

```
import numpy as np
```

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 ?

1. create an random integer numpy array for heights in inches in the range 67 to 83 of size 1015
2. create an random integer numpy array for wights in lbs in the range 150 to 290 of size 1015

```
heights_sample = [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,
weights_sample = [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
```

```
# heights = np.random.randint(67, 83, 1015)
# weights = np.random.randint(150, 290, 1015)
heights = np.array(heights_sample)
weights = np.array(weights_sample)
print("Heights")
print(heights);
print(f"Shape: {heights.shape}")
print(f"dim: {heights.ndim}")
print(f"Size: {heights.size}")
print(f"Datatype: {heights.dtype}")
print("\n")
print("Weights")
print(weights);
print(f"Shape: {weights.shape}")
print(f"dim: {weights.ndim}")
print(f"Size: {weights.size}")
print(f"Datatype: {weights.dtype}")
```

```
Heights
[74 74 72 ... 75 75 73]
Shape: (1015,)
dim: 1
Size: 1015
Datatype: int64
```

```
Weights
[180 215 210 ... 205 190 195]
Shape: (1015,)
dim: 1
Size: 1015
Datatype: int64
```

Convert the heights from inches to meters

```
heights * 0.0254
```

```
heightsMeters = heights * 0.0254
print("Heights in meters:", heightsMeters)
```

```
Heights in meters: [1.8796 1.8796 1.8288 ... 1.905 1.905 1.8542]
```

Converting weights in lbs to kg =weights_lb * 0.453592

```
weightsKgs = weights * 0.453592
print("Weights in lbs:", weightsKgs)
```

```
Weights in lbs: [81.64656 97.52228 95.25432 ... 92.98636 86.18248 88.45044]
```

Calculate the BMI: bmi

```
bmi = weightsKgs / (heightsMeters**2)
print("Bmi", bmi)
```

```
Bmi [23.11037639 27.60406069 28.48080465 ... 25.62295933 23.74810865
25.72686361]
```

Fetch the first element from the bmi array

```
bmi[0]

23.11037638875862
```

Fetch the last element from the bmi array

```
bmi[-1]

25.726863613607133
```

Fetch the first five element from the bmi array

```
bmi[:5]

array([23.11037639, 27.60406069, 28.48080465, 28.48080465, 24.80333518])
```

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

```
underweights = bmi[bmi < 21]
print(f"No. of underweights: {underweights.size}")

No. of underweights: 11
```

Double-click (or enter) to edit

▼ Practice Exercise 1

You are provided with 2 lists that contain the data of an ecommerce website. The first 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,
price = [195, 225, 150, 150, 90, 60, 75, 255, 270, 225, 135, 195, 30, 15, 210, 105, 15, 30, 1
```

```
# Type your code here
import numpy as np
import math
```

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

- 99
- 100
- 101
- 102

```
productQuantities = np.array(number)
productPrices = np.array(price)

print(f"Unique products: {productQuantities.size}")

Unique products: 102
```

▼ How many items were sold in total?

- 460
- 490
- 500
- 520

```
# Type your code here

np.sum(productQuantities)

490
```

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

- 139
- 151
- 142
- 128

```
# Type your code here
avgProductPrice = np.average(productPrices)
math.floor(avgProductPrice)
```

139

▼ What is the price of the costliest item sold?

- 225
- 310
- 280
- 285

```
# Type your code here
np.max(productPrices)
```

285

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

- 67100
- 53900
- 45300
- 71200

```
sum(productPrices * productQuantities)
```

67100

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

- True
- False
- Can't be calculated

```
print(f"20th product {productQuantities[20]}")  
print(f"50th product {productQuantities[50]}")  
productQuantities[20] > productQuantities[50]
```

```
20th product 5  
50th product 4  
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

```
productPrices[productPrices > avgProductPrice].size
```

```
52
```

```
import numpy as np
```

Extracting Elements from Array Description From a given array, extract all the elements which are greater than 'm' and less than 'n'. Note: 'm' and 'n' are integer values provided as input.

Input format:

A list of integers on line one

Integer 'm' on line two

Integer 'n' on line three

Output format:

1-D array containing integers greater than 'm' and smaller than 'n'.

Sample input:

```
[ 1, 5, 9, 12, 15, 7, 12, 9 ] (array)
```

6 (m)

12 (n)

Sample output:

```
[ 9 7 9 ]
```

```
# sample input
# 1 5 9 12 15 7 12 9
# 6
# 12
```

```
arr = np.array(list(map(int, input().split())))
m = int(input())
n = int(input())
```

```
print("Arr: ", arr)
print("m: ", m)
print("n: ", n)
```

```
1 5 9 12 15 7 12 9
6
12
Arr: [ 1 5 9 12 15 7 12 9]
m: 6
n: 12
```

```
print("Original Arr: ", arr)
print("Filtered: ", arr[np.logical_and(arr > m, arr < n)])
```

```
Original Arr: [ 1 5 9 12 15 7 12 9]
Filtered: [9 7 9]
```

Print "+" Description Given a single positive odd integer 'n' greater than 2, create a NumPy array of size (n x n) with all zeros and ones such that the ones make a shape like '+'. The lines of the plus must be present at the middle row and column.

Hint: Start by creating a (n x n) array with all zeroes using the np.zeros() function and then fill in the ones at the appropriate indices. Use integer division (//) to access the middle rows and columns

Examples:

Input 1:

3

Output 1:

```
[[0 1 0]
```

```
[1 1 1]
```

```
[0 1 0]]
```

Input 2:

5

Output 1:

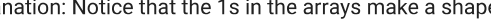
[[0 0 1 0 0]

[0 0 1 0 0]

[1 1 1 1 1]

[0 0 1 0 0]

[0 0 1 0 0]]

Explanation: Notice that the 1s in the arrays make a shape like '+'.


```
def makePlusMatrix(n):
    mid = int(n/2)
    result = np.zeros((n,n))
    result[mid, :] = 1
    result[:, mid] = 1
    return result
```

```
n = int(input())
print("n: ", n)
print("Plus matrix:")
print(makePlusMatrix(n))
```

```
5
n: 5
Plus matrix:
[[0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [1. 1. 1. 1. 1.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]]
```

```
n = int(input())
print("n: ", n)
print("Plus matrix:")
print(makePlusMatrix(n))
```

```
3
n: 3
Plus matrix:
[[0. 1. 0.]
 [1. 1. 1.]
 [0. 1. 0.]]
```