x)
y[1] = "mango"
x = tuple(y)

print(x)
```

```
('apple', 'mango', 'strawberry')
```

```
#dictionaries
thisdict = {
    "brand_name" : "Haldirams",
    "type" : "Food",
    "year" : 1972,
}

print(thisdict["brand_name"])

print("\n")

#Access items in dictionaries
car = {
"brand": "hyundai",
"model": "verna",
"year": 2020
}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change
print("\n")

for x in thisdict.values():
  print(x)
```

```
Haldirams


dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])


Haldirams
Food
1972
```

2.c) Programs on Strings, String Operations and Regular Expressions

```
#strings
print("--------------------STRINGS---------------------")
mystr="my name is nikhil"

#print(mystr[0], mystr[-1])
#print(type(mystr))
print(mystr[0:6])     #excludes last string
print(mystr[:])       #initial index to be 0 and last index is equal
print(mystr[::2])     #step size of two
print(mystr[::-2])    #reverse above string

print("\n")
#string operations
print("---------------------STRING OPERATIONS------------------------")

print(mystr.isalnum())  #is alpha numeric or not

print(mystr.endswith("nikhil"))

print(mystr.count("n"))

print(mystr.capitalize())

print(mystr.lower())

print(mystr.upper())

print(mystr.replace("nikhil","nikki"))
print("\n")
#regular expresions
print("----------------------REGULAR EXPRESSIONS------------------------")

import re

#Check if the string starts with "The" and ends with "Spain":

txt = "The rain in my village"
x = re.search("^The.*Spain$", txt)

if x:
  print("YES! We have a match!")
else:
  print("No match")

print("\n")

x = re.findall("ai", txt)
print(x)

x = re.search("\s", txt)

print("\n The first white-space character is located in position:", x.start())
print("\n")
x = re.split("\s", txt)
print(x)
```

```
print("\n")

x = re.sub("\s", "9", txt)
print(x)
```

```
     --------------------STRINGS---------------------
     my nam
     my name is nikhil
     m aei ihl
     lhi iea m


     ---------------------STRING OPERATIONS------------------------
     False
     True
     2
     My name is nikhil
     my name is nikhil
     MY NAME IS NIKHIL
     my name is nikki


     -----------------------REGULAR EXPRESSIONS------------------------
     No match


     ['ai']

      The first white-space character is located in position: 3


     ['The', 'rain', 'in', 'my', 'village']


     The9rain9in9my9village
```

Exercise 3

a) Write a Program to implement Class and Object.

b) Write a Program to implement Static and Instance methods, Abstract Classes and Interfaces.

c) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem)

```
#Write a Program to implement Class and Object.
class emp:
  no_of_emp = 0
  def __init__(self, name, sal):
    no_of_emp = 0
    self.name = name
    self.sal = sal
    no_of_emp += 1
  def show(self):
    return self.name,self.sal
```

```
        return self.name,self.sal

e1= emp("sara",100)
e2= emp("nara",200)
print(e1.no_of_emp)
print(e1.show)
print(e2.show)
print(e1.name)
print(e2.sal)
```

```
0
<bound method emp.show of <__main__.emp object at 0x7f549cdaded0>>
<bound method emp.show of <__main__.emp object at 0x7f549cdadcd0>>
sara
200
```

```
# Write a Program to implement Static and Instance methods, Abstract Classes and Interface
print("--------------------# Instance Method #-------------------")
class Student:

    def __init__(self, a, b):
        self.a = a
        self.b = b

    def avg(self):
        return (self.a + self.b) / 2

s1 = Student(10, 20)
print( s1.avg() )
print('\n')

print("--------------------# Static Method #-------------------")
class Student:
    name = 'Student'
    def __init__(self, a, b):
        self.a = a
        self.b = b

    @staticmethod
    def info():
        return "This is a student class"

print(Student.info())
```

```
--------------------# Instance Method #-------------------
15.0


--------------------# Static Method #-------------------
This is a student class
```

```
#abstract class and method
from abc import ABC, abstractclassmethod
class Employee(ABC):
    @abstractclassmethod
```

```
  @abstractclassmethod
  def work(self):
    pass
class HR(Employee):
  def work(self):
    print("HR manages the teams")
class manager(Employee):
  def work(self):
    print("Manager work is to manage office")
class TL(Employee):
  def work(self):
    print("TL manage his group")
T_L = TL()
T_L.work()
print("\n")
H_R = HR()
H_R.work()
print("\n")
MG = manager()
MG.work()
```

```
TL manage his group


HR manages the teams


Manager work is to manage office
```

```
#Write a program to compute distance between two points taking input from the user (Pythag
import math

a=input("enter first coordinate : ")

point1 = a.split(",")

b=input("enter second coordinate : ")

point2 = b.split(",")

distance = math.sqrt( ((int(point1[0])-int(point2[0]))**2)+((int(point1[1])-int(point2[1])

print("distance between ", a,"and", b, "is",distance)
```

```
enter first coordinate : 2,4
enter second coordinate : 5,8
distance between  2,4 and 5,8 is 5.0
```

Exercise 4

a) Write a program to implement Inheritance and Polymorphism.

b) Write a program to implement Files.

c) Write a program to illustrate Handling.

program on inheritance and polymorphism

[ ] ↳ *1 cell hidden*

program on illustrate handling

```
# TRY AND EXCEPT

num1=int(input("number 1:"))
num2=int(input("number 2:"))
try:
  print(num1//num2)
except Exception as e:
  #print(e)
  print("you divided it by zero")
```

```
number 1:4
number 2:2
2
```

program on files

```
#opening a file
file_1 = open ("file.txt","w+") #w+ means write

#inserting data into file
for i in range(10):
     file_1.write("This is a line %d\r\n" % (i+1))



file_1.close()


def main():
  file_1=open("file.txt", "r") #read mode
  if file_1.mode == "r":
    info = file_1.read()
    print(info)

if __name__ == "__main__":
  main()
```

```
This is a line 1
This is a line 2
This is a line 3
This is a line 4
```

```
This is a line 5
This is a line 6
This is a line 7
This is a line 8
This is a line 9
This is a line 10
```

Exercise 5

a) Write a program using scikit-learn to implement K-means Clustering.

b) Program to calculate the entropy and the information gain.

c) Program to implement perceptron.

5.a) Write a program using scikit-learn to implement K-means Clustering.

Dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
sns.set_style('whitegrid')
%matplotlib inline
```

Loading Dataset

```
from google.colab import drive
#file_data = files.upload()
drive.mount("/content/gdrive")
```

```
Mounted at /content/gdrive
```

```
df2 = pd.read_csv('/content/gdrive/MyDrive/Projects/ML with python 15/College_Data.csv', i
```

```
df2.head()
```

College	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Underg
Abilene Christian	Yes	1660	1232	721	23	52	2885	
Adelphi	Yes	2186	1924	512	16	29	2683	
Adrian	Yes	1428	1097	336	22	50	1036	

importing kmeans from scikit library

```
from sklearn.cluster import KMeans
```

```
#create an instance of Kmeans model with private and govt cluster
c_cluster = KMeans(2)
```

```
#fitting model for private label
c_cluster.fit(df2.drop('Private', axis=1))
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
c_cluster.cluster_centers_
```

```
array([[1.03631389e+04, 6.55089815e+03, 2.56972222e+03, 4.14907407e+01,
        7.02037037e+01, 1.30619352e+04, 2.46486111e+03, 1.07191759e+04,
        4.64347222e+03, 5.95212963e+02, 1.71420370e+03, 8.63981481e+01,
        9.13333333e+01, 1.40277778e+01, 2.00740741e+01, 1.41705000e+04,
        6.75925926e+01],
       [1.81323468e+03, 1.28716592e+03, 4.91044843e+02, 2.53094170e+01,
        5.34708520e+01, 2.18854858e+03, 5.95458894e+02, 1.03957085e+04,
        4.31136472e+03, 5.41982063e+02, 1.28033632e+03, 7.04424514e+01,
        7.78251121e+01, 1.40997010e+01, 2.31748879e+01, 8.93204634e+03,
        6.51195815e+01]])
```

```
df2.dtypes
```

```
Private         object
Apps             int64
Accept           int64
Enroll           int64
Top10perc        int64
Top25perc        int64
F.Undergrad      int64
P.Undergrad      int64
Outstate         int64
Room.Board       int64
Books            int64
Personal         int64
PhD              int64
Terminal         int64
S.F.Ratio      float64
perc.alumni      int64
Expend           int64
```

```
Grad.Rate        int64
```

Evaluation

```
df2['Cluster'] = df2['Private'].apply(lambda x: 1 if x == 'Yes' else 0)
```

```
df2.head()
```

College	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F.Undergrad	P.Underg
Abilene Christian University	Yes	1660	1232	721	23	52	2885	
Adelphi University	Yes	2186	1924	512	16	29	2683	
Adrian College	Yes	1428	1097	336	22	50	1036	
Agnes								

```
from sklearn.metrics import confusion_matrix, classification_report
```

```
print(confusion_matrix(df2['Cluster'], c_cluster.labels_))
```

```
[[ 74 138]
 [ 34 531]]
```

```
print(classification_report(df2['Cluster'], c_cluster.labels_))
```

```
              precision    recall  f1-score   support

           0       0.69      0.35      0.46       212
           1       0.79      0.94      0.86       565

    accuracy                           0.78       777
   macro avg       0.74      0.64      0.66       777
weighted avg       0.76      0.78      0.75       777
```

so here 0=govt, 1=private

5.b)program to calculate entropy and the information gain

```
#entropy
!pip install scipy
```

```
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (1.4.1
```

```
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-package
```

```
import numpy as np
from scipy.stats import entropy
from math import log, e
import pandas as pd

import timeit

def entropy1(labels, base=None):
  value,counts = np.unique(labels, return_counts=True)
  return entropy(counts, base=base)

def entropy2(labels, base=None):
  """ Computes entropy of label distribution. """

  n_labels = len(labels)

  if n_labels <= 1:
    return 0

  value,counts = np.unique(labels, return_counts=True)
  probs = counts / n_labels
  n_classes = np.count_nonzero(probs)

  if n_classes <= 1:
    return 0

  ent = 0.

  # Compute entropy
  base = e if base is None else base
  for i in probs:
    ent -= i * log(i, base)

  return ent

def entropy3(labels, base=None):
  vc = pd.Series(labels).value_counts(normalize=True, sort=False)
  base = e if base is None else base
  return -(vc * np.log(vc)/np.log(base)).sum()

def entropy4(labels, base=None):
  value,counts = np.unique(labels, return_counts=True)
  norm_counts = counts / counts.sum()
  base = e if base is None else base
  return -(norm_counts * np.log(norm_counts)/np.log(base)).sum()

labels = [1,3,5,2,3,5,3,2,1,3,4,5]

print(entropy1(labels))
print(entropy2(labels))
print(entropy3(labels))
```

```
print(entropy4(labels))
```

```
1.5171063970610277
1.5171063970610277
1.5171063970610277
1.5171063970610277
```

5.c) program to implement perceptron

```
#import packages
import sklearn.datasets
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score

#load the breast cancer data
breast_cancer = sklearn.datasets.load_breast_cancer()

#convert the data to pandas dataframe.
data = pd.DataFrame(breast_cancer.data, columns = breast_cancer.feature_names)
data["class"] = breast_cancer.target
data.head()
data.describe()

#plotting a graph to see class imbalance
data['class'].value_counts().plot(kind = "barh")
plt.xlabel("Count")
plt.ylabel("Classes")
plt.show()

from sklearn.preprocessing import MinMaxScaler
#perform scaling on the data.
X = data.drop("class", axis = 1)
Y = data["class"]
mnscaler = MinMaxScaler()
X = mnscaler.fit_transform(X)
X = pd.DataFrame(X, columns=data.drop("class",axis = 1).columns)

#train test split.
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.1, stratify = Y, ra
```

Exercise 6

a) Generate a decision tree. Find the Depth of decision trees and observe the results, then propose some changes in DecisionTreeClassifier function to limit.

b) Occupancy estimator using random forest.

```
#a) Generate a decision tree. Find the Depth of decision trees and observe the results, th
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris


iris_data = load_iris()
iris=pd.DataFrame(iris_data.data)

#priting features name of iris data
print ("Features Name : ", iris_data.feature_names)

#shape of datasets
print ("Dataset Shape: ", iris.shape)

#first five sample
print ("Dataset: ",iris.head())
```

```
Features Name :  ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'peta
Dataset Shape:  (150, 4)
Dataset:       0    1    2    3
0  5.1  3.5  1.4  0.2
1  4.9  3.0  1.4  0.2
2  4.7  3.2  1.3  0.2
3  4.6  3.1  1.5  0.2
4  5.0  3.6  1.4  0.2
```

```
X = iris.values[:, 0:4]
Y = iris_data.target
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3, random_state =



classifier = DecisionTreeClassifier(max_depth=8,random_state= 44)
classifier.fit(X_train,y_train)
print('Depth' classifier.get_depth())
```

```
print('Depth',classifier.get_depth())
print('Train',classifier.score(X_train,y_train))
print('Test',classifier.score(X_test,y_test))
```

```
Depth 4
Train 1.0
Test 0.9555555555555556
```

```
#b) Occupancy estimator using random forest
```

Exercise 7

a) Calculating with matrices using numpy : inv, pinv, matrix_rank, solve, lstsq, svd, transpose, eig, sort, linspace, meshgrid, mgrid, ogrid, concatenate, tile, squeeze, integrate.

```
import numpy as np
```

inv(), pinv()

```
from numpy import linalg
matrix = np.array([
                [10,20,30,40],
                [15,25,35,45],
                [77,55,66,33],
                [20,60,70,40]
])

print("Inverse matrix:\n")
print(linalg.inv(matrix))
print("\n\n")
print("Pseudo Inverse matrix:\n")
print(linalg.pinv(matrix))
```

```
Inverse matrix:

[[-0.03333333  0.03333333  0.01515152 -0.01666667]
 [-0.78333333  0.70833333 -0.04166667  0.02083333]
 [ 0.76666667 -0.71666667  0.03787879  0.00833333]
 [-0.15        0.175      -0.01136364 -0.0125    ]]



Pseudo Inverse matrix:

[[-0.03333333  0.03333333  0.01515152 -0.01666667]
 [-0.78333333  0.70833333 -0.04166667  0.02083333]
 [ 0.76666667 -0.71666667  0.03787879  0.00833333]
 [-0.15        0.175      -0.01136364 -0.0125    ]]
```

matrix_rank()

```
matrix = np.eye(5)
print(matrix)

print("Rank of matrix: ",linalg.matrix_rank(matrix))
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
Rank of matrix:  5
```

solve equation()

2x - 8y = 1

4x + 6y = 5

```
coeff_matrix = np.array([
                        [2, -8],
                        [4, 6]
])

const_matrix = np.array([1, 5])

result = linalg.solve(coeff_matrix, const_matrix)

print(result)
```

```
[1.04545455 0.13636364]
```

lstsq()

```
coeff_matrix = np.array([
                        [2, -8],
                        [4, 6]
])

const_matrix = np.array([1, 5])

linalg.lstsq(coeff_matrix, const_matrix)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: FutureWarning: `rconc
To use the future default and silence this warning we advise to pass `rcond=None`, tc

(array([1.04545455, 0.13636364]),
 array([], dtype=float64),
 2,
 array([10.03952968,  4.38267543]))
```

```
X = np.array([
              [10,20,30,40],
                [15,25,35,45],
                [77,55,66,33],
                [20,60,70,40]
])

u, s, v = linalg.svd(X)

print("Matirx:")
print(X)
print("\n")
print("U:")
print(u)
print("\n")
print("Sigma:")
print(s)
print("\n")
print("V*")
print(v)

print("\nOriginal Matrix")
print((u@np.diag(s))@v)
```

```
Matirx:
[[10 20 30 40]
 [15 25 35 45]
 [77 55 66 33]
 [20 60 70 40]]


U:
[[-0.29260264 -0.41488959  0.45148805  0.73376349]
 [-0.3503221  -0.40643284  0.50193007 -0.67834576]
 [-0.67393947  0.7195486   0.16320286  0.037686  ]
 [-0.58091627 -0.38069515 -0.71943698 -0.00423418]]


Sigma:
[170.94884286  46.28094391  25.16202878   0.6630708 ]


V*
[[-0.41937998 -0.50618507 -0.62114194 -0.42670809]
 [ 0.81126156 -0.03727557 -0.12597604 -0.56973262]
 [ 0.40623561 -0.50123338 -0.33689676  0.68573924]
 [-0.03078554 -0.70082259  0.69628536 -0.15194291]]

Original Matrix
[[10. 20. 30. 40.]
 [15. 25. 35. 45.]
 [77. 55. 66. 33.]
 [20. 60. 70. 40.]]
```

transpose()

```
matrix = np.random.randint(1,25,24).reshape(4,6)

print("Matrix:")
print(matrix)
print("\n\n")

print("Transpose of Matrix:")
print(matrix.transpose())
```

```
Matrix:
[[19  1 13 10  8  9]
 [ 7 16 10  3 24 22]
 [10 19 20 17  1  9]
 [15 17 24  8  1 18]]



Transpose of Matrix:
[[19  7 10 15]
 [ 1 16 19 17]
 [13 10 20 24]
 [10  3 17  8]
 [ 8 24  1  1]
 [ 9 22  9 18]]
```

eig()

```
matrix = np.array([
                  [10,20,30,40],
                [15,25,35,45],
                [77,55,66,33],
                [20,60,70,40]
])

print("Matrix: ")
print(matrix)

eval, evec = linalg.eig(matrix)

print("\nEigen values:")
print(eval)

print("\nEigen vectors:")
print(evec)
```

```
Matrix:
[[10 20 30 40]
 [15 25 35 45]
 [77 55 66 33]
```

```
 [20 60 70 40]]

Eigen values:
[166.75440837 +0.j         -13.55231861+20.06472663j
 -13.55231861-20.06472663j   1.35022886 +0.j        ]

Eigen vectors:
[[-0.3203412 +0.j         -0.36427748+0.18738853j -0.36427748-0.18738853j
  -0.01277138+0.j        ]
 [-0.37807282+0.j         -0.33482075+0.22794015j -0.33482075-0.22794015j
  -0.69780157+0.j        ]
 [-0.64260184+0.j          0.67449206+0.j          0.67449206-0.j
   0.69568367+0.j        ]
 [-0.58438458+0.j         -0.21796668-0.40703414j -0.21796668+0.40703414j
  -0.17010023+0.j        ]]
```

sort()

```
matrix = np.random.randint(1,20,16).reshape(4,4)
print(matrix)

print("\n\nsorted:")
print(np.sort(matrix, axis=None))

print("\n\nsorted row wise:")
print(np.sort(matrix, axis=0))

print("\n\nsorted column wise:")
print(np.sort(matrix, axis=1))
```

```
[[ 2 14 18  5]
 [ 7  9  4 10]
 [13  9 13 17]
 [ 8 17 19  1]]


sorted:
[ 1  2  4  5  7  8  9  9 10 13 13 14 17 17 18 19]


sorted row wise:
[[ 2  9  4  1]
 [ 7  9 13  5]
 [ 8 14 18 10]
 [13 17 19 17]]


sorted column wise:
[[ 2  5 14 18]
 [ 4  7  9 10]
 [ 9 13 13 17]
 [ 1  8 17 19]]
```

linspace()

```
a = np.linspace(0,100,num=10)
print(a)
```

```
[  0.          11.11111111  22.22222222  33.33333333  44.44444444
  55.55555556  66.66666667  77.77777778  88.88888889 100.        ]
```

meshgrid()

```
a = np.linspace(1, 5, 5)
b = np.linspace(1,8,8)

x,y = np.meshgrid(a,b)

print(x)
print(y)
```

```
[[1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]
 [1. 2. 3. 4. 5.]]
[[1. 1. 1. 1. 1.]
 [2. 2. 2. 2. 2.]
 [3. 3. 3. 3. 3.]
 [4. 4. 4. 4. 4.]
 [5. 5. 5. 5. 5.]
 [6. 6. 6. 6. 6.]
 [7. 7. 7. 7. 7.]
 [8. 8. 8. 8. 8.]]
```

mgrid()

```
a = np.mgrid[1:10,6:12]
print(a)
```

```
[[[ 1  1  1  1  1  1]
  [ 2  2  2  2  2  2]
  [ 3  3  3  3  3  3]
  [ 4  4  4  4  4  4]
  [ 5  5  5  5  5  5]
  [ 6  6  6  6  6  6]
  [ 7  7  7  7  7  7]
  [ 8  8  8  8  8  8]
  [ 9  9  9  9  9  9]]

 [[ 6  7  8  9 10 11]
  [ 6  7  8  9 10 11]
  [ 6  7  8  9 10 11]
```

```
        [ 6  7  8  9 10 11]
        [ 6  7  8  9 10 11]
        [ 6  7  8  9 10 11]
        [ 6  7  8  9 10 11]
        [ 6  7  8  9 10 11]
        [ 6  7  8  9 10 11]]]
```

ogrid()

```
a = np.ogrid[1:8,5:10]
print(a)
```

```
    [array([[1],
           [2],
           [3],
           [4],
           [5],
           [6],
           [7]]), array([[5, 6, 7, 8, 9]])]
```

concatenate()

```
a = np.array([
             [1,2,3],
             [4,5,6],
             [7,8,9]
])

b = np.array([
             [10,20,30],
               [15,25,35],
               [77,55,66]
])
x = np.concatenate((a,b), axis=0)
print(x)

print("\n\n")
x = np.concatenate((a,b), axis=1)
print(x)
```

```
    [[ 1  2  3]
     [ 4  5  6]
     [ 7  8  9]
     [10 20 30]
     [15 25 35]
     [77 55 66]]



    [[ 1  2  3 10 20 30]
     [ 4  5  6 15 25 35]
     [ 7  8  9 77 55 66]]
```

tile()

```
arr = np.array([
             [10,20],
             [30,40]
])

print(np.tile(arr,3))
print("\n\n")
print(np.tile(arr,(3,3)))
```

```
     [[10 20 10 20 10 20]
      [30 40 30 40 30 40]]



     [[10 20 10 20 10 20]
      [30 40 30 40 30 40]
      [10 20 10 20 10 20]
      [30 40 30 40 30 40]
      [10 20 10 20 10 20]
      [30 40 30 40 30 40]]
```

squeeze()

```
a = np.array([
             [420],
             [530],
             [940]
])

print(np.squeeze(a))
```

```
     [420 530 940]
```

Integrate()

```
from scipy import integrate
def integrand(x, a, b):
  return a*x**3 + b*x**2

result = integrate.quad(integrand, 0, 2, args=(4,8))

print(result)
```

```
     (37.333333333333336, 4.1448326252672513e-13)
```

Exercise 8

a) Program using pandas.

b) Program using matplotlib – use minimum 5 plotting techniques.

program using pandas

Pandas Series

```
import pandas as pd

sdata = pd.Series([2,4,6,8,10],index=[103,104,100,101,102])
print(type(sdata))
print(sdata)
```

```
<class 'pandas.core.series.Series'>
103     2
104     4
100     6
101     8
102    10
dtype: int64
```

```
data = {'a':1,'b':2,'c':3,'d':4}
s_data = pd.Series(data = data)
print(s_data)
```

```
a    1
b    2
c    3
d    4
dtype: int64
```

```
s_data = pd.Series(400, index = ['a','b','c','d','e'])
print(s_data)
```

```
a    400
b    400
c    400
d    400
e    400
dtype: int64
```

```
dates = pd.date_range('20200801',periods=10)
print(dates)
```

```
DatetimeIndex(['2020-08-01', '2020-08-02', '2020-08-03', '2020-08-04',
               '2020-08-05', '2020-08-06', '2020-08-07', '2020-08-08',
               '2020-08-09', '2020-08-10'],
              dtype='datetime64[ns]', freq='D')
```

Pandas Dataframes

```
data=[(1,'pavan',10,100),
      (2,"chintu",10,100),
      (3,"nihith",10,100),
      (4,"krishna",10,80),
      (5,"vishnu",10,80)
]

df=pd.DataFrame(data,columns=['S.no','Name','Class','Marks'])
df
```

	S.no	Name	Class	Marks
0	1	pavan	10	100
1	2	chintu	10	100
2	3	nihith	10	100
3	4	krishna	10	78
4	5	vishnu	10	80

```
#creating a dataframe with a list of dicionaries
data=[
      {'sno':1,'name':'pavan','class':10,'marks':100},
    {'sno':2,'name':'chintu','class':10,'marks':100},
    {'sno':3,'name':'nihith','class':10,'marks':100},
    {'sno':4,'name':'krishna','class':10,'marks':80},
    {'sno':5,'name':'vishnu','class':10,'marks':80},
]

df=pd.DataFrame(data)
df
```

	sno	name	class	marks
0	1	pavan	10	100
1	2	chintu	10	100
2	3	nihith	10	100
3	4	krishna	10	78
4	5	vishnu	10	80

```
data_dict={
    'sno':[1,2,3,4,5],
     'name':['pavan','chintu','nihith','krishna','vishnu'],
     'class':[10,10,10,10,10],
     'marks':[40,45,50,50,40]
}

df = pd.DataFrame(data=data_dict)

df.head()
```

	sno	name	class	marks
0	1	pavan	10	40
1	2	chintu	10	45
2	3	nihith	10	50
3	4	krishna	10	50
4	5	vishnu	10	40

```
s.no = np.arange(1,51)
age = np.random.randint(2,48,50)
salary = np.random.randint(10000,70000,50)

df = pd.DataFrame()
df['Id'] = s.no
df['age'] = age
df['salary'] = salary


df.head()
```

	Id	age	salary
0	1	10	37544
1	2	12	65946
2	3	46	12404
3	4	10	62831
4	5	2	28068

```
df.head()
```

	Id	age	salary
0	1	10	37544

```
df.describe()
```

	Id	age	salary
count	50.00000	50.000000	50.000000
mean	25.50000	25.700000	42820.540000
std	14.57738	12.797082	17727.424053
min	1.00000	2.000000	10822.000000
25%	13.25000	16.250000	30317.250000
50%	25.50000	25.500000	41012.000000
75%	37.75000	35.750000	58823.500000
max	50.00000	47.000000	69860.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Id      50 non-null     int64
 1   age     50 non-null     int64
 2   salary  50 non-null     int64
dtypes: int64(3)
memory usage: 1.3 KB
```

```
df.iloc[:10,:2]
```

	Id	age

```
print(df['age'].max())
print(df['age'].mean())
print(df['age'].median())
print(df['age'].std())
```

```
47
25.7
25.5
12.797081937787576
```

```
df[df.age>=50]
```

	Id	age	salary

```
df[df.age==df.age.max()]
```

	Id	age	salary
22	23	47	65721

```
df['salary'][df.age==df.age.max()]
```

```
22    65721
Name: salary, dtype: int64
```

```
df.to_csv("data.csv")
```

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

```
df.isnull().sum()
```

```
Unnamed: 0    0
Id            0
age           0
salary        0
dtype: int64
```

```
df['age'].value_counts()
```

```
25    3
41    3
6     3
10    2
20    2
21    2
16    2
13    2
46    2
26    2
```

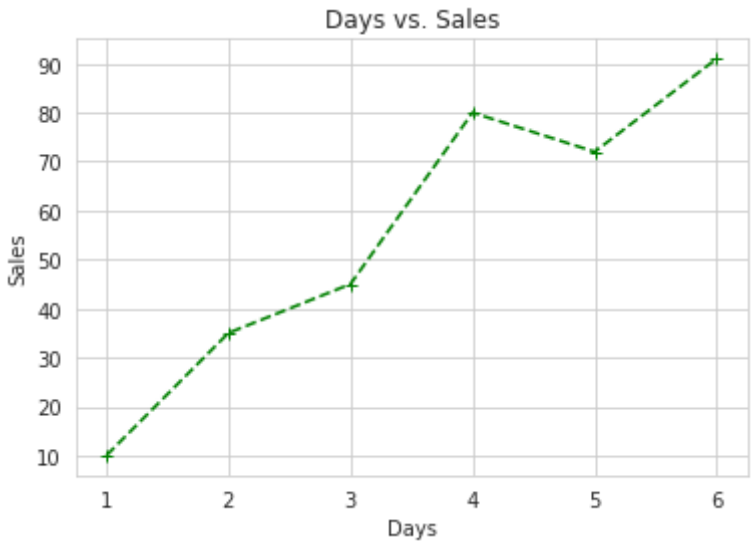
```
29    2
31    2
34    2
36    2
40    2
43    2
19    2
17    1
12    1
5     1
4     1
47    1
22    1
23    1
27    1
30    1
33    1
35    1
44    1
2     1
Name: age, dtype: int64
```

Program using matplotlib – use minimum 5 plotting techniques.

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

Lineplot

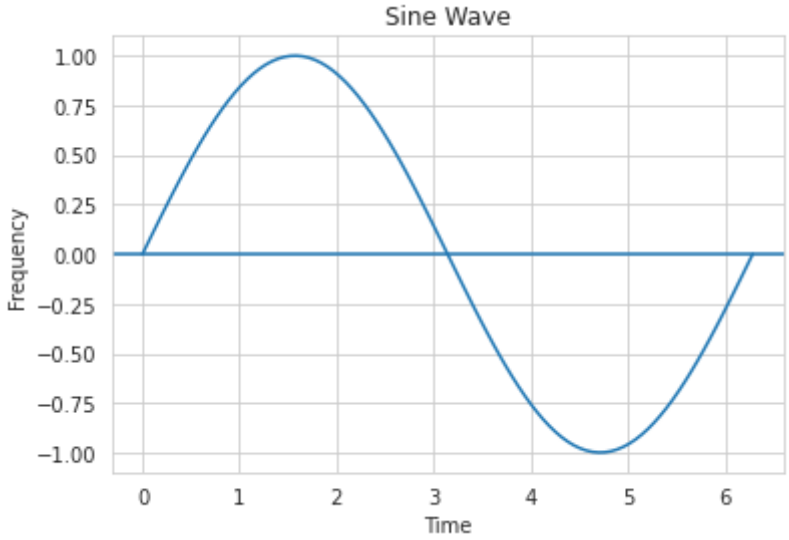
```
days = [1,2,3,4,5,6]
sales = [10,35,45,80,72,91]
plt.plot(days,sales,'g+--')
plt.title("Days vs. Sales")
plt.xlabel("Days")
plt.ylabel("Sales")
plt.show()
```




```
import math
```

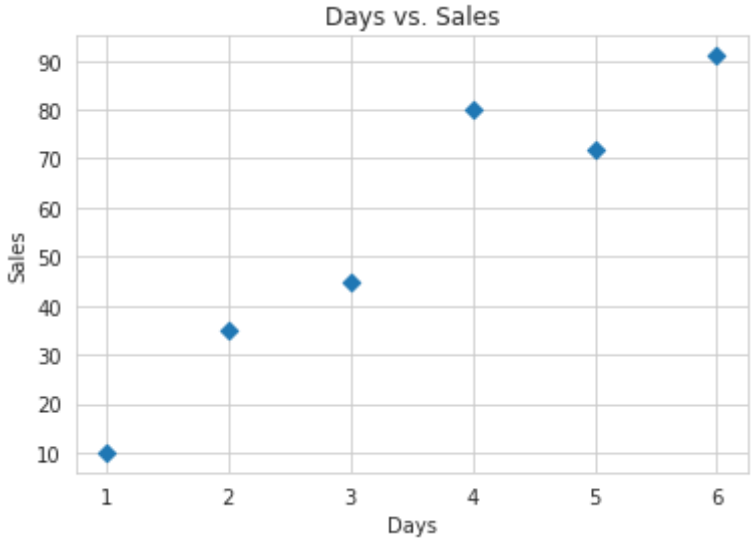
```
import math
x = np.linspace(0,2*math.pi,100)
y = np.sin(x)
plt.plot(x,y)
plt.axhline(y=0)

plt.title("Sine Wave")
plt.xlabel("Time")
plt.ylabel("Frequency")
```

```
Text(0, 0.5, 'Frequency')
```



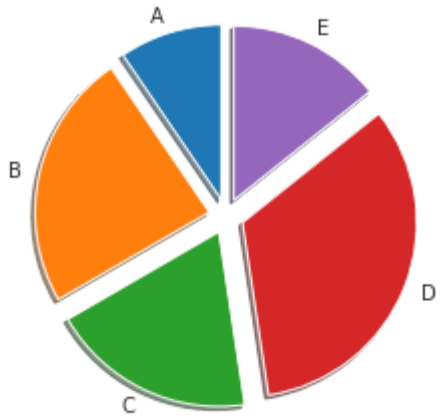
Scatter Plot

```
days = [1,2,3,4,5,6]
sales = [10,35,45,80,72,91]
plt.scatter(days,sales,marker="D")
plt.title("Days vs. Sales")
plt.xlabel("Days")
plt.ylabel("Sales")
plt.show()
```



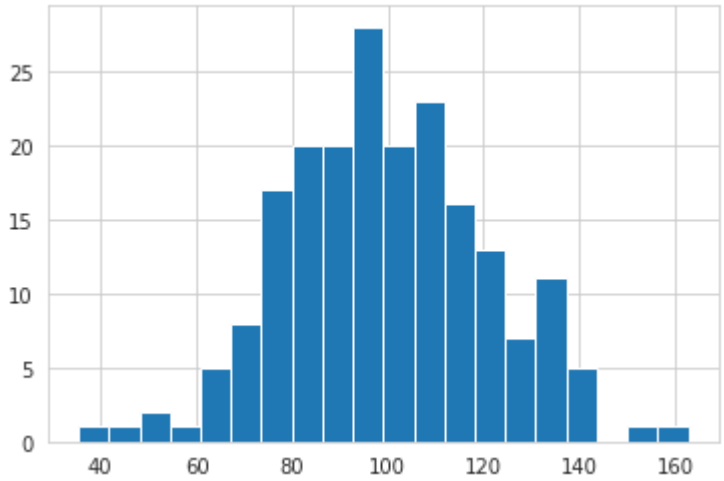
Pie Chart

```
x = [20,50,40,70,30]

labels = ['A','B','C','D','E']
plt.pie(x,labels=labels,shadow=True,startangle=90,explode=[0.1,0.1,0.1,0.1,0.1
                                                           ])
plt.show()
```



Histogram

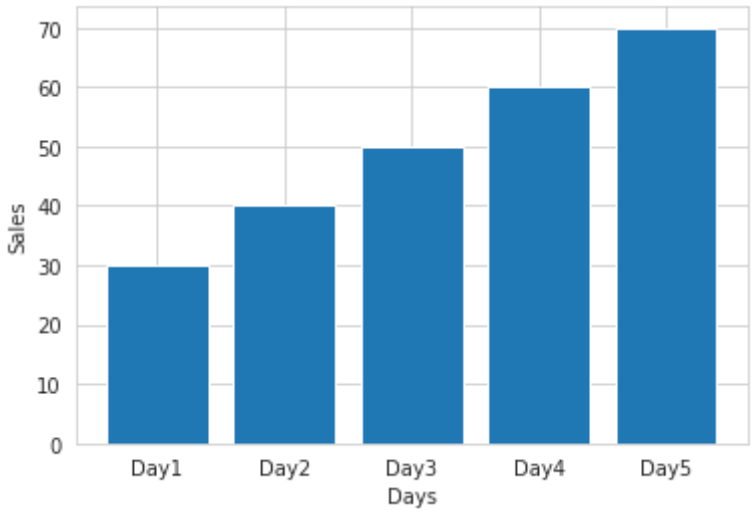
```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(100,20,200)

plt.hist(x,bins=20)
plt.show()
```



Bar Graph

```
y = [30,40,50,60,70]
x = ["Day1","Day2","Day3","Day4","Day5"]
```

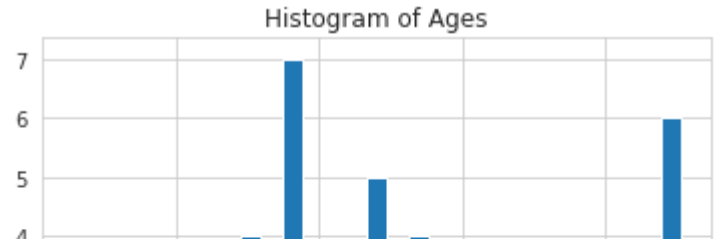
```
x = [ 'Day1' , 'Day2' , 'Day3' , 'Day4' , 'Day5' ]
plt.xlabel("Days")
plt.ylabel("Sales")

plt.bar(x,y)
plt.show()
```



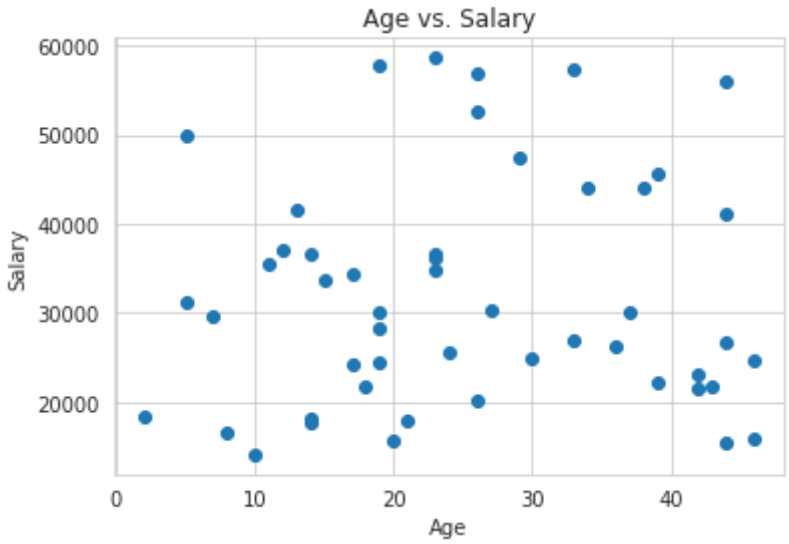
Exercise 9

a) Graph using matplotlib

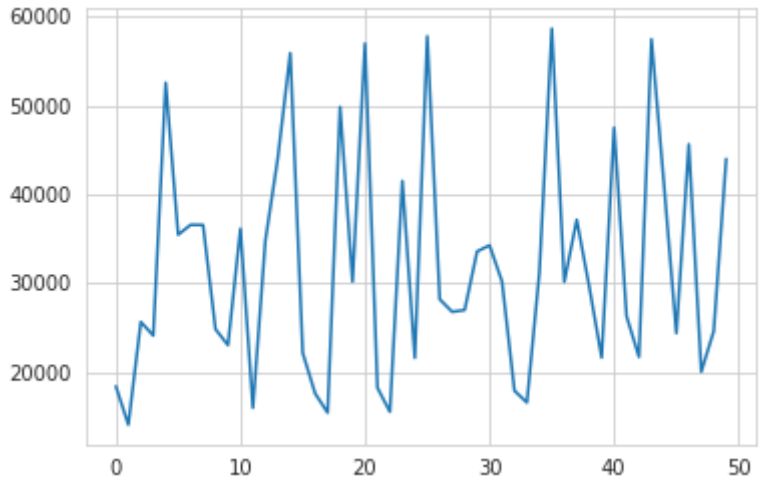
```
import pandas as pd


sno = np.arange(1,51)
age = np.random.randint(2,48,50)
salary = np.random.randint(10000,60000,50)

df = pd.DataFrame()
df['Sno'] = sno
df['Age'] = age
df['Salary'] = salary


plt.hist(df.Age,bins=15,rwidth=0.5)
plt.title("Histogram of Ages")
plt.show()
```



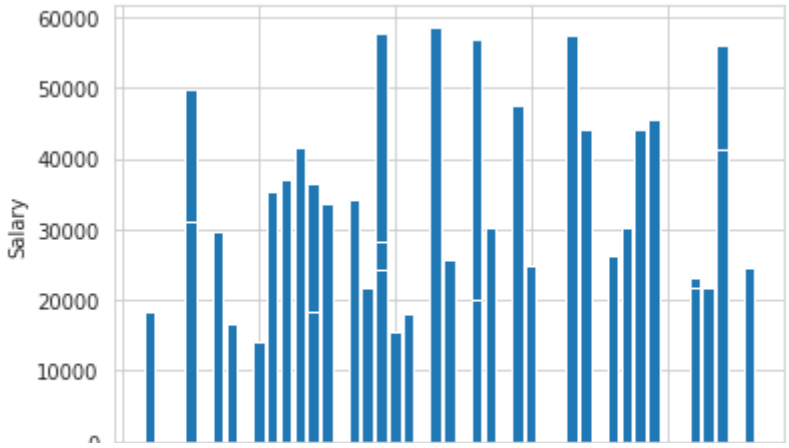

```
plt.scatter(df.Age,df.Salary)
plt.title("Age vs. Salary")
plt.xlabel("Age")
plt.ylabel("Salary")
plt.show()
```




```
plt.plot(df.Salary)
plt.show()
```



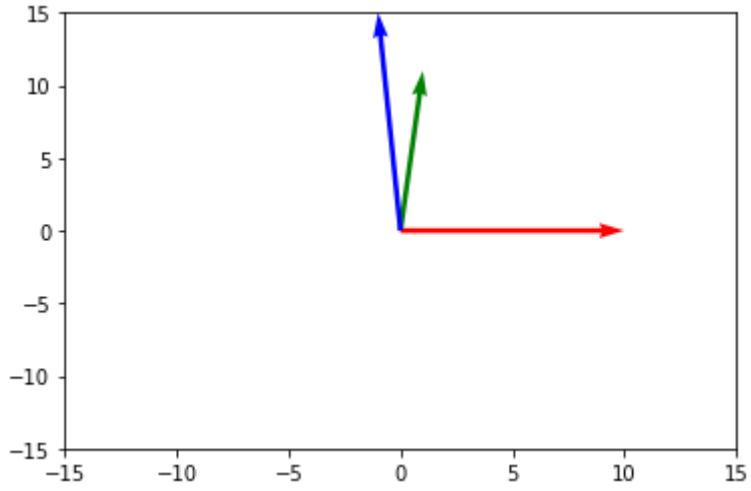

```
plt.bar(df.Age,df.Salary)
plt.xlabel("Age")
plt.ylabel("Salary")
plt.show()
```



Exercise 10

a) Vector using matplotlib

```
import matplotlib.pyplot as plt

#a) Vector using matplotlib
plt.quiver(0,0,1,11,scale_units='xy', angles='xy', scale=1, color='g')
plt.quiver(0,0,10,0,scale_units='xy', angles='xy', scale=1, color='r')
plt.quiver(0,0,-1,15,scale_units='xy', angles='xy', scale=1, color='b')
plt.xlim(-15,15)
plt.ylim(-15,15)
plt.show()
```



Exercise 11

a) Program to estimating occupancy using decision tree

✓ 0s completed at 14:04