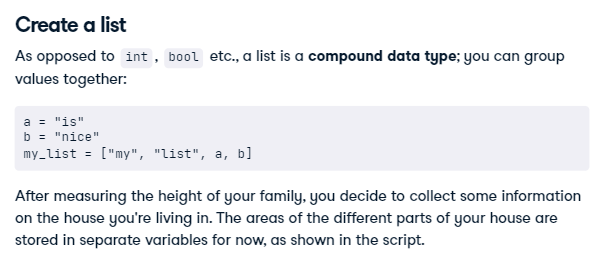
**Incorrect Submission**

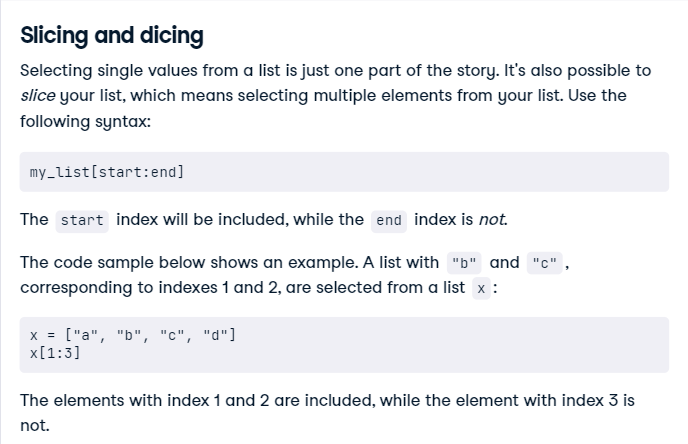
It's perfectly possible to 'multiply strings' in Python...

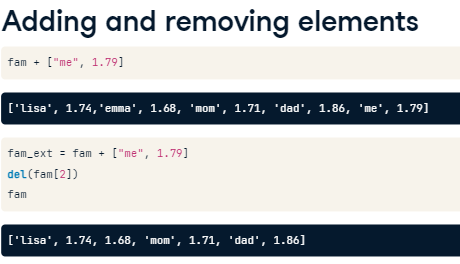


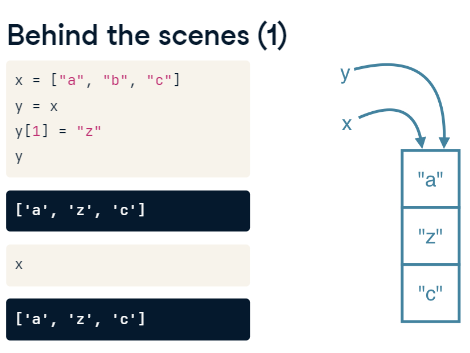


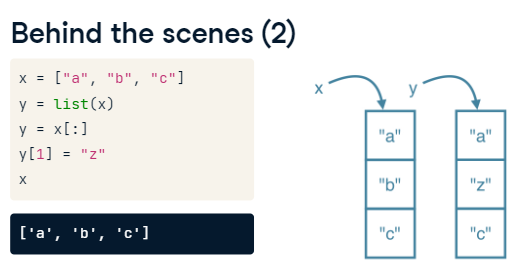


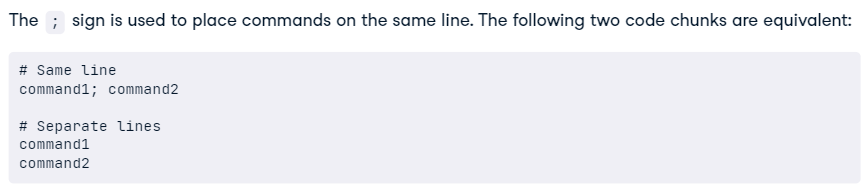
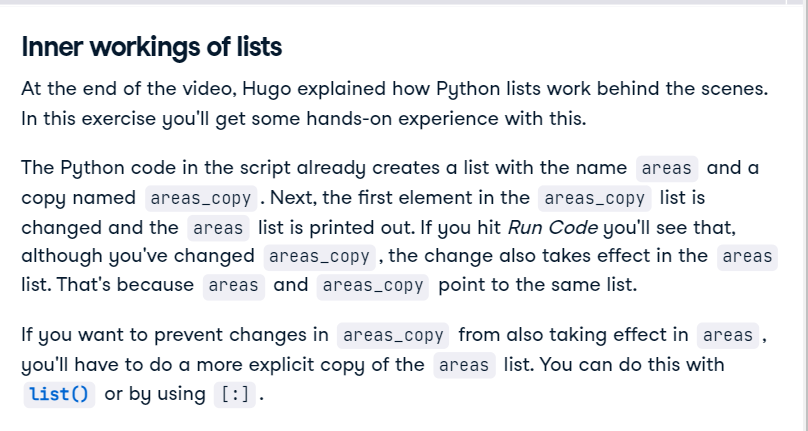
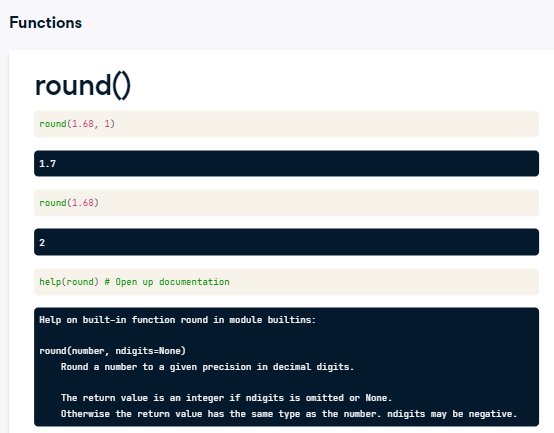


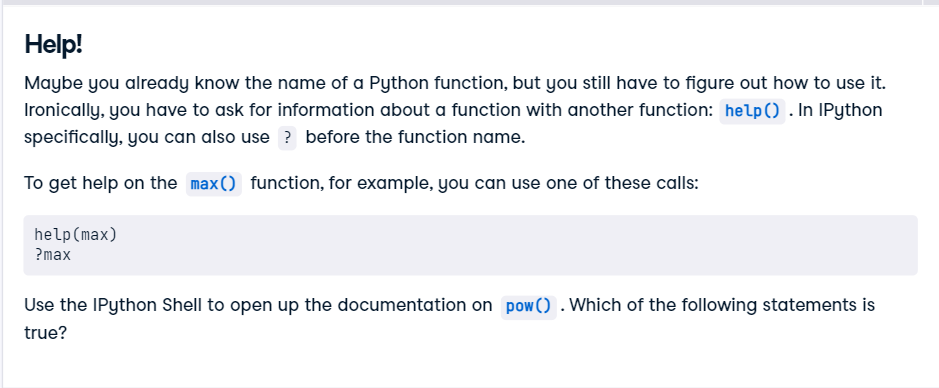


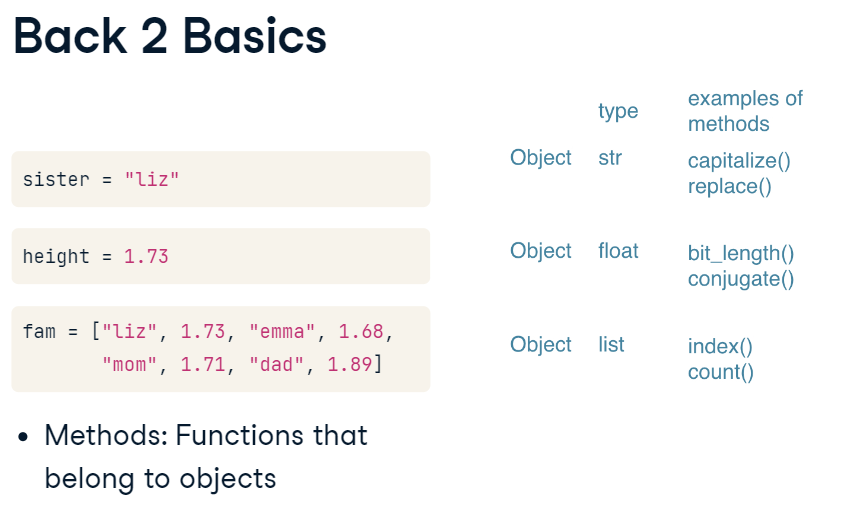




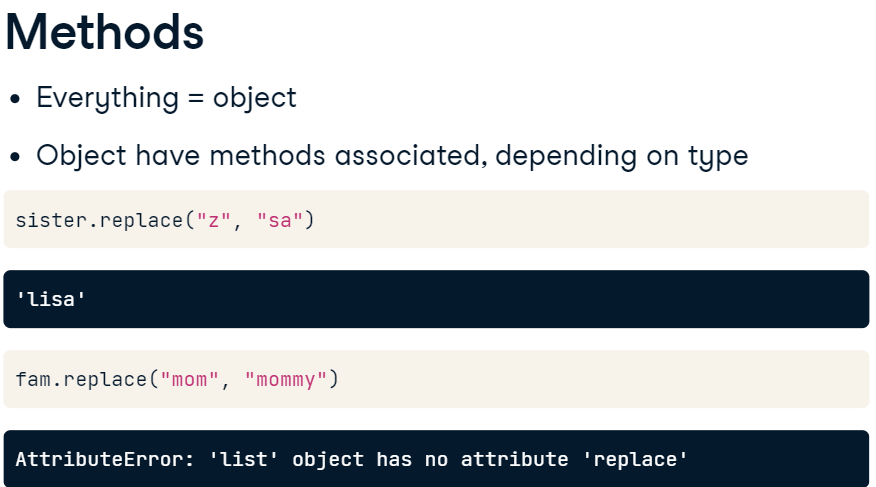


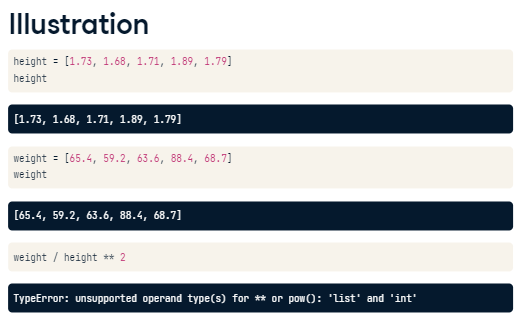


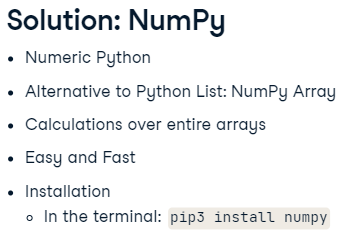


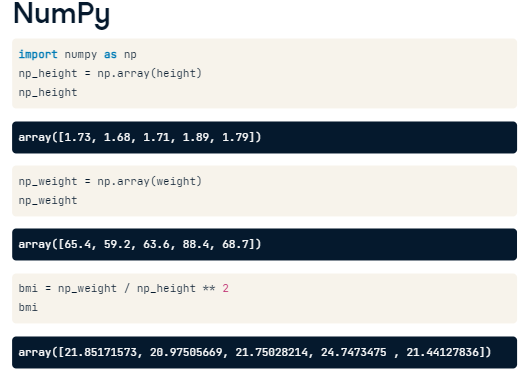






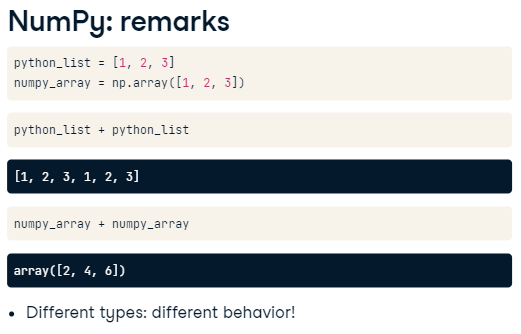


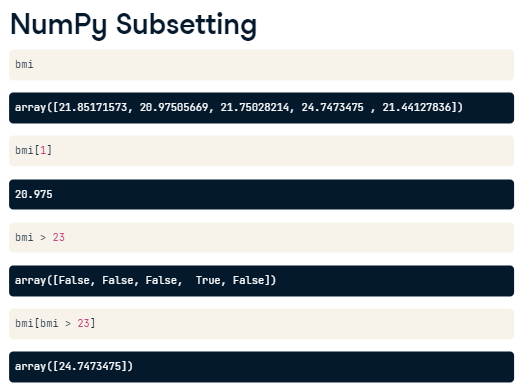










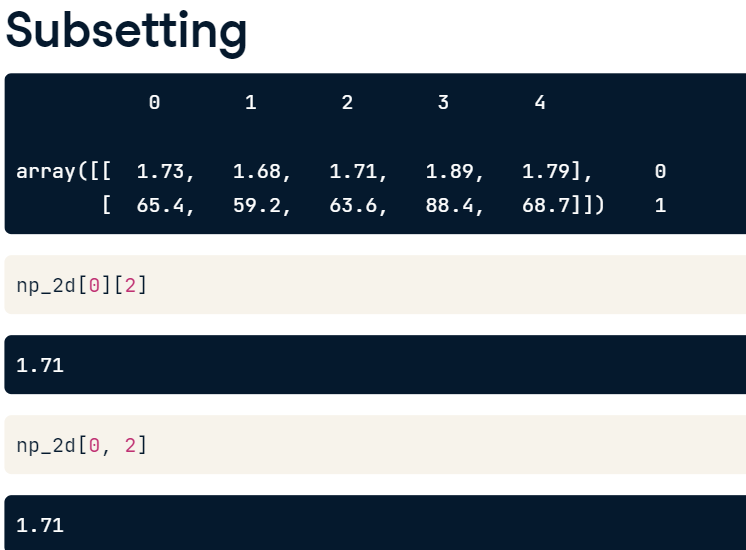


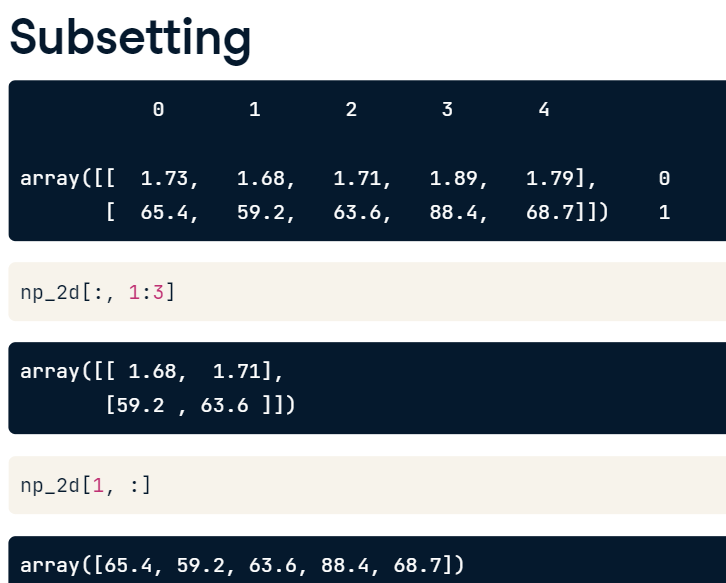
# NumPy Side Effects

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.





**#Import numpy**

**import numpy as np**

**# Create baseball, a list of lists**

**baseball = [[180, 78.4],**

**[215, 102.7],**

**[210, 98.5],**

**[188, 75.2]]**

**# Create a 2D numpy array from baseball: np\_baseball**

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

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

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

**# Print out the shape of np\_baseball**

**<class 'numpy.ndarray'>**

**<script.py> output:**

**<class 'numpy.ndarray'>**

**print(np\_baseball.shape)**

**(4, 2)**

**NameError: name 'np\_baseball' is not defined**

**# Import numpy package**

**import numpy as np**

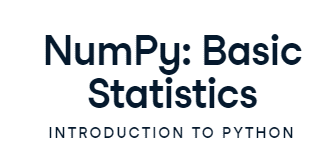
**# Create a 2D numpy array from baseball: np\_baseball**

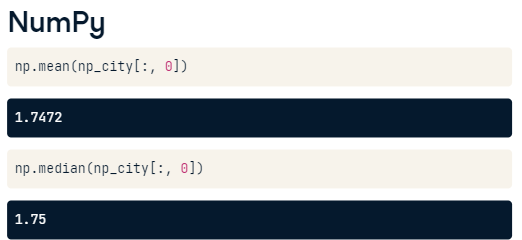
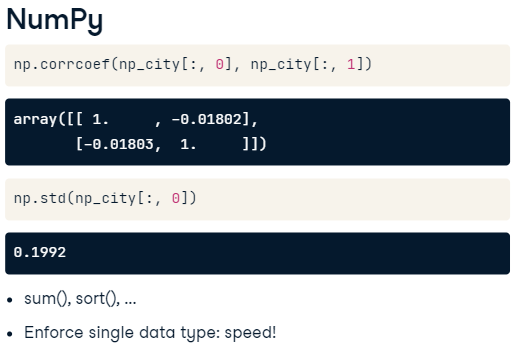
**np\_baseball = np.array(baseball)**

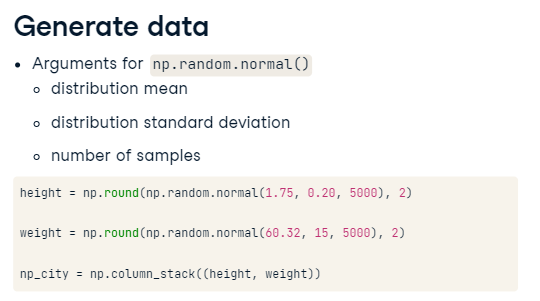
**# Print out the shape of np\_baseball**

**print(np\_baseball.shape)**

**(1015, 2)**







* 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" \t "_blank)**. Replace None with the correct code.
* Do the same for the other players. Print out their median height. Replace None with the correct code.

# Import numpy

import numpy as np

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

np\_heights = np.array(height)

np\_positions = np.array(positions)

# 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)))