1. **Your First 2D NumPy Array**

Before working on the actual MLB data, let's try to create a 2D numpy array from a small list of lists.

In this exercise, baseball is a list of lists. The main list contains 4 elements. Each of these elements is a list containing the height and the weight of 4 baseball players, in this order. baseball is already coded for you in the script.

* Use **[np.array()](http://docs.scipy.org/doc/numpy-1.10.0/glossary.html" \l "term-array" \t "_blank)** to create a 2D numpy array from baseball. Name it np\_baseball.
* Print out the type of np\_baseball.
* Print out the shape attribute of np\_baseball. Use np\_baseball.shape.

# Baseball data in 2D form

You have another look at the MLB data and realize that it makes more sense to restructure all this information in a 2D numpy array. This array should have 1015 rows, corresponding to the 1015 baseball players you have information on, and 2 columns (for height and weight).

The MLB was, again, very helpful and passed you the data in a different structure, a Python list of lists. In this list of lists, each sublist represents the height and weight of a single baseball player. The name of this embedded list is baseball.

Can you store the data as a 2D array to unlock numpy's extra functionality?

* Use **[np.array()](http://docs.scipy.org/doc/numpy-1.10.0/glossary.html" \l "term-array" \t "_blank)** to create a 2D numpy array from baseball. Name it np\_baseball.
* Print out the shape attribute of np\_baseball.

# Subsetting 2D NumPy Arrays

If your 2D numpy array has a regular structure, i.e. each row and column has a fixed number of values, complicated ways of subsetting become very easy. Have a look at the code below where the elements "a" and "c" are extracted from a list of lists.

# regular list of lists

x = [["a", "b"], ["c", "d"]]

[x[0][0], x[1][0]]

# numpy

import numpy as np

np\_x = np.array(x)

np\_x[:,0]

For regular Python lists, this is a real pain. For 2D numpy arrays, however, it's pretty intuitive! The indexes before the comma refer to the rows, while those after the comma refer to the columns. The : is for slicing; in this example, it tells Python to include all rows.

The code that converts the pre-loaded baseball list to a 2D numpy array is already in the script. The first column contains the players' height in inches and the second column holds player weight, in pounds. Add some lines to make the correct selections. Remember that in Python, the first element is at index 0!

* Print out the 50th row of np\_baseball.
* Make a new variable, np\_weight, containing the entire second column of np\_baseball.
* Select the height (first column) of the 124th baseball player in np\_baseball and print it out.

# 2D Arithmetic

Remember how you calculated the Body Mass Index for all baseball players? numpy was able to perform all calculations element-wise (i.e. element by element). For 2D numpy arrays this isn't any different! You can combine matrices with single numbers, with vectors, and with other matrices.

Execute the code below in the IPython shell and see if you understand:

import numpy as np

np\_mat = np.array([[1, 2],

[3, 4],

[5, 6]])

np\_mat \* 2

np\_mat + np.array([10, 10])

np\_mat + np\_mat

np\_baseball is coded for you; it's again a 2D numpy array with 3 columns representing height (in inches), weight (in pounds) and age (in years).

* You managed to get hold of the changes in height, weight and age of all baseball players. It is available as a 2D numpy array, updated. Add np\_baseball and updated and print out the result.
* You want to convert the units of height and weight to metric (meters and kilograms respectively). As a first step, create a numpy array with three values: 0.0254, 0.453592 and 1. Name this array conversion.
* Multiply np\_baseball with conversion and print out the result.