1.Write a python program for the following Data visualization techniques.

a) Line Chart

b) Bar chart

c) Pie chart

d)Scatter plot

e) Histogram

f) Box plot

**a) Line chart:**

#LINE GRAPH

import matplotlib.pyplot as plt

x=[10,20,30,40,50]

y=[65,98,170,220,310]

plt.plot(x,y,color='r',marker="D",mec="k",mfc="w",linestyle="dashed",

linewidth="2")

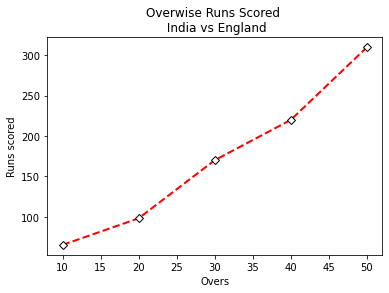
plt.xlabel("Overs")

plt.ylabel("Runs scored")

plt.title("Overwise Runs Scored \n India vs England")

plt.show()

**Output:**



**b)Bar Graph**

#Bar Graph

import matplotlib.pyplot as plt

import numpy as np

Overs=["1-10","11-20","21-30","31-40","41-50"]

Runs=[65,55,70,50,80]

plt.bar(Overs,Runs,color=["r","g","b","c","k"],width=[0.1,0.2,0.3,0.4,0.5])

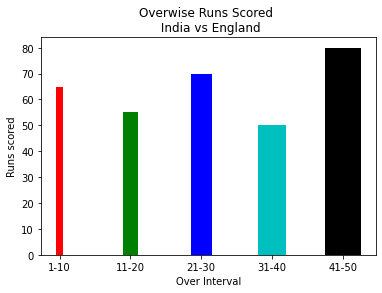
plt.xlabel("Over Interval")

plt.ylabel("Runs scored")

plt.title("Overwise Runs Scored \n India vs England")

plt.show()

**Output:**



**c) Pie chart**

#pie chart

import matplotlib.pyplot as plt

import numpy as np

branches=['cse','it','eee','ece','csebs']

data=[90,40,15,45,21]

fig=plt.figure(figsize=(9,10))

mycolors=['red','green','cyan','black','hotpink']

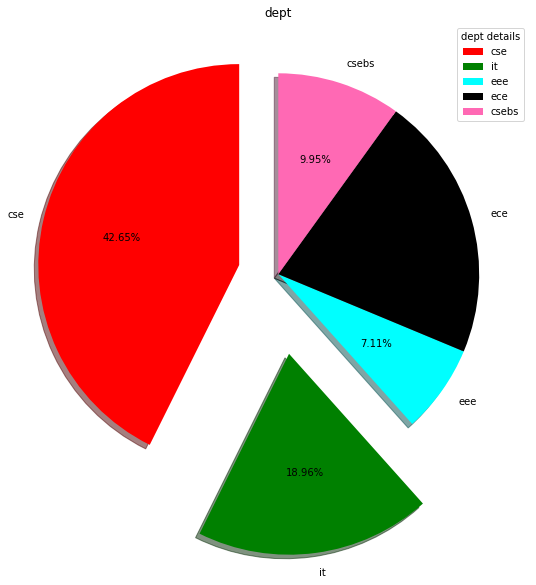
plt.pie(data,labels=branches,startangle=90,explode=[0.2,0.4,0,0,0],shadow=True,colors=mycolors,autopct="%0.2f%%")

plt.title("dept")

plt.legend(title="dept details",loc="upper right")

plt.show()

Output:



**d) Scatter plot**

#Scatter plot

import matplotlib.pyplot as plt

import numpy as np

x=[2,2.5,3,3.5,4.5,4.7,5.0]

y=[7.5,8,8.5,9,9.5,10,10.5]

x1=[9,8.5,9,9.5,10,10.5,12]

y1=[3,3.5,4.7,4,4.5,5,5.2]

plt.scatter(x,y,label="low savings high income",color='g',marker="D",linewidth=4,edgecolor="k")

plt.scatter(x1,y1,label="high savings low income",color='r',marker="s",linewidth=2,edgecolor="g")

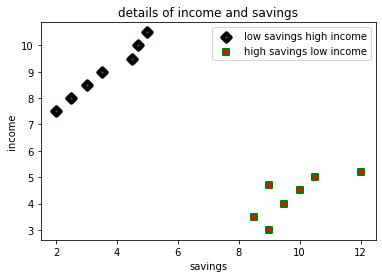
plt.xlabel("savings")

plt.ylabel("income")

plt.title("details of income and savings")

plt.legend()

plt.show()



**e) Histogram**

#Histogram

import matplotlib.pyplot as plt

age=[22,32,35,45,55,14,26,19,56,44,48,33,38,28]

years=[0,10,20,30,40,50,60]

plt.hist(age,bins=years,color='magenta',histtype='bar',edgecolor='black',rwidth=0.6)

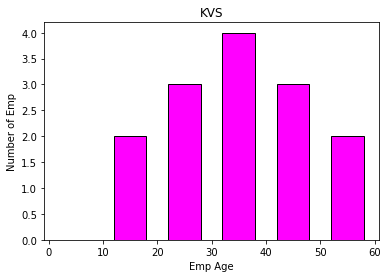
plt.xlabel("Emp Age")

plt.ylabel("Number of Emp")

plt.title("KVS")

plt.show()

Output:



f)Box plot:

#Box plot

import matplotlib.pyplot as plt

import matplotlib.pyplot as plt

value1 = [82,76,24,40,67,62,75,78,71,32,98,89,78,67,72,82,87,66,56,52]

value2=[62,5,91,25,36,32,96,95,3,90,95,32,27,55,100,15,71,11,37,21]

value3=[23,89,12,78,72,89,25,69,68,86,19,49,15,16,16,75,65,31,25,52]

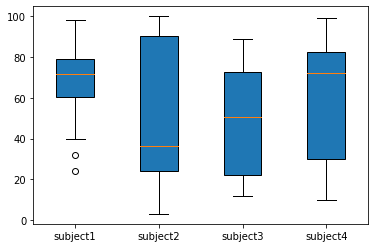
value4=[59,73,70,16,81,61,88,98,10,87,29,72,16,23,72,88,78,99,75,30]

box\_plot\_data=[value1,value2,value3,value4]

plt.boxplot(box\_plot\_data,patch\_artist=True,labels=['subject1','subject2','subject3','subject4'])

plt.show()

Output:



2. Write python programs to implement indexing, slicing and splitting, iterating a list.

**import** numpy **as** np

array1d **=** np**.**array([1, 2, 3, 4, 5, 6,7,8])

array2d **=** np**.**array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

array3d **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(array1d)

print("---------Indexing operation ----------")

print("---------Indexing on 1D----------")

print(array1d[0]) *# Get first value*

print(array1d[**-**1]) *# Get last value*

print(array1d[3]) *# Get 4th value from first*

print(array1d[**-**5]) *# Get 5th value from last*

print(array1d[[0,2,1]]) *#Get multiple values from array*

print("---------Indexing on 2D----------")

print(array2d)

print(array2d[0, 0]) *# Get first row first col*

print(array2d[0, 1]) *# Get first row second col*

print(array2d[0, 2]) *# Get first row third col*

print(array2d[0, 1]) *# Get first row second col*

print(array2d[**-**1, 1]) *# Get second row second col*

print(array2d[2, **-**2]) *# Get third row second col*

print(array2d[[0,1,2],[1,0,2]])*#Get multiple values from array*

print(array2d[[0,1,2],[**-**1,0,**-**3]])*#Get multiple values from array*

print(array2d[0,2]**+**array2d[1,1])*#adding the elements on array*

print("---------Indexing on 3D----------")

print(array3d)

print(array3d[0, 1, 2])

print(array3d[0, **-**2, 1])

print(array3d[0, 0, 2])

print(array3d[[0,1,0],[1,0,1],[1,0,2]])

print(array3d[[0,1,0,1],[1,0,1,**-**1],[1,0,2,**-**3]])

print(array3d[1,**-**1,2]**+**array3d[0, **-**1, **-**3])

print("---------Slicing operation----------")

print("---------Slicing on 1D----------")

print(array1d)

print(array1d[4:]) *# From index 4 to last index*

print(array1d[:4]) *# From index 0 to 4 index*

print(array1d[4:7]) *# From index 4(included) up to index 7(excluded)*

print(array1d[:**-**1]) *# Excluded last element*

print(array1d[:**-**2]) *# Up to second last index(negative index)*

print(array1d[::**-**1]) *# From last to first in reverse order(negative step)*

print(array1d[::**-**2]) *# All odd numbers in reversed order*

print(array1d[**-**2::**-**2]) *# All even numbers in reversed order*

print(array1d[::]) *# All elements*

array1d[0]**=**10 *#modifying the element in array*

print(array1d)

print("---------Slicing on 2D----------")

array2d **=** np**.**array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

print(array2d)

print(array2d[:, 0:2]) *# 2nd and 3rd col*

print(array2d[1:3, 0:3]) *# 2nd and 3rd row*

print(array2d[**-**1::**-**1, **-**1::**-**1]) *# Reverse an array*

print(array2d[0:2, 0:2:2]) *#start,stop,step*

print(array2d[1,0:2:2])

print("---------Slicing on 3D----------")

array3d **=** np**.**array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

print(array3d)

print(array3d[0,1,2])

print(array3d[1,**-**1,2])

print(array3d[1,0:1,0:2])

print(array3d[0:2,0:2,0:2])

print(array3d[0:2,1,**-**3])

print("---------Splitting operations----------")

print("---------Splitting on 1D----------")

print(array1d)

print("-------splitting on Horizontal---------")

np**.**hsplit(array1d,2)*#split divides into equal division*

*#or*

np**.**split(array1d,2)

*#np.split(array1d,3) error*

np**.**array\_split(array1d,3)

*#splitting on vertical on 1D not possible it requires 2D or More Dimensional*

print("---------Splitting on 2D----------")

print(array2d)

print("---------splitting on horizontal on 2D (column-wise)-----")

print(np**.**hsplit(array2d,3))*#split divides into equal division*

*#or*

print(np**.**split(array2d,3,axis**=**1))

*#np.split(array2d,2) error use array\_split()*

np**.**array\_split(array2d,2,axis**=**1)

np**.**array\_split(array2d,4,axis**=**1)

print("-"**\***10)

print("---------splitting on vertical on 2D (row-wise)---------")

print(np**.**vsplit(array2d,3))*#split divides into equal division*

*#or*

print(np**.**split(array2d,3,axis**=**0))

print(np**.**array\_split(array2d,2,axis**=**0))

print(np**.**array\_split(array2d,4,axis**=**0))

print("--------------splitting on 3D--------------")

print(array3d)

print("-------------split on horizontal----------------")

print(np**.**hsplit(array3d,2))*#split divides into equal division*

*#or*

print(np**.**split(array3d,2,axis**=**1))

*#print(np.split(array3d,3,axis=1)) error not in equal division so go for array\_split()*

print(np**.**array\_split(array3d,3,axis**=**1))

print(np**.**array\_split(array3d,4,axis**=**1))

print("---------split on vertical------------ ")

print(np**.**vsplit(array3d,2))*#split divides into equal division*

*#or*

print(np**.**split(array3d,2,axis**=**0))

*#print(np.split(array3d,3,axis=1)) error not in equal division so go for array\_split()*

print(np**.**array\_split(array3d,3,axis**=**0))

print(np**.**array\_split(array3d,4,axis**=**0))

print("-----Iterating operations------")

print("-----Iterating Arrays Using nditer()--------")

print("-----Iterating operations on 1D------")

print(array1d)

**for** x **in** np**.**nditer(array1d,flags**=**['buffered'],op\_dtypes**=**['S']):

print(x)

print("-----Iterating operations on 2D------")

print(array2d)

**for** x **in** np**.**nditer(array2d[:,::2]):

print(x)

print("-----Iterating operations on 3D------")

print(array3d)

**for** x **in** np**.**nditer(array3d[:,::2,2]):

print(x)

print("---------modifying array values----------")

**for** x **in** np**.**nditer(array3d):

x**=**5**\***x

print(x)

print("----------using ndenumerate()--------")

for idx,x in np.ndenumerate(array3d):

print(idx,x)

Output:

[1 2 3 4 5 6 7 8]

---------Indexing operation ----------

---------Indexing on 1D----------

1

8

4

4

[1 3 2]

---------Indexing on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

1

2

3

2

8

8

[2 4 9]

[3 4 7]

8

---------Indexing on 3D----------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

6

2

3

[5 7 6]

[ 5 7 6 10]

16

---------Slicing operation----------

---------Slicing on 1D----------

[1 2 3 4 5 6 7 8]

[5 6 7 8]

[1 2 3 4]

[5 6 7]

[1 2 3 4 5 6 7]

[1 2 3 4 5 6]

[8 7 6 5 4 3 2 1]

[8 6 4 2]

[7 5 3 1]

[1 2 3 4 5 6 7 8]

[10 2 3 4 5 6 7 8]

---------Slicing on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

[[1 2]

[4 5]

[7 8]]

[[4 5 6]

[7 8 9]]

[[9 8 7]

[6 5 4]

[3 2 1]]

[[1]

[4]]

[4]

---------Slicing on 3D----------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

6

12

[[7 8]]

[[[ 1 2]

[ 4 5]]

[[ 7 8]

[10 11]]]

[ 4 10]

---------Splitting operations----------

---------Splitting on 1D----------

[10 2 3 4 5 6 7 8]

-------splitting on Horizontal---------

---------Splitting on 2D----------

[[1 2 3]

[4 5 6]

[7 8 9]]

---------splitting on horizontal on 2D (column-wise)-----

[array([[1],

[4],

[7]]), array([[2],

[5],

[8]]), array([[3],

[6],

[9]])]

[array([[1],

[4],

[7]]), array([[2],

[5],

[8]]), array([[3],

[6],

[9]])]

----------

---------splitting on vertical on 2D (row-wise)---------

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

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

[array([[1, 2, 3],

[4, 5, 6]]), array([[7, 8, 9]])]

[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]]), array([], shape=(0, 3), dtype=int32)]

--------------splitting on 3D--------------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

-------------split on horizontal----------------

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]])]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]])]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]]), array([], shape=(2, 0, 3), dtype=int32)]

[array([[[1, 2, 3]],

[[7, 8, 9]]]), array([[[ 4, 5, 6]],

[[10, 11, 12]]]), array([], shape=(2, 0, 3), dtype=int32), array([], shape=(2, 0, 3), dtype=int32)]

---------split on vertical------------

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]])]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]])]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]]), array([], shape=(0, 2, 3), dtype=int32)]

[array([[[1, 2, 3],

[4, 5, 6]]]), array([[[ 7, 8, 9],

[10, 11, 12]]]), array([], shape=(0, 2, 3), dtype=int32), array([], shape=(0, 2, 3), dtype=int32)]

-----Iterating operations------

-----Iterating Arrays Using nditer()--------

-----Iterating operations on 1D------

[10 2 3 4 5 6 7 8]

b'10'

b'2'

b'3'

b'4'

b'5'

b'6'

b'7'

b'8'

-----Iterating operations on 2D------

[[1 2 3]

[4 5 6]

[7 8 9]]

1

3

4

6

7

9

-----Iterating operations on 3D------

[[[ 1 2 3]

[ 4 5 6]]

[[ 7 8 9]

[10 11 12]]]

3

9

---------modifying array values----------

5

10

15

20

25

30

35

40

45

50

55

60

----------using ndenumerate()--------

(0, 0, 0) 1

(0, 0, 1) 2

(0, 0, 2) 3

(0, 1, 0) 4

(0, 1, 1) 5

(0, 1, 2) 6

(1, 0, 0) 7

(1, 0, 1) 8

(1, 0, 2) 9

(1, 1, 0) 10

(1, 1, 1) 11

(1, 1, 2) 12

3.Write a python program on Indexing, Slicing, Splitting & Iterating on Dataset.

DATASET.csv file

65.78331 ,112.9925 ,345.89 ,234.23

71.51521 ,136.4873 ,346.89, 235.23

69.39874 ,153.0269 ,347.89 ,236.23

68.2166 ,142.3354 ,348.89, 237.23

67.78781 ,144.2971 ,349.89 ,238.23

68.69784 ,123.3024, 350.89, 239.23

69.80204 ,141.4947 ,351.89 ,240.23

70.01472, 136.4623 ,352.89, 241.23

67.90265 ,112.3723 ,353.89, 242.23

66.78236, 120.6672, 354.89 ,243.23

66.48769, 127.4516 ,355.89 ,244.23

67.62333, 114.143 ,356.89 ,245.23

68.30248, 125.6107 ,357.89 ,246.23

67.11656, 122.4618 ,358.89, 247.23

68.27967, 116.0866 ,359.89 ,248.23

71.0916, 139.9975, 360.89 ,249.23

66.461 ,129.5023, 361.89, 250.23

68.64927 ,142.9733 ,362.89, 251.23

71.23033, 137.9025 ,363.89 ,252.23

67.13118, 124.0449 ,364.89, 253.23

67.83379, 141.2807 ,365.89, 254.23

68.87881, 143.5392 ,366.89, 255.23

63.48115, 97.90191 ,367.89, 256.23

68.42187, 129.5027 ,368.89, 257.23

67.62804, 141.8501, 369.89 ,258.23

67.20864, 129.7244, 370.89 ,259.23

70.84235, 142.4235, 371.89 ,260.23

67.49434, 131.5502 ,372.89, 261.23

66.53401, 108.3324, 373.89 ,262.23

65.44098, 113.8922, 374.89 ,263.23

69.5233, 103.3016, 375.89 ,264.23

65.8132,120.7536, 376.89 ,265.23

67.8163, 125.7886, 377.89 ,266.23

70.59505,136.2225, 378.89 ,267.23

71.80484 ,140.1015, 379.89, 268.23

69.20613, 128.7487, 380.89, 269.23

66.80368, 141.7994, 381.89 ,270.23

67.65893, 121.2319, 382.89, 271.23

67.80701, 131.3478, 383.89 ,272.23

64.04535, 106.7115, 384.89, 273.23

68.57463, 124.3598, 385.89 ,274.23

65.18357, 124.8591, 386.89 ,275.23

69.65814, 139.6711, 387.89, 276.23

67.96731, 137.3696, 388.89 ,277.23

65.98088, 106.4499, 389.89 ,278.23

68.67249, 128.7639, 390.89 ,279.23

66.88088, 145.6837, 391.89, 280.23

67.69868, 116.819, 392.89, 281.23

69.82117, 143.6215, 393.89 ,282.23

69.08817, 134.9325, 394.89 ,283.23

69.91479, 147.0219, 395.89, 284.23

67.33182, 126.3285, 396.89, 285.23

70.26939,125.4839, 397.89 ,286.23

69.10344, 115.7084, 398.89, 287.23

65.38356, 123.4892, 399.89 ,288.23

70.18447, 147.8926, 400.89 ,289.23

70.40617, 155.8987, 401.89 ,290.23

66.54376, 128.0742, 402.89 ,291.23

66.36418, 119.3701, 403.89, 292.23

67.537, 133.8148, 404.89, 293.23

#Load the dataset and perform indexing,Slicing,Splitting & Iterating

import numpy as np

dataset =np.genfromtxt('C:/Users/pc/Desktop/abcd.csv',delimiter=',')

print(dataset)

print("The rows & columns on dataset:", dataset.shape)

print("---------Perform Indexing operation on Dataset------ ")

print(dataset[0]) # index single element in outermost dimension

print(dataset[-1]) # index in reversed order in outermost dimension

print(dataset[1, 1]) # index single element in two-dimensional data

print(dataset[-1, -1]) # index in reversed order in two-dimensional data

print(dataset[8,3])

print(dataset[[0,7,3,34],[1,-1,2,3]])

print(dataset[0,3]+dataset[2,-1])

print(dataset[dataset>70])#booloean operation

print("---------Perform Slicing operation on Dataset------ ")

print(dataset[1:3]) # rows 1 and 2

print(dataset[:2, :2]) # 2x2 subset of the data

print(dataset[-1, ::-1]) # last row with elements reversed

print(dataset[-5:-1, :6:2])# last 4 rows,every other element up to index 6

print("---------Perform Splitting operation on Dataset------ ")

print("-----------Horizontal splitting-----------")

print(np.hsplit(dataset,2)) # split horizontally in 3 equal lists

print("-----------vertical splitting-----------")

print(np.vsplit(dataset, 2)) # split vertically in 2 equal lists

print("---------Perform Iterating operation on Dataset------ ")

print("----------using nditer()--------")

for x in np.nditer(dataset):

print(x)

print("----------using ndenumerate()--------")

for idx,x in np.ndenumerate(dataset):

print(idx,x)

Output:

[[ 65.78331 112.9925 345.89 234.23 ]

[ 71.51521 136.4873 346.89 235.23 ]

[ 69.39874 153.0269 347.89 236.23 ]

[ 68.2166 142.3354 348.89 237.23 ]

[ 67.78781 144.2971 349.89 238.23 ]

[ 68.69784 123.3024 350.89 239.23 ]

[ 69.80204 141.4947 351.89 240.23 ]

[ 70.01472 136.4623 352.89 241.23 ]

[ 67.90265 112.3723 353.89 242.23 ]

[ 66.78236 120.6672 354.89 243.23 ]

[ 66.48769 127.4516 355.89 244.23 ]

[ 67.62333 114.143 356.89 245.23 ]

[ 68.30248 125.6107 357.89 246.23 ]

[ 67.11656 122.4618 358.89 247.23 ]

[ 68.27967 116.0866 359.89 248.23 ]

[ 71.0916 139.9975 360.89 249.23 ]

[ 66.461 129.5023 361.89 250.23 ]

[ 68.64927 142.9733 362.89 251.23 ]

[ 71.23033 137.9025 363.89 252.23 ]

[ 67.13118 124.0449 364.89 253.23 ]

[ 67.83379 141.2807 365.89 254.23 ]

[ 68.87881 143.5392 366.89 255.23 ]

[ 63.48115 97.90191 367.89 256.23 ]

[ 68.42187 129.5027 368.89 257.23 ]

[ 67.62804 141.8501 369.89 258.23 ]

[ 67.20864 129.7244 370.89 259.23 ]

[ 70.84235 142.4235 371.89 260.23 ]

[ 67.49434 131.5502 372.89 261.23 ]

[ 66.53401 108.3324 373.89 262.23 ]

[ 65.44098 113.8922 374.89 263.23 ]

[ 69.5233 103.3016 375.89 264.23 ]

[ 65.8132 120.7536 376.89 265.23 ]

[ 67.8163 125.7886 377.89 266.23 ]

[ 70.59505 136.2225 378.89 267.23 ]

[ 71.80484 140.1015 379.89 268.23 ]

[ 69.20613 128.7487 380.89 269.23 ]

[ 66.80368 141.7994 381.89 270.23 ]

[ 67.65893 121.2319 382.89 271.23 ]

[ 67.80701 131.3478 383.89 272.23 ]

[ 64.04535 106.7115 384.89 273.23 ]

[ 68.57463 124.3598 385.89 274.23 ]

[ 65.18357 124.8591 386.89 275.23 ]

[ 69.65814 139.6711 387.89 276.23 ]

[ 67.96731 137.3696 388.89 277.23 ]

[ 65.98088 106.4499 389.89 278.23 ]

[ 68.67249 128.7639 390.89 279.23 ]

[ 66.88088 145.6837 391.89 280.23 ]

[ 67.69868 116.819 392.89 281.23 ]

[ 69.82117 143.6215 393.89 282.23 ]

[ 69.08817 134.9325 394.89 283.23 ]

[ 69.91479 147.0219 395.89 284.23 ]

[ 67.33182 126.3285 396.89 285.23 ]

[ 70.26939 125.4839 397.89 286.23 ]

[ 69.10344 115.7084 398.89 287.23 ]

[ 65.38356 123.4892 399.89 288.23 ]

[ 70.18447 147.8926 400.89 289.23 ]

[ 70.40617 155.8987 401.89 290.23 ]

[ 66.54376 128.0742 402.89 291.23 ]

[ 66.36418 119.3701 403.89 292.23 ]

[ 67.537 133.8148 404.89 293.23 ]]

The rows & columns on dataset: (60, 4)

---------Perform Indexing operation on Dataset------

[ 65.78331 112.9925 345.89 234.23 ]

[ 67.537 133.8148 404.89 293.23 ]

136.4873

293.23

242.23

[112.9925 241.23 348.89 268.23 ]

470.46

[112.9925 345.89 234.23 71.51521 136.4873 346.89 235.23

153.0269 347.89 236.23 142.3354 348.89 237.23 144.2971

349.89 238.23 123.3024 350.89 239.23 141.4947 351.89

240.23 70.01472 136.4623 352.89 241.23 112.3723 353.89

242.23 120.6672 354.89 243.23 127.4516 355.89 244.23

114.143 356.89 245.23 125.6107 357.89 246.23 122.4618

358.89 247.23 116.0866 359.89 248.23 71.0916 139.9975

360.89 249.23 129.5023 361.89 250.23 142.9733 362.89

251.23 71.23033 137.9025 363.89 252.23 124.0449 364.89

253.23 141.2807 365.89 254.23 143.5392 366.89 255.23

97.90191 367.89 256.23 129.5027 368.89 257.23 141.8501

369.89 258.23 129.7244 370.89 259.23 70.84235 142.4235

371.89 260.23 131.5502 372.89 261.23 108.3324 373.89

262.23 113.8922 374.89 263.23 103.3016 375.89 264.23

120.7536 376.89 265.23 125.7886 377.89 266.23 70.59505

136.2225 378.89 267.23 71.80484 140.1015 379.89 268.23

128.7487 380.89 269.23 141.7994 381.89 270.23 121.2319

382.89 271.23 131.3478 383.89 272.23 106.7115 384.89

273.23 124.3598 385.89 274.23 124.8591 386.89 275.23

139.6711 387.89 276.23 137.3696 388.89 277.23 106.4499

389.89 278.23 128.7639 390.89 279.23 145.6837 391.89

280.23 116.819 392.89 281.23 143.6215 393.89 282.23

134.9325 394.89 283.23 147.0219 395.89 284.23 126.3285

396.89 285.23 70.26939 125.4839 397.89 286.23 115.7084

398.89 287.23 123.4892 399.89 288.23 70.18447 147.8926

400.89 289.23 70.40617 155.8987 401.89 290.23 128.0742

402.89 291.23 119.3701 403.89 292.23 133.8148 404.89

293.23 ]

---------Perform Slicing operation on Dataset------

[[ 71.51521 136.4873 346.89 235.23 ]

[ 69.39874 153.0269 347.89 236.23 ]]

[[ 65.78331 112.9925 ]

[ 71.51521 136.4873 ]]

[293.23 404.89 133.8148 67.537 ]

[[ 70.18447 400.89 ]

[ 70.40617 401.89 ]

[ 66.54376 402.89 ]

[ 66.36418 403.89 ]]

---------Perform Splitting operation on Dataset------

-----------Horizontal splitting-----------

[array([[ 65.78331, 112.9925 ],

[ 71.51521, 136.4873 ],

[ 69.39874, 153.0269 ],

[ 68.2166 , 142.3354 ],

[ 67.78781, 144.2971 ],

[ 68.69784, 123.3024 ],

[ 69.80204, 141.4947 ],

[ 70.01472, 136.4623 ],

[ 67.90265, 112.3723 ],

[ 66.78236, 120.6672 ],

[ 66.48769, 127.4516 ],

[ 67.62333, 114.143 ],

[ 68.30248, 125.6107 ],

[ 67.11656, 122.4618 ],

[ 68.27967, 116.0866 ],

[ 71.0916 , 139.9975 ],

[ 66.461 , 129.5023 ],

[ 68.64927, 142.9733 ],

[ 71.23033, 137.9025 ],

[ 67.13118, 124.0449 ],

[ 67.83379, 141.2807 ],

[ 68.87881, 143.5392 ],

[ 63.48115, 97.90191],

[ 68.42187, 129.5027 ],

[ 67.62804, 141.8501 ],

[ 67.20864, 129.7244 ],

[ 70.84235, 142.4235 ],

[ 67.49434, 131.5502 ],

[ 66.53401, 108.3324 ],

[ 65.44098, 113.8922 ],

[ 69.5233 , 103.3016 ],

[ 65.8132 , 120.7536 ],

[ 67.8163 , 125.7886 ],

[ 70.59505, 136.2225 ],

[ 71.80484, 140.1015 ],

[ 69.20613, 128.7487 ],

[ 66.80368, 141.7994 ],

[ 67.65893, 121.2319 ],

[ 67.80701, 131.3478 ],

[ 64.04535, 106.7115 ],

[ 68.57463, 124.3598 ],

[ 65.18357, 124.8591 ],

[ 69.65814, 139.6711 ],

[ 67.96731, 137.3696 ],

[ 65.98088, 106.4499 ],

[ 68.67249, 128.7639 ],

[ 66.88088, 145.6837 ],

[ 67.69868, 116.819 ],

[ 69.82117, 143.6215 ],

[ 69.08817, 134.9325 ],

[ 69.91479, 147.0219 ],

[ 67.33182, 126.3285 ],

[ 70.26939, 125.4839 ],

[ 69.10344, 115.7084 ],

[ 65.38356, 123.4892 ],

[ 70.18447, 147.8926 ],

[ 70.40617, 155.8987 ],

[ 66.54376, 128.0742 ],

[ 66.36418, 119.3701 ],

[ 67.537 , 133.8148 ]]), array([[345.89, 234.23],

[346.89, 235.23],

[347.89, 236.23],

[348.89, 237.23],

[349.89, 238.23],

[350.89, 239.23],

[351.89, 240.23],

[352.89, 241.23],

[353.89, 242.23],

[354.89, 243.23],

[355.89, 244.23],

[356.89, 245.23],

[357.89, 246.23],

[358.89, 247.23],

[359.89, 248.23],

[360.89, 249.23],

[361.89, 250.23],

[362.89, 251.23],

[363.89, 252.23],

[364.89, 253.23],

[365.89, 254.23],

[366.89, 255.23],

[367.89, 256.23],

[368.89, 257.23],

[369.89, 258.23],

[370.89, 259.23],

[371.89, 260.23],

[372.89, 261.23],

[373.89, 262.23],

[374.89, 263.23],

[375.89, 264.23],

[376.89, 265.23],

[377.89, 266.23],

[378.89, 267.23],

[379.89, 268.23],

[380.89, 269.23],

[381.89, 270.23],

[382.89, 271.23],

[383.89, 272.23],

[384.89, 273.23],

[385.89, 274.23],

[386.89, 275.23],

[387.89, 276.23],

[388.89, 277.23],

[389.89, 278.23],

[390.89, 279.23],

[391.89, 280.23],

[392.89, 281.23],

[393.89, 282.23],

[394.89, 283.23],

[395.89, 284.23],

[396.89, 285.23],

[397.89, 286.23],

[398.89, 287.23],

[399.89, 288.23],

[400.89, 289.23],

[401.89, 290.23],

[402.89, 291.23],

[403.89, 292.23],

[404.89, 293.23]])]

-----------vertical splitting-----------

[array([[ 65.78331, 112.9925 , 345.89 , 234.23 ],

[ 71.51521, 136.4873 , 346.89 , 235.23 ],

[ 69.39874, 153.0269 , 347.89 , 236.23 ],

[ 68.2166 , 142.3354 , 348.89 , 237.23 ],

[ 67.78781, 144.2971 , 349.89 , 238.23 ],

[ 68.69784, 123.3024 , 350.89 , 239.23 ],

[ 69.80204, 141.4947 , 351.89 , 240.23 ],

[ 70.01472, 136.4623 , 352.89 , 241.23 ],

[ 67.90265, 112.3723 , 353.89 , 242.23 ],

[ 66.78236, 120.6672 , 354.89 , 243.23 ],

[ 66.48769, 127.4516 , 355.89 , 244.23 ],

[ 67.62333, 114.143 , 356.89 , 245.23 ],

[ 68.30248, 125.6107 , 357.89 , 246.23 ],

[ 67.11656, 122.4618 , 358.89 , 247.23 ],

[ 68.27967, 116.0866 , 359.89 , 248.23 ],

[ 71.0916 , 139.9975 , 360.89 , 249.23 ],

[ 66.461 , 129.5023 , 361.89 , 250.23 ],

[ 68.64927, 142.9733 , 362.89 , 251.23 ],

[ 71.23033, 137.9025 , 363.89 , 252.23 ],

[ 67.13118, 124.0449 , 364.89 , 253.23 ],

[ 67.83379, 141.2807 , 365.89 , 254.23 ],

[ 68.87881, 143.5392 , 366.89 , 255.23 ],

[ 63.48115, 97.90191, 367.89 , 256.23 ],

[ 68.42187, 129.5027 , 368.89 , 257.23 ],

[ 67.62804, 141.8501 , 369.89 , 258.23 ],

[ 67.20864, 129.7244 , 370.89 , 259.23 ],

[ 70.84235, 142.4235 , 371.89 , 260.23 ],

[ 67.49434, 131.5502 , 372.89 , 261.23 ],

[ 66.53401, 108.3324 , 373.89 , 262.23 ],

[ 65.44098, 113.8922 , 374.89 , 263.23 ]]), array([[ 69.5233 , 103.3016 , 375.89 , 264.23 ],

[ 65.8132 , 120.7536 , 376.89 , 265.23 ],

[ 67.8163 , 125.7886 , 377.89 , 266.23 ],

[ 70.59505, 136.2225 , 378.89 , 267.23 ],

[ 71.80484, 140.1015 , 379.89 , 268.23 ],

[ 69.20613, 128.7487 , 380.89 , 269.23 ],

[ 66.80368, 141.7994 , 381.89 , 270.23 ],

[ 67.65893, 121.2319 , 382.89 , 271.23 ],

[ 67.80701, 131.3478 , 383.89 , 272.23 ],

[ 64.04535, 106.7115 , 384.89 , 273.23 ],

[ 68.57463, 124.3598 , 385.89 , 274.23 ],

[ 65.18357, 124.8591 , 386.89 , 275.23 ],

[ 69.65814, 139.6711 , 387.89 , 276.23 ],

[ 67.96731, 137.3696 , 388.89 , 277.23 ],

[ 65.98088, 106.4499 , 389.89 , 278.23 ],

[ 68.67249, 128.7639 , 390.89 , 279.23 ],

[ 66.88088, 145.6837 , 391.89 , 280.23 ],

[ 67.69868, 116.819 , 392.89 , 281.23 ],

[ 69.82117, 143.6215 , 393.89 , 282.23 ],

[ 69.08817, 134.9325 , 394.89 , 283.23 ],

[ 69.91479, 147.0219 , 395.89 , 284.23 ],

[ 67.33182, 126.3285 , 396.89 , 285.23 ],

[ 70.26939, 125.4839 , 397.89 , 286.23 ],

[ 69.10344, 115.7084 , 398.89 , 287.23 ],

[ 65.38356, 123.4892 , 399.89 , 288.23 ],

[ 70.18447, 147.8926 , 400.89 , 289.23 ],

[ 70.40617, 155.8987 , 401.89 , 290.23 ],

[ 66.54376, 128.0742 , 402.89 , 291.23 ],

[ 66.36418, 119.3701 , 403.89 , 292.23 ],

[ 67.537 , 133.8148 , 404.89 , 293.23 ]])]

---------Perform Iterating operation on Dataset------

----------using nditer()--------

65.78331

112.9925

345.89

234.23

71.51521

136.4873

346.89

235.23

69.39874

153.0269

347.89

236.23

68.2166

142.3354

348.89

237.23

67.78781

144.2971

349.89

238.23

68.69784

123.3024

350.89

239.23

69.80204

141.4947

351.89

240.23

70.01472

136.4623

352.89

241.23

67.90265

112.3723

353.89

242.23

66.78236

120.6672

354.89

243.23

66.48769

127.4516

355.89

244.23

67.62333

114.143

356.89

245.23

68.30248

125.6107

357.89

246.23

67.11656

122.4618

358.89

247.23

68.27967

116.0866

359.89

248.23

71.0916

139.9975

360.89

249.23

66.461

129.5023

361.89

250.23

68.64927

142.9733

362.89

251.23

71.23033

137.9025

363.89

252.23

67.13118

124.0449

364.89

253.23

67.83379

141.2807

365.89

254.23

68.87881

143.5392

366.89

255.23

63.48115

97.90191

367.89

256.23

68.42187

129.5027

368.89

257.23

67.62804

141.8501

369.89

258.23

67.20864

129.7244

370.89

259.23

70.84235

142.4235

371.89

260.23

67.49434

131.5502

372.89

261.23

66.53401

108.3324

373.89

262.23

65.44098

113.8922

374.89

263.23

69.5233

103.3016

375.89

264.23

65.8132

120.7536

376.89

265.23

67.8163

125.7886

377.89

266.23

70.59505

136.2225

378.89

267.23

71.80484

140.1015

379.89

268.23

69.20613

128.7487

380.89

269.23

66.80368

141.7994

381.89

270.23

67.65893

121.2319

382.89

271.23

67.80701

131.3478

383.89

272.23

64.04535

106.7115

384.89

273.23

68.57463

124.3598

385.89

274.23

65.18357

124.8591

386.89

275.23

69.65814

139.6711

387.89

276.23

67.96731

137.3696

388.89

277.23

65.98088

106.4499

389.89

278.23

68.67249

128.7639

390.89

279.23

66.88088

145.6837

391.89

280.23

67.69868

116.819

392.89

281.23

69.82117

143.6215

393.89

282.23

69.08817

134.9325

394.89

283.23

69.91479

147.0219

395.89

284.23

67.33182

126.3285

396.89

285.23

70.26939

125.4839

397.89

286.23

69.10344

115.7084

398.89

287.23

65.38356

123.4892

399.89

288.23

70.18447

147.8926

400.89

289.23

70.40617

155.8987

401.89

290.23

66.54376

128.0742

402.89

291.23

66.36418

119.3701

403.89

292.23

67.537

133.8148

404.89

293.23

----------using ndenumerate()--------

(0, 0) 65.78331

(0, 1) 112.9925

(0, 2) 345.89

(0, 3) 234.23

(1, 0) 71.51521

(1, 1) 136.4873

(1, 2) 346.89

(1, 3) 235.23

(2, 0) 69.39874

(2, 1) 153.0269

(2, 2) 347.89

(2, 3) 236.23

(3, 0) 68.2166

(3, 1) 142.3354

(3, 2) 348.89

(3, 3) 237.23

(4, 0) 67.78781

(4, 1) 144.2971

(4, 2) 349.89

(4, 3) 238.23

(5, 0) 68.69784

(5, 1) 123.3024

(5, 2) 350.89

(5, 3) 239.23

(6, 0) 69.80204

(6, 1) 141.4947

(6, 2) 351.89

(6, 3) 240.23

(7, 0) 70.01472

(7, 1) 136.4623

(7, 2) 352.89

(7, 3) 241.23

(8, 0) 67.90265

(8, 1) 112.3723

(8, 2) 353.89

(8, 3) 242.23

(9, 0) 66.78236

(9, 1) 120.6672

(9, 2) 354.89

(9, 3) 243.23

(10, 0) 66.48769

(10, 1) 127.4516

(10, 2) 355.89

(10, 3) 244.23

(11, 0) 67.62333

(11, 1) 114.143

(11, 2) 356.89

(11, 3) 245.23

(12, 0) 68.30248

(12, 1) 125.6107

(12, 2) 357.89

(12, 3) 246.23

(13, 0) 67.11656

(13, 1) 122.4618

(13, 2) 358.89

(13, 3) 247.23

(14, 0) 68.27967

(14, 1) 116.0866

(14, 2) 359.89

(14, 3) 248.23

(15, 0) 71.0916

(15, 1) 139.9975

(15, 2) 360.89

(15, 3) 249.23

(16, 0) 66.461

(16, 1) 129.5023

(16, 2) 361.89

(16, 3) 250.23

(17, 0) 68.64927

(17, 1) 142.9733

(17, 2) 362.89

(17, 3) 251.23

(18, 0) 71.23033

(18, 1) 137.9025

(18, 2) 363.89

(18, 3) 252.23

(19, 0) 67.13118

(19, 1) 124.0449

(19, 2) 364.89

(19, 3) 253.23

(20, 0) 67.83379

(20, 1) 141.2807

(20, 2) 365.89

(20, 3) 254.23

(21, 0) 68.87881

(21, 1) 143.5392

(21, 2) 366.89

(21, 3) 255.23

(22, 0) 63.48115

(22, 1) 97.90191

(22, 2) 367.89

(22, 3) 256.23

(23, 0) 68.42187

(23, 1) 129.5027

(23, 2) 368.89

(23, 3) 257.23

(24, 0) 67.62804

(24, 1) 141.8501

(24, 2) 369.89

(24, 3) 258.23

(25, 0) 67.20864

(25, 1) 129.7244

(25, 2) 370.89

(25, 3) 259.23

(26, 0) 70.84235

(26, 1) 142.4235

(26, 2) 371.89

(26, 3) 260.23

(27, 0) 67.49434

(27, 1) 131.5502

(27, 2) 372.89

(27, 3) 261.23

(28, 0) 66.53401

(28, 1) 108.3324

(28, 2) 373.89

(28, 3) 262.23

(29, 0) 65.44098

(29, 1) 113.8922

(29, 2) 374.89

(29, 3) 263.23

(30, 0) 69.5233

(30, 1) 103.3016

(30, 2) 375.89

(30, 3) 264.23

(31, 0) 65.8132

(31, 1) 120.7536

(31, 2) 376.89

(31, 3) 265.23

(32, 0) 67.8163

(32, 1) 125.7886

(32, 2) 377.89

(32, 3) 266.23

(33, 0) 70.59505

(33, 1) 136.2225

(33, 2) 378.89

(33, 3) 267.23

(34, 0) 71.80484

(34, 1) 140.1015

(34, 2) 379.89

(34, 3) 268.23

(35, 0) 69.20613

(35, 1) 128.7487

(35, 2) 380.89

(35, 3) 269.23

(36, 0) 66.80368

(36, 1) 141.7994

(36, 2) 381.89

(36, 3) 270.23

(37, 0) 67.65893

(37, 1) 121.2319

(37, 2) 382.89

(37, 3) 271.23

(38, 0) 67.80701

(38, 1) 131.3478

(38, 2) 383.89

(38, 3) 272.23

(39, 0) 64.04535

(39, 1) 106.7115

(39, 2) 384.89

(39, 3) 273.23

(40, 0) 68.57463

(40, 1) 124.3598

(40, 2) 385.89

(40, 3) 274.23

(41, 0) 65.18357

(41, 1) 124.8591

(41, 2) 386.89

(41, 3) 275.23

(42, 0) 69.65814

(42, 1) 139.6711

(42, 2) 387.89

(42, 3) 276.23

(43, 0) 67.96731

(43, 1) 137.3696

(43, 2) 388.89

(43, 3) 277.23

(44, 0) 65.98088

(44, 1) 106.4499

(44, 2) 389.89

(44, 3) 278.23

(45, 0) 68.67249

(45, 1) 128.7639

(45, 2) 390.89

(45, 3) 279.23

(46, 0) 66.88088

(46, 1) 145.6837

(46, 2) 391.89

(46, 3) 280.23

(47, 0) 67.69868

(47, 1) 116.819

(47, 2) 392.89

(47, 3) 281.23

(48, 0) 69.82117

(48, 1) 143.6215

(48, 2) 393.89

(48, 3) 282.23

(49, 0) 69.08817

(49, 1) 134.9325

(49, 2) 394.89

(49, 3) 283.23

(50, 0) 69.91479

(50, 1) 147.0219

(50, 2) 395.89

(50, 3) 284.23

(51, 0) 67.33182

(51, 1) 126.3285

(51, 2) 396.89

(51, 3) 285.23

(52, 0) 70.26939

(52, 1) 125.4839

(52, 2) 397.89

(52, 3) 286.23

(53, 0) 69.10344

(53, 1) 115.7084

(53, 2) 398.89

(53, 3) 287.23

(54, 0) 65.38356

(54, 1) 123.4892

(54, 2) 399.89

(54, 3) 288.23

(55, 0) 70.18447

(55, 1) 147.8926

(55, 2) 400.89

(55, 3) 289.23

(56, 0) 70.40617

(56, 1) 155.8987

(56, 2) 401.89

(56, 3) 290.23

(57, 0) 66.54376

(57, 1) 128.0742

(57, 2) 402.89

(57, 3) 291.23

(58, 0) 66.36418

(58, 1) 119.3701

(58, 2) 403.89

(58, 3) 292.23

(59, 0) 67.537

(59, 1) 133.8148

(59, 2) 404.89

(59, 3) 293.23