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 ?

In [5]:
```
# Define list
height_in = [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,
```

In [6]:
```
import numpy as np

heights = np.array(heights)
```

In [7]:
```
heights
```

Out[7]:
```
array([74, 74, 72, ..., 75, 75, 73])
```

In [8]:
```
type(heights)
```

Out[8]:
```
numpy.ndarray
```

Example - 2

Count the number of pariticipants

In [9]:
```
len(heights)
```

Out[9]:
```
1015
```

In [10]:
```
heights.size
```

Out[10]:
```
1015
```

In [11]:
```
heights.shape
```

Out[11]:
```
(1015,)
```

Example - 3

Convert the heights from inches to meters

In [12]:
```
heights_m = heights * 0.0254

heights_m
```

Out[12]:
```
array([1.8796, 1.8796, 1.8288, ..., 1.905 , 1.905 , 1.8542])
```

Example - 4

A list of weights (in lbs) of the players is provided. Convert it to kg and calculate BMI

In [13]:
```
weights_lb = [180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185, 160, 180, 185, 189, 185, 219, 230, 205
```

In [14]:
```
# Converting weights in lbs to kg

weights_kg = np.array(weights_lb) * 0.453592

weights_kg
```

Out[14]:
```
array([81.64656, 97.52228, 95.25432, ..., 92.98636, 86.18248, 88.45044])
```

In [15]:
```
# Calculate the BMI: bmi

bmi = weights_kg / (heights_m ** 2)

bmi
```

Out[15]:
```
array([23.11037639, 27.60406069, 28.48080465, ..., 25.62295933,
       23.74810865, 25.72686361])
```

Sub-Setting Arrays

Fetch the first element from the bmi array

In [16]:
```
bmi[0]
```

Out[16]:
```
23.11037638875862
```

Fetch the last element from the bmi array

In [17]:
```
bmi[-1]
```

Out[17]:
```
25.726863613607133
```

Fetch the first 5 elements from the bmi array

In [18]:
```
bmi[0:5]
```

Out[18]:
```
array([23.11037639, 27.60406069, 28.48080465, 28.48080465, 24.80333518])
```

Fetch the last 5 elements from the bmi array

In [19]:
```
bmi[-5:]
```

Out[19]:
```
array([25.06720044, 23.11037639, 25.62295933, 23.74810865, 25.72686361])
```

Conditional Sub-Setting Arrays

Count the number of pariticipants who are underweight i.e. bmi < 21

In [20]:
```
bmi < 21
```

Out[20]:
```
array([False, False, False, ..., False, False, False])
```

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

Out[21]:
```
array([20.54255679, 20.54255679, 20.69282047, 20.69282047, 20.34343189,
       20.34343189, 20.69282047, 20.15883472, 19.4984471 , 20.69282047,
       20.9205219 ])
```

In [22]:
```
underweight_players = bmi [ bmi<21]
underweight_players
```

Out[22]:
```
array([20.54255679, 20.54255679, 20.69282047, 20.69282047, 20.34343189,
       20.34343189, 20.69282047, 20.15883472, 19.4984471 , 20.69282047,
       20.9205219 ])
```

In [23]:
```
underweight_players.size
```

Out[23]:
```
11
```

NumPy Functions

Find the largest BMI value

In [24]:
```
max(bmi)
```

Out[24]:
```
35.26194861031698
```

In [25]:
```
bmi.max()
```

Out[25]:
```
35.26194861031698
```

Find lowest BMI value

In [26]:
```
bmi.min()
```

Out[26]:
```
19.498447103560874
```

Find average BMI value

In [27]:
```
bmi.mean()
```

Out[27]:
```
26.05684565448554
```

In [ ]:
```

```