Functions

INTRODUCTION TO PYTHON

Hoai Thuan TRAN
Gia Dinh University

Functions

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

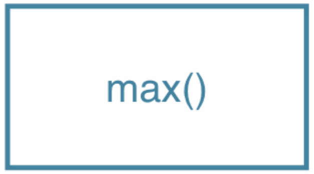
Example

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

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

```python
max(fam)
```

```
1.89
```



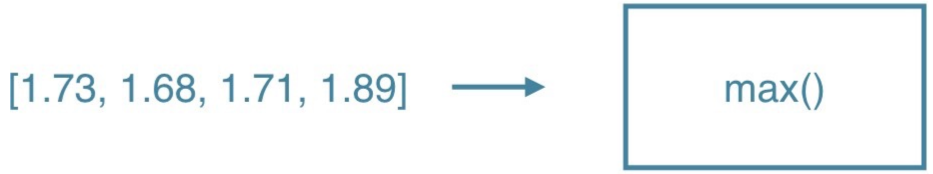
Example

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

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

```
max(fam)
```

```
1.89
```



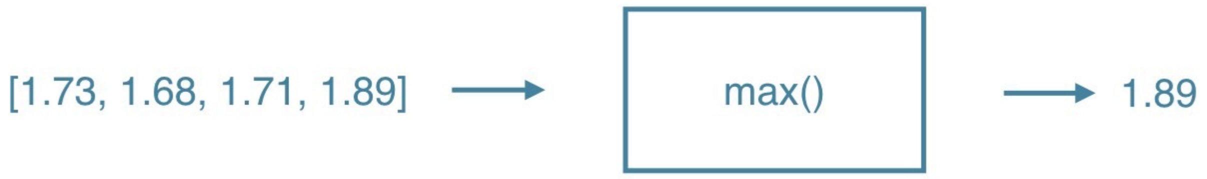
Example

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

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

```python
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
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```

round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```

round( )

```python
help(round)
```

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


round(number, ndigits=None)
    Round a number to a given precision in decimal digits.


    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```

round(1.68, 1)
round()

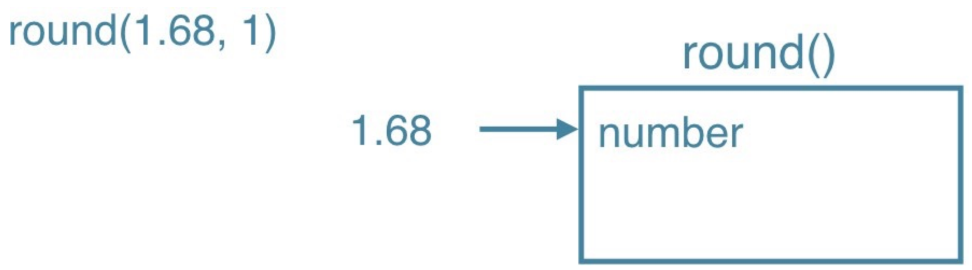
round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



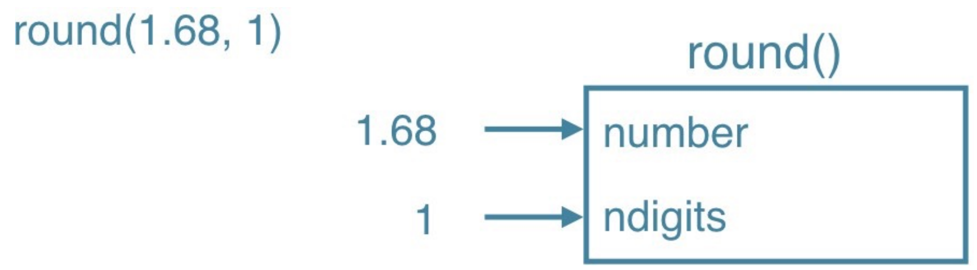
round( )

```python
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



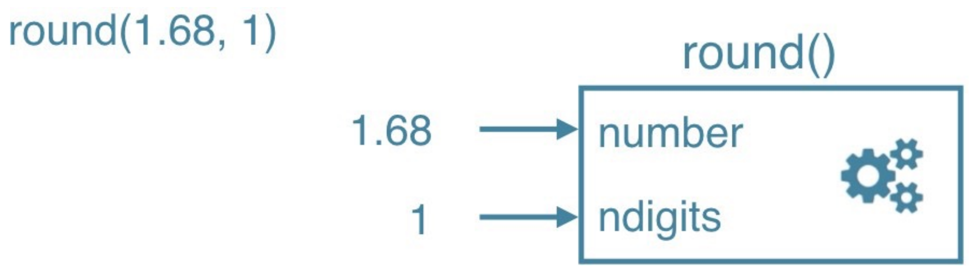
round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



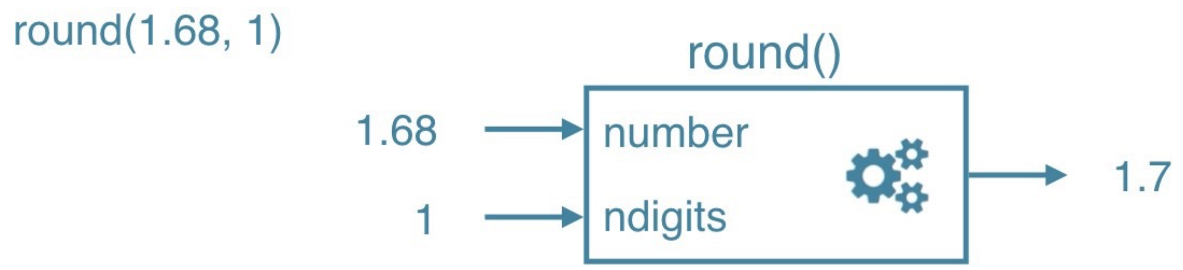
round()

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```

round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```

round(1.68)
round()

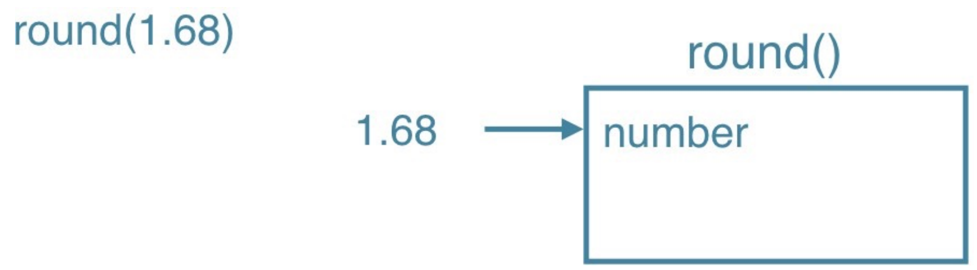
round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



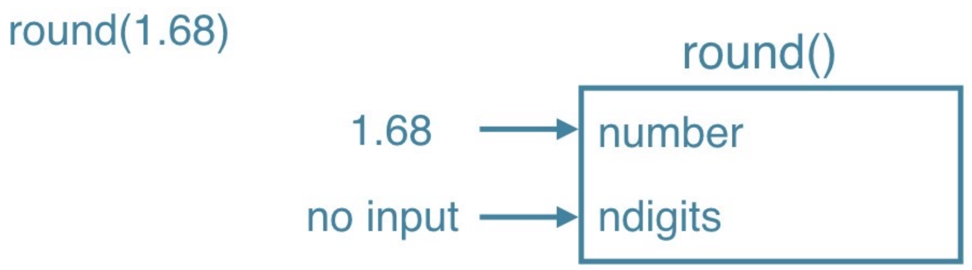
round( )

```python
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



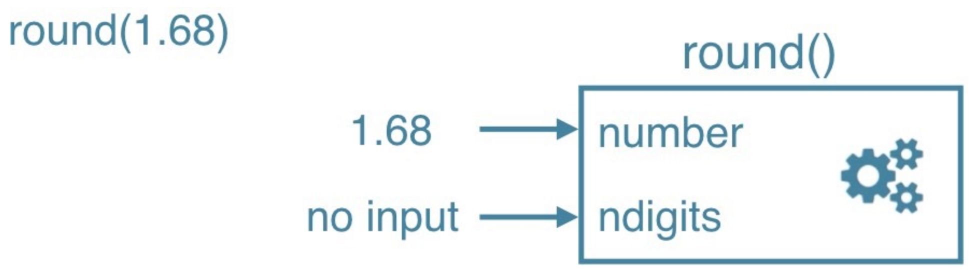
round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



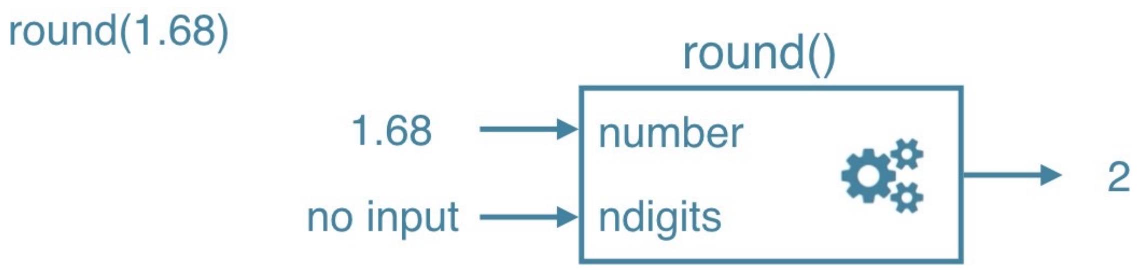
round( )

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has the same type as the number. ndigits may be negative.
```



round()

```
help(round)
```

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

round(number, ndigits=None)
    Round a number to a given precision in decimal digits.

    The return value is an integer if ndigits is omitted or None.
    Otherwise the return value has 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

Familiar functions

You already know two such functions: print() and type(). You've also used the functions str(), int(), bool() and float() to switch between data types. These are built-in functions as well.

Instructions

- Use print() in combination with type() to print out the type of var1.
- Use len() to get the length of the list var1. Wrap it in a print() call to directly print it out.
- Use int() to convert var2 to an integer. Store the output as out2.

```
# Create variables var1 and var2
var1 = [1, 2, 3, 4]
var2 = True

# Print out type of var1


# Print out length of var1


# Convert var2 to an integer: out2
```

Multiple arguments

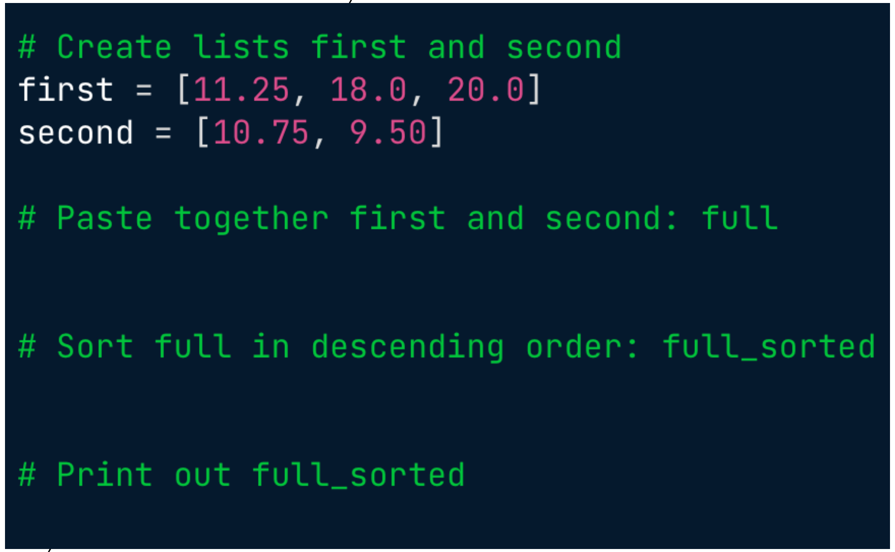
Have a look at the documentation of sorted() by typing help(sorted) in your Python script.

You'll see that sorted() takes three arguments: **iterable**, **key**, and **reverse**.

key = None means that if you don't specify the key argument, it will be None. reverse = False means that if you don't specify the reverse argument, it will be False, by default.

Instructions

- Use + to merge the contents of first and second into a new list: full.
- Call sorted() on full and specify the **reverse** argument to be True. Save the sorted list as full_sorted
- Finish off by printing out full_sorted.



```
# Create lists first and second
first = [11.25, 18.0, 20.0]
second = [10.75, 9.50]

# Paste together first and second: full


# Sort full in descending order: full_sorted


# Print out full_sorted
```

Methods

INTRODUCTION TO PYTHON

Hoai Thuan TRAN
Gia Dinh University

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"
```

```
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]
```

- Methods: Functions that belong to objects

	type	Example of methods

Object	str	capitalize()
		replace()

Object	float	bit_length()
		conjugate()

Object	list	index()
		count()

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

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

```
2
```

```python
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
```

Functions Methods: call functions on objects

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

```
6
```

Let's practice!

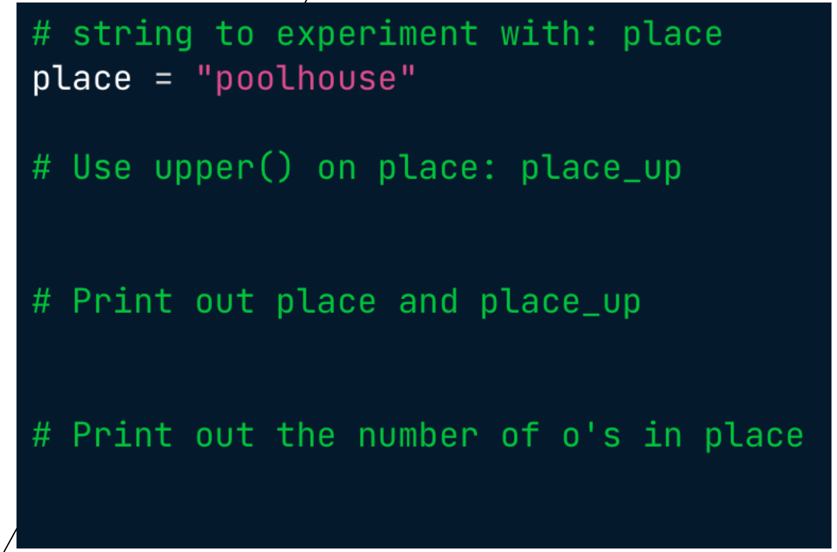
INTRODUCTION TO PYTHON

String methods

Strings come with a bunch of methods. Follow the instructions closely to discover some of them. If you want to discover them in more detail, you can always type help(str) in your Python script.

Instructions

- Use the upper() method on place and store the result in place_up.
- Print out place and place_up. Did both change?
- Print out the number of o's on the variable place by calling count().



```
# string to experiment with: place
place = "poolhouse"

# Use upper() on place: place_up


# Print out place and place_up


# Print out the number of o's in place
```

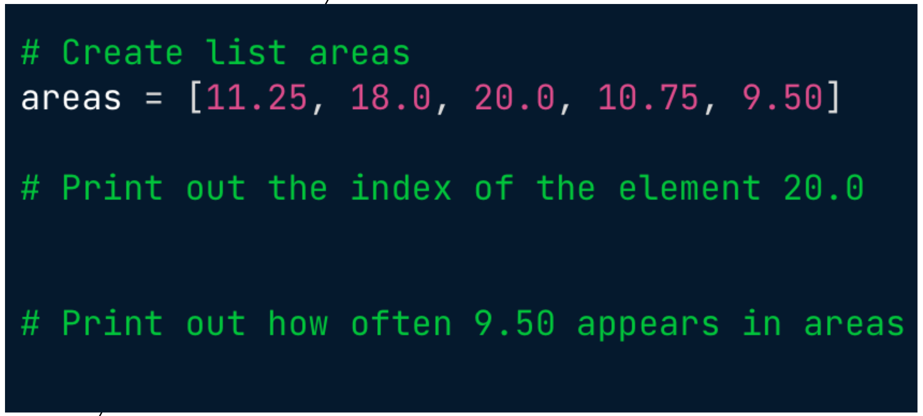
List methods

Lists, floats, integers and booleans are also types that come packaged with a bunch of useful methods. In this exercise, you'll be experimenting with:

- index(), to get the index of the first element of a list that matches its input
- count(), to get the number of times an element appears in a list.

Instructions

- Use the index() method to get the index of the element in areas that is equal to `20.0`. Print out this index.
- Call count() on areas to find out how many times `9.50` appears in the list. Again, simply print out this number.



```python
# Create list areas
areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Print out the index of the element 20.0


# Print out how often 9.50 appears in areas
```

List methods (2)

Most list methods will change the list they're called on.

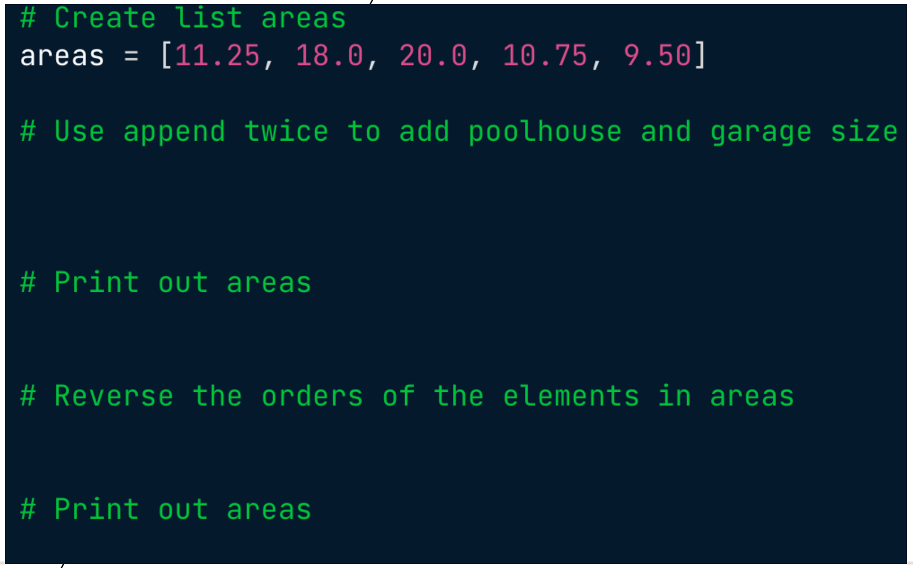
For example,

- append(), that adds an element to the list it is called on,
- remove(), that removes the first element of a list that matches the input.
- reverse(), that reverses the order of the elements in the list it is called on.

Instructions

- Use append() twice to add the size of the poolhouse and the garage again: 24.5 and 15.45, respectively. Make sure to add them in this order. Print out areas.

- Use the reverse() method to reverse the order of the elements in areas. Print out areas once more.



```python
# Create list areas
areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Use append twice to add poolhouse and garage size


# Print out areas


# Reverse the orders of the elements in areas


# Print out areas
```

Packages

INTRODUCTION TO PYTHON

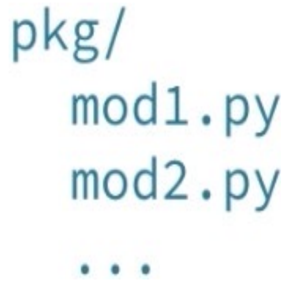
Hoai Thuan TRAN
Gia Dinh University

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 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`

```python
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

Import package

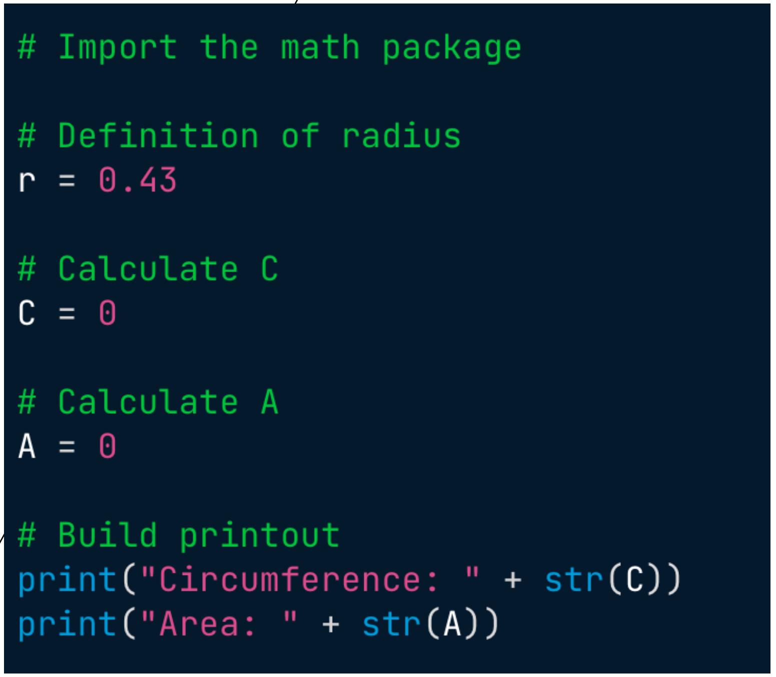
You want to find the circumference, C, and area, A, of a circle. When the radius of the circle is r, you can calculate C and A as:

$$C = 2\pi r \qquad A = \pi r^2$$

To use the constant pi, you'll need the **math** package

Instructions

- Import the **math** package. Now you can access the constant pi with math.pi.
- Calculate the circumference of the circle and store it in C.
- Calculate the area of the circle and store it in A.



```python
# Import the math package

# Definition of radius
r = 0.43

# Calculate C
C = 0

# Calculate A
A = 0

# Build printout
print("Circumference: " + str(C))
print("Area: " + str(A))
```

Selective import

General imports, like import math, make all functionality from the **math** package available to you. However, if you decide to only use a specific part of a package, you can always make your import more selective:

```
from math import pi
```

Instructions

- Perform a selective import from the **math** package where you only import the radians function.
- Calculate the distance travelled by the Moon over 12 degrees of its orbit. Assign the result to dist. You can calculate this as **r * phi**, where **r** is the radius and **phi** is the angle in radian.
- Print out dist.

```
# Import radians function of math package


# Definition of radius
r = 192500

# Travel distance of Moon over 12 degrees.
# Store in dist.


# Print out dist
```

Different ways of importing

There are several ways to import packages and modules into Python. Depending on the import call, you'll have to use different Python code.

Suppose you want to use the function inv(), which is in the **linalg** subpackage of the **scipy** package. You want to be able to use this function as follows:

```
my_inv([[1,2], [3,4]])
```

Which **import** statement will you need in order to run the above code without an error?

- ○ `import scipy`
- ○ `import scipy.linalg`
- ○ `from scipy.linalg import my_inv`
- ○ `from scipy.linalg import inv as my_inv`