Variables and Types

INTRODUCTION TO PYTHON

Variable

- Specific, case-sensitive name
- Call up value through variable name

Variable

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

Variable

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

```
height = 1.79
weight = 68.7
```

Variable

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

```
height = 1.79
weight = 68.7

height
```

```
1.79
```

Calculate BMI

```
height = 1.79
weight = 68.7
height
```

```
1.79
```

Calculate BMI

```
height = 1.79
weight = 68.7
height
```

```
1.79
```

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

Calculate BMI

```
height = 1.79
weight = 68.7
height
```

```
1.79
```

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

```
68.7 / 1.79 ** 2
```

```
21.4413
```

Calculate BMI

```
height = 1.79
weight = 68.7
height
```

```
1.79
```

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

```
68.7 / 1.79 ** 2
```

```
21.4413
```

```
weight / height ** 2
```

```
21.4413
```

Calculate BMI

```
height = 1.79
weight = 68.7
height
```

```
1.79
```

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

```
68.7 / 1.79 ** 2
```

```
21.4413
```

```
weight / height ** 2
```

```
21.4413
```

```
bmi = weight / height ** 2
bmi
```

```
21.4413
```

Reproducibility

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

```
21.4413
```

Reproducibility

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

```
23.1578
```

Python Types

Python Types

```
type(bmi)
```

```
float
```

Python Types

```
type(bmi)
```

```
float
```

```
day_of_week = 5
type(day_of_week)
```

```
int
```

Python Types (2)

Python Types (2)

```
x = "body mass index"
y = 'this works too'
```

Python Types (2)

```
x = "body mass index"
y = 'this works too'

type(y)
```

```
str
```

Python Types (2)

```
x = "body mass index"
y = 'this works too'
type(y)
```

```
str
```

```
z = True
type(z)
```

```
bool
```

Python Types (3)

Python Types (3)

```
2 + 3
```

```
5
```

Python Types (3)

```
2 + 3
```

```
5
```

```
'ab' + 'cd'
```

```
'abcd'
```

Python Types (3)

```
2 + 3
```

```
5
```

```
'ab' + 'cd'
```

```
'abcd'
```

- Different type = different behavior!

Python Lists

INTRODUCTION TO PYTHON

Python Data Types

Python Data Types

- float - real numbers

Python Data Types

- float - real numbers
- int - integer numbers

Python Data Types

- float - real numbers
- int - integer numbers
- str - string, text

Python Data Types

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

Python Data Types

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

```
height = 1.73
tall = True
```

Python Data Types

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

```
height = 1.73
tall = True
```

  - Each variable represents single value

Problem

Problem

- Data Science: many data points

Problem

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

Problem

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

```
height1 = 1.73
height2 = 1.68
height3 = 1.71
height4 = 1.89
```

Problem

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

```
height1 = 1.73
height2 = 1.68
height3 = 1.71
height4 = 1.89
```

    - Inconvenient

Python List

- `[a, b, c]`

Python List

- `[a, b, c]`

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

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

Python List

- `[a, b, c]`

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

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

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

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

Python List

- `[a, b, c]`

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

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

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

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

- Name a collection of values

Python List

- `[a, b, c]`

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

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

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

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

- Name a collection of values
- Contain any type

Python List

- `[a, b, c]`

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

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

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

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

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

Python List

- `[a, b, c]`

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

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

Python List

- `[a, b, c]`

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

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

```
fam2 = [["liz", 1.73],
        ["emma", 1.68],
        ["mom", 1.71],
        ["dad", 1.89]]
```

Python List

- `[a, b, c]`

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

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

```
fam2 = [["liz", 1.73],
        ["emma", 1.68],
        ["mom", 1.71],
        ["dad", 1.89]]
fam2
```

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

List type

```
type(fam)
```

```
list
```

```
type(fam2)
```

```
list
```

List type

```
type(fam)
```

```
list
```

```
type(fam2)
```

```
list
```

- Specific functionality
- Specific behavior

Subsetting Lists

INTRODUCTION TO PYTHON

Subsetting lists

Subsetting lists

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

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

Subsetting lists

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

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

```
fam[3]
```

```
1.68
```

Subsetting lists

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

Subsetting lists

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

```
fam[6]
```

```
'dad'
```

Subsetting lists

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

```
fam[6]
```

```
'dad'
```

```
fam[-1]
```

```
1.89
```

Subsetting lists

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

```
fam[6]
```

```
'dad'
```

```
fam[-1]  # <-
```

```
1.89
```

```
fam[7] # <-
```

```
1.89
```

List slicing

```
fam
```

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

List slicing

```
fam
```

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

```
fam[3:5]
```

List slicing

```
fam
```

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

```
fam[3:5]
```

```
[1.68, 'mom']
```

List slicing

```
fam
```

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

```
fam[3:5]
```

```
[1.68, 'mom']
```

List slicing

```
fam
```

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

```
fam[3:5]
```

```
[1.68, 'mom']
```

```
fam[1:4]
```

List slicing

```
fam
```

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

```
fam[3:5]
```

```
[1.68, 'mom']
```

```
fam[1:4]
```

```
[1.73, 'emma', 1.68]
```

List slicing

```
fam
```

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

List slicing

```
fam
```

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

```
fam[:4]
```

```
['liz', 1.73, 'emma', 1.68]
```

List slicing

```
fam
```

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

```
fam[:4]
```

```
['liz', 1.73, 'emma', 1.68]
```

```
fam[5:]
```

```
[1.71, 'dad', 1.89]
```

Manipulating Lists

INTRODUCTION TO PYTHON

List Manipulation

List Manipulation

- Change list elements

List Manipulation

- Change list elements
- Add list elements

List Manipulation

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

Changing list elements

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

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

Changing list elements

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

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

```
fam[7] = 1.86
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

Changing list elements

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

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

```
fam[7] = 1.86
fam
```

```
['liz', 1.73, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

```
fam[0:2] = ["lisa", 1.74]
fam
```

```
['lisa', 1.74, 'emma', 1.68, 'mom', 1.71, 'dad', 1.86]
```

Adding and removing elements

Adding and removing elements

```
fam + ["me", 1.79]
```

```
['lisa', 1.74,'emma', 1.68, 'mom', 1.71, 'dad', 1.86, 'me', 1.79]
```

Adding and removing elements

```
fam + ["me", 1.79]
```

```
['lisa', 1.74,'emma', 1.68, 'mom', 1.71, 'dad', 1.86, 'me', 1.79]
```

```
fam_ext = fam + ["me", 1.79]
```

Adding and removing elements

```
fam + ["me", 1.79]
```

```
['lisa', 1.74,'emma', 1.68, 'mom', 1.71, 'dad', 1.86, 'me', 1.79]
```

```
fam_ext = fam + ["me", 1.79]
del(fam[2])
```

Adding and removing elements

```
fam + ["me", 1.79]
```

```
['lisa', 1.74,'emma', 1.68, 'mom', 1.71, 'dad', 1.86, 'me', 1.79]
```

```
fam_ext = fam + ["me", 1.79]
del(fam[2])
fam
```

```
['lisa', 1.74, 1.68, 'mom', 1.71, 'dad', 1.86]
```

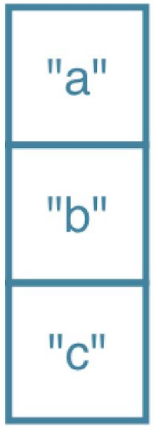
Behind the scenes (1)

Behind the scenes (1)

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

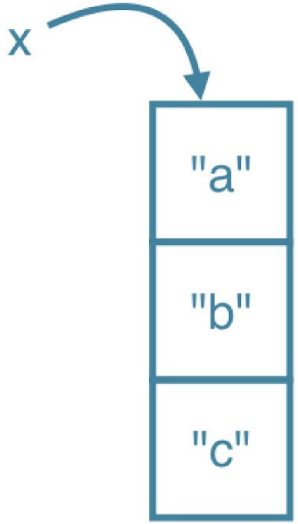
Behind the scenes (1)

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



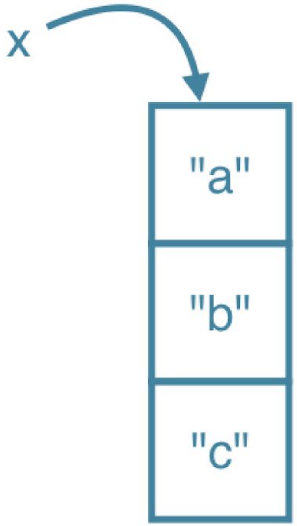
Behind the scenes (1)

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



Behind the scenes (1)

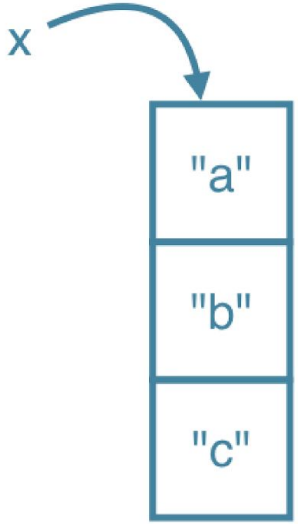
```
x = ["a", "b", "c"]

y = x
```



Behind the scenes (1)

```
x = ["a", "b", "c"]
y = x
y[1] = "z"
y
```

```
['a', 'z', 'c']
```



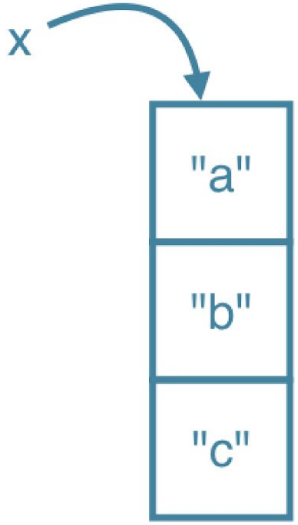
Behind the scenes (1)

```
x = ["a", "b", "c"]
y = x
y[1] = "z"
y
```

```
['a', 'z', 'c']
```

```
x
```

```
['a', 'z', 'c']
```



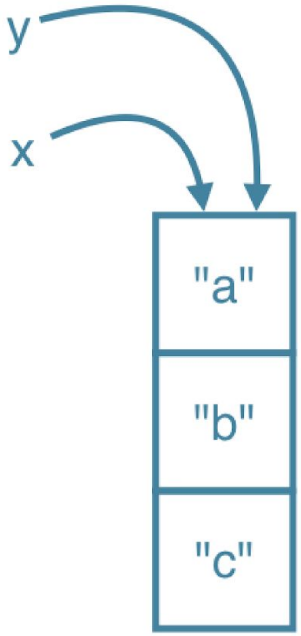
Behind the scenes (1)

```
x = ["a", "b", "c"]
y = x
y[1] = "z"
y
```

```
['a', 'z', 'c']
```

```
x
```

```
['a', 'z', 'c']
```



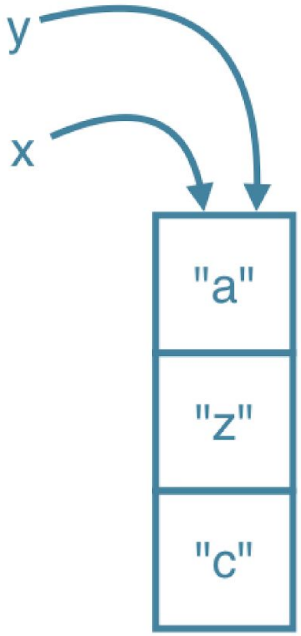
Behind the scenes (1)

```
x = ["a", "b", "c"]
y = x
y[1] = "z"
y
```

```
['a', 'z', 'c']
```

```
x
```

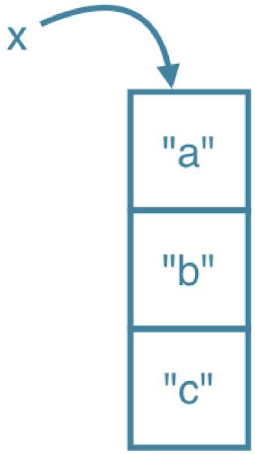
```
['a', 'z', 'c']
```



© DECODR. All right reserved.

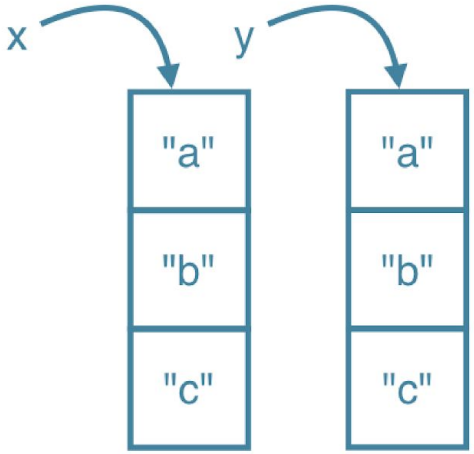
Behind the scenes (2)

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



Behind the scenes (2)

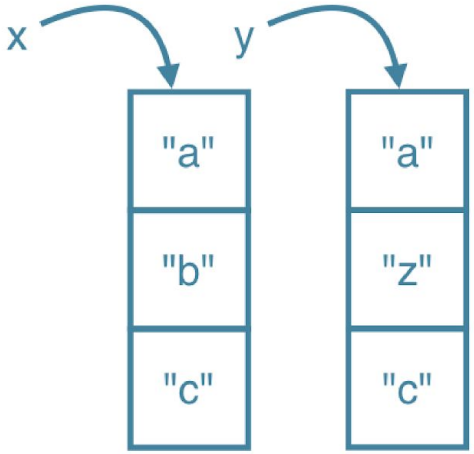
```
x = ["a", "b", "c"]
y = list(x)
y = x[:]
```



Behind the scenes (2)

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

y = list(x)
y = x[:]

y[1] = "z"
x
```

```
['a', 'b', 'c']
```



Functions

INTRODUCTION TO PYTHON

Functions

- Nothing new!

Functions

- Nothing new!
- type()

Functions

- Nothing new!
- type()
- Piece of reusable code

Functions

- Nothing new!
- type()
- Piece of reusable code
- Solves particular task

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]
```

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
```

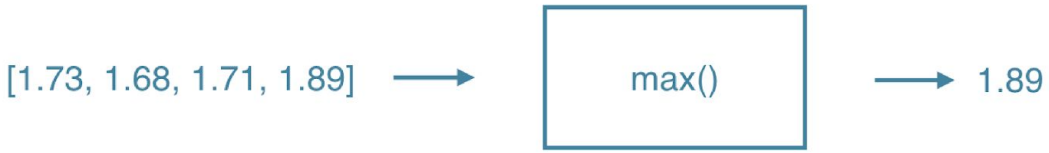
Example

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

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

```
max(fam)
```

```
1.89
```



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()

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

```
1.7
```

round()

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

```
1.7
```

```
round(1.68)
```

```
2
```

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()

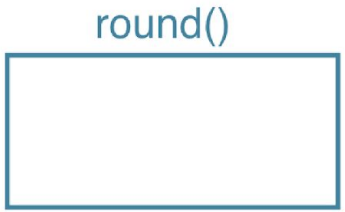
```
help(round)
```

```
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.
```

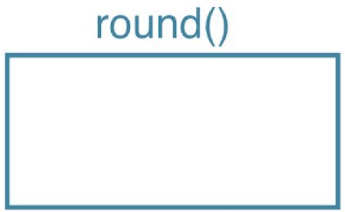


round()

```
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)
```

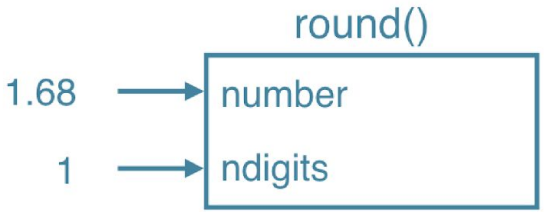
```
round(...)
    round(number[, ndigits]) -> number

    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(1.68, 1)

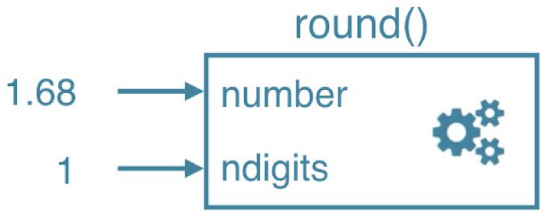


round()

```
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)

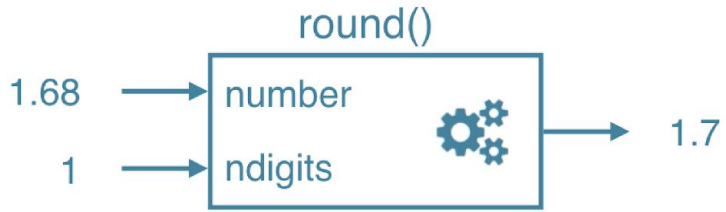


round()

```
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.
```

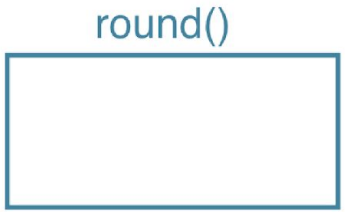
round(1.68, 1)



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.
```

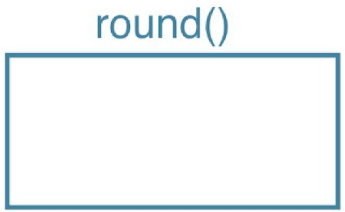


round()

```
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(1.68)



round()

```
help(round)
```

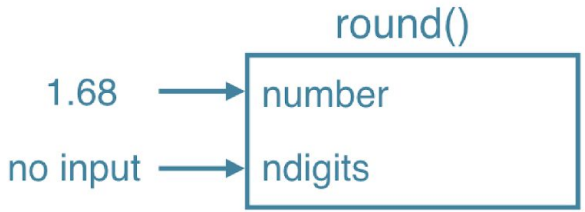
```
round(...)
    round(number[, ndigits]) -> number

    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.
```

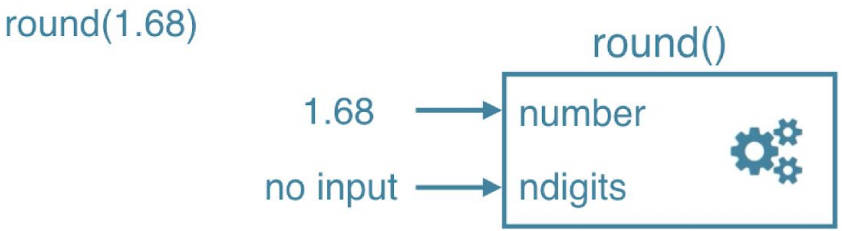
round(1.68)



round()

```
help(round)
```

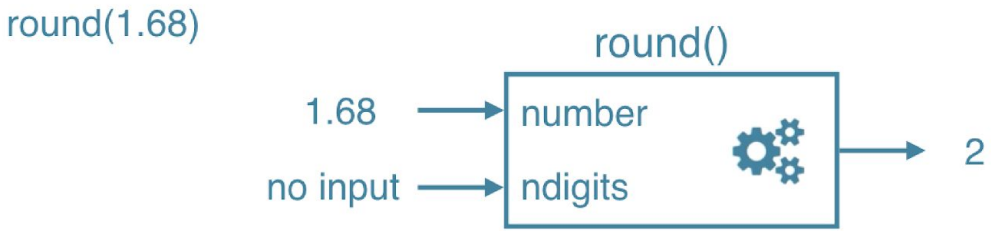
```
round(...)
    round(number[, ndigits]) -> number

    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(number)`
- `round(number, ndigits)`

Find functions

Find functions

- How to know?

Find functions

- How to know?
- Standard task -> probably function exists!

Find functions

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

Methods

INTRODUCTION TO PYTHON

Built-in Functions

Built-in Functions

- Maximum of list: max()

Built-in Functions

- Maximum of list: max()
- Length of list or string: len()

Built-in Functions

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

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"
```

Back 2 Basics

```
sister = "liz"
```

```
height = 1.73
```

Back 2 Basics

```
sister = "liz"
```

```
height = 1.73
```

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

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"
```

```
height = 1.73
```

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

	type
Object	str
Object	float
Object	list

Back 2 Basics

```
sister = "liz"
```

```
height = 1.73
```

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

	type
Object	str
Object	float
Object	list

- 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() replace()
Object	float	bit_length() conjugate()
Object	list	index() count()

- Methods: Functions that belong to objects

list methods

```
fam
```

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

list methods

```
fam
```

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

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

list methods

```
fam
```

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

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

```
4
```

list methods

```
fam
```

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

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

```
4
```

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)
```

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

str methods

```
sister
```

```
'liz'
```

str methods

```
sister
```

```
'liz'
```

```
sister.capitalize()
```

str methods

```
sister
```

```
'liz'
```

```
sister.capitalize()
```

```
'Liz'
```

str methods

```
sister
```

```
'liz'
```

```
sister.capitalize()
```

```
'Liz'
```

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

str methods

```
sister
```

```
'liz'
```

```
sister.capitalize()
```

```
'Liz'
```

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

```
'lisa'
```

Methods

- Everything = object

Methods

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

Methods

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

```
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

Methods

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

```
2
```

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

```
4
```

Methods (2)

Methods (2)

```
fam
```

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

Methods (2)

```
fam
```

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

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

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']
```

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

Summary

Functions

Summary

Functions

```
type(fam)
```

```
list
```

Summary

Functions

```
type(fam)
```

```
list
```

Methods: call functions on objects

Summary

Functions

```
type(fam)
```

```
list
```

Methods: call functions on objects

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

```
6
```

Packages

INTRODUCTION TO PYTHON

Motivation

- Functions and methods are powerful

Motivation

- Functions and methods are powerful
- All code in Python distribution?
  - Huge code base: messy

Motivation

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

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

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

Packages

- Directory of Python Scripts

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

Packages

- Directory of Python Scripts
- Each script = module

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

Packages

- Directory of Python Scripts
- Each script = module
- Specify functions, methods, types

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

Packages

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

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

Packages

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

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

Packages

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

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

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

Install package

- http://pip.readthedocs.org/en/stable/installing/

Install package

- http://pip.readthedocs.org/en/stable/installing/
- Download `get-pip.py`

Install package

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

Install package

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

```
import numpy
```

Import package

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

Import package

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

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

Import package

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

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

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

Import package

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

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

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

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

Import package

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

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

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

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

```
import numpy as np
```

Import package

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

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

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

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

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

Import package

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

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

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

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

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

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

Import package

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

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

```
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
```

Import package

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

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

```
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")

...
```

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)
```

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")

...
```

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
```