Introduction to Python
for non-programmers

Research Computing Services

Janna Nugent: janna.nugent@northwestern.edu

http://bit.ly/intro-python-2018

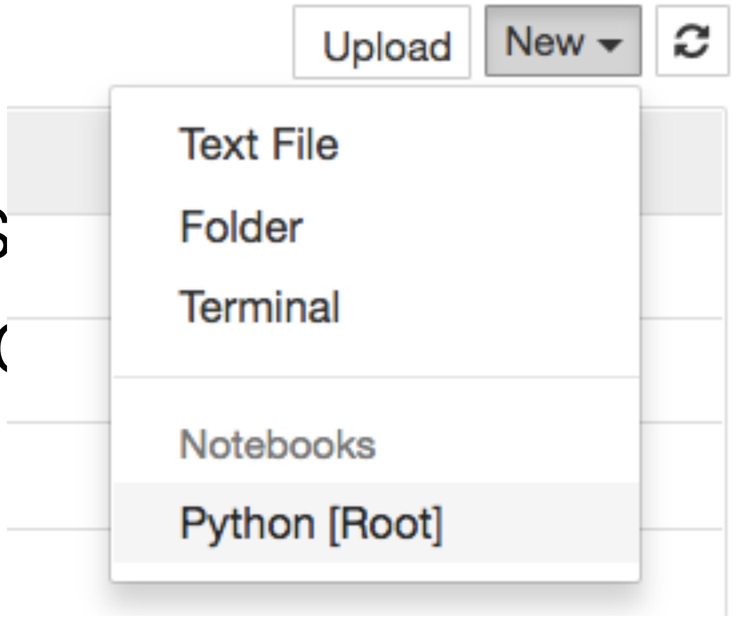
What we've covered so far:

First Session
Why Python
Running Python
Python as a calculator
Variables
Strings
Types
Indexing & Slicing
Lists
If statements
boolean values & expressions

Second Session
Jupyter
Loops - for, pass, while, range, break, continue
Dictionaries
Tuples
Sets
Functions

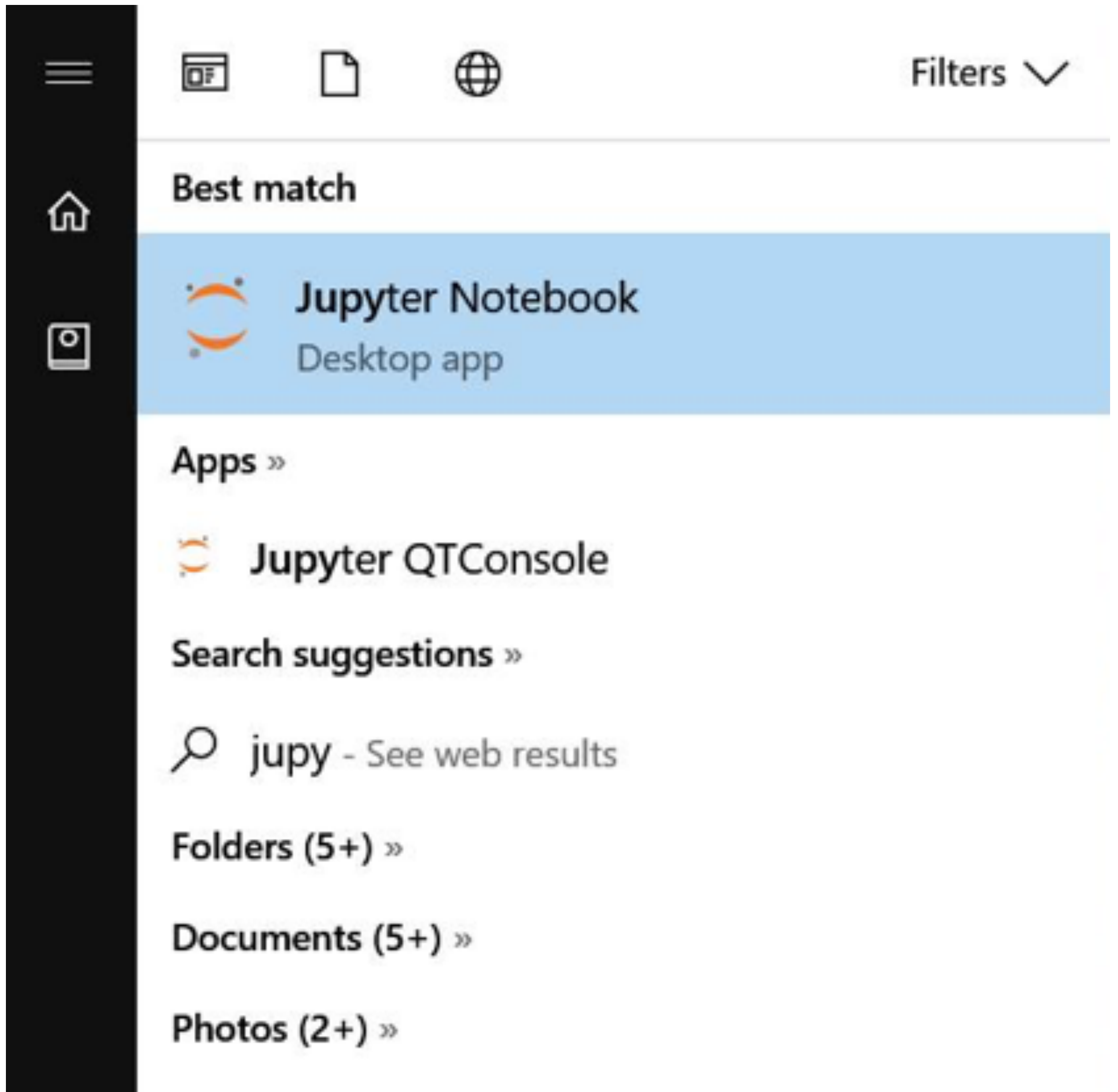
Let's get started: Jupyter (mac)

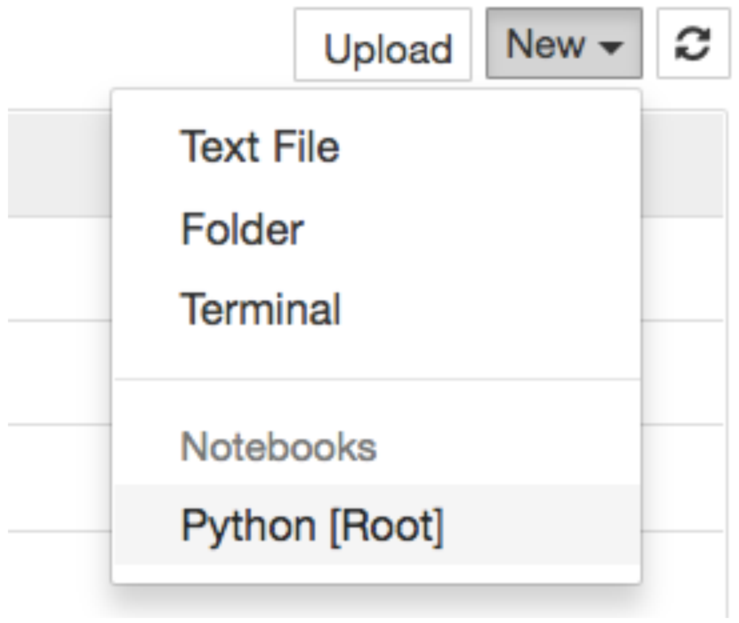
- Open Terminal
- type “jupyter notebook”
- New -> Python [Root]
- quit: Use Control-C to stop this s
down all kernels (twice to skip c



Let’s get started: Jupyter (PC)

- Open: Installed as a program - click on the icon
- New -> Python [Root]





Functions

Also called “subprograms”, a function is a block of organized, reusable code that is used to perform a single, related action.

- Each subprogram has a single entry point
- The calling program is suspended during execution of the called subprogram
- Control always returns to the caller when the called subprogram’s execution terminates

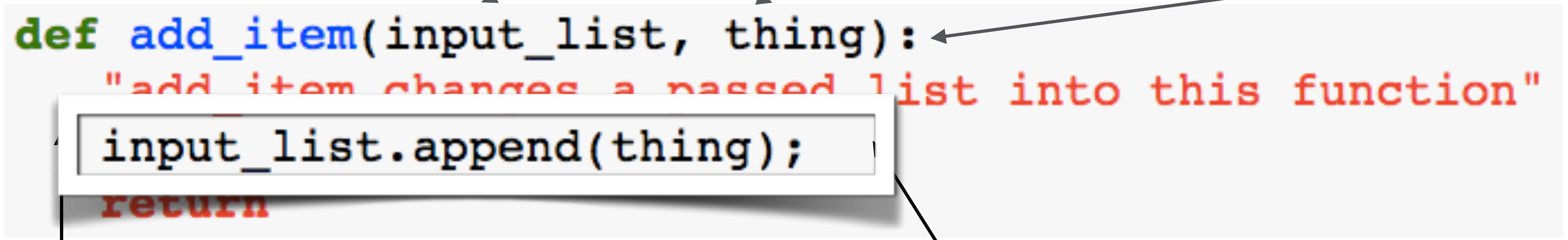
Functions you’ve already used: print(), len(), type()

Writing Functions

“Behold, this is a **function definition**: from here on out when you see this name, it means do this function”

Parameters passed in to the function

do



```
def add_item(input_list, thing):
    "add_item changes a passed list into this function"
    input_list.append(thing);
    return
```

End of function definition.
May include value to return.

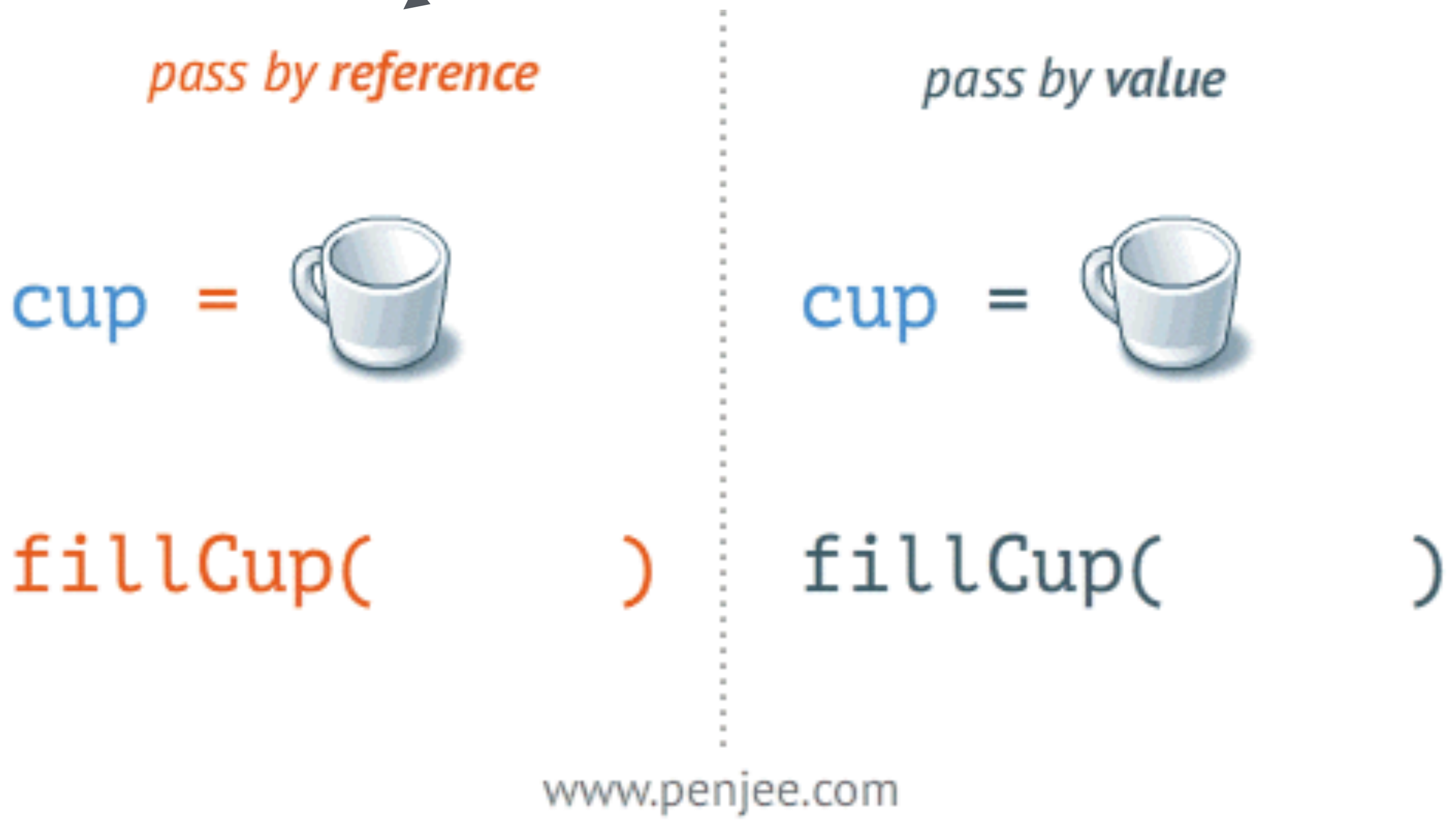
Documentation for your function, access via **add_item??** or **help(add_item)**

Must be indented 4 spaces

Functions

All parameters in the **Python** language are **passed by reference**.

Pass the parameter by sending the memory address it's located at in RAM



Pass a copy of the parameter

Java, C

Object **attributes** and **methods**

“nouns” “verbs”

```
class cup:
    """An object with .name, .state, .color, .temp ()"""

    def __init__(self, name, color):
        self.state = "empty"
        self.color = color
        self.name = name
        self.temp = 72

    def fill(self):
        if (self.state == "empty"):
            self.state = "full"
            self.temp = self.temp + 50
        return

    def drain(self):
        if (self.state == "full"):
            self.state = "empty"
            self.temp = self.temp - 50
        return
```

methods

attributes

Opening Files: IDE

An integrated development environment (IDE) is a software suite that consolidates the basic tools developers need to write and test software. Typically, an IDE contains a code editor, a compiler or interpreter and a debugger that the developer accesses through a single graphical user interface (GUI).

At the command line type: “spyder”

Solving Problems

“The **most common mistake** I see when conducting interviews or watching someone try to solve a programming problem **is they try to start writing code as soon as possible**.

You must resist this urge.

You really want to make sure you **take enough time to understand the problem completely** before attempting to solve it.”

-Matt Sonmez

https://simpleprogrammer.com/2011/01/08/solving-problems-breaking-it-down/

Opening Files: IDE

Research Question:
Is the number of jobs on Quest trending upward, downward or staying flat?

Opening Files: IDE

The Problem:
The number of jobs is recorded every 15 minutes, except sometimes it isn't - how to get average job numbers per day when the sample size is different every day?

Opening Files: IDE

The solution:

Write a python program that:

- Pulls data from the total_jobs.dat file
- Calculates the average number of jobs for each day
- Plot to see trends

Opening Files: IDE

Pseudocode:

Open the data file

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
        increment the number of job samples for the day
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
        increment the number of job samples for the day
    elif the current day has changed from the last record:
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
        increment the number of job samples for the day
    elif the current day has changed from the last record:
        calculate the average number of jobs for the day
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
        increment the number of job samples for the day
    elif the current day has changed from the last record:
        calculate the average number of jobs for the day
        put the ave for the day into a list
```

Opening Files: IDE

Pseudocode:

```
Open the data file
for each line of data:
    get the current day
    get the number of jobs
    if the current day is still the day from the last record:
        add the # of jobs to the total # of jobs for the day
        increment the number of job samples for the day
    elif the current day has changed from the last record:
        calculate the average number of jobs for the day
        put the ave for the day into a list
        reset the variables for the next iteration
```

Opening Files: IDE

https://matplotlib.org/users/pyplot_tutorial.html

Importing Packages

Many Data Science libraries are available for Python, and they are easy to install (using pip):

- DataFrame- Pandas
- Machine Learning- scikitlearn
- Statistics- Scipy
- Arrays- Numpy
- Plots- matplotlib
- Science- anaconda distribution
- DataScrapping- BeautifulSoup
- FetchUrldetails- urllib

Installing Packages

Use “pip”:

In Jupiter notebooks or console:

```
!pip install <package_name>
```

on the command line:

```
pip install <package_name>
```

Package Documentation

help(<package_name>)
or
<package_name>??

also
google <package_name>

Solving Problems

Some approaches aren't good ideas

```
my_list=[1,2,3…n]
for number in my_list:
    for number on my_list:
        for number on my_list:
            for number on my_list:
                for number on my_list:
```

for number in my_list: $\longleftarrow$	n times
for number on my_list: $\longleftarrow$	n^2 times
for number on my_list: $\longleftarrow$	n^3 times
for number on my_list: $\longleftarrow$	n^4 times
for number on my_list: $\longleftarrow$	n^5 times

$n = 20{,}000$

$n^2 = 400{,}000{,}000$

$n^3 = 8{,}000{,}000{,}000{,}000$

$n^4 = 160{,}000{,}000{,}000{,}000{,}000$

$n^5 = 3{,}200{,}000{,}000{,}000{,}000{,}000{,}000{,}000$

Introduction to Python
for non-programmers

Research Computing Services

Janna Nugent: janna.nugent@northwestern.edu

http://bit.ly/intro-python-2018