**PRACTICE TEST**

**You have to solve each of them in jupyter notebook. Even if you don’t have a dataset, just write the demo code and I will check it. This exercise is related to the Numpy arrays. There are a total of 5 questions. First, write solutions and match them with the solutions below.**

**our First NumPy Array**

In this chapter, we're going to dive into the world of baseball. Along the way, you'll get comfortable with the basics of numpy, a powerful package to do data science.

A list baseball has already been defined in the Python script, representing the height of some baseball players in centimeters. Can you add some code here and there to create a numpy array from it?

**Instructions**

**100 XP**

* Import the numpy package as np, so that you can refer to numpy with np.
* Use [**np.array()**](http://docs.scipy.org/doc/numpy-1.10.0/glossary.html#term-array) to create a numpy array from baseball. Name this array np\_baseball.
* Print out the type of np\_baseball to check that you got it right.

**Baseball players' height**

You are a huge baseball fan. You decide to call the MLB (Major League Baseball) and ask around for some more statistics on the height of the main players. They pass along data on more than a thousand players, which is stored as a regular Python list: height\_in. The height is expressed in inches. Can you make a numpy array out of it and convert the units to meters?

height\_in is already available and the numpy package is loaded, so you can start straight away (Source: stat.ucla.edu).

**Instructions**

**100 XP**

* Create a numpy array from height\_in. Name this new array np\_height\_in.
* Print np\_height\_in.
* Multiply np\_height\_in with 0.0254 to convert all height measurements from inches to meters. Store the new values in a new array, np\_height\_m.
* Print out np\_height\_m and check if the output makes sense.

**Baseball player's BMI**

The MLB also offers to let you analyze their weight data. Again, both are available as regular Python lists: height\_in and weight\_lb. height\_in is in inches and weight\_lb is in pounds.

It's now possible to calculate the BMI of each baseball player. Python code to convert height\_in to a numpy array with the correct units is already available in the workspace. Follow the instructions step by step and finish the game! height\_in and weight\_lb are available as regular lists.

**Instructions**

**100 XP**

**Instructions**

**100 XP**

* Create a numpy array from the weight\_lb list with the correct units. Multiply by 0.453592 to go from pounds to kilograms. Store the resulting numpy array as np\_weight\_kg.
* Use np\_height\_m and np\_weight\_kg to calculate the BMI of each player. Use the following equation:BMI=weight(kg)height(m)2Save the resulting numpy array as bmi.
* Print out bmi.

**Lightweight baseball players**

To subset both regular Python lists and numpy arrays, you can use square brackets:

x = [4 , 9 , 6, 3, 1]

x[1]

import numpy as np

y = np.array(x)

y[1]

For numpy specifically, you can also use boolean numpy arrays:

high = y > 5

y[high]

The code that calculates the BMI of all baseball players is already included. Follow the instructions and reveal interesting things from the data! height\_in and weight\_lb are available as regular lists.

**Instructions**

**100 XP**

* 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 light.
* Print the array light.
* Print out a numpy array with the BMIs of all baseball players whose BMI is below 21. Use light inside square brackets to do a selection on the bmi array.

# NumPy Side Effects

As Hugo explained before, numpy is great for doing vector arithmetic. If you compare its functionality with regular Python lists, however, some things have changed.

First of all, numpy arrays cannot contain elements with different types. If you try to build such a list, some of the elements' types are changed to end up with a homogeneous list. This is known as *type coercion*.

Second, the typical arithmetic operators, such as +, -, \* and / have a different meaning for regular Python lists and numpy arrays.

Have a look at this line of code:

np.array([True, 1, 2]) + np.array([3, 4, False])

Can you tell which code chunk builds the exact same Python object? The numpy package is already imported as np, so you can start experimenting in the IPython Shell straight away!

## Instructions

**50 XP**

### Possible answers



np.array([True, 1, 2, 3, 4, False])



np.array([4, 3, 0]) + np.array([0, 2, 2])



np.array([1, 1, 2]) + np.array([3, 4, -1])



np.array([0, 1, 2, 3, 4, 5])

# Blend it all together

In the last few exercises you've learned everything there is to know about heights and weights of baseball players. Now it's time to dive into another sport: soccer.

You've contacted FIFA for some data and they handed you two lists. The lists are the following:

positions = ['GK', 'M', 'A', 'D', ...]

heights = [191, 184, 185, 180, ...]

Each element in the lists corresponds to a player. The first list, positions, contains strings representing each player's position. The possible positions are: 'GK' (goalkeeper), 'M' (midfield), 'A' (attack) and 'D' (defense). The second list, heights, contains integers representing the height of the player in cm. The first player in the lists is a goalkeeper and is pretty tall (191 cm).

You're fairly confident that the median height of goalkeepers is higher than that of other players on the soccer field. Some of your friends don't believe you, so you are determined to show them using the data you received from FIFA and your newly acquired Python skills. heights and positions are available as lists

## Instructions

**100 XP**

## Instructions

**100 XP**

* Convert heights and positions, which are regular lists, to numpy arrays. Call them np\_heights and np\_positions.
* Extract all the heights of the goalkeepers. You can use a little trick here: use np\_positions == 'GK' as an index for np\_heights. Assign the result to gk\_heights.
* Extract all the heights of all the other players. This time use np\_positions != 'GK' as an index for np\_heights. Assign the result to other\_heights.
* Print out the median height of the goalkeepers using [**np.median()**](http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.median.html). Replace None with the correct code.
* Do the same for the other players. Print out their median height. Replace None with the correct code.

**SOLUTIONS**

**QUESTION 1**

**# Import the numpy package as np**

**import numpy as np**

**# Create list baseball**

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

**# Create a numpy array from baseball: np\_baseball**

**np\_baseball=np.array(baseball)**

**# Print out type of np\_baseball**

**print(type(np\_baseball))**

QUESTION 2

# Import numpy

import numpy as np

# Create a numpy array from height\_in: np\_height\_in

np\_height\_in=np.array(height\_in)

# Print out np\_height\_in

print(np\_height\_in)

# Convert np\_height\_in to m: np\_height\_m

np\_height\_m=np\_height\_in\*0.0254

# Print np\_height\_m

print(np\_height\_m)

QUESTION 3

# Import numpy

import numpy as np

# Create array from height\_in with metric units: np\_height\_m

np\_height\_m = np.array(height\_in) \* 0.0254

# Create array from weight\_lb with metric units: np\_weight\_kg

np\_weight\_kg=np.array(weight\_lb)\*0.453592

# Calculate the BMI: bmi

bmi=np\_weight\_kg/np\_height\_m\*\*2

# Print out bmi

print(bmi)

QUESTION 4

# Import numpy

import numpy as np

# Calculate the BMI: bmi

np\_height\_m = np.array(height\_in) \* 0.0254

np\_weight\_kg = np.array(weight\_lb) \* 0.453592

bmi = np\_weight\_kg / np\_height\_m \*\* 2

# Create the light array

light=np.array(bmi<21)

# Print out light

print(light)

# Print out BMIs of all baseball players whose BMI is below 21

print(bmi[light])

QUESTION 4

np.array([4, 3, 0]) + np.array([0, 2, 2])

QUESTION 5

# Import numpy

import numpy as np

# Convert positions and heights to numpy arrays: np\_positions, np\_heights

np\_positions=np.array(positions)

np\_heights=np.array(heights)

# Heights of the goalkeepers: gk\_heights

gk\_heights = np\_heights[np\_positions == 'GK']

# Heights of the other players: other\_heights

other\_heights=np\_heights[np\_positions!='GK']

# Print out the median height of goalkeepers. Replace 'None'

print("Median height of goalkeepers: " + str(np.median(gk\_heights)))

# Print out the median height of other players. Replace 'None'

print("Median height of other players: " + str(np.median(other\_heights)))