```
a = [4,6,8,7,9]
b = sorted(a)
a.sort()
print(a)
print(b)
```

```
[4, 6, 7, 8, 9]
[4, 6, 7, 8, 9]
```

```
import numpy as np
A = np.arange(1,13)
print(A)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
```

```
import numpy as np
A = np.arange(1,13).reshape(3,4)
print(A)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

```
A = np.linspace(1,13,12).reshape(2,6)
print(A)
```

```
[[ 1.          2.09090909  3.18181818  4.27272727  5.36363636  6.45454545]
 [ 7.54545455  8.63636364  9.72727273 10.81818182 11.90909091 13.        ]]
```

```
A = np.linspace(1,13,18).reshape(3,6)
print(A)
```

```
[[ 1.          1.70588235  2.41176471  3.11764706  3.82352941  4.52941176]
 [ 5.23529412  5.94117647  6.64705882  7.35294118  8.05882353  8.76470588]
 [ 9.47058824 10.17647059 10.88235294 11.58823529 12.29411765 13.        ]]
```

```
B = np.array([45,12,78,96,32,58,47,50,14]).reshape(3,3,1)
print(B)
```

```
[[[45]
  [12]
  [78]]

 [[96]
  [32]
  [58]]

 [[47]
  [50]
  [14]]]
```

```
B = np.array([45,12,78,96,32,58,47,50,14])
B.reshape(3,3,1)#changes the shape of the array
#reshape - change needs saving in another variable
print(B)
```

```
[45 12 78 96 32 58 47 50 14]
```

```
#Arthematic Operations on the matrix
A = np.array([45,87,96,2,3]).reshape(5,1)
B = np.array([21,32,44,56,78]).reshape(1,5)
C = A-B
print("A - Arrary:\n",A)
print("\n")
print("B - Arrary:\n",B)
print("\n")
print("diff:\n",C)
print("\n")
D = A+B
print("sum:\n",D)
print("\n")
X = A*B
print("mul:",X)
print("\n")
Y = A/B
print("div:\n",Y)
print("\n")
H = A@B
print("matrix-mul:\n",H)
print("\n")
S = A.dot(B)
print("maximum :\n",S.max(axis=0))
print("\n")
print("minimum :\n",S.min(axis=1))
```

```
A - Arrary:
 [[45]
 [87]
 [96]
 [ 2]
 [ 3]]


B - Arrary:
 [[21 32 44 56 78]]


diff:
 [[ 24  13   1 -11 -33]
 [ 66  55  43  31   9]
 [ 75  64  52  40  18]
 [-19 -30 -42 -54 -76]
 [-18 -29 -41 -53 -75]]


sum:
 [[ 66  77  89 101 123]
 [108 119 131 143 165]
 [117 128 140 152 174]
 [ 23  34  46  58  80]
 [ 24  35  47  59  81]]


mul: [[ 945 1440 1980 2520 3510]
 [1827 2784 3828 4872 6786]
 [2016 3072 4224 5376 7488]
 [  42   64   88  112  156]
 [  63   96  132  168  234]]


div:
 [[2.14285714 1.40625    1.02272727 0.80357143 0.57692308]
 [4.14285714 2.71875    1.97727273 1.55357143 1.11538462]
 [4.57142857 3.         2.18181818 1.71428571 1.23076923]
 [0.0952381  0.0625     0.04545455 0.03571429 0.02564103]
 [0.14285714 0.09375    0.06818182 0.05357143 0.03846154]]


matrix-mul:
 [[ 945 1440 1980 2520 3510]
 [1827 2784 3828 4872 6786]
 [2016 3072 4224 5376 7488]
 [  42   64   88  112  156]
 [  63   96  132  168  234]]


maximum :
 [2016 3072 4224 5376 7488]
```

```
    minimum :
     [ 945 1827 2016   42   63]
```

```
#Joining The Array's
A = np.array([12,45,78,36]).reshape(2,2)
B = np.array([2,4,5,6]).reshape(2,2)
print("A-Array:\n",A)
print("\n")
print("B-Array:\n",B)
print("\n")
print("After vertical stacking:")
print(np.vstack((A,B)))
print("\n")
print("After Horizontally stacking:")
print(np.hstack((A,B)))
print("\n")
print(np.stack((A,B),axis=0))
```

```
    A-Array:
     [[12 45]
     [78 36]]


    B-Array:
     [[2 4]
     [5 6]]


    After vertical stacking:
    [[12 45]
     [78 36]
     [ 2  4]
     [ 5  6]]


    After Horizontally stacking:
    [[12 45  2  4]
     [78 36  5  6]]


    [[[12 45]
      [78 36]]

     [[ 2  4]
      [ 5  6]]]
```

```
A = np.arange(30).reshape(2,3,5)
print(A)
print("\n After dstack\n")
print(np.dstack(A))
```

```
    [[[ 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 27 28 29]]]

     After dstack

    [[[ 0 15]
      [ 1 16]
      [ 2 17]
      [ 3 18]
      [ 4 19]]

     [[ 5 20]
      [ 6 21]
      [ 7 22]
      [ 8 23]
      [ 9 24]]

     [[10 25]
      [11 26]
      [12 27]
      [13 28]
      [14 29]]]
```

```
#Splitting Array's
A = np.arange(25).reshape(5,5)
print("A-Array:\n",A)
print("\n")
print("After V-Splittting")
print(np.vsplit(A,5))
print("\n")
print("After H-Splittting")
print(np.hsplit(A,5))
print("\n")
print(np.vsplit(A,(2,4)))
```

```
A-Array:
 [[ 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]]


After V-Splittting
[array([[0, 1, 2, 3, 4]]), array([[5, 6, 7, 8, 9]]), array([[10, 11, 12, 13, 14]]), array([[15, 16, 17, 18, 19]]), array([[20, 21, 22, 2


After H-Splittting
[array([[ 0],
       [ 5],
       [10],
       [15],
       [20]]), array([[ 1],
       [ 6],
       [11],
       [16],
       [21]]), array([[ 2],
       [ 7],
       [12],
       [17],
       [22]]), array([[ 3],
       [ 8],
       [13],
       [18],
       [23]]), array([[ 4],
       [ 9],
       [14],
       [19],
       [24]])]


[array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]]), array([[10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]]), array([[20, 21, 22, 23, 24]])]
```

```
A = np.array([23,43,54,23,45,57,67,78,58,99]).reshape(2,5)
print("A-Array:\n",A)
print("\n")
print("max:\n",A.argmax(axis = 0))
print("\n")
print("min:\n",A.argmin(axis = 0))
```

```
A-Array:
 [[23 43 54 23 45]
 [57 67 78 58 99]]


max:
 [1 1 1 1 1]


min:
 [0 0 0 0 0]
```

```
#Stats
A = np.array([23,45,67,89,9,44])
print("Mean               : ",np.mean(A))
print("Median             : ",np.median(A))
print("Variance           : ",np.var(A))
print("Standard Deviation : ",np.std(A))
```

```
     Mean               :  46.166666666666664
     Median             :  44.5
     Variance           :  698.8055555555555
     Standard Deviation :  26.43493059486927
```

```
#Trigonometry
A = np.pi
print("In radius :",A)
print("In Degree :",np.rad2deg(A))

     In radius : 3.141592653589793
     In Degree : 180.0
```

```
A = np.array([np.pi/4,np.pi/3,np.pi/2,np.pi])
print("In radius :",A)
print("In Degree :",np.rad2deg(A))

     In radius : [0.78539816 1.04719755 1.57079633 3.14159265]
     In Degree : [ 45.  60.  90. 180.]
```

```
print(np.sin(A))

     [7.07106781e-01 8.66025404e-01 1.00000000e+00 1.22464680e-16]
```

```
print(np.cos(A))

     [ 7.07106781e-01  5.00000000e-01  6.12323400e-17 -1.00000000e+00]
```

```
print(np.tan(A))

     [ 1.00000000e+00  1.73205081e+00  1.63312394e+16 -1.22464680e-16]
```

```
np.arccos(1)

     0.0
```

```
np.arcsin(1)

     1.5707963267948966
```

```
np.arctan(1)

     0.7853981633974483
```

```
#Hypotneuse
A = 6
B = 4
print(np.hypot(A,B))

     7.211102550927978
```

```
#Searching in array
A = np.array([1,2,3,4,5,6,7,8,1])
print(np.where(A==1))

     (array([0, 8]),)
```

```
#Searching even numbers in Array
print(np.where(A%2==0))

     (array([0, 0, 1, 1, 2, 2]), array([1, 3, 1, 3, 1, 3]))
```

```
#Search sort
A = np.array([12,34,55,78,87])
print(np.searchsorted(A,99))

     5
```

Linear aglgebra and universal functions

```
A = np.array([23,45,12,43,67])
B = np.array([90,98,12,65,62])
print("A-Array             :",A)
print("B-Array             :",B)
print("ADD                 :",np.add(A,B))
print("SUBTRACT            :",np.subtract(A,B))
print("Remainder           :",np.mod(A,B))
print("Divide              :",np.divide(A,B))
print("mod/div             :",np.divmod(A,B))
print("Diff of A           :",np.diff(A))
print("Diff of B           :",np.diff(B))
print("Union of A,B        :",np.union1d(A,B))
print("Set of A,B          :",np.setdiff1d(A,B))
print("Intersection of A,B :",np.intersect1d(A,B))
```

```
     A-Array             : [23 45 12 43 67]
     B-Array             : [90 98 12 65 62]
     ADD                 : [113 143  24 108 129]
     SUBTRACT            : [-67 -53   0 -22   5]
     Remainder           : [23 45  0 43  5]
     Divide              : [0.25555556 0.45918367 1.         0.66153846 1.08064516]
     mod/div             : (array([0, 0, 1, 0, 1]), array([23, 45,  0, 43,  5]))
     Diff of A           : [ 22 -33  31  24]
     Diff of B           : [  8 -86  53  -3]
     Union of A,B        : [12 23 43 45 62 65 67 90 98]
     Set of A,B          : [23 43 45 67]
     Intersection of A,B : [12]
```

Rounding

```
A = np.trunc([-7.4668,7.3698])
print("Truncate :",A)
B = np.round([5.56,-6.84])
print("Round    :",B)
C = np.fix([-6.456,7.325])
print("Fix      :",C)
D = np.around(5.46695,5)
print("Around   :",D)
E = np.floor(5.46)
print("Floor    :",E)
F = np.ceil(5.46)
print("Ceiling  :",F)
```

```
     Truncate : [-7.  7.]
     Round    : [ 6. -7.]
     Fix      : [-6.  7.]
     Around   : 5.46695
     Floor    : 5.0
     Ceiling  : 6.0
```

```
A = np.array([5,78,28,19,174])
print("Cumulative Sum     :",np.cumsum(A))
print("Cumulative Product :",np.cumprod(A))
print("LCM                :",np.lcm.reduce(A))
print("GCD                :",np.gcd.reduce(A))
```

```
     Cumulative Sum     : [  5  83 111 130 304]
     Cumulative Product : [       5      390    10920   207480 36101520]
     LCM                : 3008460
     GCD                : 1
```

```
A = np.array([[1,2],[3,4]])
print("Inverse Of A Matrix :",np.linalg.inv(A))
```

```
     Inverse Of A Matrix : [[-2.   1. ]
      [ 1.5 -0.5]]
```

```
#Default limit 0 to 0.99
from numpy import random as rd
A = rd.randint(50,size=6)
print(A)
print(rd.rand(20))
```

```
     [44 17 17 21 25  2]
     [0.05557043 0.78546557 0.51199185 0.1682204  0.09096053 0.71872048
```

```
 0.60153688 0.81132752 0.83314016 0.24055208 0.68441129 0.62581491
 0.95339604 0.95035976 0.34667616 0.19583781 0.82820582 0.30148297
 0.07345499 0.99413116]
```

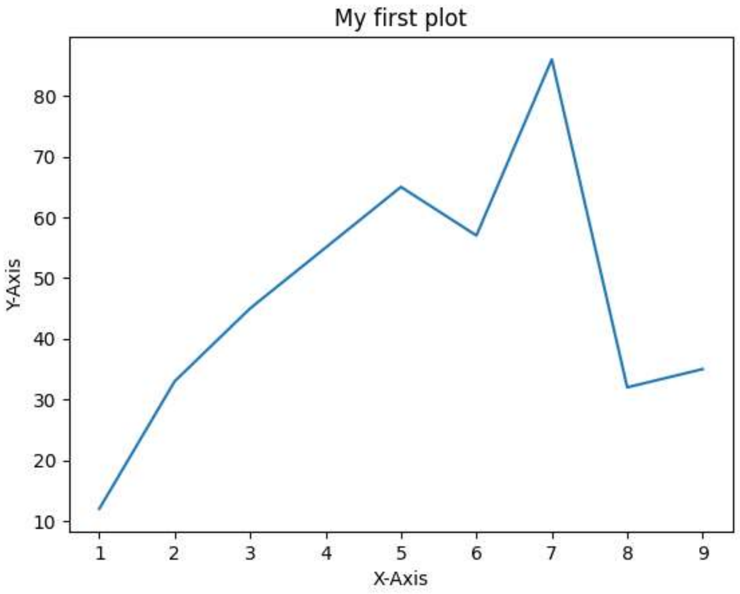
```
A = np.array([1,2,3])
B = np.array([4,5,6])
print("Inner Of A,B :",np.inner(A,B))
print("Outer Of A,B :",np.outer(A,B))
print("Cross Of A,B :",np.cross(A,B))
```

```
Inner Of A,B : 32
Outer Of A,B : [[ 4  5  6]
 [ 8 10 12]
 [12 15 18]]
Cross Of A,B : [-3  6 -3]
```

Ploting A List

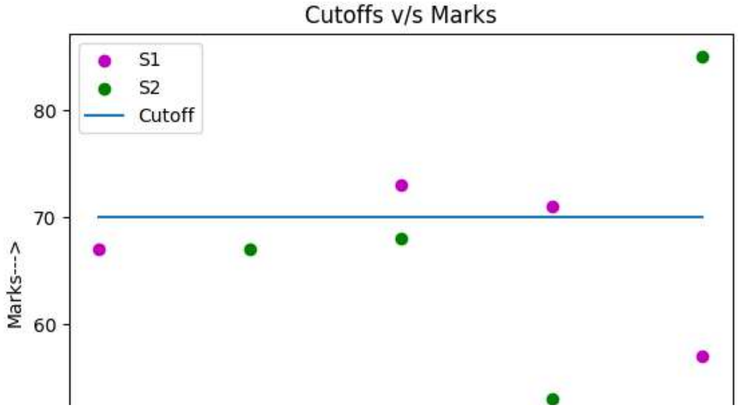
```
from matplotlib import pyplot as plt
X = [1,2,3,5,6,7,8,9]
Y = [12,33,45,65,57,86,32,35]
plt.plot(X,Y)
plt.title("My first plot")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
```

```
Text(0, 0.5, 'Y-Axis')
```



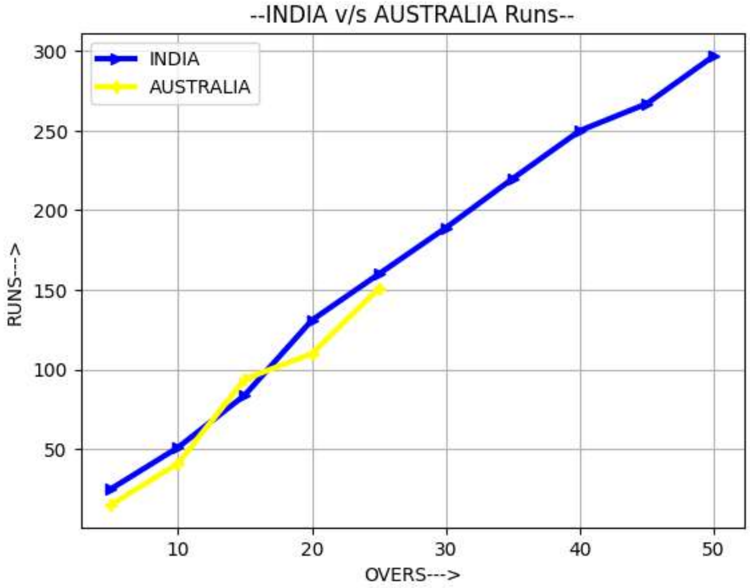
```
#There are five sudjects psychology,nanotechnology,medicine,onchology and anthropology.
#Two students have applied to do masters in any of these five fields with a line plot show in which field they can excel
#The Cutoff for each subject is 70
S1 = [67,45,73,71,57]
S2 = [43,67,68,53,85]
Marks = [70,70,70,70,70]
subs = ['psychology','nanotechnology','medicine','onchology','anthropology']
plt.scatter(subs,S1,label='S1',color='m')
plt.scatter(subs,S2,label='S2',color='green')
plt.plot(subs,Marks,label='Cutoff')
plt.title("Cutoffs v/s Marks")
plt.xlabel("Subjects--->")
plt.ylabel("Marks--->")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7ad8b14573d0>
```



```
#An India v/s Australia match is on live india has given target 268 for Australia.The runs for every 5 overs is given as follows
#India = 25,51,84,131,160,189,220,250,267,297
#Australia has completed 25 overs
#Australia = 15,41,94,110,151
#plot overs v/s runs for both the countries with as a blue line and Australia as a yellow line give proper labels and legend the Marker for
#Take overs from arange function create runs as arrays'''
import numpy as np
from matplotlib import pyplot as plt
india = np.array([25,51,84,131,160,189,220,250,267,297])
overs1=np.arange(5,51,5)
overs2=np.arange(5,26,5)
aus=np.array([15,41,94,110,151])
plt.title("--INDIA v/s AUSTRALIA Runs--")
plt.grid()
plt.xlabel("OVERS--->")
plt.ylabel("RUNS--->")
plt.plot(overs1,india,linewidth=3,marker='>',color='blue',label="INDIA")
plt.plot(overs2,aus,linewidth=3,marker='P',color="yellow",label="AUSTRALIA")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7ad8b1434d90>
```


--INDIA v/s AUSTRALIA Runs--
INDIA
AUSTRALIA
RUNS--->
OVERS--->