```
#1.NUMERIC
#1.a)int
a=int(input("enter num"))
b=int(input("enter num"))
c=a+b
print("sum is",c)
print(type(c))
print(".......")
#1.b)float
a=float(input("enter num"))
b=float(input("enter num"))
c=a+b
print("sum is",c)
print(type(c))
print(".......")
#1.c)complex
a=complex(input("enter num"))
b=complex(input("enter num"))
c=a+b
print("sum is",c)
print(type(c))
print("_______________")
#2.BOOLEAN
print(9>2)
print(9 == 2)
print(9 < 2)
print("_______________")
#3.SET
Set = {1, 2, 3}# set of integers
print(Set)
Set = {2.0, "Hello", (3, 4, 5)}# set of mixed datatypes
print(Set)
print(type(Set))
print("_______________")
#4.SEQUENCE TYPE
#4.a)Strings
my_string = "hello"
print(my_string)
my_string = """Hello, I'm balu
           from Vidya Jyothi Institute of Technology """
print(my_string)# triple quotes string can extend multiple lines
print(my_string[0:15])
print(my_string[:-10])
print(type(my_string))
print(".......")
#4.b)list
my_list = ['d','e','f','g','h']
print(my_list)
print(my_list[2])
print(my_list[-1])
print(type(my_list))
print(".......")
#4.c)Tuple
tup=('hari','ganga','shiva','hema','ravi','sweety')
print(tup)
print(tup[0])
print(tup[1:4])
print(type(tup))
print("_______________")
```

```
#5.DICTIONARY
my_dict = {'name': 'balu', 'age': 19}
print(my_dict)
print(my_dict['name'])
print(my_dict.get('age'))
print(type(my_dict))
```

```
enter num5
enter num7
sum is 12
<class 'int'>
.......
enter num2
enter num7
sum is 9.0
<class 'float'>
.......
enter num4
enter num8
sum is (12+0j)
<class 'complex'>
_______________
True
False
False
_______________
{1, 2, 3}
{(3, 4, 5), 2.0, 'Hello'}
<class 'set'>
_______________
hello
Hello, I'm balu
           from Vidya Jyothi Institute of Technology
Hello, I'm balu
Hello, I'm balu
           from Vidya Jyothi Institute of T
<class 'str'>
.......
['d', 'e', 'f', 'g', 'h']
f
h
<class 'list'>
.......
('hari', 'ganga', 'shiva', 'hema', 'ravi', 'sweety')
hari
('ganga', 'shiva', 'hema')
<class 'tuple'>
_______________
{'name': 'balu', 'age': 19}
balu
19
<class 'dict'>
```

```
#1.OPERATORS
#1.a)Arithmetic operators
a = int(input("Enter a number"))
b = int(input("Enter a number"))
add = a + b # Addition of numbers
sub = a - b # Subtraction of numbers
mul = a * b # Multiplication of number
div1 = a / b # Division(float) of number
div2 = a // b # Division(floor) of number
mod = a % b # Modulo of both number
p = a ** b# Power
print(add)
print(sub)
print(mul)
print(div1)
print(div2)
print(mod)
print(p)
print("..................")
#1.b)Relatinal operators
print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
print("..................")
#1.c)Bitwise operators
print(a & b)#bitwise AND operation
print(a | b)#bitwise OR operation
print(~a)#bitwise NOT operation
print(a ^ b)#bitwise XOR operation
print(a >> 2)#bitwise right shift operation
print(a << 2)#bitwise left shift operation
print("..................")
#1.d)Identity operators
print(a is not b)
print(a is b)
print("..................")
#1.e)Logical operators
S=True
T=False
print(S and T)
print(S or T)
print(not S)
print(not T)
print("..................")
#1.f)Assignment operators
a+=b
print(a)
a-=b
print(a)
a*=b
print(a)
a/=b
print(a)
a%=b
print(a)
a//=b
```

```
print(a)
a**=b
print(a)
print("..................")
#1.g)Membership operators
x = 'Vidya Jyothi Institute of Technology'
y = {1:'a',2:'b'}
print('I' in x)
print('of' not in x)
print('Of' not in x)
print(1 in y)
print('o' in y)
```

```
Enter a number2
Enter a number4
6
-2
8
0.5
0
2
16
..................
False
True
False
True
False
True
..................
0
6
-3
6
0
8
..................
True
False
..................
False
True
False
True
..................
6
2
8
2.0
2.0
0.0
0.0
..................
True
False
True
True
False
```

```
#function without any parameters
def my_function():#creating a function
  print("Hi!I'm from Vidya Jyothi Institute of Technology")
my_function()#calling a function
print("..................")
#function with parameters
def my_dept(clg):
  print("Hi!I'm from "+clg)
my_dept("Vidya Jyothi Institute of Technology")
my_dept("CBIT")
my_dept("CVSR")
```

```
Hi!I'm from Vidya Jyothi Institute of Technology
..................
Hi!I'm from Vidya Jyothi Institute of Technology
Hi!I'm from CBIT
Hi!I'm from CVSR
```

```
#1)list
my_list = ['a','b','c','d','e']
print(my_list)
print(my_list[2])
print(my_list[-1])
print(my_list[1:3])
print(my_list[ :3])
print(my_list[1:])
print(my_list[:-3])
print(my_list[-1:])
print(type(my_list))
print("_____________")
#2)Tuple
tup=('sita','geeta','shiva','priya','ravi','sweety')
tup1=(5,6)
tup2=(1,8)
print(tup)
print(tup[0])
print(tup[1:4])
print(tup[:3])
print(tup[2:])
print(tup[:-2])
print(tup[-1:])
if (tup1>tup2):print( "tup1 is bigger")
else: print("tup2 is bigger")
print(type(tup))
print("_____________")
#3)DICTIONARY
my_dict = {'name': 'balu', 'age': 19}
print(my_dict)
print(my_dict['name'])
print(my_dict.get('age'))
my_dict.update({'Branch':'Aie','marks':20})
print(my_dict)
del my_dict['marks']
print(my_dict)
my_dict['marks']='20'
print(my_dict)
my_dict.pop("marks")
```

```
print(my_dict)
print(type(my_dict))
```

```
['a', 'b', 'c', 'd', 'e']
c
e
['b', 'c']
['a', 'b', 'c']
['b', 'c', 'd', 'e']
['a', 'b']
['e']
<class 'list'>
_______________
('sita', 'geeta', 'shiva', 'priya', 'ravi', 'sweety')
sita
('geeta', 'shiva', 'priya')
('sita', 'geeta', 'shiva')
('shiva', 'priya', 'ravi', 'sweety')
('sita', 'geeta', 'shiva', 'priya')
('sweety',)
tup1 is bigger
<class 'tuple'>
_______________
{'name': 'balu', 'age': 19}
balu
19
{'name': 'balu', 'age': 19, 'Branch': 'Aie', 'marks': 20}
{'name': 'balu', 'age': 19, 'Branch': 'Aie'}
{'name': 'balu', 'age': 19, 'Branch': 'Aie', 'marks': '20'}
{'name': 'balu', 'age': 19, 'Branch': 'Aie'}
<class 'dict'>
```

```
#classes and objects without constructor
print("WITHOUT CONSTRUCTOR")
class person:
    pass
male=person()#object1
female=person()#object2
male.name='balu'
male.age=19
female.name='balaa'
female.age=19
print(male.__dict__)
print(female.age)
print(female.__dict__)
#classes and objects with constructor
print("WITH CONSTRUCTOR")
class person:
    def __init__(self,name,age):
        self.name=name
        self.age=age
male=person('balu',19)
female=person('balaa',19)
male.lastname='chinna'
print(male.__dict__)#dict keyword gives all d info about object
print(female.age)
print(female.__dict__)
```

```
WITHOUT CONSTRUCTOR
```

```
      {'name': 'balu', 'age': 19}
      19
      {'name': 'balaa', 'age': 19}
      WITH CONSTRUCTOR
      {'name': 'balu', 'age': 19, 'lastname': 'chinna'}
      19
      {'name': 'balaa', 'age': 19}


x1=int(input("enter x1 : "))
x2=int(input("enter x2 : "))
y1=int(input("enter y1 : "))
y2=int(input("enter y2 : "))
result= ((((x2 - x1 )**2) + ((y2-y1)**2) )**3.5)
print("distance between",(x1,x2),"and",(y1,y2),"is : ",result)

      enter x1 : 3
      enter x2 : 6
      enter y1 : 4
      enter y2 : 8
      distance between (3, 6) and (4, 8) is :  78125.0


#1.)INHERITENCE
#1.a)Single inheritence
print("SINGLE INHERITENCE")
class parent:
    def fun1(self):
        print("function 1")
class child(parent):
    def fun2(self):
        print("function 2")
ob=child()
ob.fun1()
#1.b)Multiple inheritence
print("MULTIPLE INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2:
    def fun2(self):
        print("function 2")
class child(parent1,parent2):
    def fun3(self):
        print("function 3")
ob=child()
ob.fun1()
ob.fun2()
#1.c)Multilevel inheritence
print("MULTILEVEL INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2(parent1):
    def fun2(self):
        print("function 2")
class child(parent2):
    def fun3(self):
        print("function 3")
ob=child()
```

```
ob.fun1()
ob.fun2()
#1.d)hierarichal inheritence
print("HIERARICHAL INHERITENCE")
class parent:
    def fun1(self):
        print("function 1")
class child1(parent):
    def fun2(self):
        print("function 2")
class child2(parent):
    def fun3(self):
        print("function 3")
ob=child2()
ob.fun1()
ob.fun3()
#1.e)hybrid inheritence
print("HYBRID INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2:
    def fun4(self):
        print("function 4")
class child1(parent1):
    def fun2(self):
        print("function 2")
class child2(parent1,parent2):
    def fun3(self):
        print("function 3")
ob=child2()
ob.fun1()
ob.fun4()
print("__________________")
#2.)polymorphism
print("POLYMORPHISM")
#overriding a variable
class parent:
    name="balaa"
class child(parent):
    name="chinna"#overriding is done here
ob=child()
ob.name
#overriding a method
class parent:
    def name(self):
        return 0
class child(parent):
    def name(self):
        return 1#overriding is done here
ob=child()
print(ob.name())
ob=parent()
print(ob.name())
```

```
SINGLE INHERITENCE
function 1
MULTIPLE INHERITENCE
function 1
function 2
```

```
      MULTILEVEL INHERITENCE
      function 1
      function 2
      HIERARICHAL INHERITENCE
      function 1
      function 3
      HYBRID INHERITENCE
      function 1
      function 4
      ____________________
      POLYMORPHISM
      1
      0
```

```
#Implement static and instance methods
print("STATIC AND INSTANCE METHODS")
class Shape:
    def rectArea(self,l,b):
        return(l*b)
    @classmethod
    def sqArea(cls,s):
        return(s*s)
    @staticmethod
    def CircleArea(r):
        return(3.14*r*r)
s=Shape()
print("Area of Rectangle = ",s.rectArea(2,4))
#calling a class method
print("Area of Square = ",Shape.sqArea(3))
#calling a static method
print("Area of Circle = ",Shape.CircleArea(2))
print("_____________________")
#Abstract class
print("ABSTRACT CLASS")
from abc import ABC,abstractmethod
class Animal(ABC):
  @abstractmethod
  def eat(self):
    pass
class Tiger(Animal):
  def eat(self):
    print("non-veg")
class Cow(Animal):
  def eat(self):
    print("veg")
t=Tiger()
t.eat()
c=Cow()
c.eat()
```

```
      STATIC AND INSTANCE METHODS
      Area of Rectangle =  8
      Area of Square =  9
      Area of Circle =  12.56
      ____________________
      ABSTRACT CLASS
      non-veg
      veg
```

```
x1=int(input("enter x1 : "))
x2=int(input("enter x2 : "))
y1=int(input("enter y1 : "))
y2=int(input("enter y2 : "))
result= ((((x2 - x1 )**2) + ((y2-y1)**2) )**0.5)
print("distance between",(x1,x2),"and",(y1,y2),"is : ",result)
```

```
enter x1 : 6
enter x2 : 8
enter y1 : 2
enter y2 : 6
distance between (6, 8) and (2, 6) is :  4.47213595499958
```

```
#1.)INHERITENCE
#1.a)Single inheritence
print("SINGLE INHERITENCE")
class parent:
    def fun1(self):
        print("function 1")
class child(parent):
    def fun2(self):
        print("function 2")
ob=child()
ob.fun1()
#1.b)Multiple inheritence
print("MULTIPLE INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2:
    def fun2(self):
        print("function 2")
class child(parent1,parent2):
    def fun3(self):
        print("function 3")
ob=child()
ob.fun1()
ob.fun2()
#1.c)Multilevel inheritence
print("MULTILEVEL INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2(parent1):
    def fun2(self):
        print("function 2")
class child(parent2):
    def fun3(self):
        print("function 3")
ob=child()
ob.fun1()
ob.fun2()
#1.d)hierarichal inheritence
print("HIERARICHAL INHERITENCE")
class parent:
    def fun1(self):
        print("function 1")
class child1(parent):
    def fun2(self):
```

```
        print("function 2")
class child2(parent):
    def fun3(self):
        print("function 3")
ob=child2()
ob.fun1()
ob.fun3()
#1.e)hybrid inheritence
print("HYBRID INHERITENCE")
class parent1:
    def fun1(self):
        print("function 1")
class parent2:
    def fun4(self):
        print("function 4")
class child1(parent1):
    def fun2(self):
        print("function 2")
class child2(parent1,parent2):
    def fun3(self):
        print("function 3")
ob=child2()
ob.fun1()
ob.fun4()
print("__________________")
#2.)polymorphism
print("POLYMORPHISM")
#overriding a variable
class parent:
    name="balu"
class child(parent):
    name="chinna"#overriding is done here
ob=child()
ob.name
#overriding a method
class parent:
    def name(self):
        return 0
class child(parent):
    def name(self):
        return 1#overriding is done here
ob=child()
print(ob.name())
ob=parent()
print(ob.name())
```

```
SINGLE INHERITENCE
function 1
MULTIPLE INHERITENCE
function 1
function 2
MULTILEVEL INHERITENCE
function 1
function 2
HIERARICHAL INHERITENCE
function 1
function 3
HYBRID INHERITENCE
function 1
function 4
```

```
_____________________
POLYMORPHISM
1
0
```

```
file=open("/content/cleans.csv.zip","r")
print("Name of the file:",file.name)
print("File open status:",file.closed)
print("File opened mode:",file.mode)
print("Content in the file:")
file.close()
print("File open status:",file.closed)
```

```
import os
f1=open("text2.txt","w")
f1.write("Welcome to Artificial intelligence")
f1.close()
print("New file \"text2.txt\"is created in the path.")
f1=open("text2.txt","a")
f1.write(" Happy Learning")
f1.close()
f1=open("text2.txt","r")
print("Filepointer position when it was opened: ", f1.tell())
f1.seek(3)
print("Filepointer position after seek(): ", f1.tell())
print("Contents in file: " , f1.read())
print("Filepointer position after read(): ", f1.tell())
f1.close()
os.rename("text2.txt","text3.txt")
os.remove("text3.txt")
```

```
New file "text2.txt"is created in the path.
Filepointer position when it was opened:  0
Filepointer position after seek():  3
Contents in file:  come to Artificial intelligence Happy Learning
Filepointer position after read():  49
```

```
try:
    num=int(input("Enter numerator: "))
    den=int(input("enter denominator: "))
    q=num/den
    print("Qutioent= ",q)
except ZeroDivisionError:
    print("Denominator is zero")
except ValueError:
    print("Please eneter only digits")
finally:
    print("this is finally block. this will be executed all the time..");
```

```
Enter numerator: 54
```

```
      enter denominator: 6
      Qutioent=  9.0
      this is finally block. this will be executed all the time..
```

```
from sklearn.cluster import KMeans
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from matplotlib import pyplot as plt
%matplotlib inline
```

```
from math import log2

def entropy(class0, class1):
    return -(class0 * log2(class0) + class1 * log2(class1))

class0 = 14 /20
class1 = 8 /20
# calculate entropy before the change
s_entropy = entropy(class0, class1)
print('Dataset Entropy: %.3f bits' % s_entropy)

# split 1 (split via value1)
s1_class0 = 6 / 8
s1_class1 = 2 / 8
# calculate the entropy of the first group
s1_entropy = entropy(s1_class0, s1_class1)
print('Group1 Entropy: %.3f bits' % s1_entropy)

# split 2  (split via value2)
s2_class0 = 5 / 10
s2_class1 = 5 / 10
# calculate the entropy of the second group
s2_entropy = entropy(s2_class0, s2_class1)
print('Group2 Entropy: %.3f bits' % s2_entropy)

# calculate the information gain
gain = s_entropy - (8/20 * s1_entropy + 12/20 * s2_entropy)
print('Information Gain: %.3f bits' % gain)
```

```
      Dataset Entropy: 0.889 bits
      Group1 Entropy: 0.811 bits
      Group2 Entropy: 1.000 bits
      Information Gain: -0.036 bits
```

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


class Perceptron():
    def __init__(self,eta=0.01,n_iter=10):
        self.eta = eta
        self.n_iter = n_iter

    def fit(self,X,y):
        self.w_ = np.zeros(1+X.shape[1])
        self.errors_ = []
```

```
        h = 0
        for _ in range(self.n_iter):
            errors = 0
            for xi, target in zip(X,y):
                update = self.eta*(target-self.predict(xi))
                self.w_[1:]+=update*xi
                self.w_[0]+=update
                errors+=int(update != 0.0)
                h+=1
            self.errors_.append(errors)
        print("value of h: ",h)
        return self

    def net_input(self,X):
        return np.dot(X,self.w_[1:])+self.w_[0]

    def predict(self,X):
        return np.where(self.net_input(X)>=0.0,1,-1)


data = pd.read_csv("/content/iris.csv",header=None)
y = data.iloc[0:100,4].values
y = np.where(y == 'Iris-setosa',-1,1)
X = data.iloc[0:100,[0,2]].values



plt.scatter(X[:50,0],X[:50,1],color='red',marker='o',label='setosa')
plt.scatter(X[50:100,0],X[50:100,1],color='blue',marker='x',label='versicolor')
plt.xlabel('petal length')
plt.ylabel('sepal length')
plt.legend(loc='upper left')
plt.show()
```


```
ppn = Perceptron(eta=0.1, n_iter=10)
ppn.fit(X,y)
plt.plot(range(1, len(ppn.errors_)+1), ppn.errors_,marker='o')
plt.xlabel('Epochs')
plt.ylabel("Number of misclassifications")
plt.show()
```

```
      value of h:  1000
```

```
import numpy as np


from numpy import linalg
matrix = np.array([
                [10,20,30,40],
                [34,54,36,90],
                [48,83,94,23],
                [23,45,36,89]
])

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

```
      Inverse of matrix:

      [[ 1.33295973e-01  1.79088944e-01 -2.08950444e-02 -2.35609639e-01]
       [-1.65086863e-01 -1.08618205e-01  2.64190217e-02  1.77207589e-01]
       [ 7.29245056e-02  2.60187335e-03 -8.00576415e-05 -3.53854775e-02]
       [ 1.95260588e-02  7.58546153e-03 -7.92570651e-03 -3.16227684e-03]]



      Pseudo Inverse of matrix:

      [[ 1.33295973e-01  1.79088944e-01 -2.08950444e-02 -2.35609639e-01]
       [-1.65086863e-01 -1.08618205e-01  2.64190217e-02  1.77207589e-01]
       [ 7.29245056e-02  2.60187335e-03 -8.00576415e-05 -3.53854775e-02]
       [ 1.95260588e-02  7.58546153e-03 -7.92570651e-03 -3.16227684e-03]]
```

```
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
```

```
coeff_matrix = np.array([
                         [4, -8],
                         [6, 5]
])

const_matrix = np.array([2, -1])

solution = linalg.solve(coeff_matrix, const_matrix)

print(solution)
```

```
      [ 0.02941176 -0.23529412]
```

```
coeff_matrix = np.array([
                         [5, -8],
                         [4, 5]
])

const_matrix = np.array([1, -6])

linalg.lstsq(coeff_matrix, const_matrix)
```

```
      /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: FutureWarning:
      To use the future default and silence this warning we advise to pass `rcond=Non

      (array([-0.75438596, -0.59649123]),
       array([], dtype=float64),
       2,
       array([9.81024968, 5.81024968]))
```

```
X = np.array([
              [10,20,30,40],
              [23,45,89,90],
              [2,5,4,9],
              [9,8,0,1]
])

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]
 [23 45 89 90]
 [ 2  5  4  9]
 [ 9  8  0  1]]


U:
[[-0.36980609  0.27872639 -0.75618115 -0.46232577]
 [-0.92569789 -0.15882766  0.32292537  0.11651779]
 [-0.07215797  0.17759612 -0.43664915  0.8789712 ]
 [-0.03346788  0.93034671  0.36502985 -0.00938707]]


Sigma:
[147.19709745  11.55464586   6.80590319   0.42935323]


V*
[[-0.17279308 -0.33751385 -0.63703652 -0.67112698]
 [ 0.68046572  0.58487621 -0.43822073 -0.05337436]
 [ 0.33462749  0.02130674  0.63302793 -0.69774358]
 [-0.62859638  0.7372586   0.03771874 -0.24473179]]

Original Matrix
[[ 1.00000000e+01  2.00000000e+01  3.00000000e+01  4.00000000e+01]
 [ 2.30000000e+01  4.50000000e+01  8.90000000e+01  9.00000000e+01]
 [ 2.00000000e+00  5.00000000e+00  4.00000000e+00  9.00000000e+00]
 [ 9.00000000e+00  8.00000000e+00 -2.14477821e-15  1.00000000e+00]]
```

```
matrix = np.random.randint(100,200,24).reshape(3,8)
print("Matrix:")
print(matrix)
print("\n\n")
print("Transpose of Matrix:")
print(matrix.transpose())
```

```
Matrix:
[[133 127 111 138 142 165 121 179]
 [157 109 183 120 152 192 181 135]
 [175 131 147 114 157 144 152 103]]



Transpose of Matrix:
[[133 157 175]
 [127 109 131]
 [111 183 147]
 [138 120 114]
 [142 152 157]
 [165 192 144]
 [121 181 152]
 [179 135 103]]
```

```
matrix = np.array([
```

```
                    [23,0,0,0],
                    [0,90,0,0],
                    [0,0,87,0],
                    [0,0,0,76]
])

print("Matrix: ")
print(matrix)

evl, evc = linalg.eig(matrix)

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

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

```
Matrix:
[[23  0  0  0]
 [ 0 90  0  0]
 [ 0  0 87  0]
 [ 0  0  0 76]]

Eigen values:
[23. 90. 87. 76.]

Eigen vectors:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

```
matrix = np.random.randint(100,140,24).reshape(6,4)
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))
```

```
[[131 102 122 100]
 [100 132 113 136]
 [110 134 124 120]
 [134 128 121 101]
 [129 122 123 129]
 [108 132 115 128]]


sorted:
[100 100 101 102 108 110 113 115 120 121 122 122 123 124 128 128 129 129
 131 132 132 134 134 136]


sorted row wise:
```

```
[[100 102 113 100]
 [108 122 115 101]
 [110 128 121 120]
 [129 132 122 128]
 [131 132 123 129]
 [134 134 124 136]]


sorted column wise:
[[100 102 122 131]
 [100 113 132 136]
 [110 120 124 134]
 [101 121 128 134]
 [122 123 129 129]
 [108 115 128 132]]
```

```
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.        ]
```

```
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.]]
```

```
a = np.mgrid[1:10,5:20]
print(a)
```

```
[[[ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1]
  [ 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2]
  [ 3  3  3  3  3  3  3  3  3  3  3  3  3  3  3]
  [ 4  4  4  4  4  4  4  4  4  4  4  4  4  4  4]
  [ 5  5  5  5  5  5  5  5  5  5  5  5  5  5  5]
```

```
 [ 6  6  6  6  6  6  6  6  6  6  6  6  6  6  6]
 [ 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7]
 [ 8  8  8  8  8  8  8  8  8  8  8  8  8  8  8]
 [ 9  9  9  9  9  9  9  9  9  9  9  9  9  9  9]]

[[ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
 [ 5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]]]
```

```
a = np.ogrid[1:10,5:20]
print(a)
```

```
[array([[1],
        [2],
        [3],
        [4],
        [5],
        [6],
        [7],
        [8],
        [9]]), array([[ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
```

```
a = np.array([
              [10,20,30],
              [40,50,60],
              [70,80,90]
])

b = np.array([
              [2,3,4],
              [500,69,94],
              [45,25,65]
])
x = np.concatenate((a,b), axis=0)
print(x)

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

```
[[ 10  20  30]
 [ 40  50  60]
 [ 70  80  90]
 [  2   3   4]
 [500  69  94]
 [ 45  25  65]]



[[ 10  20  30   2   3   4]
```

```
       [ 40  50  60 500  69  94]
       [ 70  80  90  45  25  65]]
```

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

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

```
a = np.array([
              [4],
              [5],
              [6]
])

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

```
      [4 5 6]
```

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

result = integrate.quad(integrand, 0, 2, args=(3,9))

print(result)
```

```
      (36.0, 3.9968028886505635e-13)
```

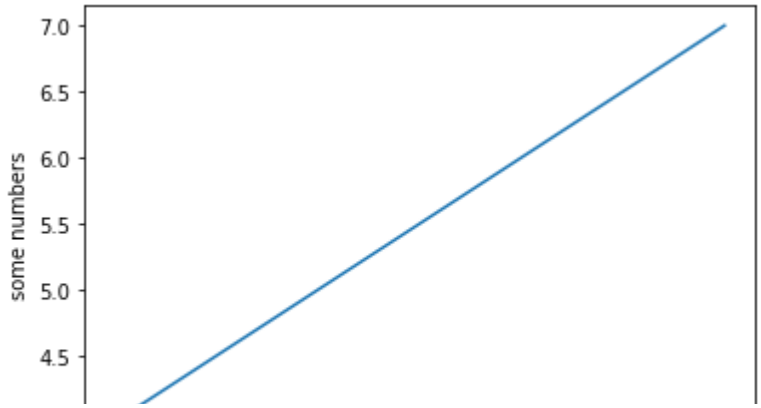
```
#Series
print("CREATING SERIES")
import pandas as pd
import numpy as np
t=pd.Series([1,3,5,6,8])#in series indexing will be there
s=pd.Series([1,3,5,6,8],index=(10,11,12,13,14))#we can mention inex of our wish also
print(t)
print(s)
print(s[:3])
print(s[:-3])
print(s[-3:])
print(".............")
#coverting numpy array to series
```

```
data=np.array(['a','b','c','d'])
s=pd.Series(data)
t=pd.Series(data,index=[100,101,102,103])
print(s)
print(t)
print(".............")
#converting dictionary to series
data={'a':0.,'b':1.,'c':2.}
s=pd.Series(data,dtype=int)
t=pd.Series(data,index=[100,101,102,103])
print(s)
print(t)
print("__________")
#Dataframe
print("CREATING DATAFRAME")
data=[['divya',18],['tinku',16],['shrav',17]]
print(data)
print('......')
x=pd.DataFrame(data)#normal dataframe
print(x)
print('.......')
x=pd.DataFrame(data,index=['row1','row2','row3'])#dataframe with index changes
print(x)
print('......')
x=pd.DataFrame(data,columns=['Name','Age'])#dataframe with col names changed
print(x)
print('......')
x=pd.DataFrame(data,index=['row1','row2','row3'],columns=['Name','Age'])#dataframe with bo
print(x)
print('......')
print(x.shape)#shape
print('.......')
print(x.T)#transpose of matrix
print('.......')
print(x.dtypes)
print("__________")
#Adding a new column using pandas
d = {'one' : pd.Series([1, 2, 3], index=['a', 'b', 'c']),
   'two' : pd.Series([1, 2, 3, 4], index=['a', 'b', 'c', 'd'])}
df = pd.DataFrame(d)
print(df)
print('...........')
print ("Adding a new column by passing as Series:")
df['three']=pd.Series([10,20,30],index=['a','b','c'])
df['four'] = pd.Series([100, 200, 300])
print(df)
print('.....')
print ("Adding a new column using the existing columns in DataFrame:")
df['five']=df['one']*df['three']
print(df)
```

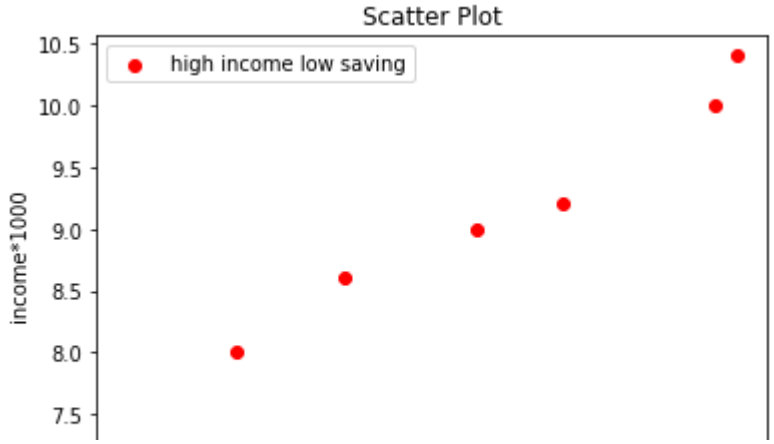
```
CREATING SERIES
0    1
1    3
2    5
3    6
4    8
dtype: int64
10    1
11    3
```

```
12    5
13    6
14    8
dtype: int64
10    1
11    3
12    5
dtype: int64
10    1
11    3
dtype: int64
12    5
13    6
14    8
dtype: int64
.............
0    a
1    b
2    c
3    d
dtype: object
100    a
101    b
102    c
103    d
dtype: object
.............
a    0
b    1
c    2
dtype: int64
100   NaN
101   NaN
102   NaN
103   NaN
dtype: float64
__________
CREATING DATAFRAME
[['divya', 18], ['tinku', 16], ['shrav', 17]]
......
       0   1
0  divya  18
1  tinku  16
2  shrav  17
.......
          0   1
row1  divya  18
row2  tinku  16
row3  shrav  17
......
```

```
import matplotlib.pyplot as plt

x= [4,5,6,7]
plt.plot(x)

plt.ylabel('some numbers')
plt.show()
```

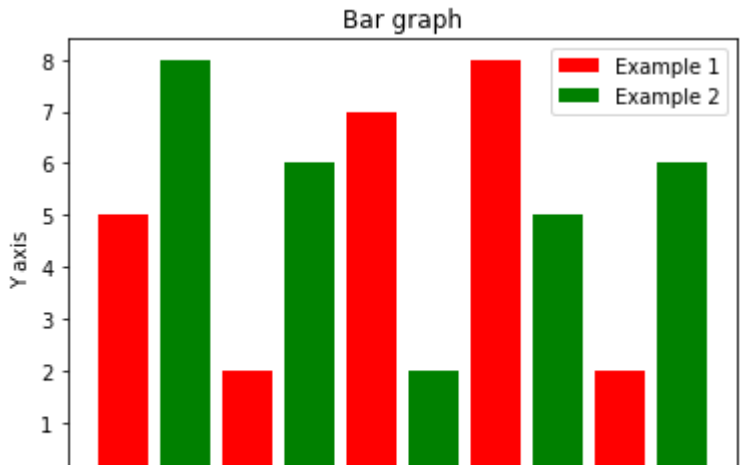



```
x = [1,1.5,2,2.6,3,3.7,3.8]
y = [7.2,8,8.6,9,9.2,10,10.4]
plt.scatter(x,y, label='high income low saving',color='r')

plt.xlabel('saving*100')
plt.ylabel('income*1000')

plt.title('Scatter Plot')

plt.legend()
plt.grid(False)
plt.show()
```

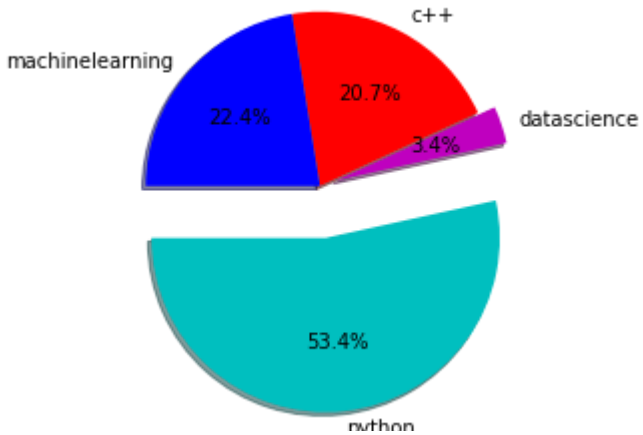



```
x1 = [1,3,5,7,9]
y1 = [5,2,7,8,2]
x2 = [2,4,6,8,10]
y2 = [8,6,2,5,6]
plt.bar(x1, y1, label="Example 1",color='r')
plt.bar(x2, y2, label="Example 2", color="g")
plt.legend()
plt.xlabel("X axis")
plt.ylabel("Y axis")
```

```
plt.title("Bar graph")
plt.show()
```

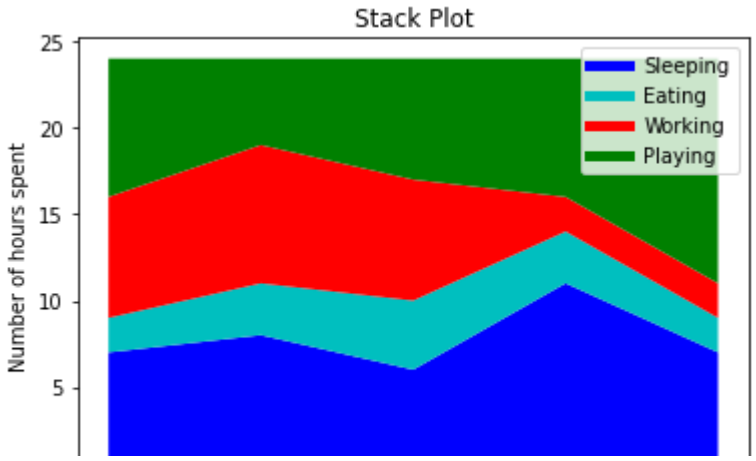



```
slices = [31,2,12,13]
activities = ['python','datascience','c++','machinelearning']
cols = ['c','m','r','b']

plt.pie(
    slices,
    labels=activities,
    colors=cols,
    startangle=180,
    shadow=True,
    explode=(0.3, 0.1, 0, 0),
    autopct="%2.1f%%"
)
plt.show()
```

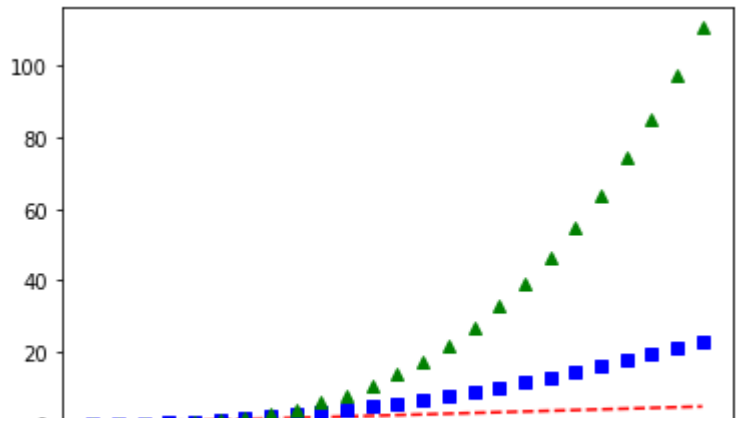



```
days = [1,2,3,4,5]

sleeping =[7,8,6,11,7]
eating = [2,3,4,3,2]
working =[7,8,7,2,2]
playing = [8,5,7,8,13]
```

```
plt.plot([],[],color='b', label='Sleeping', linewidth=5)
plt.plot([],[],color='c', label='Eating', linewidth=5)
plt.plot([],[],color='r', label='Working', linewidth=5)
plt.plot([],[],color='g', label='Playing', linewidth=5)

plt.stackplot(days, sleeping,eating,working,playing, colors=['b','c','r','g'])

plt.xlabel('Day')
plt.ylabel('Number of hours spent')
plt.title('Stack Plot')
plt.legend()
plt.show()
```



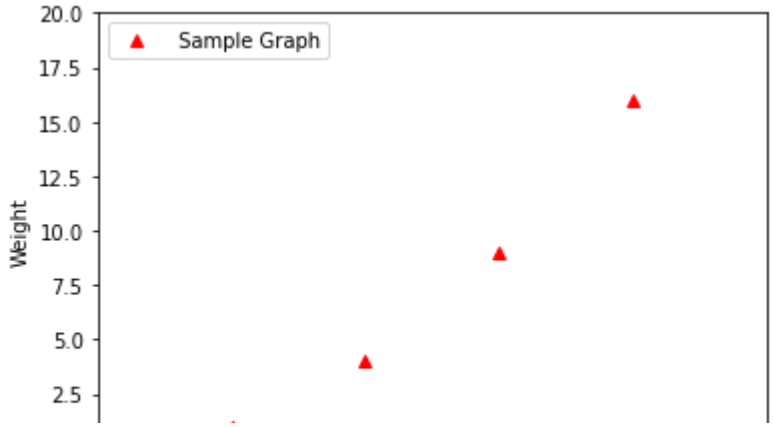

```
import numpy as np
t = np.arange(0., 5., 0.2)

#Two or more lines
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.show()
```



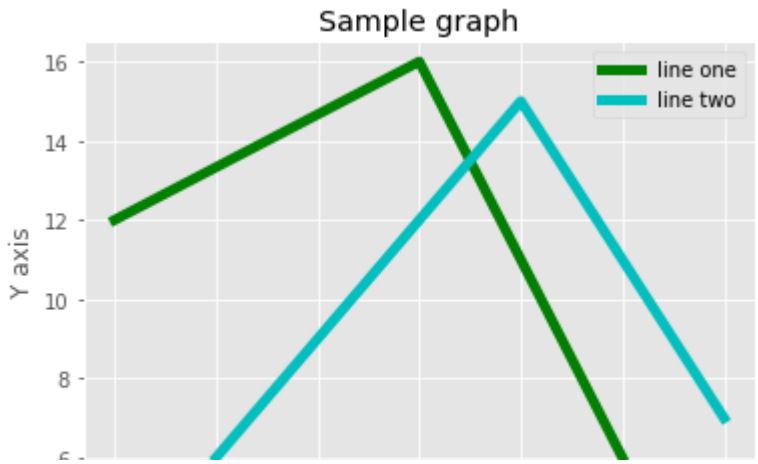

```
#With legend
plt.plot([1, 2, 3, 4], [1, 4, 9, 16], 'r^', label=" Sample Graph")
plt.legend(loc="upper left")
plt.xlabel(' Height')
```

```
plt.ylabel('Weight')
plt.axis([0, 5, 0, 20])
```

```
(0.0, 5.0, 0.0, 20.0)
```

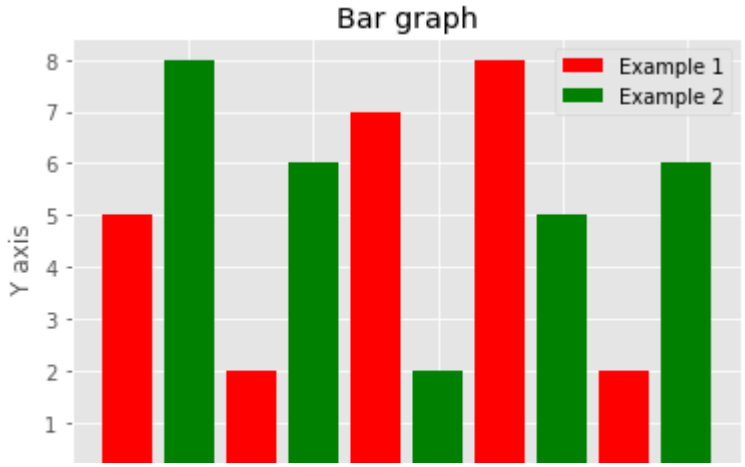



```
#Adding style to graph
from matplotlib import style
style.use("ggplot")#this is used for grid look
x = [5, 8, 10]
y = [12, 16, 6]
x2 = [6, 9, 11]
y2 = [6, 15, 7]
plt.plot(x, y, 'g', label="line one", linewidth=5)
plt.plot(x2, y2, 'c', label="line two", linewidth=5)
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Sample graph")
plt.legend() # Bydeafult upper right
#plt.grid(True, color="b")#this is used for color of lines in grid
plt.grid(True)
plt.show()
```

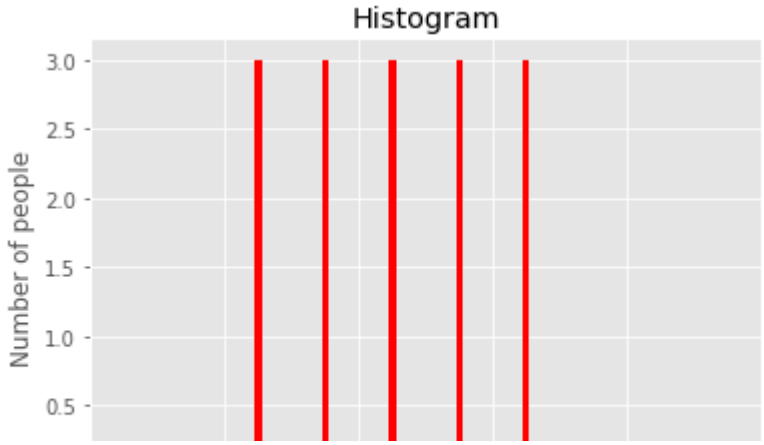



```
# bar graph for categorical values
x1 = [1,3,5,7,9]
```

```
y1 = [5,2,7,8,2]
x2 = [2,4,6,8,10]
y2 = [8,6,2,5,6]
plt.bar(x1, y1, label="Example 1",color='r')
plt.bar(x2, y2, label="Example 2", color="g")
plt.legend()
plt.xlabel("X axis")
plt.ylabel("Y axis")
plt.title("Bar graph")
plt.show()
```

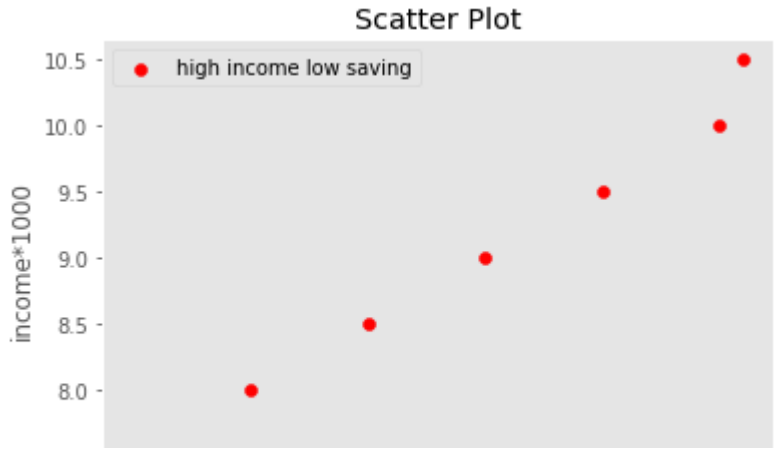



```
# histogram for numeric values
population_age = [20,20,20,30,30,30,40,40,40,50,50,50,60,60,60]
bins = [0,10,20,30,40,50,60,70,80,90,100]
plt.hist(population_age, bins, color="r", rwidth=0.1) # histogram is representing frequnec
plt.xlabel('age groups')
plt.ylabel('Number of people')
plt.title('Histogram')
plt.show()
```




```
#scatter plot
x = [1,1.5,2,2.5,3,3.5,3.6]
```

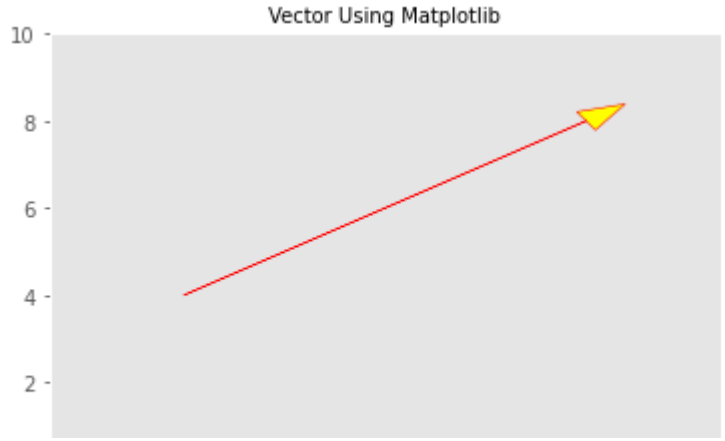
```
y = [7.5,8,8.5,9,9.5,10,10.5]
plt.scatter(x,y, label='high income low saving',color='r')

plt.xlabel('saving*100')
plt.ylabel('income*1000')

plt.title('Scatter Plot')

plt.legend()
plt.grid(False)
plt.show()
```



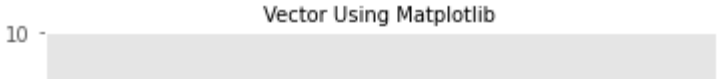

```
#sine ans cosine graphs
x = np.arange(0, 3 * np.pi, 0.1)
print(x)
y_sin = np.sin(x)
y_cos = np.cos(x)
# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

```
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8 2.9 3.  3.1 3.2 3.3 3.4 3.5
 3.6 3.7 3.8 3.9 4.  4.1 4.2 4.3 4.4 4.5 4.6 4.7 4.8 4.9 5.  5.1 5.2 5.3
 5.4 5.5 5.6 5.7 5.8 5.9 6.  6.1 6.2 6.3 6.4 6.5 6.6 6.7 6.8 6.9 7.  7.1
 7.2 7.3 7.4 7.5 7.6 7.7 7.8 7.9 8.  8.1 8.2 8.3 8.4 8.5 8.6 8.7 8.8 8.9]
```

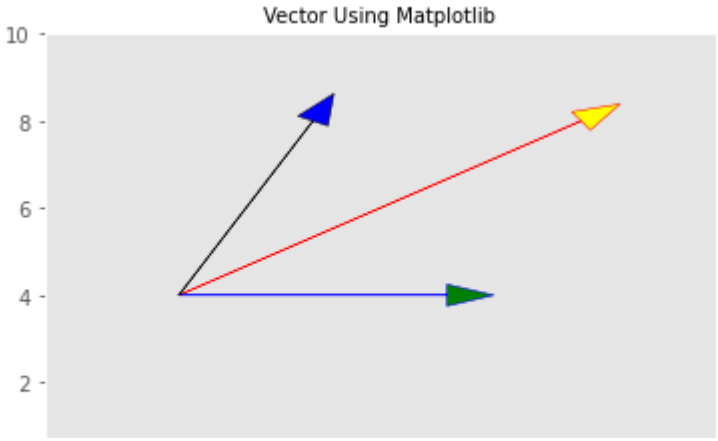
```
#Plotting Of Vector In 2D
import matplotlib.pyplot as plt
ax = plt.axes()
#(x,y,dx,dy):the starting point of vectors is (x,y) and the end point of vector is(x+dy,x+
#fc=face color,ec=edge color
ax.arrow(2.0, 4.0, 6.0, 4.0, head_width=0.5, head_length=0.7, fc='yellow', ec='red')
plt.grid()
plt.xlim(0,10)
plt.ylim(0,10)
plt.title('Vector Using Matplotlib',fontsize=10)
plt.show()
```




```
#Plotting Of 2 Vectors In 2D
ax = plt.axes()
ax.arrow(2.0, 4.0, 6.0, 4.0, head_width=0.5, head_length=0.7, fc='yellow', ec='red')
ax.arrow(2.0, 4.0, 4.0, 0.0, head_width=0.5, head_length=0.7, fc='green', ec='blue')
plt.grid()
plt.xlim(0,10)
plt.ylim(0,10)
plt.title('Vector Using Matplotlib',fontsize=10)
plt.show()
```




```
#Plotting Of 3 Vectors In 2D
ax = plt.axes()
ax.arrow(2.0, 4.0, 6.0, 4.0, head_width=0.5, head_length=0.7, fc='yellow', ec='red')
ax.arrow(2.0, 4.0, 4.0, 0.0, head_width=0.5, head_length=0.7, fc='green', ec='blue')
ax.arrow(2.0, 4.0, 2.0, 4.0, head_width=0.5, head_length=0.7, fc='blue', ec='black')
plt.grid()
plt.xlim(0,10)
plt.ylim(0,10)
plt.title('Vector Using Matplotlib',fontsize=10)
plt.show()
```




```
#Plotting of single vector in 3D
fig=plt.figure()
ax=plt.axes(projection='3d')
ax.set_xlim([-1,10])
ax.set_ylim([-10,10])
ax.set_zlim([1,10])
start=[0,0,0] #starting of the vector
end=[2,3,1]   # Ending of the vector
ax.quiver(start[0],start[1],start[2],end[0],end[1],end[2])
```

```
<mpl_toolkits.mplot3d.art3d.Line3DCollection at 0x7f8cfe345d10>
```