```
1  import numpy as np
```

NUMPY MINI PROJECT 1

1 Inject data into array, find mean, median, IQR for Sachin, Rahul and India

2 Find the histogram of sachin's Scores with 10 bins

3 Find mean of sachin's scores grouped by 25 matches

4 Find mean of sachin's scores where he has scored a century

5 Find mean of sachin's scores when Rahul has scored less then 10

6 Find mean of sachin's scores based on which quartile India's score falls in

7 For every match findout who has scored more Sachin or Rahul

8 How many more runs does sachin score on an average after scoring X runs

9 How many matches did sachin take to score first 1000 runs and then next 1000

```
1 !head cric_data.tsv
```

```
          Sachin Tendulkar        Rahul Dravid    India
0       100     78      342
1       11      62      191
2       8       85      252
3       71      24      307
4       104     17      229
5       18      104     246
6       8       76      226
7       86      74      288
8       12      60      216
```

```
1 cric_data = np.loadtxt("cric_data.tsv",skiprows=1)
```

```
1 cric_data
```

```
          [167.,  31.,  54., 314.],
          [168.,   2.,  52., 419.],
          [169.,  44.,   9., 229.],
          [170.,  47.,   0., 133.],
          [171.,   6.,  18., 228.],
          [172.,  17.,   3., 143.],
          [173.,  35.,   4., 116.],
          [174.,  88.,   6., 194.],
          [175., 114.,  16., 249.],
          [176.,   7.,  71., 319.],
          [177., 127.,  64., 205.],
          [178.,   0.,   7., 191.],
          [179.,  45.,  61., 213.],
          [180.,  33.,   1., 104.],
          [181., 110.,   7., 210.],
          [182.,  146.,   0.,  392.]
```

```
[182., 140.,   0., 392.],
[183.,   7.,   0.,  57.],
[184.,  25.,  26., 213.],
[185.,   9.,  19., 158.],
[186.,   2.,  63., 163.],
[187.,  11.,   3., 166.],
[188.,  17.,  46., 164.],
[189.,  14.,  31., 238.],
[190.,   1.,  12., 118.],
[191.,  45.,  49., 189.],
[192.,  94.,   4., 346.],
[193.,  28.,  99., 390.],
[194.,   4.,  47., 319.],
[195.,  62.,  56., 276.],
[196.,   1.,  73., 324.],
[197.,   9.,  74., 213.],
[198.,  67.,  31., 207.],
[199.,  10.,  29., 158.],
[200.,  99.,  92., 304.],
[201.,  55.,   1., 199.],
[202.,   0.,   0.,   0.],
[203.,   1.,  30., 195.],
[204., 101.,  16., 207.],
[205.,   2., 145., 352.],
[206.,   1.,  22., 225.],
[207.,  16.,  31., 236.],
[208., 140., 104., 301.],
[209.,  23.,  20., 136.],
[210., 113.,  13., 327.],
[211.,  10.,  52., 296.],
[212.,   1.,  31., 176.],
[213.,  41.,  12., 201.],
[214.,   1.,  50., 171.],
[215.,  28.,   3., 191.],
[216.,   2.,   9.,  11.],
[217.,  22.,  53., 211.],
[218.,  29.,  49., 225.],
[219.,   1.,  21., 290.],
[220.,  25.,  14., 101.],
[221., 102.,  50., 470.],
[222.,   0.,  22., 165.],
[223.,  27.,   0., 192.],
[224.,  40.,   0., 213.]])
```

```
#typecasting float to integer array
cric_data = cric_data.astype(dtype=int)
cric_data
```

```
[167,  31,  54, 314],
[168,   2,  52, 419],
[169,  44,   9, 229],
[170,  47,   0, 133],
[171,   6,  18, 228],
[172,  17,   3, 143],
[173,  35,   4, 116],
[174,  88,   6, 194],
[175, 114,  16, 249],
[176,   7,  71, 319],
[177, 127,  64, 205],
[178,   0,   7, 191],
```

```
       [179,  45,  61, 213],
       [180,  33,   1, 104],
       [181, 110,   7, 210],
       [182, 146,   0, 392],
       [183,   7,   0,  57],
       [184,  25,  26, 213],
       [185,   9,  19, 158],
       [186,   2,  63, 163],
       [187,  11,   3, 166],
       [188,  17,  46, 164],
       [189,  14,  31, 238],
       [190,   1,  12, 118],
       [191,  45,  49, 189],
       [192,  94,   4, 346],
       [193,  28,  99, 390],
       [194,   4,  47, 319],
       [195,  62,  56, 276],
       [196,   1,  73, 324],
       [197,   9,  74, 213],
       [198,  67,  31, 207],
       [199,  10,  29, 158],
       [200,  99,  92, 304],
       [201,  55,   1, 199],
       [202,   0,   0,   0],
       [203,   1,  30, 195],
       [204, 101,  16, 207],
       [205,   2, 145, 352],
       [206,   1,  22, 225],
       [207,  16,  31, 236],
       [208, 140, 104, 301],
       [209,  23,  20, 136],
       [210, 113,  13, 327],
       [211,  10,  52, 296],
       [212,   1,  31, 176],
       [213,  41,  12, 201],
       [214,   1,  50, 171],
       [215,  28,   3, 191],
       [216,   2,   9,  11],
       [217,  22,  53, 211],
       [218,  29,  49, 225],
       [219,   1,  21, 290],
       [220,  25,  14, 101],
       [221, 102,  50, 470],
       [222,   0,  22, 165],
       [223,  27,   0, 192],
       [224,  40,   0, 213]])
```

```
#converting into numpy array
cric_data = np.array(cric_data)
cric_data
```

```
       [167,  31,  54, 314],
       [168,   2,  52, 419],
       [169,  44,   9, 229],
       [170,  47,   0, 133],
       [171,   6,  18, 228],
       [172,  17,   3, 143],
       [173,  35,   4, 116],
       [174,  88,   6, 194],
       [175, 114,  16, 249],
```

```
[175, 114,  16, 249],
[176,   7,  71, 319],
[177, 127,  64, 205],
[178,   0,   7, 191],
[179,  45,  61, 213],
[180,  33,   1, 104],
[181, 110,   7, 210],
[182, 146,   0, 392],
[183,   7,   0,  57],
[184,  25,  26, 213],
[185,   9,  19, 158],
[186,   2,  63, 163],
[187,  11,   3, 166],
[188,  17,  46, 164],
[189,  14,  31, 238],
[190,   1,  12, 118],
[191,  45,  49, 189],
[192,  94,   4, 346],
[193,  28,  99, 390],
[194,   4,  47, 319],
[195,  62,  56, 276],
[196,   1,  73, 324],
[197,   9,  74, 213],
[198,  67,  31, 207],
[199,  10,  29, 158],
[200,  99,  92, 304],
[201,  55,   1, 199],
[202,   0,   0,   0],
[203,   1,  30, 195],
[204, 101,  16, 207],
[205,   2, 145, 352],
[206,   1,  22, 225],
[207,  16,  31, 236],
[208, 140, 104, 301],
[209,  23,  20, 136],
[210, 113,  13, 327],
[211,  10,  52, 296],
[212,   1,  31, 176],
[213,  41,  12, 201],
[214,   1,  50, 171],
[215,  28,   3, 191],
[216,   2,   9,  11],
[217,  22,  53, 211],
[218,  29,  49, 225],
[219,   1,  21, 290],
[220,  25,  14, 101],
[221, 102,  50, 470],
[222,   0,  22, 165],
[223,  27,   0, 192],
[224,  40,   0, 213]])
```

1 Inject data into array, find mean, median, IQR for Sachin, Rahul and **India**

```
#can also do
```

```
#cric_data=cric_data[:,[1,2,3]]
#ALSO DO COPY NOT = this will ensure data is not canges
```

```
arSch = np.array(cric_data[:,1])
arSch
```

```
array([100,  11,   8,  71, 104,  18,   8,  86,  12,  85,  18,   4,   7,
        37,  14,   0,   4,   0,  21,   1,  62,   0, 138,  38,   2,  46,
        65,   0,  39,  48, 141,  62,  12,   1,  41,  11,   3, 186,  11,
        27,  27,  51,  18,  32, 146,   5,  45, 141,  12,  65,  27,   7,
        16,   2,  28,   6, 123, 120,   7,   3,   0,  81,   2,  54, 122,
         4,  14,   0, 100,  15,   0,  57,  99,  37,  38,  32,  21,  32,
        40,   0,   5,   8,   5,   0,  50,  30,  37,  89,   4,  98,  83,
        93,   0,  52, 152,   1,   8,  93,  45,  26,   0,   1,   0,  16,
        47,  89,   3,   1,  53,  16,   0,  81,  14,  78,   6, 105, 122,
         9,   8,  28,  35,  69,  13,  97,  93,   2,  36,  39,   2,  29,
        12,  19,  34,   2, 100,  44,  82,   0,  79,   6,   9,   8,  23,
        93,  35,  63,  74,   8, 117,  39,  49,  64,  43,  72,   5,  17,
         0,  65,  20, 141,  28,  44,  27,  60,  68,   3, 139,  31,   2,
        44,  47,   6,  17,  35,  88, 114,   7, 127,   0,  45,  33, 110,
       146,   7,  25,   9,   2,  11,  17,  14,   1,  45,  94,  28,   4,
        62,   1,   9,  67,  10,  99,  55,   0,   1, 101,   2,   1,  16,
       140,  23, 113,  10,   1,  41,   1,  28,   2,  22,  29,   1,  25,
       102,   0,  27,  40])
```

```
arRah = np.array(cric_data[:,2])
arRah
```

```
array([ 78,  62,  85,  24,  17, 104,  76,  74,  60,  12,  63, 107,  76,
         4,   5,  33,   7,   0,  36,  66,   0, 123,  39,   9,  11,  14,
         0,   0,  26,   4,  48,   7,  73,  86,  32,  82,  25, 153,  26,
         1,   6,   3,   1,  39,  30,  32,  84,  36,  31,   0,  47,  13,
        49,  28,   0,  28,  19,  13,  14,  11, 103,  43,   5,   5,  15,
         0,   5,  30,   4,  53,  60,   7,  74,  71,  54,  13,  69,  33,
        29,   2,   8,  36,  51,   8,  62,   0,  22,  39, 104,  44,   1,
         2,   0,  17,   0,  14,  62,  15,  68,  17,   0,  22,  25,  29,
        60,  34,  65,   3,   7,  16,   0,   4,  53,   5,  23,  82,   3,
         0,  20,  21,  80,  22,   8,  18,  24,  50,   1,  42,  61,   0,
         6,  64,   9,  46,   0,   0,   0,   0,  13,   0,  72,  12,  38,
        79,  18,  16,  16,  56,  21,  58,   8,   0,   0,   7,  13,  77,
        29,  20,  90,  26,  54,  17,  78,  57,  59,  10,  15,  54,  52,
         9,   0,  18,   3,   4,   6,  16,  71,  64,   7,  61,   1,   7,
         0,   0,  26,  19,  63,   3,  46,  31,  12,  49,   4,  99,  47,
        56,  73,  74,  31,  29,  92,   1,   0,  30,  16, 145,  22,  31,
       104,  20,  13,  52,  31,  12,  50,   3,   9,  53,  49,  21,  14,
        50,  22,   0,   0])
```

```
arInd = np.array(cric_data[:,3])
arInd
```

```
array([342, 191, 252, 307, 229, 246, 226, 288, 216, 224, 161, 276, 283,
       297, 139, 224, 178,   0, 193, 231, 134, 246, 299, 242, 214, 152,
       104,   4, 155, 168, 282, 228, 231, 238, 255, 273, 143, 345, 134,
       292, 299, 233, 332, 276, 264, 213, 224, 306, 259, 141, 155, 183,
       309, 208, 124, 208, 305, 273, 186, 163, 239, 274, 182, 256, 227,
       157, 353, 271, 211, 141, 243, 499, 229, 234, 259, 242, 223, 184,
       125, 195, 154, 142, 197, 213, 256, 197, 177, 167, 366, 269, 292,
```

```
218,   0, 199, 319, 245, 274, 208, 201, 275, 134,  68, 149, 216,
407, 302, 197, 222, 224, 321,   0, 162, 221, 204, 213, 286, 272,
382, 229, 175, 360, 256, 165, 335, 188, 146, 113, 241, 211, 225,
 35, 163, 200, 176, 269, 284, 175,  27, 282, 141, 229, 213, 205,
279,  88, 284, 207, 224, 204, 280, 197, 205, 230, 323, 209, 210,
162, 272, 251, 286, 228, 148, 272, 309, 202, 136, 281, 314, 419,
229, 133, 228, 143, 116, 194, 249, 319, 205, 191, 213, 104, 210,
392,  57, 213, 158, 163, 166, 164, 238, 118, 189, 346, 390, 319,
276, 324, 213, 207, 158, 304, 199,   0, 195, 207, 352, 225, 236,
301, 136, 327, 296, 176, 201, 171, 191,  11, 211, 225, 290, 101,
470, 165, 192, 213])
```

mean for Sachin, Rahul and **India**

```
#CAN DO MEAN MEDIAN MODE IN FUNCTION
```

```
SchMean = np.mean(arSch)
SchMean
```

```
39.87555555555556
```

```
RahMean = np.mean(arRah)
RahMean
```

```
32.062222222222225
```

```
IndMean = np.mean(arInd)
IndMean
```

```
220.79555555555555
```

Median for Sachin, Rahul and **India**

```
SchMed = np.median(arSch)
SchMed
```

```
27.0
```

```
RahMed = np.median(arRah)
RahMed
```

```
22.0
```

```
IndMed = np.median(arInd)
IndMed
```

```
216.0
```

IQR for Sachin, Rahul and **India**

```
SchIQR = np.percentile(arSch,75)-np.percentile(arSch,25)
SchIQR
```

```
57.0
```

```
RahIQR = np.percentile(arRah,75)-np.percentile(arRah,25)
RahIQR
```

```
46.0
```

```
IndIQR = np.percentile(arInd,75)-np.percentile(arInd,25)
IndIQR
```

```
98.0
```

ALTERNATE EFFICIENT WAY

```
np.mean(cric_data,axis=0)
```

```
array([112.        ,  39.87555556,  32.06222222, 220.79555556])
```

```
np.median(cric_data,axis=0)
```

```
array([112.,  27.,  22., 216.])
```

```
np.percentile(cric_data,75,axis=0)-np.percentile(cric_data,25,axis=0)
```

```
array([112.,  57.,  46.,  98.])
```

2 Find the histogram of sachin's Scores with 10 bins

```
SchHist = np.histogram(arSch,bins=10)
SchHist
```

```
(array([99, 36, 28, 16, 11, 17,  8,  8,  1,  1]),
 array([  0. ,  18.6,  37.2,  55.8,  74.4,  93. , 111.6, 130.2, 148.8,
        167.4, 186. ]))
```

3 Find mean of sachin's scores grouped by 25 matches

```
k= 9 #225/25
i=0
j=25
SchM25 =[]
for m in range(k) :
```

```
    Mean=np.mean(arSch[i:j])
    SchM25.append(Mean)
    i=i+25
    j=j+25
SchM25
```

```
[33.96, 49.4, 38.48, 40.16, 39.36, 38.2, 44.6, 39.52, 35.2]
```

ALTERNATE EFFICIENT WAY WAY

```
arSch.shape
```

```
(225,)
```

```
arSch25Match= arSch.reshape(9,25)
arSch25Match
```

```
array([[100,  11,   8,  71, 104,  18,   8,  86,  12,  85,  18,   4,   7,
         37,  14,   0,   4,   0,  21,   1,  62,   0, 138,  38,   2],
       [ 46,  65,   0,  39,  48, 141,  62,  12,   1,  41,  11,   3, 186,
         11,  27,  27,  51,  18,  32, 146,   5,  45, 141,  12,  65],
       [ 27,   7,  16,   2,  28,   6, 123, 120,   7,   3,   0,  81,   2,
         54, 122,   4,  14,   0, 100,  15,   0,  57,  99,  37,  38],
       [ 32,  21,  32,  40,   0,   5,   8,   5,   0,  50,  30,  37,  89,
          4,  98,  83,  93,   0,  52, 152,   1,   8,  93,  45,  26],
       [  0,   1,   0,  16,  47,  89,   3,   1,  53,  16,   0,  81,  14,
         78,   6, 105, 122,   9,   8,  28,  35,  69,  13,  97,  93],
       [  2,  36,  39,   2,  29,  12,  19,  34,   2, 100,  44,  82,   0,
         79,   6,   9,   8,  23,  93,  35,  63,  74,   8, 117,  39],
       [ 49,  64,  43,  72,   5,  17,   0,  65,  20, 141,  28,  44,  27,
         60,  68,   3, 139,  31,   2,  44,  47,   6,  17,  35,  88],
       [114,   7, 127,   0,  45,  33, 110, 146,   7,  25,   9,   2,  11,
         17,  14,   1,  45,  94,  28,   4,  62,   1,   9,  67,  10],
       [ 99,  55,   0,   1, 101,   2,   1,  16, 140,  23, 113,  10,   1,
         41,   1,  28,   2,  22,  29,   1,  25, 102,   0,  27,  40]])
```

```
arSch25Match.mean(axis=0)#wrong as it gave 25 means
```

```
array([52.11111111, 29.66666667, 29.44444444, 27.        , 45.22222222,
       35.88888889, 37.11111111, 53.88888889, 26.88888889, 53.77777778,
       28.11111111, 38.22222222, 37.44444444, 42.33333333, 39.55555556,
       28.88888889, 53.11111111, 21.88888889, 40.55555556, 47.33333333,
       33.33333333, 40.22222222, 57.55555556, 52.77777778, 44.55555556])
```

```
arSch25Match.mean(axis=1)#Correct
```

```
array([33.96, 49.4 , 38.48, 40.16, 39.36, 38.2 , 44.6 , 39.52, 35.2 ])
```

4 Find mean of sachin's scores where he has scored a century

```
arSch100 = np.array([ ele for ele in arSch if ele >=100])
arSch100
```

```
array([100, 104, 138, 141, 186, 146, 141, 123, 120, 122, 100, 152, 105,
       122, 100, 117, 141, 139, 114, 127, 110, 146, 101, 140, 113, 102])
```

```
arSch100Mean= np.mean(arSch100)
arSch100Mean
```

```
125.0
```

EFFICIENT WAY

```
arSch>=100
```

```
array([ True, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False,  True, False, False, False, False, False, False,  True,
       False, False,  True, False, False, False, False, False, False,
       False, False,  True,  True, False, False, False, False, False,
       False,  True, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False,  True,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False, False, False, False,  True, False, False, False, False,
       False, False, False, False,  True, False,  True, False, False,
       False,  True,  True, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False,  True, False, False,
       False,  True, False,  True, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False])
```

```
arSch[arSch>=100]
```

```
array([100, 104, 138, 141, 186, 146, 141, 123, 120, 122, 100, 152, 105,
       122, 100, 117, 141, 139, 114, 127, 110, 146, 101, 140, 113, 102])
```

```
np.mean(arSch[arSch>=100])
```

```
125.0
```

5 Find mean of sachin’s scores when Rahul has scored less then 10

```
SchRl10=[]
for i in range(len(arSch)) :
```

```
    if (arRah[i]<10) :
        x=arSch[i]
        SchRl10.append(x)
arSchRl10 = np.array(SchRl10)
arSchRl10
```

```
array([ 37,  14,   4,   0,  62,  38,  65,   0,  48,  62,  27,  27,  51,
        18,  65,  28,   2,  54,   4,  14, 100,  57,   0,   5,   0,  30,
        83,  93,   0, 152,   0,   1,  53,   0,  81,  78, 122,   9,  13,
        36,  29,  12,  34, 100,  44,  82,   0,   6,  49,  64,  43,  72,
        44,  47,  17,  35,  88,   0,  33, 110, 146,   7,  11,  94,  55,
         0,  28,   2,  27,  40])
```

```
arSchRl10Mean = np.mean(arSchRl10)
arSchRl10Mean
```

```
40.74285714285714
```

EFFICIENT WAY

```
arSch[arRah<10]
```

```
array([ 37,  14,   4,   0,  62,  38,  65,   0,  48,  62,  27,  27,  51,
        18,  65,  28,   2,  54,   4,  14, 100,  57,   0,   5,   0,  30,
        83,  93,   0, 152,   0,   1,  53,   0,  81,  78, 122,   9,  13,
        36,  29,  12,  34, 100,  44,  82,   0,   6,  49,  64,  43,  72,
        44,  47,  17,  35,  88,   0,  33, 110, 146,   7,  11,  94,  55,
         0,  28,   2,  27,  40])
```

```
np.mean(arSch[arRah<10])
```

```
40.74285714285714
```

6 Find mean of sachin’s scores based on which quartile India’s score falls in

```
IndIQR0 = np.percentile(arInd,0)
IndIQR0
```

```
0.0
```

```
IndIQR1 = np.percentile(arInd,25)
IndIQR1
```

```
175.0
```

```
IndIQR2 = np.percentile(arInd,50)
IndIQR2
```

```
216.0
```

```
IndIQR3 = np.percentile(arInd,75)
```

```
IndIQR3
```

```
273.0
```

```
IndIQR4 = np.percentile(arInd,100)
IndIQR4
```

```
499.0
```

```
arSchIQR1 = []
arSchIQR2 = []
arSchIQR3 = []
arSchIQR4 = []
for i in range(len(arSch)) :
    if(arInd[i] > IndIQR0 and arInd[i] < IndIQR1) :
        x= arSch[i]
        arSchIQR1.append(x)
    elif(arInd[i] > IndIQR1 and arInd[i] < IndIQR2) :
        x= arSch[i]
        arSchIQR2.append(x)
    elif(arInd[i] > IndIQR2 and arInd[i] < IndIQR3) :
        x= arSch[i]
        arSchIQR3.append(x)
    elif(arInd[i] > IndIQR3 and arInd[i] < IndIQR4) :
        x= arSch[i]
        arSchIQR4.append(x)
arSchIQR1 = np.array(arSchIQR1)
arSchIQR2 = np.array(arSchIQR2)
arSchIQR3 = np.array(arSchIQR3)
arSchIQR4 = np.array(arSchIQR4)
print(arSchIQR1)
print("__________________")
print(arSchIQR2)
print("__________________")
print(arSchIQR3)
print("__________________")
print(arSchIQR4 )
```

```
[18 14 62 46 65  0 39 48  3 11 65 27 28  3  4 15 40  5  8 89  0  1  0 81
 13  2 36 12 19  0  6 35  0 44  3 47 17 35 33  7  9  2 11 17  1 10 23  1
  2 25  0]
__________________
[ 11   4  21   2   5   7   2   6   7   2 100  32   0   5   0  30  37  52
  93  45   3  78   6  93   2  34   2   8  23  74 117  49  64   5  17  68
  88 127   0  45 110  25  45   9  67  55   1 101   1  41  28  22  27  40]
__________________
[  8 104  18   8  85   0   1   0  38  62  12   1  41  51 146  45  12   0
  54 122   0   0  99  37  38  32  21  50  98  93   1   1  53  14 122   8
  69  39  29 100   9   8  43  65  20  28  27  44   6 114  14   1  16  29]
__________________
[100  71  86   4   7  37 138 141 186  27  27  18  32 141  16 123  81  14
   4  83 152   8  26  47  89  16 105   9  35  97  44  79  93  63  39  72
 141  60 139  31   2   7 146  94  28   4  62   1  99   2 140 113  10   1
 102]
```

```
SchMeanIndIQR1 = np.mean(arSchIQR1)
SchMeanIndIQR2 = np.mean(arSchIQR2)
SchMeanIndIQR3 = np.mean(arSchIQR3)
SchMeanIndIQR4 = np.mean(arSchIQR4)
```

```
SchMeanIndIQR1
```

```
21.215686274509803
```

```
SchMeanIndIQR2
```

```
35.851851851851855
```

```
SchMeanIndIQR3
```

```
39.55555555555556
```

```
SchMeanIndIQR4
```

```
63.49090909090909
```

EFFICIENT WAY

```
arIndQuar=np.percentile(arInd,[25,50,75,100])
arIndQuar
```

```
array([175., 216., 273., 499.])
```

```
  #broadcast rule one should match , one should be 0
  np.mean(arSch[arInd<arIndQuar[0]])
```

```
19.672727272727272
```

```
  np.mean(arSch[arInd<arIndQuar[1]])
```

```
28.18018018018018
```

```
  np.mean(arSch[arInd<arIndQuar[2]])
```

```
31.688622754491018
```

```
  np.mean(arSch[arInd<arIndQuar[3]])
```

```
39.799107142857146
```

7 For every match findout who has scored more Sachin or Rahul

```
SachVsRah = []
#1 if Sachin more, 0 if Rahul More
for i in range(0,len(arSch)) :
    if arSch[i]>arRah[i] :
        SachVsRah.append(1)
    else :
        SachVsRah.append(0)
SachVsRah= np.array(SachVsRah)
SachVsRah
```

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

EFFICIENT WAY

```
np.max(arSch)
```

```
186
```

```
np.argmax(arSch) #return index of max value
```

```
37
```

```
SachRahData = cric_data[:,1:3]
SachRahData
```

```
       [ 31,  54],
       [  2,  52],
       [ 44,   9],
       [ 47,   0],
       [  6,  18],
       [ 17,   3],
       [ 35,   4],
       [ 88,   6],
       [114,  16],
       [  7,  71],
       [127,  64],
       [  0,   7],
       [ 45,  61],
       [ 33,   1],
       [110,   7],
       [146,   0],
       [  7,   0],
       [ 25,  26],
       [  9,  19],
       [  2,  63],
       [ 11,   3],
```

```
       [ 17,  46],
       [ 14,  31],
       [  1,  12],
       [ 45,  49],
       [ 94,   4],
       [ 28,  99],
       [  4,  47],
       [ 62,  56],
       [  1,  73],
       [  9,  74],
       [ 67,  31],
       [ 10,  29],
       [ 99,  92],
       [ 55,   1],
       [  0,   0],
       [  1,  30],
       [101,  16],
       [  2, 145],
       [  1,  22],
       [ 16,  31],
       [140, 104],
       [ 23,  20],
       [113,  13],
       [ 10,  52],
       [  1,  31],
       [ 41,  12],
       [  1,  50],
       [ 28,   3],
       [  2,   9],
       [ 22,  53],
       [ 29,  49],
       [  1,  21],
       [ 25,  14],
       [102,  50],
       [  0,  22],
       [ 27,   0],
       [ 40,   0]])
```

```
1 is_rahul = np.argmax(SachRahData,axis=1)
2 is_rahul
```

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

```
1 np.where(is_rahul ==0 ,"Sachin" ,"Rahul")
```

```
array(['Sachin', 'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Rahul',
       'Sachin', 'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin',
       'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Sachin',
```

```
       'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Sachin',
       'Sachin', 'Sachin', 'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Sachin',
       'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Sachin',
       'Sachin', 'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Rahul',
       'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Rahul',
       'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Rahul',
       'Sachin', 'Sachin', 'Sachin', 'Sachin', 'Rahul', 'Sachin', 'Rahul',
       'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Rahul',
       'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Rahul', 'Rahul',
       'Rahul', 'Sachin', 'Sachin', 'Sachin', 'Rahul', 'Sachin', 'Sachin',
       'Sachin', 'Sachin', 'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Sachin',
       'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Rahul',
       'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Sachin', 'Sachin',
       'Rahul', 'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Sachin', 'Rahul',
       'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Sachin', 'Sachin', 'Rahul',
       'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Sachin',
       'Rahul', 'Sachin', 'Sachin', 'Sachin', 'Sachin', 'Sachin',
       'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Sachin',
       'Sachin', 'Rahul', 'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Sachin',
       'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Sachin',
       'Rahul', 'Sachin', 'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Sachin',
       'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Rahul', 'Sachin', 'Sachin',
       'Sachin', 'Sachin', 'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Sachin',
       'Sachin', 'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin',
       'Rahul', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Rahul',
       'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Sachin', 'Sachin',
       'Sachin', 'Rahul', 'Sachin', 'Rahul', 'Rahul', 'Rahul', 'Sachin',
       'Sachin', 'Sachin', 'Rahul', 'Rahul', 'Sachin', 'Rahul', 'Sachin',
       'Rahul', 'Rahul', 'Rahul', 'Rahul', 'Sachin', 'Sachin', 'Rahul',
       'Sachin', 'Sachin'], dtype='<U6')
```

```
1 np.count_nonzero(is_rahul)
2 #rahul more in
```

```
    106
```

```
1 #we can also use arSch>arRah , then apply where
```

Out of 225 matches

Sachin Scored more than Rahul in 111 matches

Rahul Scored more than Sachin in 114 matches

8 How many more runs does sachin score on an average after scoring X runs

```
1  #if 10 , how many more runs he score ' #20, 30
2  arSchsort = np.sort(arSch)
3  arSchsort
```

```
    array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
             0,   0,   0,   0,   0,   1,   1,   1,   1,   1,   1,   1,   1,
             1,   1,   1,   1,   2,   2,   2,   2,   2,   2,   2,   2,   2,
             2,   3,   3,   3,   3,   4,   4,   4,   4,   4,   5,   5,   5,
```

```
 5,   6,   6,   6,   6,   7,   7,   7,   7,   7,   8,   8,   8,
 8,   8,   8,   8,   9,   9,   9,   9,  10,  10,  11,  11,  11,
11,  12,  12,  12,  12,  13,  14,  14,  14,  14,  15,  16,  16,
16,  16,  17,  17,  17,  18,  18,  18,  19,  20,  21,  21,  22,
23,  23,  25,  25,  26,  27,  27,  27,  27,  27,  28,  28,  28,
28,  28,  29,  29,  30,  31,  32,  32,  32,  33,  34,  35,  35,
35,  36,  37,  37,  37,  38,  38,  39,  39,  39,  40,  40,  41,
41,  43,  44,  44,  44,  45,  45,  45,  45,  46,  47,  47,  48,
49,  50,  51,  52,  53,  54,  55,  57,  60,  62,  62,  62,  63,
64,  65,  65,  65,  67,  68,  69,  71,  72,  74,  78,  79,  81,
81,  82,  83,  85,  86,  88,  89,  89,  93,  93,  93,  93,  94,
97,  98,  99,  99, 100, 100, 100, 101, 102, 104, 105, 110, 113,
114, 117, 120, 122, 122, 123, 127, 138, 139, 140, 141, 141, 141,
146, 146, 152, 186])
```

```
SchFreqCounter = np.histogram(arSchsort,bins=[0,20,40,60,80,100,120,140,160,180,200])
SchFreqCounter
```

```
(array([100,  40,  24,  17,  18,  11,   7,   7,   0,   1]),
 array([  0,  20,  40,  60,  80, 100, 120, 140, 160, 180, 200]))
```

```
SchFreqCounterCount = SchFreqCounter.count
SchFreqCounterCount
```

```
<function tuple.count>
```

```
SchFreqCounterCount = np.array([100,  40,  24,  17,  18,  11,   7,   7,   0,   1])
SchFreqCounterCount
```

```
array([100,  40,  24,  17,  18,  11,   7,   7,   0,   1])
```

```
np.sum(SchFreqCounterCount)
```

```
225
```

```
SchRFraw=(SchFreqCounterCount/(np.sum(SchFreqCounterCount)))
SchRFraw
```

```
array([0.44444444, 0.17777778, 0.10666667, 0.07555556, 0.08      ,
       0.04888889, 0.03111111, 0.03111111, 0.        , 0.00444444])
```

```
SchRFbinend= np.array([0,20,40,60,80,100,120,140,160,180])
SchRFbinend
```

```
array([  0,  20,  40,  60,  80, 100, 120, 140, 160, 180])
```

```
(SchRFraw*SchRFbinend)
```

```
array([0.        , 3.55555556, 4.26666667, 4.53333333, 6.4       ,
       4.88888889, 3.73333333, 4.35555556, 0.        , 0.8       ])
```

```
(SchRFraw*SchRFbinend)+SchRFbinend
```

```
array([  0.        ,  23.55555556,  44.26666667,  64.53333333,
        86.4       , 104.88888889, 123.73333333, 144.35555556,
       160.        , 180.8       ])
```

CORRECT AND EFFICIENT WAY OF ANSWER

```
# WHEN INTO GROUND EXPECATION IS AVG =40
#0 - 40
#1-10 MORE VULNEERABLE
#10 MORE THAN 43
#0 - AFTER 0 AVERAGE
# 10 AFTER 10 AVG , EXCLUDING
print("Runs scored + More Runs He Score = expected Average score")
for i in range(0,max(arSch),5) :
    print(i,"+",(int(np.mean(arSch[arSch>=i]-i))),"=",int(np.mean(arSch[arSch>=i])))
```

```
Runs scored + More Runs He Score = expected Average score
0 + 39 = 39
5 + 45 = 50
10 + 47 = 57
15 + 47 = 62
20 + 46 = 66
25 + 44 = 69
30 + 45 = 75
35 + 43 = 78
40 + 44 = 84
45 + 43 = 88
50 + 43 = 93
55 + 42 = 97
60 + 38 = 98
65 + 37 = 102
70 + 37 = 107
75 + 34 = 109
80 + 30 = 110
85 + 28 = 113
90 + 27 = 117
95 + 26 = 121
100 + 25 = 125
105 + 27 = 132
110 + 23 = 133
115 + 22 = 137
120 + 18 = 138
125 + 20 = 145
130 + 17 = 147
135 + 12 = 147
140 + 9 = 149
145 + 12 = 157
150 + 19 = 169
155 + 31 = 186
160 + 26 = 186
165 + 21 = 186
170 + 16 = 186
175 + 11 = 186
180 + 6 = 186
185 + 1 = 186
```

9 How many matches did sachin take to score first 1000 runs and then next 1000

```
1 cumsumSch = np.cumsum(arSch)
2 cumsumSch
```

```
array([ 100,  111,  119,  190,  294,  312,  320,  406,  418,  503,  521,
        525,  532,  569,  583,  583,  587,  587,  608,  609,  671,  671,
        809,  847,  849,  895,  960,  960,  999, 1047, 1188, 1250, 1262,
       1263, 1304, 1315, 1318, 1504, 1515, 1542, 1569, 1620, 1638, 1670,
       1816, 1821, 1866, 2007, 2019, 2084, 2111, 2118, 2134, 2136, 2164,
       2170, 2293, 2413, 2420, 2423, 2423, 2504, 2506, 2560, 2682, 2686,
       2700, 2700, 2800, 2815, 2815, 2872, 2971, 3008, 3046, 3078, 3099,
       3131, 3171, 3171, 3176, 3184, 3189, 3189, 3239, 3269, 3306, 3395,
       3399, 3497, 3580, 3673, 3673, 3725, 3877, 3878, 3886, 3979, 4024,
       4050, 4050, 4051, 4051, 4067, 4114, 4203, 4206, 4207, 4260, 4276,
       4276, 4357, 4371, 4449, 4455, 4560, 4682, 4691, 4699, 4727, 4762,
       4831, 4844, 4941, 5034, 5036, 5072, 5111, 5113, 5142, 5154, 5173,
       5207, 5209, 5309, 5353, 5435, 5435, 5514, 5520, 5529, 5537, 5560,
       5653, 5688, 5751, 5825, 5833, 5950, 5989, 6038, 6102, 6145, 6217,
       6222, 6239, 6239, 6304, 6324, 6465, 6493, 6537, 6564, 6624, 6692,
       6695, 6834, 6865, 6867, 6911, 6958, 6964, 6981, 7016, 7104, 7218,
       7225, 7352, 7352, 7397, 7430, 7540, 7686, 7693, 7718, 7727, 7729,
       7740, 7757, 7771, 7772, 7817, 7911, 7939, 7943, 8005, 8006, 8015,
       8082, 8092, 8191, 8246, 8246, 8247, 8348, 8350, 8351, 8367, 8507,
       8530, 8643, 8653, 8654, 8695, 8696, 8724, 8726, 8748, 8777, 8778,
       8803, 8905, 8905, 8932, 8972])
```

```
1   np.histogram(cumsumSch,bins=range(0,max(cumsumSch),1000))
```

```
(array([29, 18, 26, 25, 26, 26, 23, 22]),
 array([   0, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000]))
```