9/12/2023

# *Project Title:*

*Analysis of Cricket Tournament for select players for an upcoming league match based on their fitness.*

**Name: K.Pranavi**

**Branch: CSE**

**Year: 4th year**

**Project Guide**

**Mr. Lokesh B. Sir**

## Problem Statement

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.

**Tools:**

1. Jupyter Notebook.
2. Python Programing Language.
3. Numpy (Python Libery_)
4. Google Colab.

# K.Pranavi

# 20X01A0584

# NRCM

## Case Study: Cricket Tournament

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

Height of the payers is stored as a regular Python list: height_in. The height is expressed in inches.

Weight of the payers is stored as a regular Python list: weight_lb. The weight is expressed in pounds.

## project Title: Analysis and Manupluation of cricket tournament data set for BCCI Organisation

```
heights_in = [74, 74, 72, 72, 73, 69, 69, 71, 76, 71, 73, 73, 74, 74, 69, 70, 73, 75, 78, 79,
weights_lb = [180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185, 160, 180, 185,
```

```
# Importing the numpy library
import numpy as np
```

```
# Convert the lists into NumPy arrays
heights_in = np.array(heights_in)
weights_lb = np.array(weights_lb)
```

```
# Print the type of the created arrays
heights_in
```

```
weights_lb
```

```
array([180, 215, 210, ..., 205, 190, 195])
```

```
# Convert the lists into NumPy arrays
heights_in = np.array(heights_in)
weights_lb = np.array(weights_lb)
```

```
# Convert the lists into NumPy arrays
heights_in = np.array(heights_in)
weights_lb = np.array(weights_lb)
```

```
# Print the length of the created arrays
weights_lb.size
```

```
1015
```

## Multiplication with a factor

Now, let's convert the height and weight of the players into more generic units.

- Height in metres
- Weight in kilograms

```
# Convert the units of height and weight using appropriate factors (look in the video for the

heights_m = heights_in*0.025
weights_kg = weights_lb*0.45
```

```
# Print the newly created arrays
print(heights_m)
```

```
[1.85  1.85  1.8   ... 1.875 1.875 1.825]
```

As you can see, the NumPy way is clearly more concise. Even simple mathematical operations on lists required loops, unlike with arrays.

## Deriving new data from existing

Now, let us try to calculate the Body Mass Index (BMI) value for the players. The formula for BMI is:

$$\text{BMI} = \frac{\text{Weight of the individual}}{(\text{Height of the individual})_2}$$

```
# Calculate the bmi value based on the formula above

bmi = weights_kg/heights_m**2


# Check the newly created array 'bmi'
print(bmi)
```

```
[23.66691015 28.26880935 29.16666667 ... 26.24       24.32
 26.34640646]
```

## Indexing through arrays

For **one-dimensional arrays**, indexing is **similar to Python lists** - indexing starts at 0.

```
# Obtain the 5th element from the array 'bmi'

bmi[5]
```

```
26.61625708884688
```

```
# Obtain the 2nd last element from the array 'bmi'
bmi[-2]
```

```
24.32
```

```
# Obtain the first five elements from the array 'bmi'

bmi[:6]
bmi
```

```
array([23.66691015, 28.26880935, 29.16666667, ..., 26.24      ,
       24.32      , 26.34640646])
```

```
# Obtain the last three elements from the array 'bmi'
bmi[-3:]
```

```
    array([26.24      , 24.32      , 26.34640646])
```

## Subsets

```
# Check for the elements where bmi value is less than 21
bmi[bmi<21]
```

```
    array([20.83333333, 20.83333333, 20.64429077, 19.968     ])
```

```
# Filter the elements where bmi value is less than 21
```

```
# Count the number of elements where bmi value is less than 21
print((len(bmi[bmi<21])))
```

```
    4
```

```
# Find the maximum bmi values among the players
bmi.max()
```

```
    36.11111111111111
```

```
# Find the minimum bmi values among the players
bmi.min()
```

```
    19.968
```

```
# Find the average bmi among the players

bmi.mean()
```

```
    26.684334976283704
```

# Conclusion: According to the my conclusion the body mass

index for the players the range lies b/w 23 to 26.

minimum criteria for the height is 23m as individual player max criteria for the height is 26m as individual player

## Practice Exercise 1

You are provided with 2 lists that contain the data of an ecommerce website. The rst list contains the data for the number of items sold for a particular product and the second list contains the price of the product sold. As a part of this exercise, solve the questions that are provided below.

```
number = [8, 9, 9, 1, 6, 9, 5, 7, 3, 9, 7, 3, 4, 8, 3, 5, 8, 4, 8, 7, 5, 7, 3, 6, 1, 2, 7, 4, 7, 7, 8, 4, 3, 4, 2, 2, 2, 7, 3, 5, 6, 1, 1, 3,
price = [195, 225, 150, 150, 90, 60, 75, 255, 270, 225, 135, 195, 30, 15, 210, 105, 15, 30, 180, 60, 165, 60, 45, 225, 180, 90, 30, 210, 150,
```

### How many different products are sold by the company in total?

- 99
- 100
- 101
- 102

```
#len(price) import
numpy as np
A=np.array(price)
A.size
```

```
102
```

### How many items were sold in total?

- 460
- 490
- 500
- 520

```
# Type your code here
import numpy as np
number=np.array(number)
number.sum() 490
```

### What is the average price of the products sold by the ecommerce company?

- 139
- 151
- 142
- 128

```
# Type your code here
s=np.array(price)
s.mean()
```

```
139.01960784313727
```

### What is the price of the costliest item sold?

- 225
- 310
- 280
- 285

```
# Type your code here
s=np.array(price)
s.max()
```

```
285
```

What is the total revenue of the company? [Revenue = Price*Quantity]

- 67100
- 53900
- 45300
- 71200

```
# Type your code here
s=np.array(price)
p=np.array(number)
rev=s*p rev.sum()
```

```
67100
```

Demand for the 20th product in the list is more than the 50th product. [True/False]

- True
- False
- Can't be calculated

```
# Type your code here
if (p[19]>p[49]):
print("True") else:
  print("False")
```

```
True
```

How many products fall under the category of expensive goods?

An expensive good is that good whose price is more than the average price of the products sold by the company.

- 48
- 50
- 52
- 54

```
# Type your code here
len(A[A>A.mean()])
```

```
52
```

# Multidimensional Arrays

A multidimensional array is an array of arrays. For example, a two dimensional array would be an array with each element as one-dimensional array.

1-D array : [1, 2, 3, 4, 5]

2-D array : [ [1, 2, 3, 4, 5], [6, 7, 8, 9, 10] ]

Similary for a n-dimensional array

In NumPy, dimension is called axis. In Numpy terminology, for 2-D arrays:

- axis = 0 refers to the rows axis =
- 1 refers to the columns

image.png

**Multidimensional arrays** are indexed using as many indices as the number of dimensions or axes. For instance, to index a 2-D array, you need two indices - `array[x, y]` . Each axes has an index starting at 0.

```
# import numpy library
import numpy as np

# player information is provided in the lists - players, skills
# players: list of tuples where 1st element is height in inches and the 2nd element is weight in lbs
# skills: the skill of the player in the sport cricket

players = [(74, 180), (74, 215), (72, 210), (72, 210), (73, 188), (69, 176), (69, 209), (71, 200), (76, 231), (71, 180), (73, 188), (73, 180)
skills = np.array(['Keeper', 'Batsman', 'Bowler', 'Keeper-Batsman', 'Batsman', 'Keeper-Batsman', 'Batsman', 'Batsman', 'Batsman', 'Bowler', '
```

```
# Creating a 2-D array using the list of tuples
np_players = np.array(players)

# Checking the created array
np_players
```

```
array([[ 74, 180],
[ 74, 215],
       [ 72, 210],
...,
       [ 75, 205],
       [ 75, 190],
       [ 73, 195]])
```

```
# checking the type of created object
type(np_players) numpy.ndarray
```

```
# printing the number of columns and rows in the array
np_players.shape
```

```
(1015, 2)
```

```
# printing the dimensions of the array
np_players.ndim
```

```
2
```

```
# printing the data type of the elements in the array
np_players.dtype dtype('int64')
```

## Slicing a 2D array

```
# printing the entire second row of the 2-D array
np_players[:2]
```

```
array([[ 74, 180],
[ 74, 215]])
```

```
# printing the second column of the second row of the 2-D array
np_players[2][1]
```

```
210
```

```
# printing the first column (height of the players) with all the rows of the 2-D array np_players[1:]
```

```
array([[ 74, 215],
[ 72, 210],
       [ 72, 210],
...,
       [ 75, 205],
       [ 75, 190],
       [ 73, 195]])
```

```
# printing only those rows where height of the player is more than 75 inches
np_players[np_players[:,0]>75].size
```

```
414
```

## Slicing one array based on the other

```
# printing those rows where the skill of the player is 'Batsman'
np_players[skills=="Batsman"]
```

```
       [ 74, 190],
       [ 73, 204],
       [ 74, 165],
       [ 75, 216],
       [ 74, 210],
       [ 73, 215],
       [ 76, 229],
       [ 78, 240],
       [ 73, 205],
       [ 75, 225],
       [ 67, 180],
       [ 70, 163],
       [ 70, 175],
       [ 79, 205],
       [ 76, 211],
       [ 72, 180],
       [ 75, 205]])
```

0s completed at 12:27 PM