# Functions

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

# Example

```
fam = [1.73, 1.68, 1.71, 1.89]
fam
```

```
[1.73, 1.68, 1.71, 1.89]
```

```
max(fam)
```

```
1.89
```

# Example

```
fam = [1.73, 1.68, 1.71, 1.89]
fam
```

```
[1.73, 1.68, 1.71, 1.89]
```

```
max(fam)
```

```
1.89
```

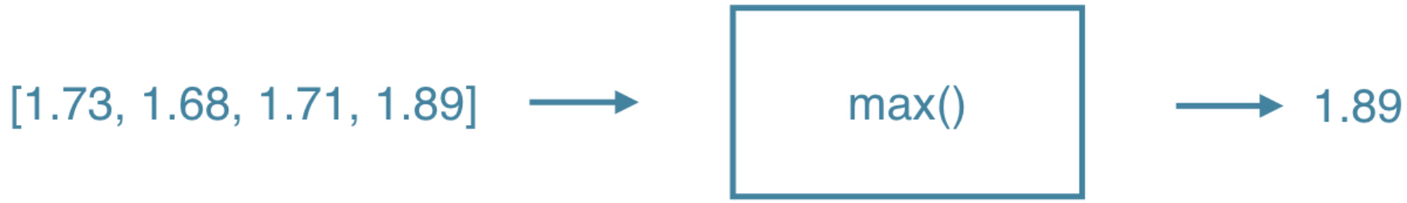
# Example

```
fam = [1.73, 1.68, 1.71, 1.89]
fam
```

```
[1.73, 1.68, 1.71, 1.89]
```

```
max(fam)
```

```
1.89
```



# Example

```
fam = [1.73, 1.68, 1.71, 1.89]
fam
```

```
[1.73, 1.68, 1.71, 1.89]
```

```
max(fam)
```

```
1.89
```

```
tallest = max(fam)
tallest
```

```
1.89
```

# round()

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

```
1.7
```

```
round(1.68)
```

```
2
```

```
help(round) # Open up documentation
```

```
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()

```
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()

```
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.
```

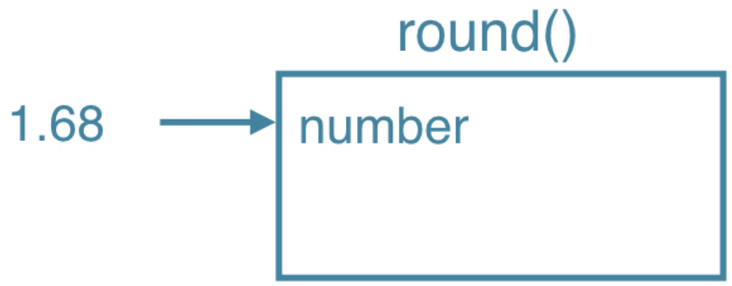
round(1.68, 1)

# round()

```
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)

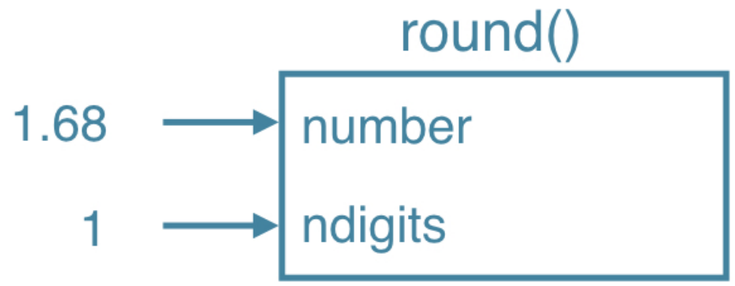


# round()

```
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.
```

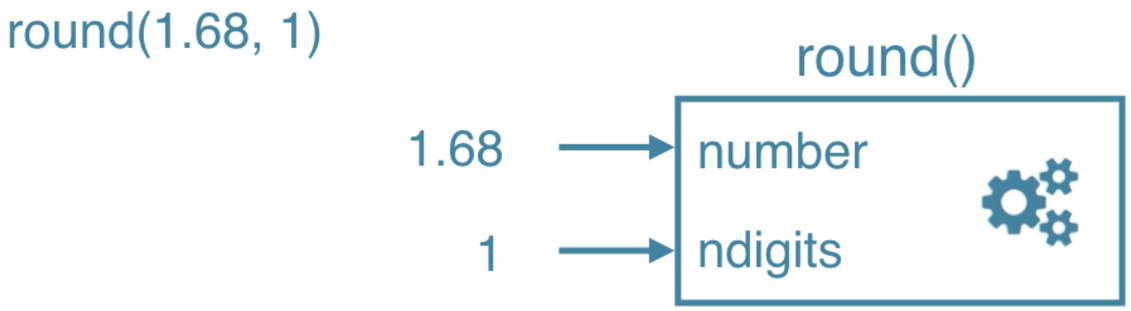
round(1.68, 1)



# round()

```
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()

```
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()

```
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()

```
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()

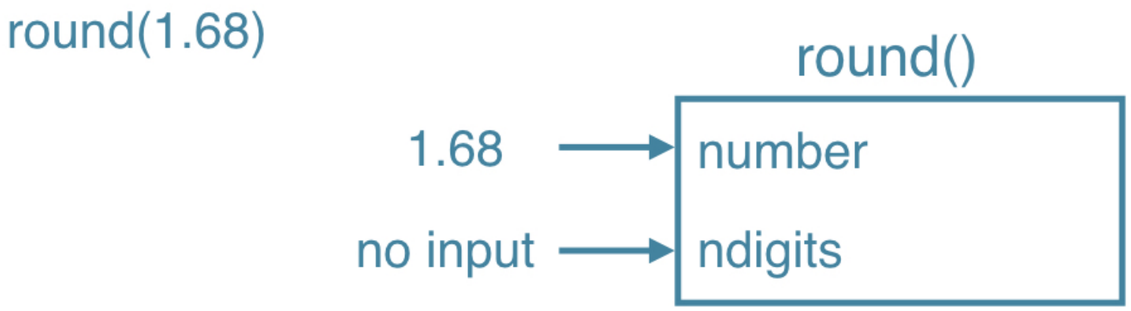
```
help(round)
```

```
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()

```
help(round)
```

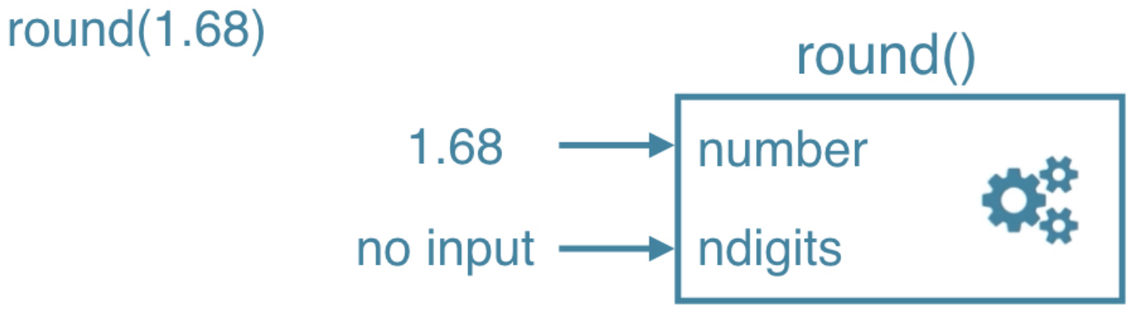
```
round(...)
    round(number[, ndigits]) -> number

    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()

```
help(round)
```

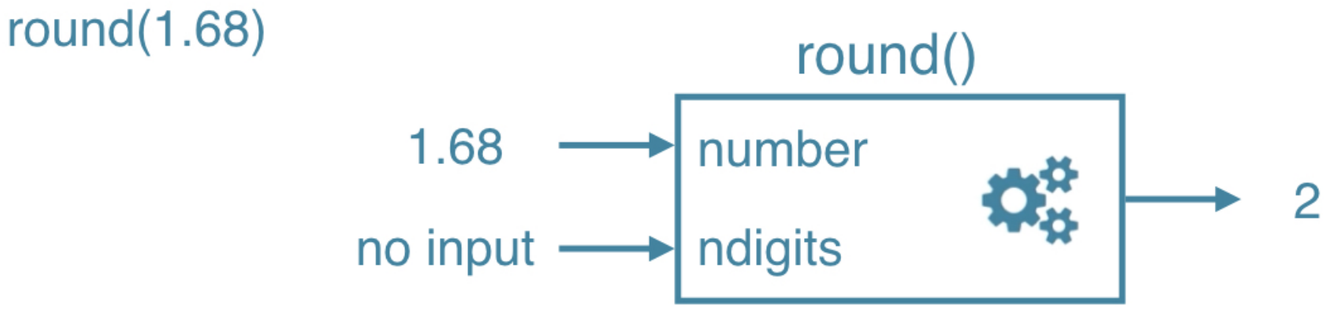
```
round(...)
    round(number[, ndigits]) -> number

    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()

```
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()

```
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

# Let's practice!

INTRODUCTION TO PYTHON

# Built-in Functions

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

# Back 2 Basics

```
sister = "liz"
```

Object

```
height = 1.73
```

Object

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

Object

# Back 2 Basics

```
sister = "liz"
```

Object str

```
height = 1.73
```

Object float

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

Object list

type

- Methods: Functions that belong to objects

# Back 2 Basics

```
sister = "liz"
```

```
height = 1.73
```

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

| | type | examples of methods |
|---|---|---|
| Object | str | capitalize()<br>replace() |
| Object | float | bit_length()<br>conjugate() |
| Object | list | index()<br>count() |

- Methods: Functions that belong to objects

# list methods

```
fam
```

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

```
fam.index("mom") # "Call method index() on fam"
```

```
4
```

```
fam.count(1.73)
```

```
1
```

# str methods

```
sister
```

```
'liz'
```

```
sister.capitalize()
```

```
'Liz'
```

```
sister.replace("z", "sa")
```

```
'lisa'
```

# Methods

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

```
sister.replace("z", "sa")
```

```
'lisa'
```

```
fam.replace("mom", "mommy")
```

```
AttributeError: 'list' object has no attribute 'replace'
```

# Methods

```
sister.index("z")
```

```
2
```

```
fam.index("mom")
```

```
4
```

# Methods (2)

```
fam
```

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

```
fam.append("me")
fam
```

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

```
fam.append(1.79)
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.89, 'me', 1.79]
```

# Summary

## Functions

```
type(fam)
```

```
list
```

## Methods: call functions on objects

```
fam.index("dad")
```

```
6
```

# Let's practice!

INTRODUCTION TO PYTHON

# 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

```
import numpy
array([1, 2, 3])
```

```
NameError: name 'array' is not define
```

```
numpy.array([1, 2, 3])
```

```
array([1, 2, 3])
```

```
import numpy as np
np.array([1, 2, 3])
```

```
array([1, 2, 3])
```

```
from numpy import array
array([1, 2, 3])
```

```
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

```
import numpy as np

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 = np.array(fam_ext) # Clearly using Numpy
```

# Let's practice!

INTRODUCTION TO PYTHON