NumPy

DS & ML Bootcamp by Ai DataYard

What is NumPy?

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

NumPy

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called `ndarray`, it provides a lot of supporting functions that make working with `ndarray` very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

NumPy

Why is NumPy Faster Than Lists?

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.

This behavior is called locality of reference in computer science.

This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

NumPy as np

NumPy is usually imported under the `np` alias.

alias: In Python alias are an alternate name for referring to the same thing.

Create an alias with the `as` keyword while importing:

```
import numpy as np
```


NumPy

Task #1 – NumPy

Task#1

baseball=[180,215,210,210,188,176,209,200]

Use np.array() to create a numpy array from baseball list. Name this array np_baseball. Print out the type of np_baseball.

```
[ ]  import numpy as np

     baseball=[180,215,210,210,188,176,209,200]

     np_baseball = np.array(baseball)

     np_baseball
```

```
array([180, 215, 210, 210, 188, 176, 209, 200])
```

```
[ ]  print(type(np_baseball))
```

```
<class 'numpy.ndarray'>
```

DataYard
Professional eLearning Partner

NumPy

Task #2 – NumPy

Task#2

Convert the height to inches and print the array


```
[ ]  height = 55 # height is in cm

     height_in_inches = height * 0.393701

     print(height_in_inches)
```

```
21.653555
```

DataYard
Professional eLearning Partner

NumPy

Task #3 – NumPy

Task# 3

Create a list of random heights of 50 baseball players ranging between 70-80

Note: The height is expressed in inches.

```
import random as rn
heights = [rn.randint(70, 81) for _ in range(50)]

print(heights)
```

```
[73, 70, 80, 70, 72, 78, 74, 76, 79, 75, 71, 70, 81, 72, 76, 77, 81, 79, 72, 77, 70, 72, 75, 76, 74, 78, 74, 72, 78, 71, 76, 77, 78, 73, 78, 73, 71, 71, 70, 79,
```

DataYard
Professional eLearning Partner

NumPy

Task #4 – NumPy

Task# 4

Create a numpy array from the list created in Task#3 & name this new array np_height_in. Print np_height_in.

```
import numpy as np

np_height_in = np.array(heights)

print(np_height_in)
```

```
[79 79 80 74 73 80 78 74 70 74 70 77 73 74 75 71 78 72 79 72 81 74 70 74
 70 80 79 71 71 73 77 80 77 74 80 73 73 72 70 80 75 73 70 76 70 76 78 73
 81 73]
```

DataYard
Professional eLearning Partner

NumPy

Task #5 – NumPy

Task# 5

a. Create a list of random weight (lb) values of 50 baseball players ranging between 160-260.

b. Create a numpy array from the weight_lb list.

```
import random as rn
weight = [rn.randint(160, 260) for _ in range(50)]
# weight
```

```
[ ]  import numpy as np

     weight_lb = np.array(weight)

     print(weight_lb)
```

```
[187 204 260 183 213 234 207 244 251 247 258 238 207 202 161 170 189 194
 205 243 182 180 198 167 178 256 250 245 190 172 198 257 247 180 198 259
 225 183 168 165 256 217 237 206 169 215 196 202 252 179]
```

DataYard
Professional eLearning Partner

NumPy

Task #5 – NumPy

c. Convert weights from pounds to kilograms.

```
[ ]  weights_kg = weight_lb * 0.4536

     print(weights_kg)
```

```
[ 84.8232  92.5344 117.936   83.0088  96.6168 106.1424  93.8952 110.6784
 113.8536 112.0392 117.0288 107.9568  93.8952  91.6272  73.0296  77.112
  85.7304  87.9984  92.988  110.2248  82.5552  81.648   89.8128  75.7512
  80.7408 116.1216 113.4    111.132   86.184   78.0192  89.8128 116.5752
 112.0392  81.648   89.8128 117.4824 102.06    83.0088  76.2048  74.844
 116.1216  98.4312 107.5032  93.4416  76.6584  97.524   88.9056  91.6272
 114.3072  81.1944]
```

DataYard
Professional eLearning Partner

NumPy

Task #6 – NumPy

Task# 6

Use np_height_m and np_weight_kg to calculate the BMI of each player. Use the following equation:

$$\text{BMI} = \frac{\text{weight(kg)}}{\text{height(m)}^2}$$

```
[ ]  BMI = weights_kg / (np_height_in ** 2)

     print(BMI)
```

```
[0.01359128 0.01482685 0.0184275  0.01515866 0.01813038 0.01658475
 0.01543314 0.02021154 0.02323543 0.02046004 0.02388343 0.01820826
 0.01761967 0.01673251 0.01298304 0.01529696 0.01409112 0.016975
 0.01489954 0.0212625  0.01258272 0.01491015 0.01832914 0.01383331
 0.01647771 0.018144   0.01817017 0.02204563 0.01709661 0.0146405
 0.01514805 0.01821487 0.01889681 0.01491015 0.01403325 0.02204586
 0.01915181 0.0160125  0.015552   0.01169437 0.02064384 0.01847086
 0.02193943 0.01617756 0.01564457 0.01688435 0.01461302 0.01719407
 0.01742222 0.01523633]
```

DataYard
Professional eLearning Partner

NumPy

Boolean Indexing - NumPy

Suppose we have an array named `array1`.

```
array1 = np.array([12, 24, 16, 21, 32, 29, 7, 15])
```

Now let's create a mask that selects all elements of `array1` that are greater than **20**.

```
boolean_mask = array1 > 20
```

Here, `array1 > 20` creates a boolean mask that evaluates to `True` for elements that are greater than **20**, and `False` for elements that are less than or equal to **20**.

The resulting mask is an array stored in the `boolean_mask` variable as:

```
[False, True, False, True, True, True, False, False]
```

DataYard
Professional eLearning Partner

NumPy

Task #7 – NumPy

Task# 7

Create a list and use a logical operator to check which values in a list are greater than 5.

```
[ ]  array1 = np.array([12, 2, 5, 0,7,11,10])

     boolean_mask = array1 > 5

     print(boolean_mask)
```

```
[ True False False False  True  True  True]
```

DataYard
Professional eLearning Partner

NumPy

Task #8 – NumPy

Task# 8

Use the BMI calculation to create a boolean numpy array: the element of the array should be True if the corresponding baseball player's BMI is below 21.

You can use the < operator for this. Name the array 'np_light.'

Print the array light.

```
[ ]  np_light = BMI < 21

     print(np_light)
```

```
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True]
```

DataYard
Professional eLearning Partner

NumPy

Task #9 – NumPy

Task# 9

Create a boolean numpy array: the element of the array should be True if the corresponding baseball player's BMI is greater then 21.

You can use the > operator for this.

Name the array np_heavy. Print np_heavy

```
np_heavy = BMI > 21

print(np_heavy)
```

```
[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 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]
```

DataYard
Professional eLearning Partner

NumPy

Task #10 – NumPy

Task# 10

a. Using the MLB data subset and print out the element at index 50 from np_weight_lb.

b. Print out a sub-array of np_height_in that contains the elements at index 30 up to and 40.

```
[ ] weights_kg[49]
```

```
np.float64(81.1944)
```

+ Code + Text

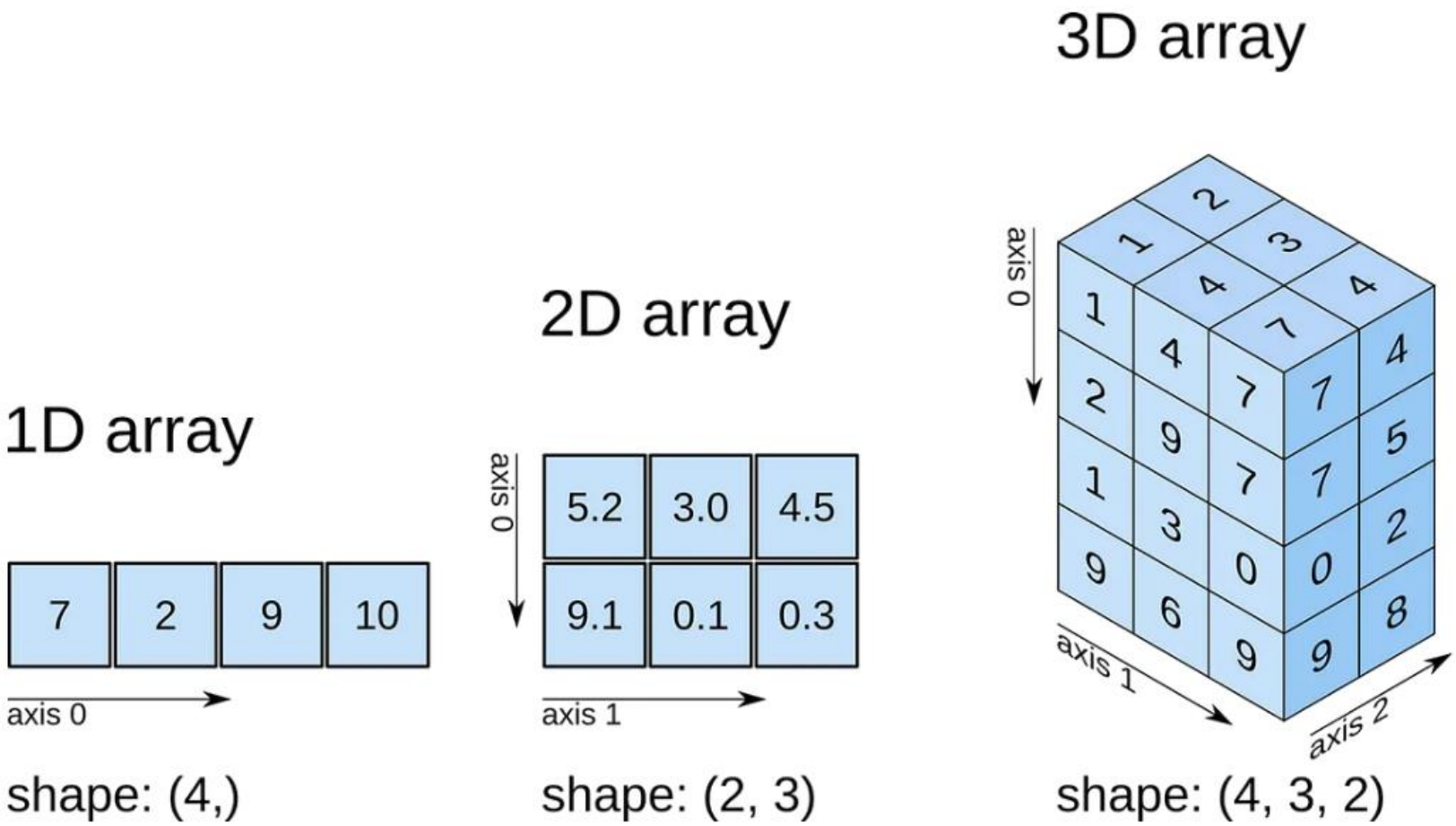
```
[ ] np_height_in[29:40]
```

```
array([73, 77, 80, 77, 74, 80, 73, 73, 72, 70, 80])
```

DataYard
Professional eLearning Partner

NumPy

NumPy – 1D, 2D, 3D Arrays



NumPy

Task #11 – NumPy

Task# 11

a. Use np.array() to create a 2D numpy array from baseball list. Name it np_baseball. Print out the shape attribute of np_baseball. Use np_baseball.shape.

```
np_baseball = np.array([[74, 180], [74, 215], [72, 210], [72, 210], [73, 188], [69, 176], [69, 209], [71, 200], [76, 231], [71, 180], [73, 188], [73, 180], [74, 185]])

np_baseball
```

```
array([[ 74, 180],
       [ 74, 215],
       [ 72, 210],
       [ 72, 210],
       [ 73, 188],
       [ 69, 176],
       [ 69, 209],
       [ 71, 200],
       [ 76, 231],
       [ 71, 180],
       [ 73, 188],
       [ 73, 180],
       [ 74, 185]])
```

```
[ ] np_baseball.shape
```

```
(13, 2)
```

DataYard
Professional eLearning Partner

NumPy

Task #12 – NumPy

Task# 12

a. Using the 2D array you created earlier, print out the 10th row of np_baseball.

b. Make a new variable, np_weight_lb_n, containing the entire second column of np_baseball. Print out np_weight_lb_n

c. Select the height (first column) of the 4th baseball player in np_baseball and print it out.

```
[ ]  np_baseball[[9]]
```

```
array([[ 71, 180]])
```

```
[ ]  np_weight_lb_n = np_baseball[:,1]

     np_weight_lb_n
```

```
array([180, 215, 210, 210, 188, 176, 209, 200, 231, 180, 188, 180, 185])
```

```
[ ]  np_baseball[3,0]
```

```
np.int64(72)
```

DataYard
Professional eLearning Partner

NumPy

Task #13 – NumPy

Task# 13

Create a Numpy Array price using the below matrix and add 100 to it.

```
price = np.array([
    [690, 199.00],
    [199, 192.00],
    [959, 913.00],
    [683, 129.00],
    [188, 510.00],
    [592, 207.00],
    [245, 507.00]
])

updated_price = price + 100

print(updated_price)
```

```
[[ 790.  299.]
 [ 299.  292.]
 [1059. 1013.]
 [ 783.  229.]
 [ 288.  610.]
 [ 692.  307.]
 [ 345.  607.]]
```

DataYard
Professional eLearning Partner

NumPy

Task #14 – NumPy

Task# 14

Multiply the Price in 2019 to the Inflation Rate (7 %) and Store the new price as price2020x_adjusted

```
[ ] price2019 = price[:, 0]

    price2020x_adjusted = price2019 * 1.07

    print(price2020x_adjusted)
```

```
[ 738.3   212.93 1026.13  730.81  201.16  633.44  262.15]
```

DataYard
Professional eLearning Partner

NumPy