DataCamp Python Courses

01-intro-to-python-for-data-science

02-intermediate-python-for-data-science

03-python-data-science-toolbox-(part-1)

04-statistical-thinking-in-python-(part1)

05-importing-data-in-python-(part-1)

06-python-data-science-toolbox-(part-2)

07-statistical-thinking-in-python-(part-2)

08-importing-data-in-python-(part-2)

09-manipulating-time-series-data-in-python

10-cleaning-data-in-python

11-pandas-foundations

12-manipulating-dataframes-with-pandas

13-introduction-to-databases-in-python

14-merging-dataframes-with-pandas

15-introduction-to-data-visualization-with-python

16-interactive-data-visualization-with-bokeh

17-supervised-learning-with-scikit-learn

18-machine-learning-with-the-experts-school-budgets

19-unsupervised-learning-in-python

20-deep-learning-in-python

21-network-analysis-in-python-(part-1)

01 Intro-to-python-for-data-science done

1 Python Basics

100%

An introduction to the basic concepts of Python. Learn how to use Python both interactively and through a script. Create your first variables and acquaint yourself with Python's basic data types.

Hello Python!	✓ 50 xp
The Python Interface	✓ 100 xp
When to use Python?	✓ 50 xp
Any comments?	✓ 100 xp
Python as a calculator	✓ 100 xp
Variables & Types	✓ 50 xp
Variable Assignment	✓ 100 xp
Calculations with variables	✓ 100 xp
Other variable types	✓ 100 xp
Guess the type	✓ 50 xp
Operations with other types	✓ 100 xp
Type conversion	✓ 100 xp
Can Python handle everything?	✓ 50 xp

INTRO TO PYTHON FOR DATA SCIENCE

Hello Python!

What you will learn

- Python
- Specifically for Data Science
- Store data
- Manipulate data
- Tools for data analysis

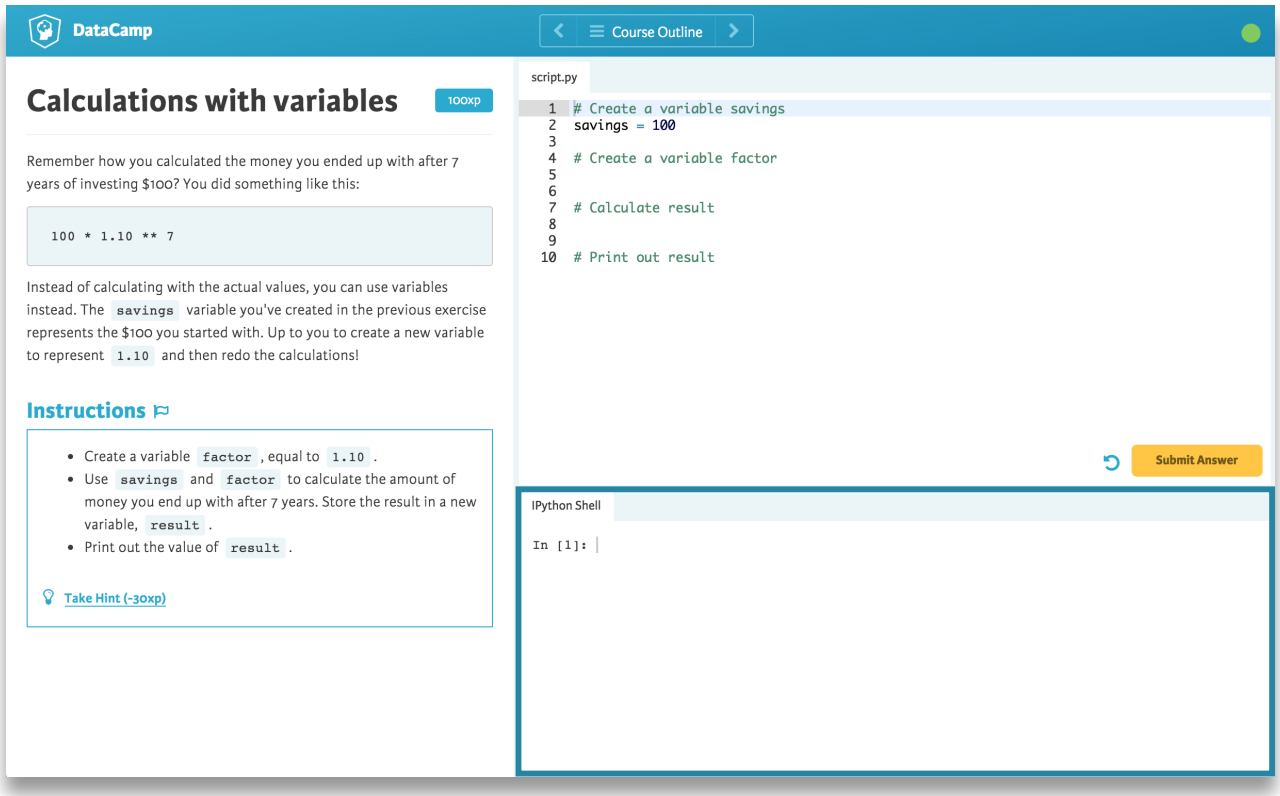
How you will learn

Python

- Guido Van Rossum
- General Purpose: build anything
- Open Source! Free!
- Python Packages, also for Data Science
    - Many applications and fields
- Version 3.x - https://www.python.org/downloads/

IPython Shell

Execute Python commands



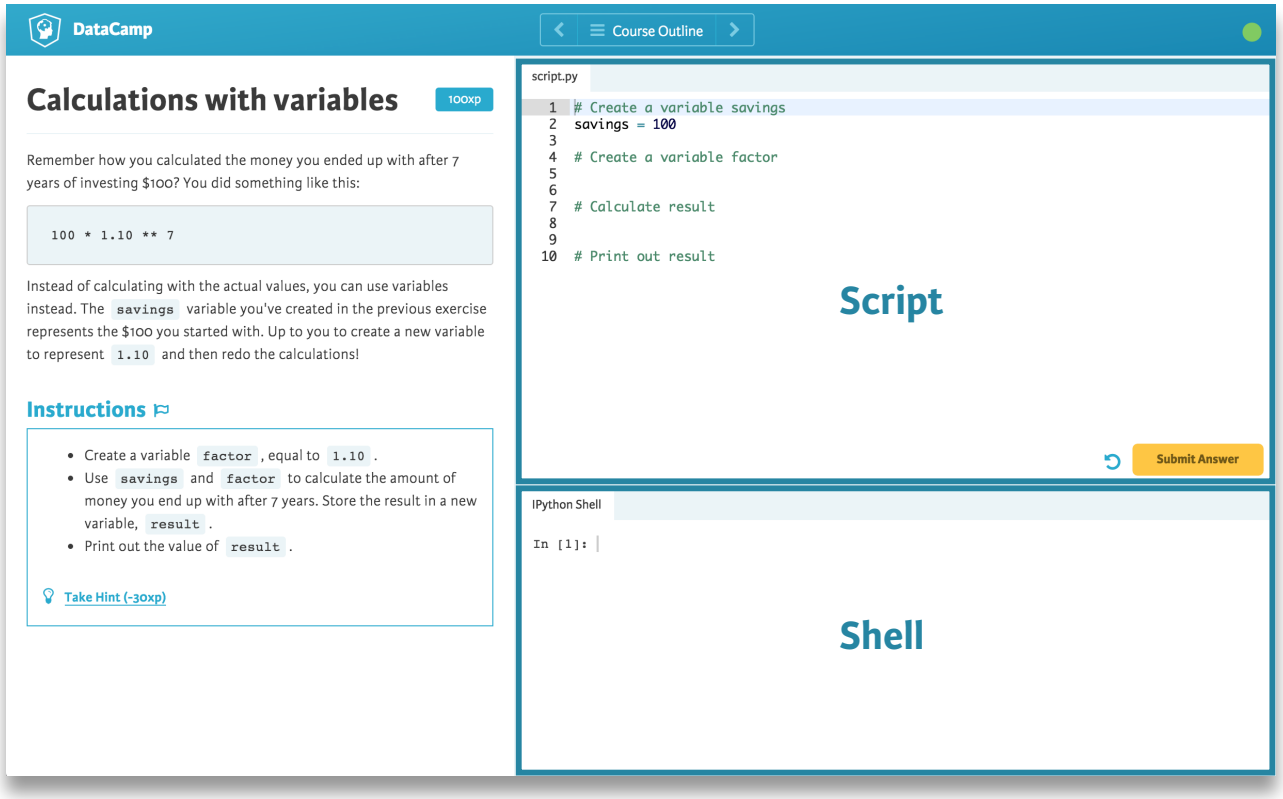
IPython Shell

Python Script

- Text Files - .py
- List of Python Commands
- Similar to typing in IPython Shell

Python Script

DataCamp Interface



INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

Variables and Types

Variable

- Specific, case-sensitive name
- Call up value through variable name
- 1.79 m - 68.7 kg

```
In [1]: height = 1.79

In [2]: weight = 68.7

In [3]: height
Out[3]: 1.79
```

Calculate BMI

```
In [1]: height = 1.79

In [2]: weight = 68.7

In [3]: height
Out[3]: 1.79

In [4]: 68.7 / 1.79 ** 2
Out[4]: 21.4413

In [5]: weight / height ** 2
Out[5]: 21.4413

In [6]: bmi = weight / height ** 2

In [7]: bmi
Out[7]: 21.4413
```

$$\text{BMI} = \frac{\text{weight}}{\text{height}^2}$$

Reproducibility

my_script.py

```
height = 1.79
weight = 68.7
bmi = weight / height ** 2
print(bmi)
```

```
Output:
21.4413
```

Reproducibility

my_script.py

```
height = 1.79
weight = 74.2
bmi = weight / height ** 2
print(bmi)
```

```
Output:
23.1578
```

Python Types

```
In [8]: type(bmi)
Out[8]: float

In [9]: day_of_week = 5

In [10]: type(day_of_week)
Out[10]: int
```

Python Types (2)

```
In [11]: x = "body mass index"

In [12]: y = 'this works too'

In [13]: type(y)
Out[13]: str

In [14]: z = True

In [15]: type(z)
Out[15]: bool
```

Python Types (3)

```
In [16]: 2 + 3
Out[16]: 5

In [17]: 'ab' + 'cd'
Out[17]: 'abcd'
```

Different type = different behavior!

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

2 Python Lists

100%

Learn to store, access and manipulate data in lists: the first step towards efficiently working with huge amounts of data.

Lists, what are they?	✓ 50 xp
Create a list	✓ 100 xp
Create list with different types	✓ 100 xp
Select the valid list	✓ 50 xp
List of lists	✓ 100 xp
Subsetting lists	✓ 50 xp
Subset and conquer	✓ 100 xp
Subset and calculate	✓ 100 xp
Slicing and dicing	✓ 100 xp
Slicing and dicing (2)	✓ 100 xp
Subsetting lists of lists	✓ 50 xp
List Manipulation	✓ 50 xp
Replace list elements	✓ 100 xp
Extend a list	✓ 100 xp
Delete list elements	✓ 50 xp
Inner workings of lists	✓ 100 xp

INTRO TO PYTHON FOR DATA SCIENCE

Python Lists

Python Data Types

- `float` - real numbers
- `int` - integer numbers
- `str` - string, text
- `bool` - `True`, `False`

```
In [1]: height = 1.73

In [2]: tall = True
```

- Each variable represents single value

Problem

- Data Science: many data points
- Height of entire family

```
In [3]: height1 = 1.73

In [4]: height2 = 1.68

In [5]: height3 = 1.71

In [6]: height4 = 1.89
```

- Inconvenient

Python List [a, b, c]

```
In [7]: [1.73, 1.68, 1.71, 1.89]
Out[7]: [1.73, 1.68, 1.71, 1.89]

In [8]: fam = [1.73, 1.68, 1.71, 1.89]

In [9]: fam
Out[9]: [1.73, 1.68, 1.71, 1.89]
```

- Name a collection of values
- Contain any type
- Contain different types

Python List [a, b, c]

```
In [10]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [11]: fam
Out[11]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
```

```
["liz", 1.73]
["emma", 1.68]
["mom", 1.71]
["dad", 1.89]
```

Python List [a, b, c]

```
In [10]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [11]: fam
Out[11]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]


In [11]: fam2 = [["liz", 1.73],
                 ["emma", 1.68],
                 ["mom", 1.71],
                 ["dad", 1.89]]

In [12]: fam2
Out[12]: [['liz', 1.73], ['emma', 1.68],
               ['mom', 1.71], ['dad', 1.89]]
```

List type

```
In [13]: type(fam)
Out[13]: list

In [14]: type(fam2)
Out[14]: list
```

- Specific functionality
- Specific behavior

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

Subsetting Lists

Subsetting lists

```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
    index:   0      1      2      3      4      5      6      7
```

"zero-based indexing"

Subsetting lists

```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
    index:   0      1      2      3      4      5      6      7


In [3]: fam[3]
Out[3]: 1.68
```

Subsetting lists

```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
    index:   0      1      2      3      4      5      6      7


In [3]: fam[3]
Out[3]: 1.68

In [4]: fam[6]
Out[4]: 'dad'
```

Subsetting lists

```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
    index:   0      1      2      3      4      5      6      7
            -8     -7     -6     -5     -4     -3     -2     -1

In [3]: fam[3]
Out[3]: 1.68

In [4]: fam[6]
Out[4]: 'dad'

In [5]: fam[-1]
Out[5]: 1.89
```

Subsetting lists

```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
    index:   0      1      2      3      4      5      6      7
            -8     -7     -6     -5     -4     -3     -2     -1

In [3]: fam[3]
Out[3]: 1.68

In [4]: fam[6]
Out[4]: 'dad'

In [5]: fam[-1]
Out[5]: 1.89

In [6]: fam[-2]
Out[6]: 'dad'
```

List slicing

```
In [7]: fam
Out[7]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
          0     1      2      3     4     5     6      7

In [8]: fam[3:5]
Out[8]: [1.68, 'mom']
```

[start : end]

inclusive **exclusive**

List slicing

```
In [7]: fam
Out[7]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
          0     1      2      3     4     5     6      7

In [8]: fam[3:5]
Out[8]: [1.68, 'mom']

In [9]: fam[1:4]
Out[9]: [1.73, 'emma', 1.68]
```

[start : end]

inclusive **exclusive**

List slicing

```
In [7]: fam
Out[7]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
          0      1       2      3     4     5      6      7

In [8]: fam[3:5]
Out[8]: [1.68, 'mom']

In [9]: fam[1:4]
Out[9]: [1.73, 'emma', 1.68]

In [10]: fam[:4]
Out[10]: ['liz', 1.73, 'emma', 1.68]
```

List slicing

```
In [7]: fam
Out[7]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]
          0     1      2      3      4     5      6      7

In [8]: fam[3:5]
Out[8]: [1.68, 'mom']

In [9]: fam[1:4]
Out[9]: [1.73, 'emma', 1.68]

In [10]: fam[:4]
Out[10]: ['liz', 1.73, 'emma', 1.68]

In [11]: fam[5:]
Out[11]: [1.71, 'dad', 1.89]
```

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

Manipulating Lists

List Manipulation

- Change list elements
- Add list elements
- Remove list elements

Changing list elements

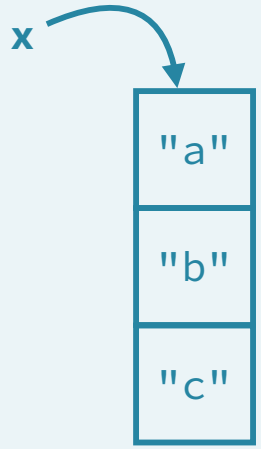
```
In [1]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]

In [2]: fam
Out[2]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

In [3]: fam[7] = 1.86

In [4]: fam
Out[4]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]

In [5]: fam[0:2] = ["lisa", 1.74]

In [6]: fam
Out[6]: ['lisa', 1.74, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

Adding and removing elements

```
In [7]: fam + ["me", 1.79]
Out[7]: ['lisa', 1.74,'emma', 1.68,
                          'mom', 1.71, 'dad', 1.86, 'me', 1.79]

In [8]: fam_ext = fam + ["me", 1.79]



In [9]: del(fam[2])

In [10]: fam
Out[10]: ['lisa', 1.74, 1.68, 'mom', 1.71, 'dad', 1.86]

In [11]: del(fam[2])

In [12]: fam
Out[12]: ['lisa', 1.74, 'mom', 1.71, 'dad', 1.86]
```

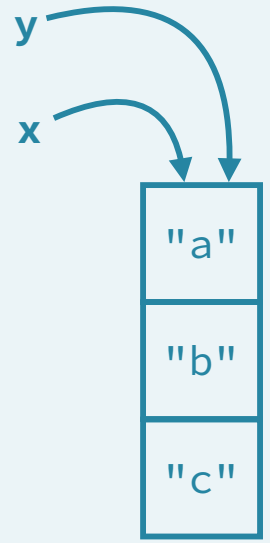
Behind the scenes (1)

```
In [13]: x = ["a", "b", "c"]

In [14]: y = x
```



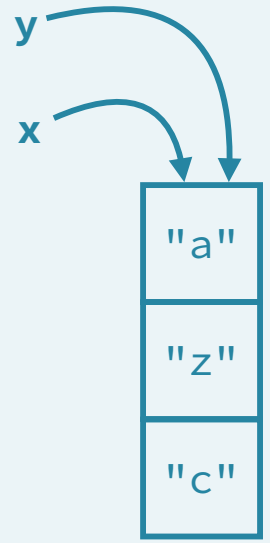
Behind the scenes (1)

```
In [13]: x = ["a", "b", "c"]

In [14]: y = x

In [15]: y[1] = "z"

In [16]: y
Out[16]: ['a', 'z', 'c']

In [17]: x
Out[17]: ['a', 'z', 'c']
```



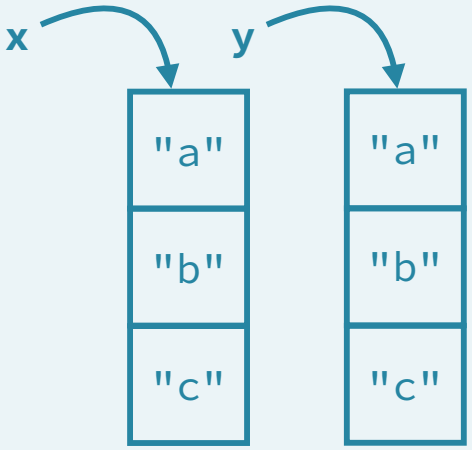
Behind the scenes (1)

```
In [13]: x = ["a", "b", "c"]

In [14]: y = x

In [15]: y[1] = "z"

In [16]: y
Out[16]: ['a', 'z', 'c']

In [17]: x
Out[17]: ['a', 'z', 'c']
```



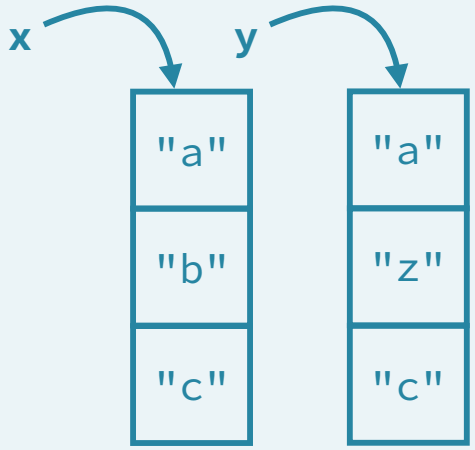
Behind the scenes (2)

```
In [18]: x = ["a", "b", "c"]

In [19]: y = list(x)

In [20]: y = x[:]

In [21]: y[1] = "z"
```



Behind the scenes (2)

```
In [18]: x = ["a", "b", "c"]

In [19]: y = list(x)

In [20]: y = x[:]

In [21]: y[1] = "z"

In [22]: x
Out[22]: ['a', 'b', 'c']
```



INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

3 Functions and Packages

100%

To leverage the code that brilliant Python developers have written, you'll learn about using functions, methods and packages. This will help you to reduce the amount of code you need to solve challenging problems!

Functions	✓ 50 xp
Familiar functions	✓ 100 xp
Help!	✓ 50 xp
Multiple arguments	✓ 100 xp
Methods	✓ 50 xp
String Methods	✓ 100 xp
List Methods	✓ 100 xp
List Methods (2)	✓ 100 xp
Packages	✓ 50 xp
Import package	✓ 100 xp
Selective import	✓ 100 xp
Different ways of importing	✓ 50 xp

INTRO TO PYTHON FOR DATA SCIENCE

Functions

Functions

- Nothing new!
- `type()`
- Piece of reusable code
- Solves particular task
- Call function instead of writing code yourself

Example

```
In [1]: fam = [1.73, 1.68, 1.71, 1.89]

In [2]: fam
Out[2]: [1.73, 1.68, 1.71, 1.89]

In [3]: max(fam)
Out[3]: 1.89
```

Example

```
In [1]: fam = [1.73, 1.68, 1.71, 1.89]

In [2]: fam
Out[2]: [1.73, 1.68, 1.71, 1.89]

In [3]: max(fam)
Out[3]: 1.89

In [4]: tallest = max(fam)

In [5]: tallest
Out[5]: 1.89
```

round()

```
In [6]: round(1.68, 1)
Out[6]: 1.7

In [7]: round(1.68)
Out[7]: 2

In [8]: help(round)
```

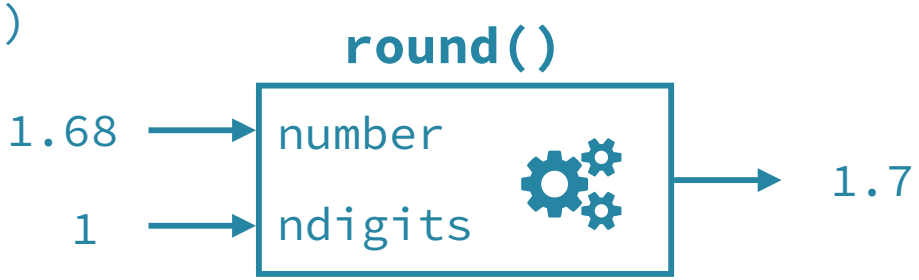
Open up documentation

```
  Help on built-in function round in module builtins:

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

round()

```
In [8]: help(round)

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

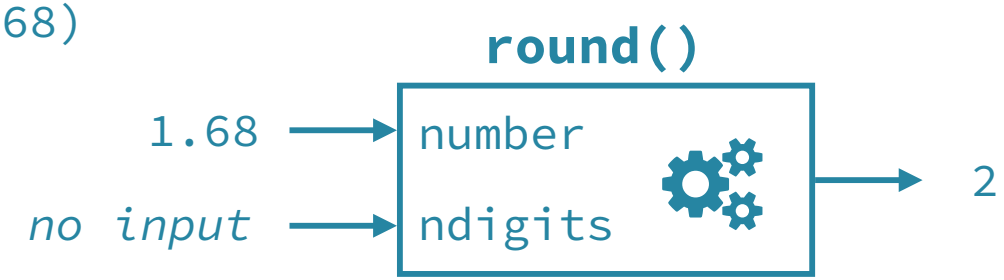
```
round(1.68, 1)
```



round()

```
In [8]: help(round)

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

round(1.68)



round()

```
In [8]: help(round)

  round(...)
      round(number[, ndigits]) -> number

      Round a number to a given precision in decimal digits
      (default 0 digits). This returns an int when called with
      one argument, otherwise the same type as the number.
      ndigits may be negative.
```

```
round(number)

round(number, ndigits)
```

Find functions

- How to know?
- Standard task -> probably function exists!
- The internet is your friend

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

Methods

Built-in Functions

- Maximum of list: `max()`
- Length of list or string: `len()`
- Get index in list: ?
- Reversing a list: ?

Back 2 Basics

		type	examples of methods
`In [1]: sister = "liz"`	Object	`str`	`capitalize()` `replace()`
`In [2]: height = 1.73`	Object	`float`	`bit_length()` `conjugate()`
`In [3]: fam = ["liz", 1.73, "emma", 1.68, "mom", 1.71, "dad", 1.89]`	Object	`list`	`index()` `count()`

Methods: Functions that belong to objects

list methods

```
In [4]: fam
Out[4]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

In [5]: fam.index("mom")
Out[5]: 4

In [6]: fam.count(1.73)
Out[6]: 1
```

"Call method index() on fam"

str methods

```
In [7]: sister
Out[7]: 'liz'

In [8]: sister.capitalize()
Out[8]: 'Liz'

In [9]: sister.replace("z", "sa")
Out[9]: 'lisa'
```

Methods

- Everything = object
- Object have methods associated, depending on type

```
In [10]: sister.replace("z", "sa")
Out[10]: 'lisa'

In [11]: fam.replace("mom", "mommy")
AttributeError: 'list' object has no attribute 'replace'

In [12]: sister.index("z")
Out[12]: 2

In [13]: fam.index("mom")
Out[13]: 4
```

Methods (2)

```
In [14]: fam
Out[14]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89]

In [15]: fam.append("me")

In [16]: fam
Out[16]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89, 'me']

In [17]: fam.append(1.79)

In [18]: fam
Out[18]: ['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89, 'me', 1.79]
```

Summary

- Functions

```
In [11]: type(fam)
Out[11]: list
```

- Methods: call functions *on* objects

```
In [12]: fam.index("dad")
Out[12]: 6
```

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

Packages

Motivation

- Functions and methods are powerful
- All code in Python distribution?
  - Huge code base: messy
  - Lots of code you won't use
  - Maintenance problem

Packages

- Directory of Python Scripts
- Each script = module
- Specify functions, methods, types
- Thousands of packages available
  - Numpy
  - Matplotlib
  - Scikit-learn

```
pkg/
  mod1.py
  mod2.py
  ...
```

Install package

- http://pip.readthedocs.org/en/stable/installing/
- Download `get-pip.py`
- Terminal:
  - `python3 get-pip.py`
  - `pip3 install numpy`

Import package

```
In [1]: import numpy

In [2]: array([1, 2, 3])
NameError: name 'array' is not defined

In [3]: numpy.array([1, 2, 3])
Out[3]: array([1, 2, 3])

In [4]: import numpy as np

In [5]: np.array([1, 2, 3])
Out[5]: array([1, 2, 3])

In [6]: from numpy import array

In [7]: array([1, 2, 3])
Out[7]: array([1, 2, 3])
```

from numpy import array

my_script.py

```
from numpy import array

fam = ["liz", 1.73, "emma", 1.68,
       "mom", 1.71, "dad", 1.89]

...

fam_ext = fam + ["me", 1.79]

...

print(str(len(fam_ext)) + " elements in fam_ext")

...

np_fam = array(fam_ext)
```

Using Numpy, but not very clear

import numpy

my_script.py

```
import numpy

fam = ["liz", 1.73, "emma", 1.68,
       "mom", 1.71, "dad", 1.89]

...

fam_ext = fam + ["me", 1.79]

...

print(str(len(fam_ext)) + " elements in fam_ext")

...

np_fam = numpy.array(fam_ext)
```

Clearly using Numpy

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

4 NumPy

100%

NumPy is a Python package to efficiently do data science. Learn to work with the NumPy array, a faster and more powerful alternative to the list, and take your first steps in data exploration.

NumPy	✓ 50 xp
Your First NumPy Array	✓ 100 xp
Baseball players' height	✓ 100 xp
Baseball player's BMI	✓ 100 xp
Lightweight baseball players	✓ 100 xp
NumPy Side Effects	✓ 50 xp
Subsetting NumPy Arrays	✓ 100 xp
2D NumPy Arrays	✓ 50 xp
Your First 2D NumPy Array	✓ 100 xp
Baseball data in 2D form	✓ 100 xp
Subsetting 2D NumPy Arrays	✓ 100 xp
2D Arithmetic	✓ 100 xp
NumPy: Basic Statistics	✓ 50 xp
Average versus median	✓ 100 xp
Explore the baseball data	✓ 0 xp
Blend it all together	✓ 100 xp

INTRO TO PYTHON FOR DATA SCIENCE

NumPy

Lists Recap

- Powerful
- Collection of values
- Hold different types
- Change, add, remove
- Need for Data Science
  - Mathematical operations over collections
  - Speed

Illustration

```
In [1]: height = [1.73, 1.68, 1.71, 1.89, 1.79]

In [2]: height
Out[2]: [1.73, 1.68, 1.71, 1.89, 1.79]

In [3]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]

In [4]: weight
Out[4]: [65.4, 59.2, 63.6, 88.4, 68.7]

In [5]: weight / height ** 2
TypeError: unsupported operand type(s) for **: 'list' and 'int'
```

Solution: NumPy

- Numeric Python
- Alternative to Python List: NumPy Array
- Calculations over entire arrays
- Easy and Fast
- Installation
  - In the terminal: `pip3 install numpy`

NumPy

```
In [6]: import numpy as np

In [7]: np_height = np.array(height)

In [8]: np_height
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [9]: np_weight = np.array(weight)

In [10]: np_weight
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])

In [11]: bmi = np_weight / np_height ** 2

In [12]: bmi
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

NumPy

Element-wise calculations

```
In [6]: import numpy as np

In [7]: np_height = np.array(height)

In [8]: np_height
Out[8]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [9]: np_weight = np.array(weight)

In [10]: np_weight
Out[10]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])

In [11]: bmi = np_weight / np_height ** 2

In [12]: bmi
Out[12]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

```
= 65.5/1.73 ** 2
```

Comparison

```
In [13]: height = [1.73, 1.68, 1.71, 1.89, 1.79]

In [14]: weight = [65.4, 59.2, 63.6, 88.4, 68.7]

In [15]: weight / height ** 2
TypeError: unsupported operand type(s) for **: 'list' and 'int'



In [16]: np_height = np.array(height)

In [17]: np_weight = np.array(weight)

In [18]: np_weight / np_height ** 2
Out[18]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])
```

NumPy: remarks

```
In [19]: np.array([1.0, "is", True])
Out[19]:
array(['1.0', 'is', 'True'],
      dtype='<U32')

In [20]: python_list = [1, 2, 3]

In [21]: numpy_array = np.array([1, 2, 3])

In [22]: python_list + python_list
Out[22]: [1, 2, 3, 1, 2, 3]

In [23]: numpy_array + numpy_array
Out[23]: array([2, 4, 6])
```

NumPy arrays: contain only one type

Different types: different behavior!

NumPy Subsetting

```
In [24]: bmi
Out[24]: array([ 21.852,  20.975,  21.75 ,  24.747,  21.441])

In [25]: bmi[1]
Out[25]: 20.975

In [26]: bmi > 23
Out[26]: array([False, False, False,  True, False], dtype=bool)

In [27]: bmi[bmi > 23]
Out[27]: array([ 24.747])
```

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

2D NumPy Arrays

Type of NumPy Arrays

```
In [1]: import numpy as np

In [2]: np_height = np.array([1.73, 1.68, 1.71, 1.89, 1.79])

In [3]: np_weight = np.array([65.4, 59.2, 63.6, 88.4, 68.7])

In [4]: type(np_height)
Out[4]: numpy.ndarray

In [5]: type(np_weight)
Out[5]: numpy.ndarray
```

ndarray = N-dimensional array

2D NumPy Arrays

```
In [6]: np_2d = np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
                          [65.4, 59.2, 63.6, 88.4, 68.7]])

In [7]: np_2d
Out[7]:
array([[  1.73,   1.68,   1.71,   1.89,   1.79],
       [ 65.4 ,  59.2 ,  63.6 ,  88.4 ,  68.7 ]])

In [8]: np_2d.shape
Out[8]: (2, 5)

In [9]: np.array([[1.73, 1.68, 1.71, 1.89, 1.79],
                  [65.4, 59.2, 63.6, 88.4, "68.7"]])
Out[9]:
array([['1.73', '1.68', '1.71', '1.89', '1.79'],
       ['65.4', '59.2', '63.6', '88.4', '68.7']],
      dtype='<U32')
```

2 rows, 5 columns

Single type!

Subsetting

```
                 0       1       2       3       4
array([[  1.73,   1.68,   1.71,   1.89,   1.79],   0
       [  65.4,   59.2,   63.6,   88.4,   68.7]])  1
```

```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71
```

Subsetting

```
                 0       1       2       3       4
array([[  1.73,   1.68,   1.71,   1.89,   1.79],   0
       [  65.4,   59.2,   63.6,   88.4,   68.7]])  1
```

```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71

In [13]: np_2d[:,1:3]
Out[13]:
array([[  1.68,   1.71],
       [ 59.2 ,  63.6 ]])
```

Subsetting

```
                0     1     2     3     4
array([[  1.73,   1.68,   1.71,   1.89,   1.79],   0
       [  65.4,   59.2,   63.6,   88.4,   68.7]])  1
```

```
In [10]: np_2d[0]
Out[10]: array([ 1.73,  1.68,  1.71,  1.89,  1.79])

In [11]: np_2d[0][2]
Out[11]: 1.71

In [12]: np_2d[0,2]
Out[12]: 1.71

In [13]: np_2d[:,1:3]
Out[13]:
array([[  1.68,   1.71],
       [ 59.2 ,  63.6 ]])

In [14]: np_2d[1,:]
Out[14]: array([ 65.4,  59.2,  63.6,  88.4,  68.7])
```

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!

INTRO TO PYTHON FOR DATA SCIENCE

NumPy: Basic Statistics

Data analysis

- Get to know your data
- Little data -> simply look at it
- Big data -> ?

City-wide survey

```
In [1]: import numpy as np

In [2]: np_city = ... # Implementation left out

In [3]: np_city
Out[3]:
array([[  1.64,  71.78],
       [  1.37,  63.35],
       [  1.6 ,  55.09],
       ...,
       [  2.04,  74.85],
       [  2.04,  68.72],
       [  2.01,  73.57]])
```

NumPy

```
In [4]: np.mean(np_city[:,0])
Out[4]: 1.7472

In [5]: np.median(np_city[:,0])
Out[5]: 1.75

In [6]: np.corrcoef(np_city[:,0], np_city[:,1])
Out[6]:
array([[ 1.     , -0.01802],
       [-0.01803,  1.     ]])

In [7]: np.std(np_city[:,0])
Out[7]: 0.1992
```

- `sum()`, `sort()`, ...
- Enforce single data type: speed!

Generate data


```
In [8]: height = np.round(np.random.normal(1.75, 0.20, 5000), 2)

In [9]: weight = np.round(np.random.normal(60.32, 15, 5000), 2)

In [10]: np_city = np.column_stack((height, weight))
```

INTRO TO PYTHON FOR DATA SCIENCE

Let's practice!