



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)
```



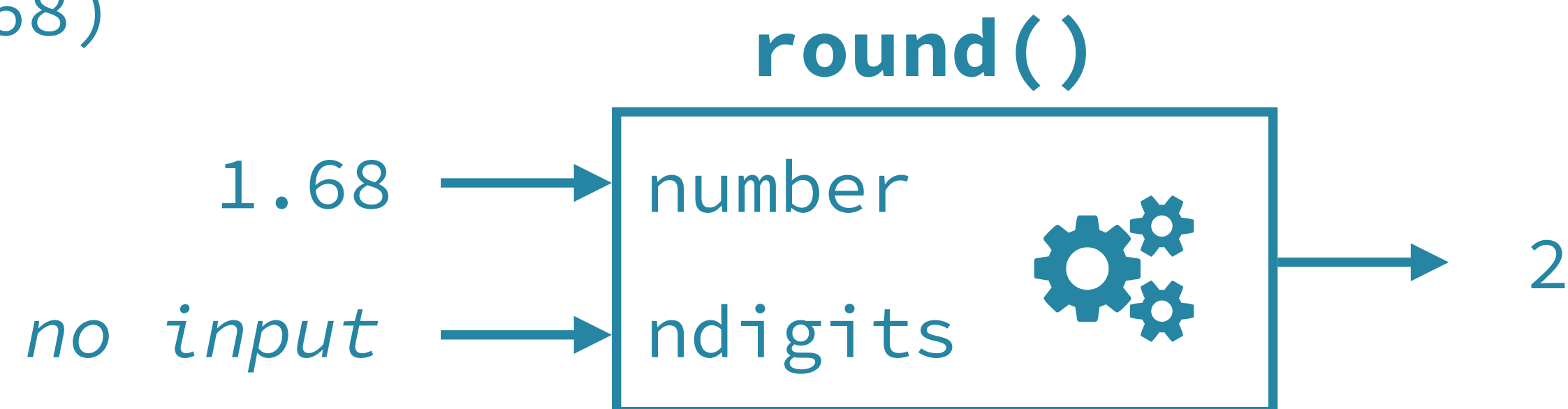
round()

```
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
```

"Call method `index()` on `fam`"

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

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!