1. **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!

* 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 bmiarray.

# NumPy Side Effects

As Filip 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!

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

# Subsetting NumPy Arrays

You've seen it with your own eyes: Python lists and numpy arrays sometimes behave differently. Luckily, there are still certainties in this world. For example, subsetting (using the square bracket notation on lists or arrays) works exactly the same. To see this for yourself, try the following lines of code in the IPython Shell:

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

x[1]

np\_x = np.array(x)

np\_x[1]

The script on the right already contains code that imports numpy as np, and stores both the height and weight of the MLB players as numpy arrays.

* Subset np\_weight by printing out the element at index 50.
* Print out a sub-array of np\_height that contains the elements at index 100 up to **and including** index 110.