Class03

September 9, 2024

1 Working with NUMPY

Case Study - Cricket Tournament

A panel wants to select players for an upcoming league match based on their fitness. Players from all significant cricket clubs have participated in a practice match, and their data is collected. Let us now explore NumPy features using the player's data.

Example - 1

Heights of the players is stored as a regular Python list: height_in. The height is expressed in inches. Can you make a numpy array out of it ?

[3]:
```
# Define list
```

```
heights = [74, 74, 72, 72, 73, 69, 69, 71, 76, 71, 73, 73, 74, 74, 69, 70, 73,␣
↪75, 78, 79, 76, 74, 76, 72, 71, 75, 77, 74, 73, 74, 78, 73, 75, 73, 75, 75,␣
↪74, 69, 71, 74, 73, 73, 76, 74, 74, 70, 72, 77, 74, 70, 73, 75, 76, 76, 78,␣
↪74, 74, 76, 77, 81, 78, 75, 77, 75, 76, 74, 72, 72, 75, 73, 73, 73, 70, 70,␣
↪70, 76, 68, 71, 72, 75, 75, 75, 75, 68, 74, 78, 71, 73, 76, 74, 74, 79, 75,␣
↪73, 76, 74, 74, 73, 72, 74, 73, 74, 72, 73, 69, 72, 73, 75, 75, 73, 72, 72,␣
↪76, 74, 72, 77, 74, 77, 75, 76, 80, 74, 74, 75, 78, 73, 73, 74, 75, 76, 71,␣
↪73, 74, 76, 76, 74, 73, 74, 70, 72, 73, 73, 73, 73, 71, 74, 74, 72, 74, 71,␣
↪74, 73, 75, 75, 79, 73, 75, 76, 74, 76, 78, 74, 76, 72, 74, 76, 74, 75, 78,␣
↪75, 72, 74, 72, 74, 70, 71, 70, 75, 71, 71, 73, 72, 71, 73, 72, 75, 74, 74,␣
↪75, 73, 77, 73, 76, 75, 74, 76, 75, 73, 71, 76, 75, 72, 71, 77, 73, 74, 71,␣
↪72, 74, 75, 73, 72, 75, 75, 74, 72, 74, 71, 70, 74, 77, 77, 75, 75, 78, 75,␣
↪76, 73, 75, 75, 79, 77, 76, 71, 75, 74, 69, 71, 76, 72, 72, 70, 72, 73, 71,␣
↪72, 71, 73, 72, 73, 74, 74, 72, 75, 74, 74, 77, 75, 73, 72, 71, 74, 77, 75,␣
↪75, 75, 78, 78, 74, 76, 78, 76, 70, 72, 80, 74, 74, 71, 70, 72, 71, 74, 71,␣
↪72, 71, 74, 69, 76, 75, 75, 76, 73, 76, 73, 77, 73, 72, 72, 77, 77, 71, 74,␣
↪74, 73, 78, 75, 73, 70, 74, 72, 73, 73, 75, 75, 74, 76, 73, 74, 75, 75, 72,␣
↪73, 73, 72, 74, 78, 76, 73, 74, 75, 70, 75, 71, 72, 78, 75, 73, 73, 71, 75,␣
↪77, 72, 69, 73, 74, 72, 70, 75, 70, 72, 72, 74, 73, 74, 76, 75, 80, 72, 75,␣
↪73, 74, 74, 73, 75, 75, 71, 73, 75, 74, 74, 72, 74, 74, 74, 73, 76, 75, 72,␣
↪73, 73, 73, 72, 72, 72, 72, 71, 75, 75, 74, 73, 75, 79, 74, 76, 73, 74, 74,␣
↪72, 74, 74, 75, 78, 74, 74, 74, 77, 70, 73, 74, 73, 71, 75, 71, 72, 77, 74,␣
↪70, 77, 73, 72, 76, 71, 76, 78, 75, 73, 78, 74, 79, 75, 76, 72, 75, 75, 70,␣
↪72, 70, 74, 71, 76, 73, 76, 71, 69, 72, 72, 69, 73, 69, 73, 74, 74, 72, 71,␣
↪72, 72, 76, 76, 76, 74, 76, 75, 71, 72, 71, 73, 75, 76, 75, 71, 75, 74, 72,␣
↪73, 73, 73, 73, 76, 72, 76, 73, 73, 73, 75, 75, 77, 73, 72, 75, 70, 74, 72,␣
↪80, 71, 71, 74, 74, 73, 75, 76, 73, 77, 72, 73, 77, 76, 71, 75, 73, 74, 77,␣
↪71, 72, 73, 69, 73, 70, 74, 76, 73, 73, 75, 73, 79, 74, 73, 74, 77, 75, 74,␣
↪73, 77, 73, 77, 74, 74, 73, 77, 74, 77, 75, 77, 75, 71, 74, 70, 79, 72, 72,␣
↪70, 74, 74, 72, 73, 72, 74, 74, 76, 82, 74, 74, 70, 73, 73, 74, 77, 72, 76,␣
↪73, 73, 72, 74, 74, 71, 72, 75, 74, 74, 77, 70, 71, 73, 76, 71, 75, 74, 72,␣
↪76, 79, 76, 73, 76, 78, 75, 76, 72, 72, 73, 73, 75, 71, 76, 70, 75, 74, 75,␣
↪73, 71, 71, 72, 73, 73, 72, 69, 73, 78, 71, 73, 75, 76, 70, 74, 77, 75, 79,␣
↪72, 77, 73, 75, 75, 75, 73, 73, 76, 77, 75, 70, 71, 71, 75, 74, 69, 70, 75,␣
↪72, 75, 73, 72, 72, 72, 76, 75, 74, 69, 73, 72, 72, 75, 77, 76, 80, 77, 76,␣
↪79, 71, 75, 73, 76, 77, 73, 76, 70, 75, 73, 75, 70, 69, 71, 72, 72, 73, 70,␣
↪76, 72, 77, 75, 72, 71, 71, 75, 72, 73, 73, 71, 70, 75, 71, 76, 73, 68, 71,␣
↪72, 74, 77, 71, 72, 76, 78, 81, 72, 73, 76, 72, 72, 74, 76, 73, 76, 75, 70, 71,␣
↪74, 72, 73, 76, 76, 73, 71, 68, 71, 71, 74, 77, 69, 72, 76, 75, 76, 75, 76,␣
↪72, 74, 76, 74, 72, 75, 78, 77, 70, 72, 79, 74, 71, 68, 77, 75, 71, 72, 70,␣
↪72, 72, 73, 72, 74, 72, 72, 75, 72, 73, 74, 72, 78, 75, 72, 74, 75, 75, 76,␣
↪74, 74, 73, 74, 71, 74, 75, 76, 74, 76, 76, 73, 75, 75, 74, 68, 72, 75, 71,␣
↪70, 72, 73, 72, 75, 74, 70, 76, 71, 82, 72, 73, 74, 71, 75, 77, 72, 74, 72,␣
↪73, 78, 77, 73, 73, 73, 73, 73, 76, 75, 70, 73, 72, 73, 75, 74, 73, 73, 76,␣
```

[4]:
```
import numpy as np
heights_in = np.array(heights)
heights_in
```

[4]:
```
array([74, 74, 72, 72, 73, 69, 69, 71, 76, 71, 73, 73, 74, 74, 69, 70, 73,
       75, 78, 79, 76, 74, 76, 72, 71, 75, 77, 74, 73, 74, 78, 73, 75, 73,
       75, 75, 74, 69, 71, 74, 73, 73, 76, 74, 74, 70, 72, 77, 74, 70, 73,
       75, 76, 76, 78, 74, 74, 76, 77, 81, 78, 75, 77, 75, 76, 74, 72, 72,
       75, 73, 73, 73, 70, 70, 70, 76, 68, 71, 72, 75, 75, 75, 75, 68, 74,
       78, 71, 73, 76, 74, 74, 79, 75, 73, 76, 74, 74, 73, 72, 74, 73, 74,
       72, 73, 69, 72, 73, 75, 75, 73, 72, 72, 76, 74, 72, 77, 74, 77, 75,
       76, 80, 74, 74, 75, 78, 73, 73, 74, 75, 76, 71, 73, 74, 76, 76, 74,
       73, 74, 70, 72, 73, 73, 73, 73, 71, 74, 74, 72, 74, 71, 74, 73, 75,
       75, 79, 73, 75, 76, 74, 76, 78, 74, 76, 72, 74, 76, 74, 75, 78, 75,
       72, 74, 72, 74, 70, 71, 70, 75, 71, 71, 73, 72, 71, 73, 72, 75, 74,
       74, 75, 73, 77, 73, 76, 75, 74, 76, 75, 73, 71, 76, 75, 72, 71, 77,
       73, 74, 71, 72, 74, 75, 73, 72, 75, 75, 74, 72, 74, 71, 70, 74, 77,
       77, 75, 75, 78, 75, 76, 73, 75, 75, 79, 77, 76, 71, 75, 74, 69, 71,
       76, 72, 72, 70, 72, 73, 71, 72, 71, 73, 72, 74, 74, 72, 75, 74, 74,
       77, 75, 73, 72, 71, 74, 77, 75, 75, 75, 78, 78, 74, 76, 78, 76, 70,
       72, 80, 74, 74, 71, 70, 72, 71, 74, 71, 72, 71, 74, 69, 76, 75, 75,
       76, 73, 76, 73, 77, 73, 72, 72, 77, 77, 71, 74, 74, 73, 78, 75, 73,
       70, 74, 72, 73, 73, 75, 75, 74, 76, 73, 74, 75, 75, 72, 73, 73, 72,
       74, 78, 76, 73, 74, 75, 70, 75, 71, 72, 78, 75, 73, 73, 71, 75, 77,
       72, 69, 73, 74, 72, 70, 75, 70, 72, 72, 74, 73, 74, 76, 75, 80, 72,
       75, 73, 74, 74, 73, 75, 75, 71, 73, 75, 74, 74, 72, 74, 74, 74, 73,
       76, 75, 72, 73, 73, 73, 72, 72, 72, 72, 71, 75, 75, 74, 73, 75, 79,
       74, 76, 73, 74, 74, 72, 74, 74, 75, 78, 74, 74, 74, 77, 70, 73, 74,
       73, 71, 75, 71, 72, 77, 74, 70, 77, 73, 72, 76, 71, 76, 78, 75, 73,
       78, 74, 79, 75, 76, 72, 75, 75, 70, 72, 70, 74, 71, 76, 73, 76, 71,
       69, 72, 72, 69, 73, 69, 73, 74, 74, 72, 71, 72, 72, 76, 76, 76, 74,
       76, 75, 71, 72, 71, 73, 75, 76, 75, 71, 75, 74, 72, 73, 73, 73, 73,
       76, 72, 76, 73, 73, 73, 75, 75, 77, 73, 72, 75, 70, 74, 72, 80, 71,
       71, 74, 74, 73, 75, 76, 73, 77, 72, 73, 77, 76, 71, 75, 73, 74, 77,
       71, 72, 73, 69, 73, 70, 74, 76, 73, 73, 75, 73, 79, 74, 73, 74, 77,
       75, 74, 73, 77, 73, 77, 74, 74, 73, 77, 74, 77, 75, 77, 75, 71, 74,
       70, 79, 72, 72, 70, 74, 74, 72, 73, 72, 74, 74, 76, 82, 74, 74, 70,
       73, 73, 74, 77, 72, 76, 73, 73, 72, 74, 74, 71, 72, 75, 74, 74, 77,
       70, 71, 73, 76, 71, 75, 74, 72, 76, 79, 76, 73, 76, 78, 75, 76, 72,
       72, 73, 73, 75, 71, 76, 70, 75, 74, 75, 73, 71, 71, 72, 73, 73, 72,
       69, 73, 78, 71, 73, 75, 76, 70, 74, 77, 75, 79, 72, 77, 73, 75, 75,
       75, 73, 73, 76, 77, 75, 70, 71, 71, 75, 74, 69, 70, 75, 72, 75, 73,
       72, 72, 72, 76, 75, 74, 69, 73, 72, 72, 75, 77, 76, 80, 77, 76, 79,
       71, 75, 73, 76, 77, 73, 76, 70, 75, 73, 75, 70, 69, 71, 72, 72, 73,
       70, 76, 72, 77, 75, 72, 71, 71, 75, 72, 73, 73, 71, 70, 75, 71, 76,
       73, 68, 71, 72, 74, 77, 71, 72, 76, 78, 81, 72, 73, 76, 72, 72, 74,
       76, 73, 76, 75, 70, 71, 74, 72, 73, 76, 76, 73, 71, 68, 71, 71, 74,
       77, 69, 72, 76, 75, 76, 75, 76, 72, 74, 76, 74, 72, 75, 78, 77, 70,
       72, 79, 74, 71, 68, 77, 75, 71, 72, 70, 72, 72, 73, 72, 74, 72, 72,
       75, 72, 73, 74, 72, 78, 75, 72, 74, 75, 75, 76, 74, 74, 73, 74, 71,
       74, 75, 76, 74, 76, 76, 73, 75, 75, 74, 68, 72, 75, 71, 70, 72, 73,
       72, 75, 74, 70, 76, 71, 82, 72, 73, 74, 71, 75, 77, 72, 74, 72, 73,
       78, 77, 73, 73, 73, 73, 73, 76, 75, 70, 73, 72, 73, 75, 74, 73, 73,
       76])
```

1.0.1 Count of participants

[6]:
```
len(heights_in)
```

[6]:
```
1015
```

[7]:
```
heights_in.size
```

[7]:
```
1015
```

[8]:
```
heights_in.shape
```

[8]:
```
(1015,)
```

1.0.2 Convert inches into meters

[10]:
```
height_m = heights_in * 0.0245
height_m
```

[10]:
```
array([1.813 , 1.813 , 1.764 , …, 1.8375, 1.8375, 1.7885])
```

1.0.3 Weights of the players

[12]:
```

```

```
weights_lb = [180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185,␣
↪160, 180, 185, 189, 185, 219, 230, 205, 230, 195, 180, 192, 225, 203, 195,␣
↪182, 188, 200, 180, 200, 200, 245, 240, 215, 185, 175, 199, 200, 215, 200,␣
↪205, 206, 186, 188, 220, 210, 195, 200, 200, 212, 224, 210, 205, 220, 195,␣
↪200, 260, 228, 270, 200, 210, 190, 220, 180, 205, 210, 220, 211, 200, 180,␣
↪190, 170, 230, 155, 185, 185, 200, 225, 225, 220, 160, 205, 235, 250, 210,␣
↪190, 160, 200, 205, 222, 195, 205, 220, 220, 170, 185, 195, 220, 230, 180,␣
↪220, 180, 180, 170, 210, 215, 200, 213, 180, 192, 235, 185, 235, 210, 222,␣
↪211, 190, 210, 190, 190, 185, 290, 175, 185, 200, 220, 170, 220, 190, 220,␣
↪205, 210, 230, 225, 215, 210, 215, 195, 200, 194, 220, 180, 180, 170, 195,␣
↪180, 170, 206, 205, 200, 225, 201, 225, 233, 180, 225, 180, 220, 180, 237,␣
↪215, 190, 235, 190, 180, 165, 195, 200, 190, 190, 185, 185, 205, 190, 205,␣
↪206, 220, 208, 170, 195, 210, 190, 211, 230, 170, 185, 185, 241, 225, 210,␣
↪200, 210, 215, 200, 205, 211, 190, 208, 200, 210, 232, 230, 210, 220, 210,␣
↪202, 212, 225, 170, 190, 200, 237, 220, 170, 193, 190, 150, 220, 200, 190,␣
↪185, 185, 200, 172, 220, 225, 190, 195, 219, 190, 197, 200, 195, 210, 177,␣
↪220, 235, 180, 195, 195, 190, 230, 190, 200, 190, 190, 200, 200, 184, 200,␣
↪180, 219, 187, 200, 220, 205, 190, 170, 160, 215, 175, 205, 200, 214, 200,␣
↪190, 180, 205, 220, 190, 215, 235, 191, 200, 181, 200, 210, 240, 185, 165,␣
↪190, 185, 175, 155, 210, 170, 175, 220, 210, 205, 200, 205, 195, 240, 150,␣
↪190, 202, 205, 220, 175, 160, 190, 200, 229, 206, 220, 180, 195, 175, 188,␣
↪230, 190, 200, 190, 219, 235, 180, 180, 180, 200, 234, 185, 220, 223, 200,␣
↪210, 200, 210, 190, 177, 227, 180, 195, 199, 175, 185, 240, 210, 180, 194,␣
↪225, 180, 205, 193, 230, 230, 220, 200, 249, 190, 208, 245, 250, 160, 192,␣
↪220, 170, 197, 155, 190, 200, 220, 210, 228, 190, 160, 184, 180, 180, 200,␣
↪176, 160, 222, 211, 195, 200, 175, 206, 240, 185, 260, 185, 221, 205, 200,␣
↪170, 201, 205, 185, 205, 245, 220, 210, 220, 185, 175, 170, 180, 200, 210,␣
↪175, 220, 206, 180, 210, 195, 200, 200, 164, 180, 220, 195, 205, 170, 240,␣
```

[13]:
```
weight_kgs = np.array(weights_lb) * 0.453592
weight_kgs
```

[13]:
```
array([81.64656 , 97.52228 , 95.25432 , ..., 92.98636 , 86.18248 , 88.45044])
```

1.0.4 Calculating BMI

[15]:
```
bmi = weight_kgs / (height_m ** 2)
bmi
```

[15]:
```
array([24.83946761, 29.66936409, 30.611705  , ..., 27.54003906,
       25.52491425, 27.65171733])
```

1.0.5 Sub setting in arrays

[17]:
```
bmi[0] # fetching the 1st element in the BMI array
```

[17]:
```
24.83946760678302
```

[18]:
```
bmi[-1] # fetching the last element in the BMI array
```

[18]:
```
27.651717332702667
```

[19]:
```
bmi[0:5] # fetching the 1st 5 elements from the BMI array
```

[19]:
```
array([24.83946761, 29.66936409, 30.611705  , 30.611705  , 26.65909158])
```

1.0.6 conditional subsettting

[21]:
```
bmi < 21
```

[21]:
```
array([False, False, False, …, False, False, False])
```

[22]:
```
bmi[bmi < 21]
```

[22]:
```
array([20.95729801])
```

[23]:
```
under_weight = bmi[bmi < 21]
under_weight
```

[23]:
```
array([20.95729801])
```

[24]:
```
under_weight.shape
```

[24]:
```
(1,)
```

1.0.7 Largest BMI

[26]:
```
max(bmi)
```

[26]:
```
37.90020618980774
```

[27]:
```
bmi.max()
```

[27]:
```
37.90020618980774
```

1.0.8 lowest BMI

[29]:
```
min(bmi)
```

[29]:
```
20.957298014716088
```

[30]:
```
bmi.min()
```

[30]:
```
20.957298014716088
```

1.0.9 Players list containing both height and weight

[32]:
```
# list of height and weight of the players. 2D arrays
```

[33]:
```
len(players)
```

[33]: 1015

[34]:
```
players[1][1]
```

[34]: 215

[35]:
```
players[0][0]
```

[35]: 74

[36]:
```
players[100][0]
```

[36]: 73

[37]:
```
players_array = np.array(players)
print(players_array)
```

```
[[ 74 180]
 [ 74 215]
 [ 72 210]
 …
 [ 75 205]
 [ 75 190]
 [ 73 195]]
```

[38]:
```
type(players_array)
```

[38]: numpy.ndarray

[39]:
```
players_array.shape
```

[39]: (1015, 2)

[40]:
```
players_array.ndim
```

[40]: 2

[41]:
```
players_array.dtype
```

[41]: dtype('int64')

[42]:
```
players_array.itemsize
```

[42]: 8

1.0.10 Convert the heights into meters and weights into kgs

[44]:
```
players_converted = players_array * [0.0245, 0.453592]
players_converted
```

[44]:
```
array([[ 1.813   , 81.64656],
       [ 1.813   , 97.52228],
       [ 1.764   , 95.25432],
       …,
       [ 1.8375  , 92.98636],
       [ 1.8375  , 86.18248],
       [ 1.7885  , 88.45044]])
```

[45]:
```
players_converted[0]
```

[45]:
```
array([ 1.813   , 81.64656])
```

[46]:
```
players_converted[0][1]
```

[46]:
```
81.64656
```

[47]:
```
players_converted[99][0] # fetching the 100th players height
```

[47]:
```
1.8130000000000002
```

[48]:
```
players_converted[99][1] # fetching the 100th players weight
```

[48]:
```
88.45044
```

[49]:
```
players_converted[999][0] # fetching the 1000th players data
```

[49]:
```
1.911
```

[50]:
```
players_converted[999][1]
```

[50]:
```
94.347136
```

[51]:
```
players_converted[:,0] # fetching the 1st column
```

[51]:
```
array([1.813 , 1.813 , 1.764 , …, 1.8375, 1.8375, 1.7885])
```

[52]:
```
players_converted[:,1] # fetching the 2nd column
```

[52]:
```
array([81.64656, 97.52228, 95.25432, …, 92.98636, 86.18248, 88.45044])
```

[53]:
```
players_converted[124]
```

[53]:
```
array([ 1.911   , 95.25432])
```

[54]:
```
players_converted[124][0]
```

[54]:
```
1.911
```

1.0.11 Conditional subseting in 2D arrays

[56]:
```
tall_players= players_converted[players_converted[:,0] > 1.8]
tall_players
```

[56]:
```
array([[  1.813   ,  81.64656 ],
       [  1.813   ,  97.52228 ],
       [  1.862   , 104.779752],
       …,
       [  1.813   ,  81.64656 ],
       [  1.8375  ,  92.98636 ],
       [  1.8375  ,  86.18248 ]])
```

[57]:
```
tall_players.shape
```

[57]:
```
(535, 2)
```

[58]:
```
len(tall_players)
```

[58]:
```
535
```

[59]:
```
over_weight = players_converted[players_converted[:,1] > 90]
over_weight
```

[59]:
```
array([[ 1.813  , 97.52228],
       [ 1.764  , 95.25432],
       [ 1.764  , 95.25432],
       …,
       [ 1.8375 , 99.79024],
       [ 1.8375 , 92.98636],
       [ 1.8375 , 92.98636]])
```

[60]:
```
over_weight.shape
```

[60]:
```
(568, 2)
```

1.0.12 Skills of the players

[62]:

```
skills
```

[62]:
```
array(['Keeper', 'Batsman', 'Bowler', …, 'Batsman', 'Bowler',
       'Keeper-Batsman'], dtype='<U14')
```

[63]:
```
skills.shape
```

[63]:
```
(1015,)
```

[64]:
```
batsmen = players_converted[skills == 'Batsman']
batsmen
```

[64]:
```
array([[  1.813   ,  97.52228 ],
       [  1.7885  ,  85.275296],
       [  1.6905  ,  94.800728],
       [  1.7395  ,  90.7184  ],
       [  1.862   , 104.779752],
       [  1.7885  ,  85.275296],
       [  1.7885  ,  81.64656 ],
       [  1.715   ,  83.91452 ],
       [  1.9355  , 104.32616 ],
       [  1.764   ,  81.64656 ],
       [  1.7395  ,  87.089664],
       [  1.8375  , 102.0582  ],
       [  1.8865  ,  92.079176],
       [  1.7885  ,  82.553744],
       [  1.8375  , 111.13004 ],
       [  1.813   ,  97.52228 ],
       [  1.7395  ,  79.3786  ],
       [  1.7885  ,  90.7184  ],
       [  1.813   ,  92.98636 ],
       [  1.862   ,  88.45044 ],
       [  1.8375  ,  95.25432 ],
       [  1.862   ,  86.18248 ],
       [  1.764   ,  81.64656 ],
       [  1.8375  ,  95.25432 ],
       [  1.7885  ,  99.79024 ],
       [  1.715   ,  81.64656 ],
       [  1.715   ,  86.18248 ],
       [  1.7395  ,  83.91452 ],
       [  1.764   ,  83.91452 ],
       [  1.8375  ,  90.7184  ],
       [  1.911   , 106.59412 ],
       [  1.7395  , 113.398   ],
       [  1.7885  ,  95.25432 ],
       [  1.7885  ,  88.45044 ],
       [  1.862   ,  92.98636 ],
```

```
[  1.813   ,  99.79024 ],
[  1.7885  ,  77.11064 ],
[  1.764   ,  83.91452 ],
[  1.7885  ,  99.79024 ],
[  1.764   ,  81.64656 ],
[  1.7885  ,  99.79024 ],
[  1.764   ,  81.64656 ],
[  1.8375  ,  95.25432 ],
[  1.8375  ,  97.52228 ],
[  1.764   ,  96.615096],
[  1.764   ,  81.64656 ],
[  1.8865  , 106.59412 ],
[  1.8865  , 100.697424],
[  1.8375  ,  95.25432 ],
[  1.862   , 104.32616 ],
[  1.96    ,  99.79024 ],
[  1.8375  ,  90.7184  ],
[  1.7885  ,  87.996848],
[  1.813   ,  86.18248 ],
[  1.8375  , 108.86208 ],
[  1.7395  ,  89.811216],
[  1.862   ,  95.25432 ],
[  1.813   ,  86.18248 ],
[  1.7885  ,  77.11064 ],
[  1.7885  ,  81.64656 ],
[  1.7395  ,  86.18248 ],
[  1.7885  ,  86.18248 ],
[  1.7885  ,  79.3786  ],
[  1.764   ,  92.98636 ],
[  1.8375  ,  97.52228 ],
[  1.715   ,  81.64656 ],
[  1.7395  ,  77.11064 ],
[  1.764   ,  92.98636 ],
[  1.8375  , 102.0582  ],
[  1.7885  ,  81.64656 ],
[  1.862   , 107.501304],
[  1.8375  ,  97.52228 ],
[  1.862   , 106.59412 ],
[  1.764   ,  86.18248 ],
[  1.7885  ,  94.347136],
[  1.8375  ,  95.25432 ],
[  1.813   ,  86.18248 ],
[  1.7395  ,  77.11064 ],
[  1.8865  , 102.0582  ],
[  1.911   , 104.32616 ],
[  1.8375  , 102.511792],
[  1.8375  ,  99.336648],
```

```
[  1.862   , 102.0582  ],
[  1.764   ,  79.832192],
[  1.764   ,  81.64656 ],
[  1.7885  ,  99.79024 ],
[  1.813   ,  86.18248 ],
[  1.813   ,  92.98636 ],
[  1.8375  ,  92.98636 ],
[  1.7885  ,  95.707912],
[  1.813   ,  90.7184  ],
[  1.8865  ,  95.25432 ],
[  1.911   ,  99.79024 ],
[  1.911   ,  95.25432 ],
[  1.813   ,  91.625584],
[  1.911   , 102.0582  ],
[  1.813   ,  99.79024 ],
[  1.764   ,  83.91452 ],
[  1.862   ,  88.45044 ],
[  1.7885  ,  99.336648],
[  1.7885  ,  88.45044 ],
[  1.8865  ,  99.79024 ],
[  1.8865  , 106.59412 ],
[  1.7395  ,  81.64656 ],
[  1.911   , 104.32616 ],
[  1.715   ,  86.18248 ],
[  1.813   ,  86.18248 ],
[  1.764   ,  90.7184  ],
[  1.7885  ,  83.460928],
[  1.8375  ,  90.7184  ],
[  1.813   ,  99.336648],
[  1.7885  ,  90.7184  ],
[  1.813   ,  99.79024 ],
[  1.7885  ,  72.57472 ],
[  1.911   ,  90.7184  ],
[  1.862   ,  97.068688],
[  1.7885  ,  90.7184  ],
[  1.8375  ,  81.64656 ],
[  1.911   , 106.59412 ],
[  1.8865  , 108.86208 ],
[  1.764   ,  83.91452 ],
[  1.6905  ,  74.84268 ],
[  1.813   ,  83.91452 ],
[  1.715   ,  77.11064 ],
[  1.813   ,  95.25432 ],
[  1.862   ,  92.98636 ],
[  1.8375  ,  90.7184  ],
[  1.7885  ,  86.18248 ],
[  1.7885  ,  72.57472 ],
```

```
[  1.7885  ,  81.64656 ],
[  1.813   ,  81.64656 ],
[  1.813   ,  90.7184  ],
[  1.715   ,  86.18248 ],
[  1.813   , 102.965384],
[  1.8375  ,  90.264808],
[  1.764   ,  83.91452 ],
[  1.8865  , 108.86208 ],
[  1.7885  , 102.0582  ],
[  1.862   , 104.32616 ],
[  1.7885  ,  90.7184  ],
[  1.8375  , 111.13004 ],
[  1.813   ,  86.18248 ],
[  1.7885  ,  95.25432 ],
[  1.7395  ,  90.7184  ],
[  1.862   ,  92.98636 ],
[  1.8375  ,  95.25432 ],
[  1.7885  ,  77.11064 ],
[  1.7885  ,  81.64656 ],
[  1.862   ,  95.25432 ],
[  1.7885  ,  93.439952],
[  1.8375  ,  88.45044 ],
[  1.8375  ,  90.7184  ],
[  1.8865  ,  90.7184  ],
[  1.813   ,  92.98636 ],
[  1.7395  ,  95.25432 ],
[  1.813   ,  90.7184  ],
[  1.813   ,  92.98636 ],
[  1.8375  ,  86.18248 ],
[  1.764   ,  92.98636 ],
[  1.7885  ,  79.3786  ],
[  1.862   ,  99.79024 ],
[  1.8865  , 111.13004 ],
[  1.7885  ,  81.64656 ],
[  1.715   ,  88.45044 ],
[  1.7885  ,  97.52228 ],
[  1.813   ,  81.64656 ],
[  1.8865  ,  86.18248 ],
[  1.8375  ,  86.18248 ],
[  1.7885  ,  95.25432 ],
[  1.8865  , 115.66596 ],
[  1.8865  , 104.32616 ],
[  1.813   ,  92.98636 ],
[  1.813   ,  97.52228 ],
[  1.8375  ,  99.79024 ],
[  1.7395  ,  89.357624],
[  1.715   ,  84.821704],
```

```
[  1.9355  , 111.13004 ],
[  1.813   ,  90.7184  ],
[  1.764   ,  90.7184  ],
[  2.009   , 104.779752],
[  1.813   ,  74.84268 ],
[  1.813   , 103.418976],
[  1.764   ,  97.52228 ],
[  1.862   , 115.212368],
[  1.7885  ,  81.64656 ],
[  1.764   ,  97.52228 ],
[  1.764   ,  77.11064 ],
[  1.715   ,  74.84268 ],
[  1.7395  ,  77.11064 ],
[  1.8375  , 101.604608],
[  1.862   , 108.408488],
[  1.862   ,  95.25432 ],
[  1.911   ,  99.79024 ],
[  1.8375  ,  90.7184  ],
[  1.862   ,  88.45044 ],
[  1.764   , 104.32616 ],
[  1.7885  ,  99.79024 ],
[  1.715   ,  87.089664],
[  1.813   ,  92.98636 ],
[  1.7885  ,  92.98636 ],
[  1.7885  ,  83.91452 ],
[  1.8375  ,  92.98636 ],
[  1.715   ,  81.64656 ],
[  1.9355  , 108.86208 ],
[  1.8375  ,  97.52228 ],
[  1.8375  ,  88.45044 ],
[  1.7885  ,  97.52228 ],
[  1.715   ,  88.45044 ],
[  1.7395  ,  88.45044 ],
[  1.8375  ,  95.25432 ],
[  1.764   ,  81.64656 ],
[  1.764   ,  81.64656 ],
[  1.862   , 106.59412 ],
[  1.8375  ,  90.7184  ],
[  1.6905  ,  81.64656 ],
[  1.764   ,  92.079176],
[  1.862   ,  90.7184  ],
[  1.7395  ,  86.18248 ],
[  1.8375  ,  77.11064 ],
[  1.8865  ,  97.52228 ],
[  1.715   ,  81.64656 ],
[  1.764   ,  84.821704],
[  1.7395  ,  81.64656 ],
```

```
[  1.813   ,  95.25432 ],
[  1.862   ,  87.543256],
[  1.764   ,  90.7184  ],
[  1.813   ,  77.11064 ],
[  1.764   ,  88.45044 ],
[  1.7395  ,  86.18248 ],
[  1.715   ,  81.64656 ],
[  1.764   ,  77.11064 ],
[  1.813   ,  95.25432 ],
[  1.813   ,  99.79024 ],
[  1.8375  ,  95.25432 ],
[  1.813   , 111.13004 ],
[  1.7395  , 100.243832],
[  1.7395  ,  83.91452 ],
[  1.862   , 104.32616 ],
[  1.7395  ,  79.3786  ],
[  1.813   ,  90.7184  ],
[  1.764   ,  90.7184  ],
[  1.862   , 113.398   ],
[  1.911   ,  95.25432 ],
[  1.764   ,  97.52228 ],
[  1.862   ,  97.52228 ],
[  1.715   ,  89.811216],
[  1.666   ,  75.749864],
[  1.7395  ,  86.18248 ],
[  1.8865  ,  97.52228 ],
[  1.6905  ,  72.57472 ],
[  1.8375  ,  79.3786  ],
[  1.8375  ,  86.18248 ],
[  1.862   , 108.86208 ],
[  1.764   ,  79.3786  ],
[  1.8865  , 113.398   ],
[  1.715   ,  86.18248 ],
[  1.9355  ,  86.18248 ],
[  1.813   ,  77.11064 ],
[  1.666   ,  68.0388  ],
[  1.8375  ,  99.79024 ],
[  1.764   , 117.93392 ],
[  1.764   ,  88.45044 ],
[  1.7885  ,  86.18248 ],
[  1.813   ,  81.64656 ],
[  1.764   ,  88.45044 ],
[  1.8375  ,  99.336648],
[  1.764   , 102.0582  ],
[  1.7885  ,  96.161504],
[  1.862   ,  86.18248 ],
[  1.813   ,  96.161504],
```

```
[  1.8375  ,  95.25432 ],
[  1.862   ,  90.7184  ],
[  1.813   ,  99.79024 ],
[  1.764   ,  83.91452 ],
[  1.7885  , 104.779752],
[  1.764   ,  83.91452 ],
[  1.7885  ,  81.64656 ],
[  1.764   ,  99.79024 ],
[  1.7885  ,  88.904032],
[  1.715   ,  72.57472 ],
[  1.764   ,  92.98636 ],
[  1.8375  ,  95.25432 ],
[  1.7885  ,  90.7184  ],
[  1.862   , 116.573144],
[  1.8865  ,  92.98636 ],
[  1.8865  ,  94.347136],
[  1.8375  ,  97.52228 ],
[  1.8375  ,  95.25432 ],
[  1.862   ,  92.079176],
[  1.764   ,  95.25432 ],
[  1.7885  ,  90.7184  ],
[  1.813   ,  92.98636 ],
[  1.764   ,  86.18248 ],
[  1.813   , 102.0582  ],
[  1.813   ,  93.893544],
[  1.8375  ,  96.161504],
[  1.7395  ,  95.25432 ],
[  1.862   ,  86.18248 ],
[  1.7885  ,  97.52228 ],
[  1.666   ,  88.904032],
[  1.764   ,  79.3786  ],
[  1.6905  ,  79.3786  ],
[  1.7885  ,  92.98636 ],
[  1.8375  ,  95.25432 ],
[  1.813   ,  89.357624],
[  1.813   , 103.418976],
[  1.813   ,  86.18248 ],
[  1.7885  ,  92.532768],
[  1.813   ,  74.84268 ],
[  1.8375  ,  97.975872],
[  1.813   ,  95.25432 ],
[  1.7885  ,  97.52228 ],
[  1.862   , 103.872568],
[  1.911   , 108.86208 ],
[  1.7885  ,  92.98636 ],
[  1.8375  , 102.0582  ],
[  1.6415  ,  81.64656 ],
```

```
       [  1.715   ,  73.935496],
       [  1.715   ,  79.3786  ],
       [  1.9355  ,  92.98636 ],
       [  1.862   ,  95.707912],
       [  1.764   ,  81.64656 ],
       [  1.8375  ,  92.98636 ]])
```

[65]:
```
batsmen.shape
```

[65]:
```
(323, 2)
```

[66]:
```
batsmen[:,0]
```

[66]:
```
array([1.813 , 1.7885, 1.6905, 1.7395, 1.862 , 1.7885, 1.7885, 1.715 ,
       1.9355, 1.764 , 1.7395, 1.8375, 1.8865, 1.7885, 1.8375, 1.813 ,
       1.7395, 1.7885, 1.813 , 1.862 , 1.8375, 1.862 , 1.764 , 1.8375,
       1.7885, 1.715 , 1.715 , 1.7395, 1.764 , 1.8375, 1.911 , 1.7395,
       1.7885, 1.7885, 1.862 , 1.813 , 1.7885, 1.764 , 1.7885, 1.764 ,
       1.7885, 1.764 , 1.8375, 1.8375, 1.764 , 1.764 , 1.8865, 1.8865,
       1.8375, 1.862 , 1.96  , 1.8375, 1.7885, 1.813 , 1.8375, 1.7395,
       1.862 , 1.813 , 1.7885, 1.7885, 1.7395, 1.7885, 1.7885, 1.764 ,
       1.8375, 1.715 , 1.7395, 1.764 , 1.8375, 1.7885, 1.862 , 1.8375,
       1.862 , 1.764 , 1.7885, 1.8375, 1.813 , 1.7395, 1.8865, 1.911 ,
       1.8375, 1.8375, 1.862 , 1.764 , 1.764 , 1.7885, 1.813 , 1.813 ,
       1.8375, 1.7885, 1.813 , 1.8865, 1.911 , 1.911 , 1.813 , 1.911 ,
       1.813 , 1.764 , 1.862 , 1.7885, 1.7885, 1.8865, 1.8865, 1.7395,
       1.911 , 1.715 , 1.813 , 1.764 , 1.7885, 1.8375, 1.813 , 1.7885,
       1.813 , 1.7885, 1.911 , 1.862 , 1.7885, 1.8375, 1.911 , 1.8865,
       1.764 , 1.6905, 1.813 , 1.715 , 1.813 , 1.862 , 1.8375, 1.7885,
       1.7885, 1.7885, 1.813 , 1.813 , 1.715 , 1.813 , 1.8375, 1.764 ,
       1.8865, 1.7885, 1.862 , 1.7885, 1.8375, 1.813 , 1.7885, 1.7395,
       1.862 , 1.8375, 1.7885, 1.7885, 1.862 , 1.7885, 1.8375, 1.8375,
       1.8865, 1.813 , 1.7395, 1.813 , 1.813 , 1.8375, 1.764 , 1.7885,
       1.862 , 1.8865, 1.7885, 1.715 , 1.7885, 1.813 , 1.8865, 1.8375,
       1.7885, 1.8865, 1.8865, 1.813 , 1.813 , 1.8375, 1.7395, 1.715 ,
       1.9355, 1.813 , 1.764 , 2.009 , 1.813 , 1.813 , 1.764 , 1.862 ,
       1.7885, 1.764 , 1.764 , 1.715 , 1.7395, 1.8375, 1.862 , 1.862 ,
       1.911 , 1.8375, 1.862 , 1.764 , 1.7885, 1.715 , 1.813 , 1.7885,
       1.7885, 1.8375, 1.715 , 1.9355, 1.8375, 1.8375, 1.7885, 1.715 ,
       1.7395, 1.8375, 1.764 , 1.764 , 1.862 , 1.8375, 1.6905, 1.764 ,
       1.862 , 1.7395, 1.8375, 1.8865, 1.715 , 1.764 , 1.7395, 1.813 ,
       1.862 , 1.764 , 1.813 , 1.764 , 1.7395, 1.715 , 1.764 , 1.813 ,
       1.813 , 1.8375, 1.813 , 1.7395, 1.7395, 1.862 , 1.7395, 1.813 ,
       1.764 , 1.862 , 1.911 , 1.764 , 1.862 , 1.715 , 1.666 , 1.7395,
       1.8865, 1.6905, 1.8375, 1.8375, 1.862 , 1.764 , 1.8865, 1.715 ,
       1.9355, 1.813 , 1.666 , 1.8375, 1.764 , 1.764 , 1.7885, 1.813 ,
       1.764 , 1.8375, 1.764 , 1.7885, 1.862 , 1.813 , 1.8375, 1.862 ,
```

```
       1.813 , 1.764 , 1.7885, 1.764 , 1.7885, 1.764 , 1.7885, 1.715 ,
       1.764 , 1.8375, 1.7885, 1.862 , 1.8865, 1.8865, 1.8375, 1.8375,
       1.862 , 1.764 , 1.7885, 1.813 , 1.764 , 1.813 , 1.813 , 1.8375,
       1.7395, 1.862 , 1.7885, 1.666 , 1.764 , 1.6905, 1.7885, 1.8375,
       1.813 , 1.813 , 1.813 , 1.7885, 1.813 , 1.8375, 1.813 , 1.7885,
       1.862 , 1.911 , 1.7885, 1.8375, 1.6415, 1.715 , 1.715 , 1.9355,
       1.862 , 1.764 , 1.8375])
```

[67]:
```
bowler = players_converted[skills == 'Bowler']
bowler
```

[67]:
```
array([[  1.764   ,  95.25432 ],
       [  1.7395  ,  81.64656 ],
       [  1.813   ,  83.91452 ],
       [  1.813   ,  72.57472 ],
       [  1.6905  ,  81.64656 ],
       [  1.8375  ,  83.91452 ],
       [  1.8375  , 108.86208 ],
       [  1.862   ,  90.7184  ],
       [  1.715   ,  84.368112],
       [  1.7885  ,  90.7184  ],
       [  1.862   ,  96.161504],
       [  1.862   , 104.32616 ],
       [  1.666   ,  70.30676 ],
       [  1.8375  , 102.0582  ],
       [  1.666   ,  72.57472 ],
       [  1.813   ,  92.98636 ],
       [  1.813   ,  90.7184  ],
       [  1.9355  ,  92.98636 ],
       [  1.813   , 104.32616 ],
       [  1.862   ,  87.089664],
       [  1.862   ,  90.7184  ],
       [  1.813   ,  88.45044 ],
       [  1.7885  ,  83.91452 ],
       [  1.813   ,  79.3786  ],
       [  1.764   ,  92.532768],
       [  1.813   ,  95.707912],
       [  1.8375  ,  86.18248 ],
       [  1.862   ,  90.7184  ],
       [  1.8375  ,  97.52228 ],
       [  1.764   ,  91.171992],
       [  1.8865  ,  99.79024 ],
       [  1.8375  ,  86.18248 ],
       [  1.7885  ,  81.64656 ],
       [  1.862   ,  88.45044 ],
       [  1.7395  ,  86.18248 ],
       [  1.8865  ,  83.91452 ],
```

```
[  1.7395  ,  86.18248 ],
[  1.8375  ,  90.7184  ],
[  1.8375  , 126.098576],
[  1.9355  ,  97.52228 ],
[  1.8865  , 104.32616 ],
[  1.862   , 108.86208 ],
[  1.7395  ,  83.460928],
[  1.7395  ,  86.18248 ],
[  1.764   ,  89.357624],
[  1.7885  ,  92.532768],
[  1.764   ,  92.98636 ],
[  1.8375  ,  90.7184  ],
[  1.862   ,  77.11064 ],
[  1.715   ,  86.18248 ],
[  1.7395  ,  87.543256],
[  1.813   ,  90.7184  ],
[  1.6905  ,  78.017824],
[  1.8375  , 102.0582  ],
[  1.7885  ,  89.357624],
[  1.8865  ,  90.7184  ],
[  1.764   ,  80.285784],
[  1.7885  ,  90.7184  ],
[  1.7885  ,  90.7184  ],
[  1.8375  ,  92.98636 ],
[  1.7885  ,  97.52228 ],
[  1.8375  ,  99.79024 ],
[  1.7885  ,  90.7184  ],
[  1.764   ,  79.3786  ],
[  1.764   ,  79.3786  ],
[  1.813   ,  90.7184  ],
[  1.8375  ,  88.45044 ],
[  1.96    , 108.86208 ],
[  1.764   ,  68.0388  ],
[  1.813   ,  91.625584],
[  1.813   ,  90.7184  ],
[  1.8375  ,  92.98636 ],
[  1.8375  ,  83.91452 ],
[  1.7885  ,  91.625584],
[  1.764   ,  99.79024 ],
[  1.862   , 106.59412 ],
[  1.813   ,  81.64656 ],
[  1.764   ,  90.7184  ],
[  1.8865  ,  95.25432 ],
[  1.7885  ,  81.64656 ],
[  1.715   ,  81.64656 ],
[  1.764   ,  81.64656 ],
[  1.911   , 104.32616 ],
```

```
[  1.8375  ,  99.79024 ],
[  1.862   , 113.398   ],
[  1.8375  ,  87.089664],
[  1.764   ,  89.357624],
[  1.7395  ,  86.18248 ],
[  1.7885  ,  90.7184  ],
[  1.764   ,  88.45044 ],
[  1.764   ,  79.3786  ],
[  1.764   ,  93.439952],
[  1.862   , 117.93392 ],
[  1.862   , 100.243832],
[  1.7885  ,  92.98636 ],
[  1.8375  ,  83.91452 ],
[  1.8375  , 111.13004 ],
[  1.764   ,  83.91452 ],
[  1.7885  ,  81.64656 ],
[  1.7885  ,  95.25432 ],
[  1.715   ,  88.45044 ],
[  1.7885  , 108.86208 ],
[  1.8865  , 113.398   ],
[  1.8375  ,  95.25432 ],
[  1.813   ,  81.64656 ],
[  1.7885  ,  92.98636 ],
[  1.7885  ,  81.64656 ],
[  1.8865  ,  99.79024 ],
[  1.8865  ,  90.7184  ],
[  1.813   , 102.0582  ],
[  1.764   ,  83.91452 ],
[  1.764   ,  83.91452 ],
[  1.715   ,  79.3786  ],
[  1.715   ,  95.25432 ],
[  1.7885  ,  86.636072],
[  1.8865  ,  90.7184  ],
[  1.8375  ,  88.45044 ],
[  1.813   ,  90.7184  ],
[  1.8865  ,  99.79024 ],
[  1.862   ,  92.98636 ],
[  1.8375  ,  90.7184  ],
[  1.7885  ,  95.25432 ],
[  1.6905  ,  90.7184  ],
[  1.7395  ,  83.91452 ],
[  1.8865  ,  86.18248 ],
[  1.8375  ,  94.347136],
[  1.764   ,  81.64656 ],
[  1.8375  ,  86.18248 ],
[  1.7885  ,  97.52228 ],
[  1.8375  , 104.32616 ],
```

```
[  1.8375  ,  94.800728],
[  1.8865  ,  77.11064 ],
[  1.715   ,  83.91452 ],
[  1.7395  ,  86.18248 ],
[  1.764   ,  72.57472 ],
[  1.7885  ,  90.7184  ],
[  1.8375  ,  99.79024 ],
[  1.813   ,  99.79024 ],
[  1.813   ,  99.336648],
[  1.764   ,  81.64656 ],
[  1.764   ,  86.18248 ],
[  1.715   ,  83.91452 ],
[  1.7395  ,  88.45044 ],
[  1.7885  ,  86.18248 ],
[  1.7395  ,  83.91452 ],
[  1.813   , 108.86208 ],
[  1.7395  ,  81.64656 ],
[  1.764   ,  88.45044 ],
[  1.7395  ,  72.57472 ],
[  1.764   ,  92.98636 ],
[  1.764   ,  83.91452 ],
[  1.813   ,  86.18248 ],
[  1.7885  ,  97.52228 ],
[  1.8375  ,  70.760352],
[  1.764   ,  86.18248 ],
[  1.7395  ,  91.625584],
[  1.7885  ,  86.18248 ],
[  1.8375  ,  86.18248 ],
[  1.764   ,  81.64656 ],
[  1.764   , 110.676448],
[  1.7885  ,  90.7184  ],
[  1.862   , 104.32616 ],
[  1.862   , 101.151016],
[  1.813   , 104.32616 ],
[  1.911   ,  86.18248 ],
[  1.764   ,  86.18248 ],
[  1.7395  ,  94.800728],
[  1.911   ,  90.7184  ],
[  1.764   ,  90.7184  ],
[  1.813   ,  88.45044 ],
[  1.813   ,  86.18248 ],
[  1.7885  ,  98.883056],
[  1.7395  ,  86.18248 ],
[  1.7885  ,  85.275296],
[  2.009   , 113.398   ],
[  1.7885  ,  81.64656 ],
[  1.715   ,  74.84268 ],
```

```
       [  1.813   ,  83.91452 ],
       [  1.8375  , 106.59412 ],
       [  1.7395  ,  68.0388  ],
       [  1.813   ,  84.368112],
       [  1.764   ,  89.811216],
       [  1.8375  ,  97.52228 ],
       [  1.8375  ,  81.64656 ],
       [  1.715   ,  87.089664],
       [  1.764   ,  87.089664],
       [  1.862   ,  99.336648],
       [  1.862   ,  99.79024 ],
       [  1.7395  ,  77.11064 ],
       [  1.7395  ,  86.18248 ],
       [  1.764   , 101.151016],
       [  1.7885  ,  85.728888],
       [  1.715   ,  81.64656 ],
       [  1.862   ,  97.52228 ],
       [  1.8375  ,  93.893544],
       [  1.813   ,  83.91452 ],
       [  1.764   ,  86.18248 ],
       [  1.764   ,  94.347136],
       [  1.7395  , 102.0582  ],
       [  1.862   , 108.86208 ],
       [  1.8375  ,  81.64656 ],
       [  1.8375  ,  90.7184  ],
       [  1.911   , 104.32616 ],
       [  1.8375  , 104.32616 ],
       [  1.8375  ,  92.98636 ],
       [  1.8375  ,  86.18248 ]])
```

[68]:
```
bowler.shape
```

[68]:
```
(206, 2)
```

[69]:
```
array1 = np.arange(12).reshape(3,4)
array1
```

[69]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

[70]:
```
array2 = np.arange(20).reshape(5,4)
array2
```

[70]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
```

```
        [12, 13, 14, 15],
        [16, 17, 18, 19]])
```

[71]:
```
print(array1, '\n', array2)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]]
```

[72]:
```
np.vstack((array1, array2))
```

[72]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11],
       [12, 13, 14, 15],
       [16, 17, 18, 19]])
```

1.0.13 Built in function in numpy

[74]:
```
np.power(array1, 3) # power = array1 ** 3
```

[74]:
```
array([[   0,    1,    8,   27],
       [  64,  125,  216,  343],
       [ 512,  729, 1000, 1331]])
```

[75]:
```
np.arange(9).reshape(3,3)
```

[75]:
```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

[76]:
```
x = np.array([-2,-1,0,1,2])
x
```

[76]:
```
array([-2, -1,  0,  1,  2])
```

[77]:
```
abs(x)
```

[77]:
```
array([2, 1, 0, 1, 2])
```

[78]:
```
np.absolute(x)
```

[78]:
```
array([2, 1, 0, 1, 2])
```

[79]:
```
np.pi
```

[79]:
```
3.141592653589793
```

[80]:
```
theta = np.linspace(0,np.pi,5)
theta
```

[80]:
```
array([0.        , 0.78539816, 1.57079633, 2.35619449, 3.14159265])
```

[81]:
```
np.sin(theta)
```

[81]:
```
array([0.00000000e+00, 7.07106781e-01, 1.00000000e+00, 7.07106781e-01,
       1.22464680e-16])
```

[82]:
```
np.cos(theta)
```

[82]:
```
array([ 1.00000000e+00,  7.07106781e-01,  6.12323400e-17, -7.07106781e-01,
       -1.00000000e+00])
```

[83]:
```
np.tan(theta)
```

[83]:
```
array([ 0.00000000e+00,  1.00000000e+00,  1.63312394e+16, -1.00000000e+00,
       -1.22464680e-16])
```

1.0.14 Exponential and lograthmic functions

[85]:
```
x = [1,2,3,4,5]
x = np.array(x)
x
```

[85]:
```
array([1, 2, 3, 4, 5])
```

[86]:
```
np.exp(x) # e^1
```

[86]:
```
array([  2.71828183,   7.3890561 ,  20.08553692,  54.59815003,
       148.4131591 ])
```

[87]:
```
np.exp2(x) # 2^1 , 2^2, 2^3
```

[87]:
```
array([ 2.,  4.,  8., 16., 32.])
```

[88]:
```
np.power(x,3)
```

[88]:
```
array([  1,   8,  27,  64, 125])
```

[89]:
```
np.log(x)
```

[89]:
```
array([0.        , 0.69314718, 1.09861229, 1.38629436, 1.60943791])
```

[90]:
```
np.log10(x)
```

[90]:
```
array([0.        , 0.30103   , 0.47712125, 0.60205999, 0.69897   ])
```

[91]:
```
np.log
```

[91]:
```
<ufunc 'log'>
```

[92]:
```
x = np.arange(5)
x
```

[92]:
```
array([0, 1, 2, 3, 4])
```

[93]:
```
y = x *10
y
```

[93]:
```
array([ 0, 10, 20, 30, 40])
```

[94]:
```
z = np.empty(5)
z
```

[94]:
```
array([2.77911610e-316, 0.00000000e+000, 3.03587213e-316, 2.93001876e-316,
       2.37151510e-322])
```

[95]:
```
np.multiply(x, 12, out=z)
```

[95]:
```
array([ 0., 12., 24., 36., 48.])
```

[96]:
```
a = np.zeros(10)
a
```

[96]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

[97]:
```
np.power(2,x, out=a[::2])
```

[97]:
```
array([ 1.,  2.,  4.,  8., 16.])
```

[98]:
```
x = np.arange(1,6)
x
```

[98]:
```
array([1, 2, 3, 4, 5])
```

[99]:
```
sum(x) # adding all the elements in the x array
```

[99]:
```
15
```

[100]:
```
np.add.reduce(x) # it reduces the given data to its simplest form using the add␣
↪function
```

[100]:
```
15
```

[101]:
```
np.add.accumulate(x) # Cummulative frequency
```

[101]:
```
array([ 1,  3,  6, 10, 15])
```

[102]:
```
np.multiply.accumulate(x)
```

[102]:
```
array([  1,   2,   6,  24, 120])
```

1.0.15 Apply basic algebra expressions

[104]:
```
# help(np.linalg)
```

[105]:
```
A = np.array([[6,1,1],
              [4, 5.5,-2],
              [2,8,0]])
```

[106]:
```
A
```

[106]:
```
array([[ 6. ,  1. ,  1. ],
       [ 4. ,  5.5, -2. ],
       [ 2. ,  8. ,  0. ]])
```

[107]:
```
np.linalg.matrix_rank(A) # Rank of a matrix
```

[107]:
```
3
```

[108]:
```
np.trace(A) # sum of elements on the diagnol of the matrix
```

[108]:
```
11.5
```

[109]:
```
np.linalg.det(A)
```

[109]:
```
113.00000000000003
```

[110]:
```
np.linalg.inv(A)
```

[110]:
```
array([[ 0.14159292,  0.07079646, -0.06637168],
       [-0.03539823, -0.01769912,  0.14159292],
```

```
       [ 0.18584071, -0.40707965,  0.25663717]])
```

[111]:
```
B = np.linalg.inv(A)
```

[112]:
```
np.matmul(A,B) # actual matrix multiplication using linear algebra rules
```

[112]:
```
array([[ 1.00000000e+00, -5.55111512e-17,  0.00000000e+00],
       [ 0.00000000e+00,  1.00000000e+00,  1.11022302e-16],
       [ 0.00000000e+00,  0.00000000e+00,  1.00000000e+00]])
```

[113]:
```
A * B # elementwise multiplication
```

[113]:
```
array([[ 0.84955752,  0.07079646, -0.06637168],
       [-0.14159292, -0.09734513, -0.28318584],
       [ 0.37168142, -3.25663717,  0.        ]])
```

[114]:
```
np.linalg.matrix_power(A,3)
```

[114]:
```
array([[338.   , 181.25 ,   3.   ],
       [311.   ,  86.375,   5.5  ],
       [420.   , 185.   , -48.   ]])
```

[115]:
```
import time

list1 = [i for i in range(1000000)]
list2 = [j**2 for j in range(1000000)]

t0 = time.time()
product_list = list(map(lambda x, y : x*y , list1, list2))

t1 = time.time()
list_time = t1 - t0
print("Time taken for list", list_time)

array1 = np.array(list1)
array2 = np.array(list2)

t0 = time.time()
product_numpy = array1 * array2
t1 = time.time()
numpy_time = t1-t0
print("Time taken for Numpy", numpy_time)

print("The ratio if time taken is {}".format(list_time//numpy_time))
```

```
Time taken for list 0.11545133590698242
Time taken for Numpy 0.015027284622192383
```

```
The ratio if time taken is 7.0
```

In this case, numpy is **an order of magnitude faster** than lists. This is with arrays of size in millions, but you may work on much larger arrays of sizes in order of billions. Then, the difference is even larger.

Some reasons for such difference in speed are: * NumPy is written in C, which is basically being executed behind the scenes * NumPy arrays are more compact than lists, i.e. they take much lesser storage space than lists

Official webpage of Numpy: https://numpy.org/doc/stable/user/absolute_beginners.html

[ ]: