Introduction to Python (for neuroscience)

NSP bootcamp
Daniel Denman
Assistant Professor, Department of Physiology and Biophysics

Plan for Thursday

- Introduction (45- 60 min)
- Set up Python environments (5 - 30 min); Intro Jupyter notebook (X min)
- Break
- [Finish Intro Jupyter notebook (with instructor) (X min)]
- Independent data analysis notebooks

Plan for Friday

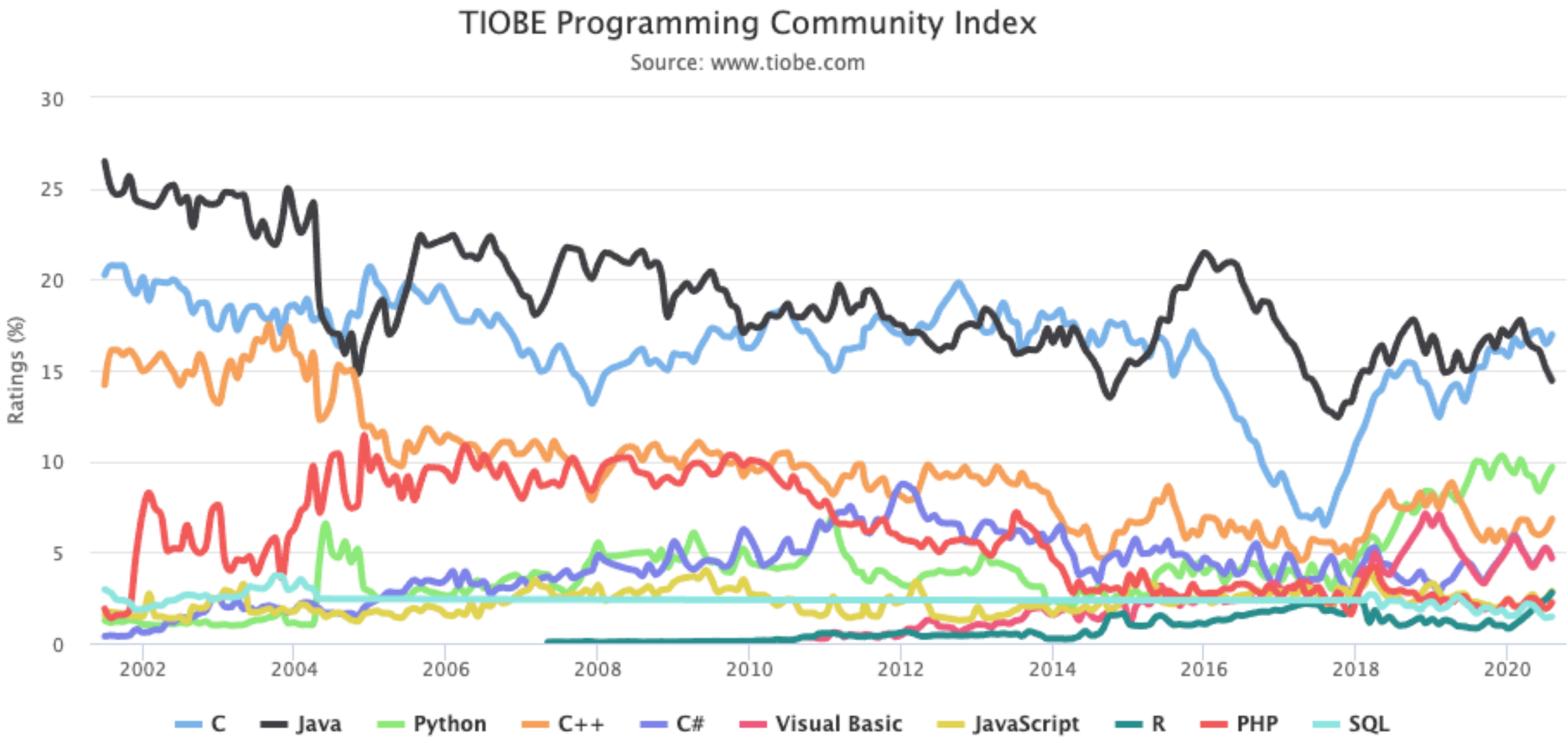
- Intro to scripting and hardware control (60 min).
- Intro to cloud-based Jupyter and team set up.
- Break
- Independent data analysis notebooks - collaboration in the cloud

why code?

- **Python was developed to make it easier for people to automate simple tasks ("scripting")**

- **You *could* do things by hand, but why not have a computer do it?**

history: languages used for neuroscience



TIOBE index

history: languages used for neuroscience

TIOBE index

history: languages used for neuroscience

Aug 2020	Aug 2019	Change	Programming Language	Ratings	Change
1	2	↑	C	16.98%	+1.83%
2	1	↓	Java	14.43%	-1.60%
3	3		Python	9.69%	-0.33%
4	4		C++	6.84%	+0.78%
5	5		C#	4.68%	+0.83%
6	6		Visual Basic	4.66%	+0.97%
7	7		JavaScript	2.87%	+0.62%
8	20	⇈	R	2.79%	+1.97%
9	8	↓	PHP	2.24%	+0.17%
10	10		SQL	1.46%	-0.17%
11	17	⇈	Go	1.43%	+0.45%
12	18	⇈	Swift	1.42%	+0.53%
13	19	⇈	Perl	1.11%	+0.25%
14	15	↑	Assembly language	1.04%	-0.07%
15	11	⇊	Ruby	1.03%	-0.28%
16	12	⇊	MATLAB	0.86%	-0.41%

TIOBE index

history: languages used for neuroscience

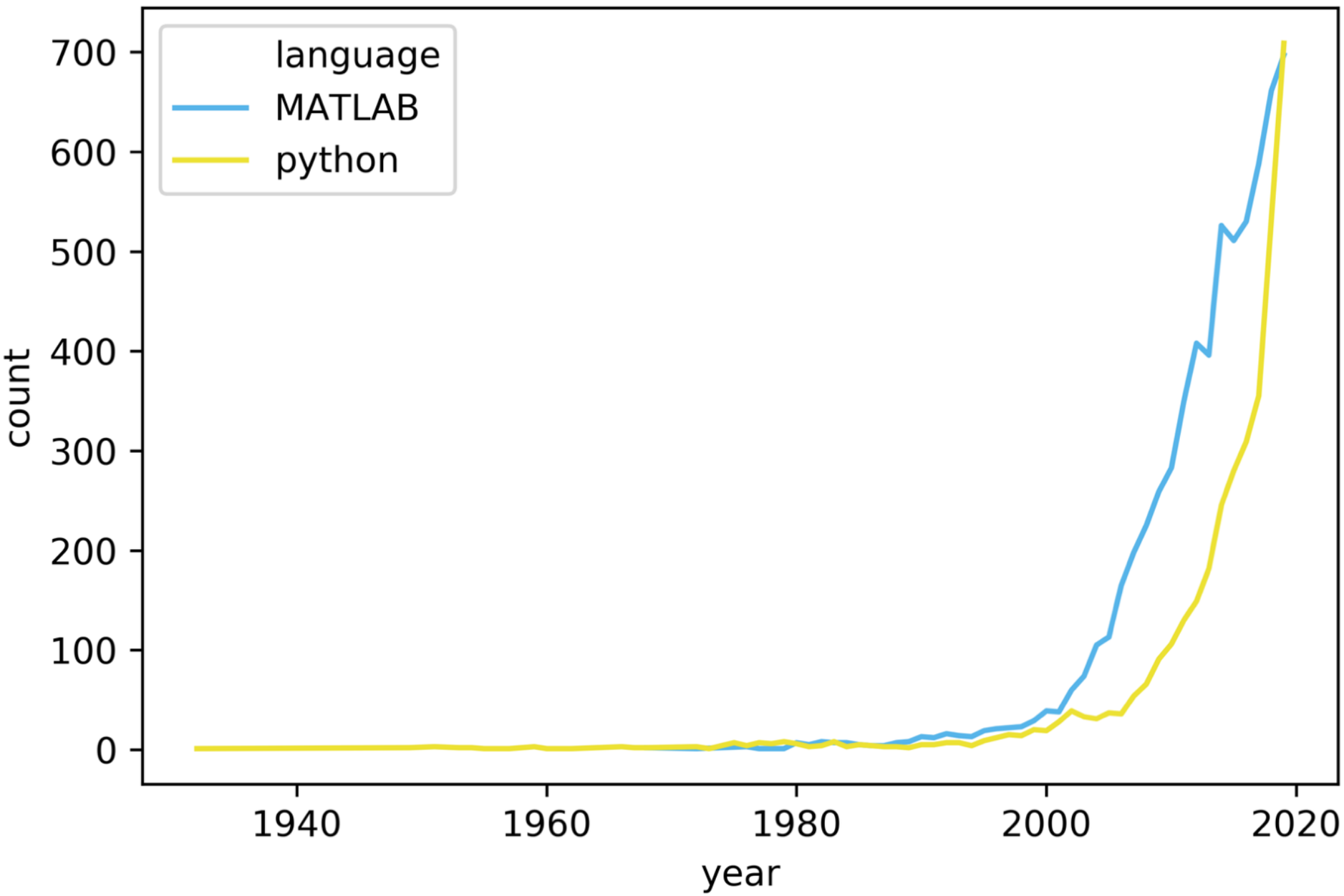
Aug 2022	Aug 2021	Change	Programming Language	Ratings	Change
1	2	↑	Python	15.42%	+3.56%
2	1	↓	C	14.59%	+2.03%
3	3		Java	12.40%	+1.96%
4	4		C++	10.17%	+2.81%
5	5		C#	5.59%	+0.45%
6	6		Visual Basic	4.99%	+0.33%
7	7		JavaScript	2.33%	-0.61%
8	9	↑	Assembly language	2.17%	+0.14%
9	10	↑	SQL	1.70%	+0.23%
10	8	↓	PHP	1.39%	-0.80%
11	16	⇈	Swift	1.27%	+0.30%
12	12		Classic Visual Basic	1.27%	+0.04%
13	22	⇈	Delphi/Object Pascal	1.22%	+0.60%
14	23	⇈	Objective-C	1.22%	+0.61%
15	18	↑	Go	0.98%	+0.08%
16	14	↓	R	0.92%	-0.13%
17	17		MATLAB	0.90%	-0.08%

TIOBE index

history: languages used for neuroscience

TIOBE index

history: languages used for neuroscience



TIOBE index

history: languages used for neuroscience

Volume 16 Issue 12, December 2019

Focus on Deep Learning in Microscopy

Artwork representing the application of deep learning methods in microscopy.

Image: National Institutes of Health/Stocktrek Images/Getty. Cover design: Erin DeWalt

TIOBE index

history: languages used for neuroscience

Volume 16 Issue 12, December 2019

Analysis | Published: 21 October 2019

Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl

Juan C. Caicedo, Allen Goodman, Kyle W. Karhohs, Beth A. Cimini, Jeanelle Ackerman, Marzieh Haghighi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, Mohammad Rohban, Shantanu Singh & Anne E. Carpenter

Nature Methods **16**, 1247–1253(2019) | Cite this article

2 out top 3 entries used python

TIOBE index

history: languages used for neuroscience

Volume 16 Issue 12, December 2019

Focus on Deep Learning in Microscopy

Artwork representing the application of deep learning methods in microscopy.

Image: National Institutes of Health/Stocktrek Images/Getty. Cover design: Erin DeWalt

5/7 software papers used python
1 used R
1 used ImageJ (Java)

TIOBE index

history: languages used for neuroscience

nature methods

Article | Published: 27 July 2020

Cumulus provides cloud-based data analysis for large-scale single-cell and single-nucleus RNA-seq

Bo Li, Joshua Gould, Yiming Yang, Siranush Sarkizova, Marcin Tabaka, Orr Ashenberg, Yanay Rosen, Michal Slyper, Monika S. Kowalczyk, Alexandra-Chloé Villani, Timothy Tickle, Nir Hacohen, Orit Rozenblatt-Rosen & Aviv Regev

TIOBE index

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum
“Benevolent Dictator for Life”

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum
“Benevolent Dictator for Life”

but, but, biology…why code at all?

history

- First released in 1991
- A “scripting” or “high-level” language, designed for readability and productivity
  - simple syntax, use of white space

- Major release: Python 2.7, July 2010
- Use increases
- “data science” after era of “Big Data”
- Support for Python2.7 ended Jan 1, 2020

Guido van Rossum
“Benevolent Dictator for Life”

but, but, biology…why code at all?

- Python was developed to make it easier for people to automate simple tasks (“scripting”)

overview

Some plusses

Some minuses

overview

Some plusses

- **Free**
- **Readable syntax**
- **Cross platform**
- **Huge community**
- **Used across science **and** outside of science**

Some minuses

overview

Some plusses

- **Free**
- **Readable syntax**
- **Cross platform**
- **Huge community**
- **Used across science **<u>*and*</u>** outside of science**

Some minuses

```
a = 1
b = 2
c = 2
if a is not b:
    print('a is not b')
if b is c:
    print('b and c are the same')
```

overview

Some plusses

- **Free**
- **Readable syntax**
- **Cross platform**
- **Huge community**
- **Used across science **<u>*and*</u>** outside of science**

Some minuses

```
a = 1
b = 2
c = 2
if a is not b:
    print('a is not b')
if b is c:
    print('b and c are the same')
```

```
a is not b
b and c are the same
```

overview

Some plusses

- **Free**
- **Readable syntax**
- **Cross platform**
- **Huge community**
- **Used across science **and** outside of science**

Some minuses

```
a = 1
b = 2
c = 2
if a is not b:
    print('a is not b')
if b is c:
    print('b and c are the same')
```

```
a is not b
b and c are the same
```

overview

Some plusses

- **Free**
- **Readable syntax**
- **Cross platform**
- **Huge community**
- **Used across science **and** outside of science**

Some minuses

- **Slow[er] to execute (than C)**
- **White space matters (which some find to be a pain)**
- ***Sometimes problematic + insular culture; see "BDFL"**

*editorial opinion

```
a = 1
b = 2
c = 2
if a is not b:
    print('a is not b')
if b is c:
    print('b and c are the same')
```

```
a is not b
b and c are the same
```

overview: Python language basics

overview: Python language basics

“high-level” :

you don’t need to know how a computer *actually* works to use Python. Memory addressing, pointers, call stacks - blah blah computer science

garbage collected:

you, the programmer do not usually need to worry about cleaning up your “garbage” - memory pointers etc. More stability, fewer resources.

overview: Python language basics

“high-level” :

you don’t need to know how a computer *actually* works to use Python. Memory addressing, pointers, call stacks - blah blah computer science

garbage collected:

you, the programmer do not usually need to worry about cleaning up your “garbage” - memory pointers etc. More stability, fewer resources.

Interpreted

Translated into steps for the computer to execute one-by-one, as opposed to translating all of the steps at the beginning, and then (e.g, Igor is a compiled language).

overview: Python language basics

“high-level” :

you don’t need to know how a computer *actually* works to use Python. Memory addressing, pointers, call stacks - blah blah computer science

garbage collected:

you, the programmer do not usually need to worry about cleaning up your “garbage” - memory pointers etc. More stability, fewer resources.

Interpreted

Translated into steps for the computer to execute one-by-one, as opposed to translating all of the steps at the beginning, and then (e.g, Igor is a compiled language).

Several programming paradigms:

Procedural programming (like C), Object-oriented programming (like Java), functional programming (Wolfram)

overview: Python language basics

“high-level” :

you don’t need to know how a computer *actually* works to use Python. Memory addressing, pointers, call stacks - blah blah computer science

garbage collected:

you, the programmer do not usually need to worry about cleaning up your “garbage” - memory pointers etc. More stability, fewer resources.

Interpreted

Translated into steps for the computer to execute one-by-one, as opposed to translating all of the steps at the beginning, and then (e.g, Igor is a compiled language).

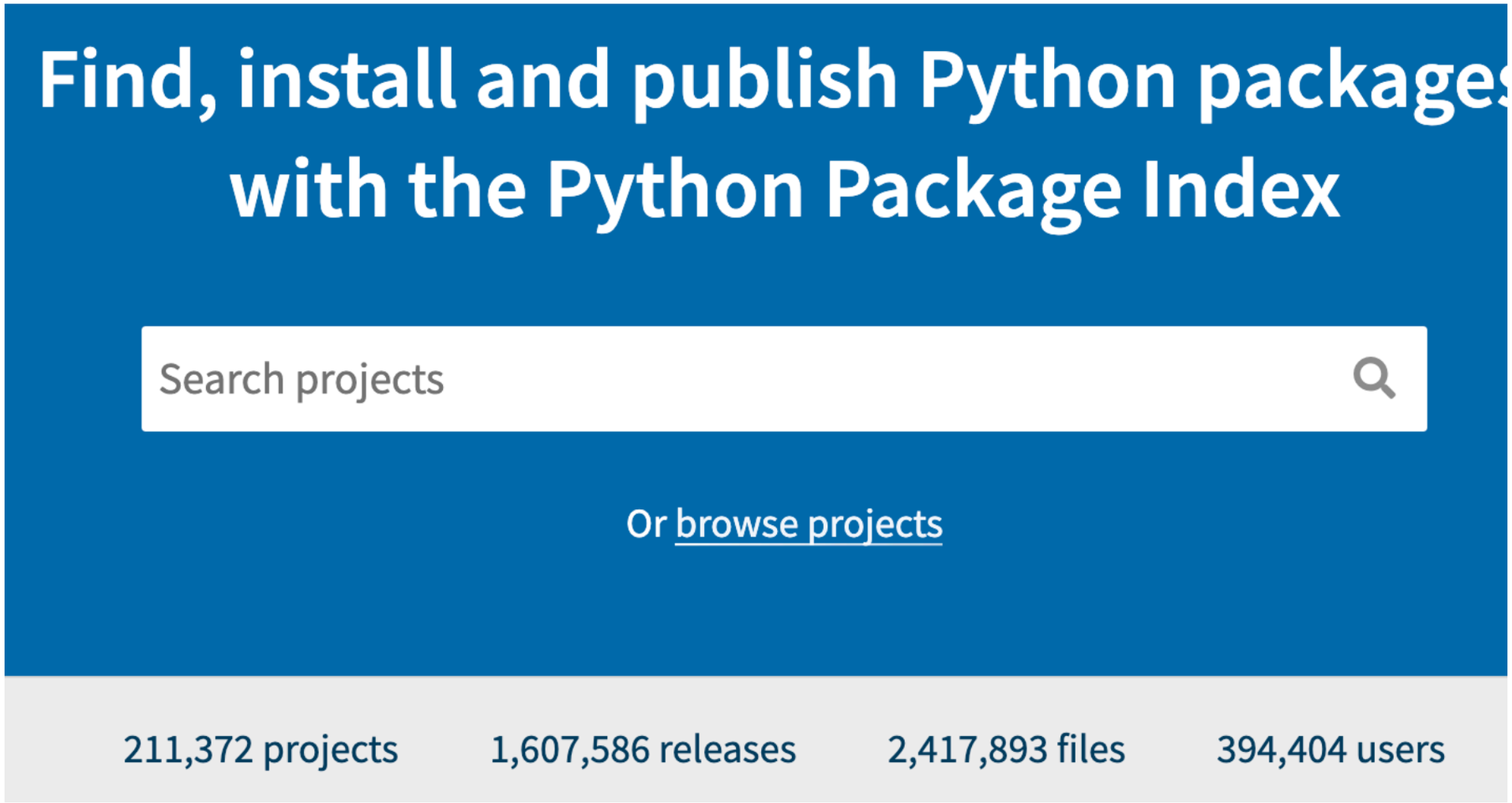
Several programming paradigms:

Procedural programming (like C), Object-oriented programming (like Java), functional programming (Wolfram)

Standard library

Often written in C (again, you, the programmer, don’t need to care about this), these are functions and tools that ship with Python. Widely useful, and already vetted. Backwards compatible.

overview: packages



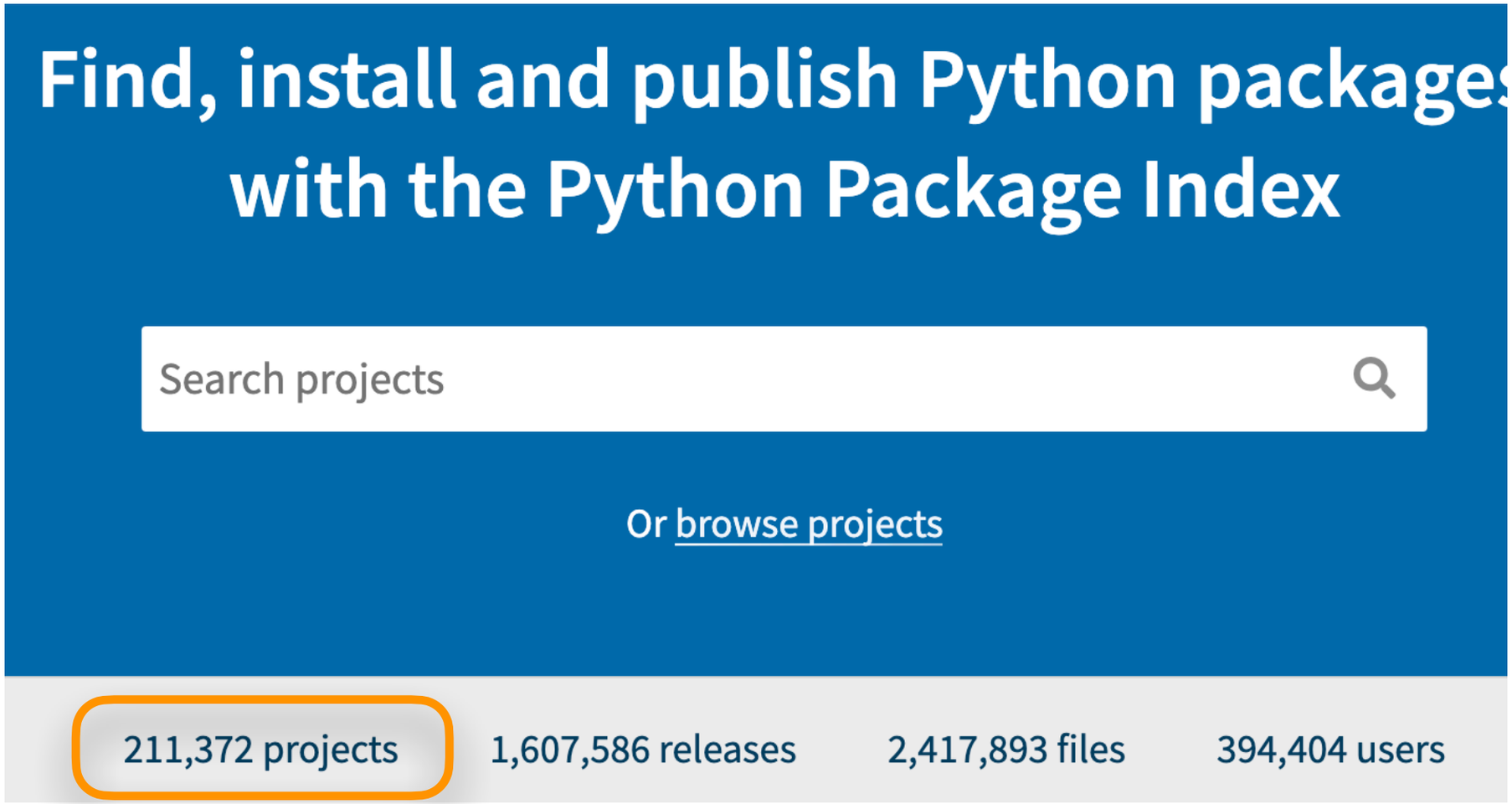
Packages

In addition to the standard library, the true power of python is the extensive world of packages available. These are sets of tools you can use with Python to do just about anything!

Some are general tools, the hammers or screwdrivers of using python for science: **numpy**, **matplotlib, pandas, seaborn**

Others are specialized: **scikit-learn, PIL, scanpy, Suite2P, DeepLabCut, PyTom**

overview: packages



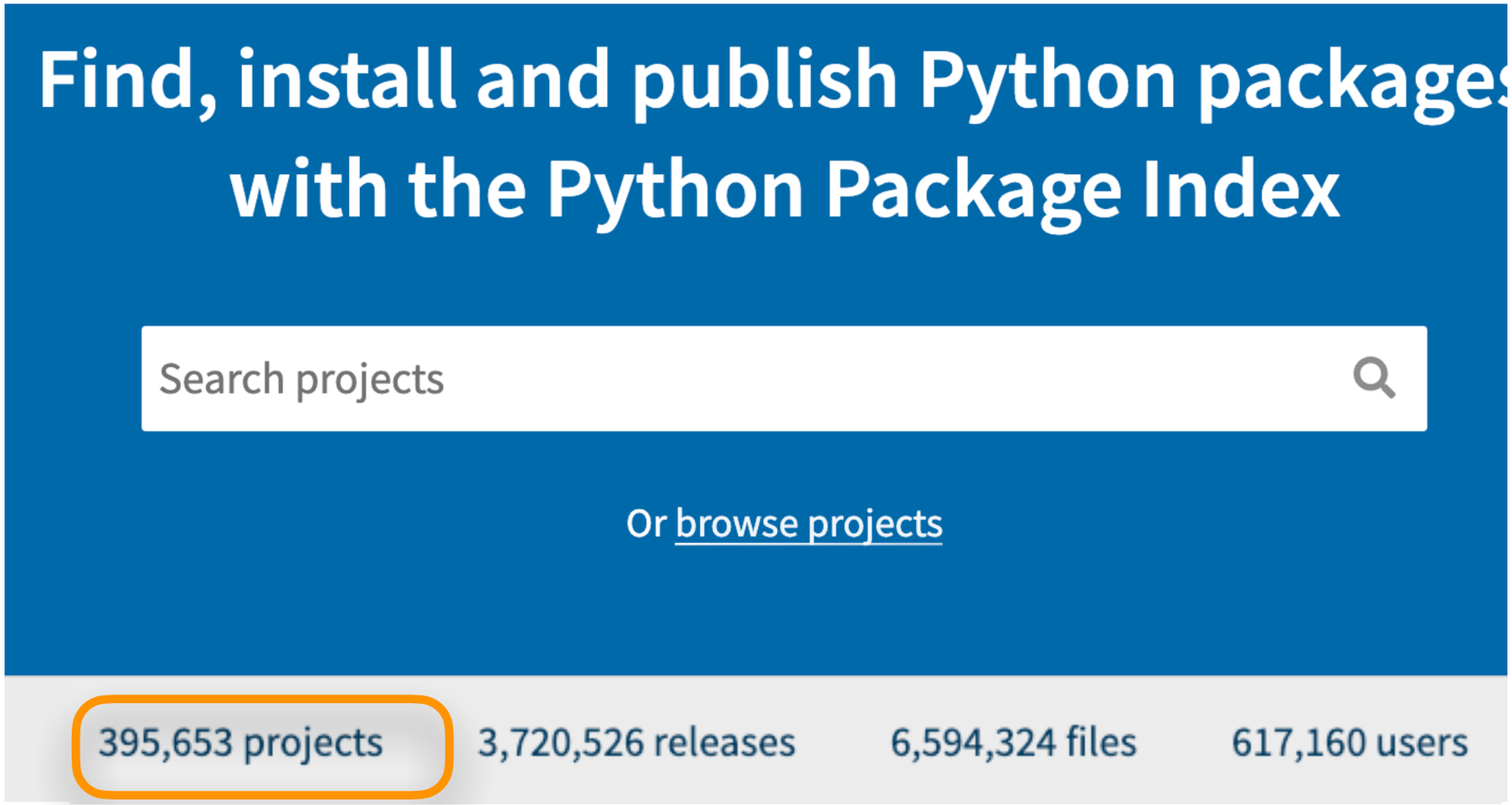
Packages

In addition to the standard library, the true power of python is the extensive world of packages available. These are sets of tools you can use with Python to do just about anything!

Some are general tools, the hammers or screwdrivers of using python for science: **numpy**, **matplotlib, pandas, seaborn**

Others are specialized: **scikit-learn, PIL, scanpy, Suite2P, DeepLabCut, PyTom**

overview: packages



Packages

In addition to the standard library, the true power of python is the extensive world of packages available. These are sets of tools you can use with Python to do just about anything!

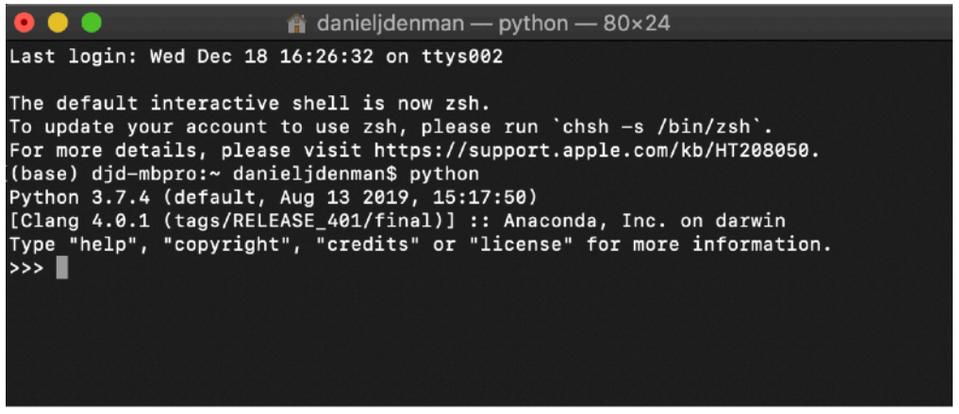
Some are general tools, the hammers or screwdrivers of using python for science: **numpy**, **matplotlib, pandas, seaborn**

Others are specialized: **scikit-learn, PIL, scanpy, Suite2P, DeepLabCut, PyTom**

overview: levels

overview: levels

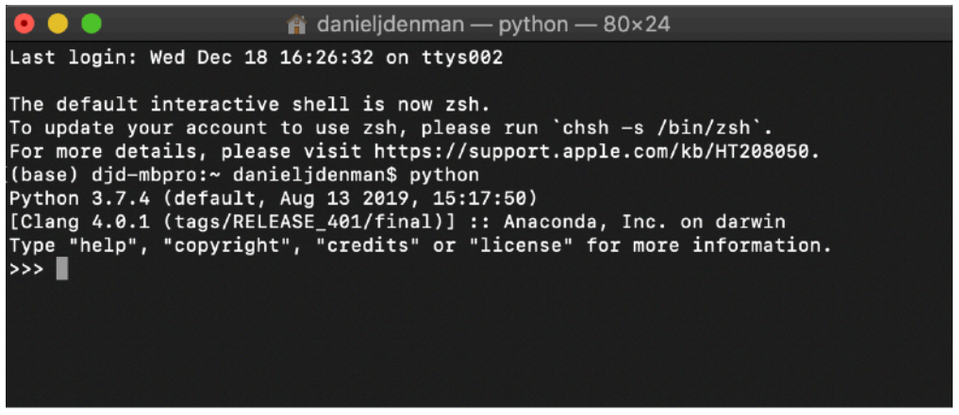
System



Native in Mac OS X, Linux; in Windows store (free)

overview: levels

System



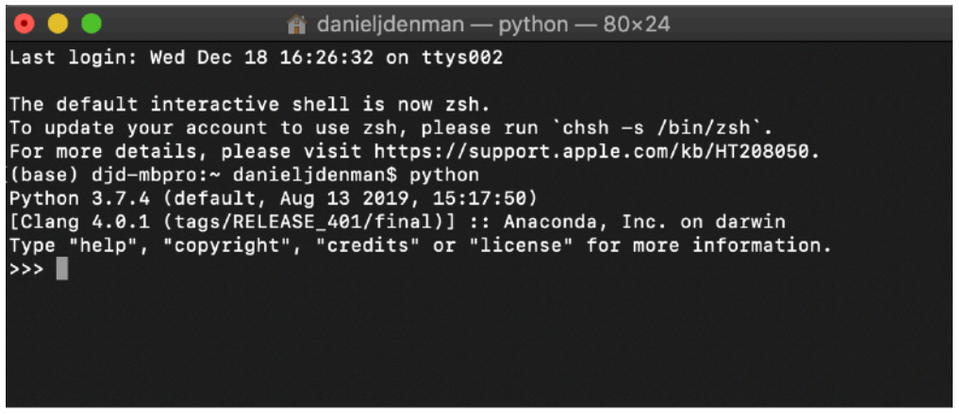
Native in Mac OS X, Linux; in Windows store (free)

Package Managers
Environments

ANACONDA®

overview: levels

System



Native in Mac OS X, Linux; in Windows store (free)

Package Managers Environments

ANACONDA®

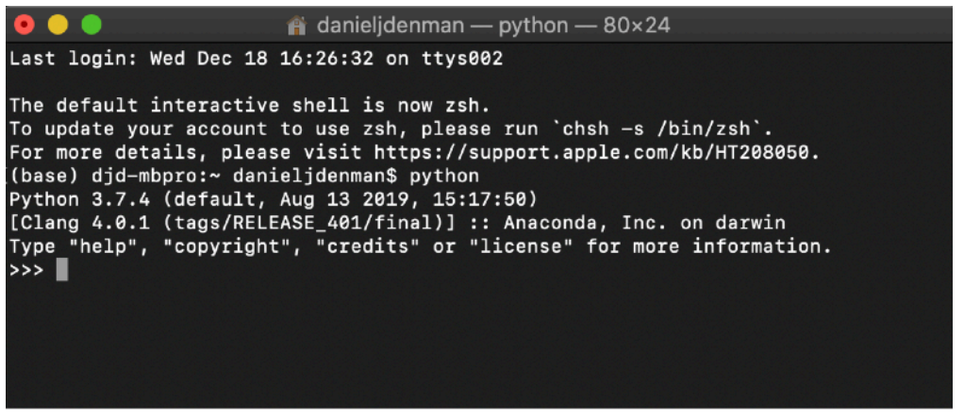
Containerized

docker

COCALC

overview: levels

System



Native in Mac OS X, Linux; in Windows store (free)

Package Managers Environments

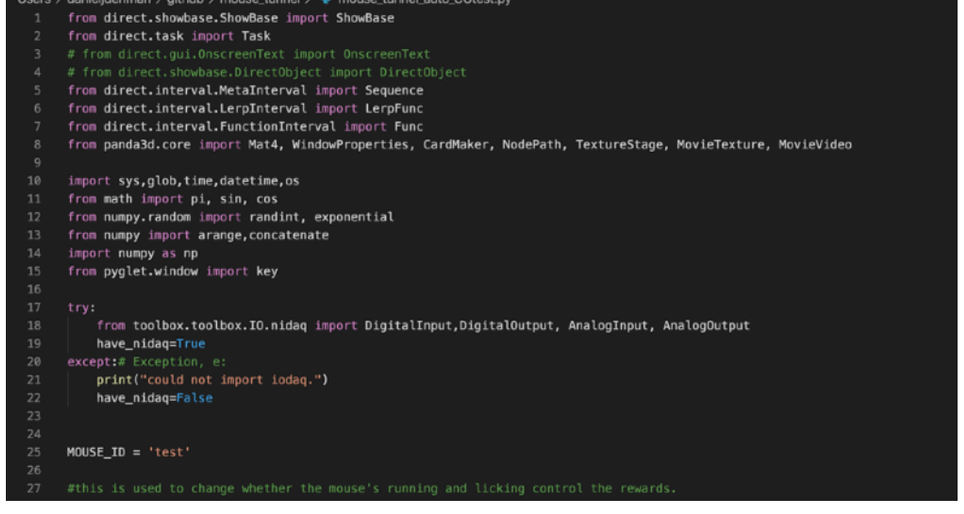
ANACONDA®

Containerized

docker

COCALC

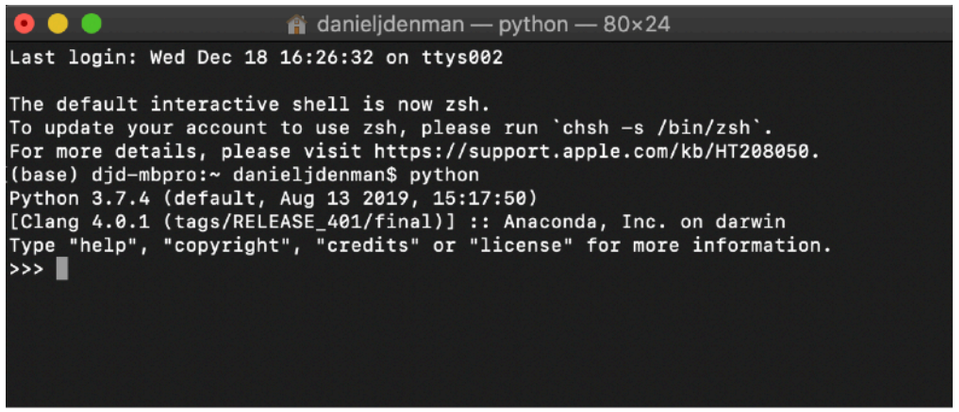
Scripts



```
Users > danieljdenman > github > mouse_tunnel > mouse_tunnel_auto_CUtest.py
from direct.showbase.ShowBase import ShowBase
from direct.task import Task
# from direct.gui.OnscreenText import OnscreenText
# from direct.showbase.DirectObject import DirectObject
from direct.interval.MetaInterval import Sequence
from direct.interval.LerpInterval import LerpFunc
from direct.interval.FunctionInterval import Func
from panda3d.core import Mat4, WindowProperties, CardMaker, NodePath, TextureStage, MovieTexture, MovieVideo

import sys,glob,time,datetime,os
from math import pi, sin, cos
from numpy.random import randint, exponential
from numpy import arange,concatenate
import numpy as np
from pyglet.window import key

try:
    from toolbox.toolbox.IO.nidaq import DigitalInput,DigitalOutput, AnalogInput, AnalogOutput
    have_nidaq=True
except:# Exception, e:
    print("could not import iodaq.")
    have_nidaq=False


MOUSE_ID = 'test'

#this is used to change whether the mouse's running and licking control the rewards.
```

overview: levels

System



Native in Mac OS X, Linux; in Windows store (free)

Package Managers
Environments

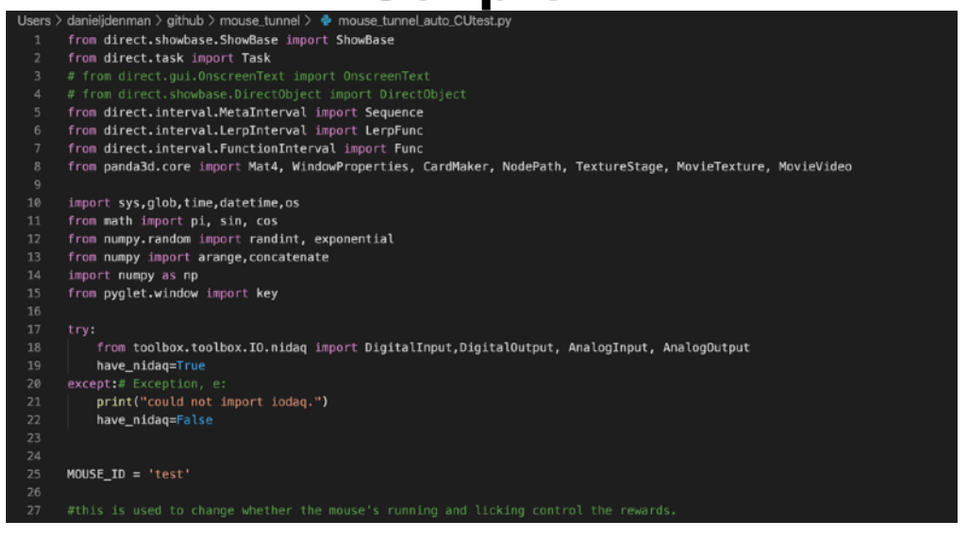
ANACONDA®

Containerized

docker

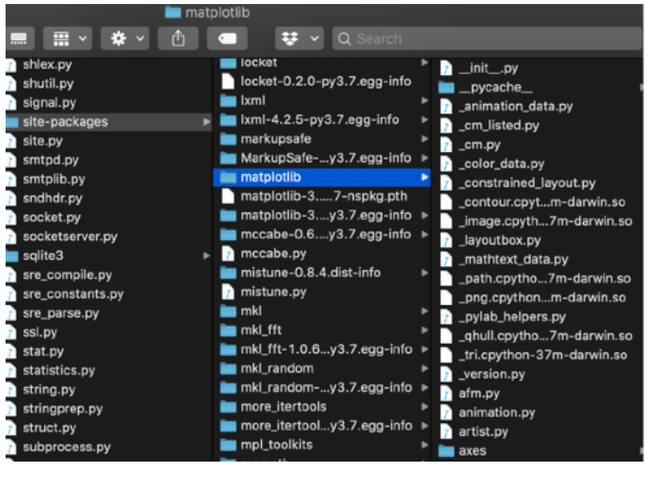
COCALC

Scripts



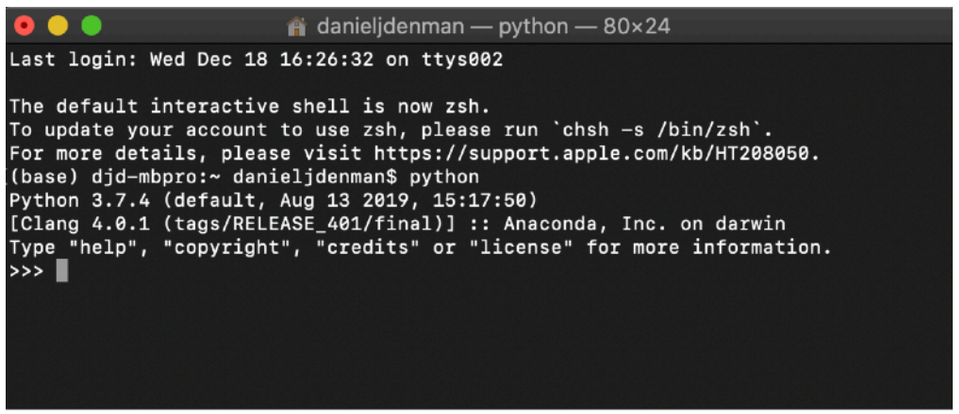
```
Users > danieljdenman > github > mouse_tunnel > mouse_tunnel_auto_CUtest.py
from direct.showbase.ShowBase import ShowBase
from direct.task import Task
# from direct.gui.OnscreenText import OnscreenText
# from direct.showbase.DirectObject import DirectObject
from direct.interval.MetaInterval import Sequence
from direct.interval.LerpInterval import LerpFunc
from direct.interval.FunctionInterval import Func
from panda3d.core import Mat4, WindowProperties, CardMaker, NodePath, TextureStage, MovieTexture, MovieVideo

import sys,glob,time,datetime,os
from math import pi, sin, cos
from numpy.random import randint, exponential
from numpy import arange,concatenate
import numpy as np
from pyglet.window import key

try:
    from toolbox.toolbox.IO.nidaq import DigitalInput,DigitalOutput, AnalogInput, AnalogOutput
    have_nidaq=True
except:# Exception, e:
    print("could not import iodaq.")
    have_nidaq=False


MOUSE_ID = 'test'

#this is used to change whether the mouse's running and licking control the rewards.
```

Packages



overview: levels

System



Native in Mac OS X, Linux; in Windows store (free)

Package Managers
Environments

ANACONDA®

Containerized

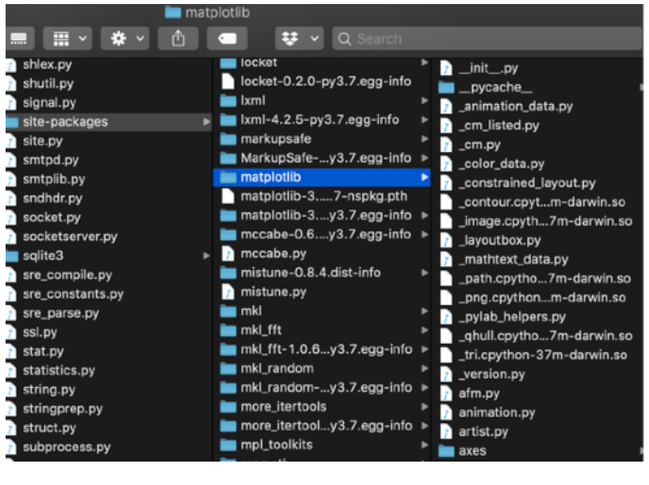
docker

COCALC

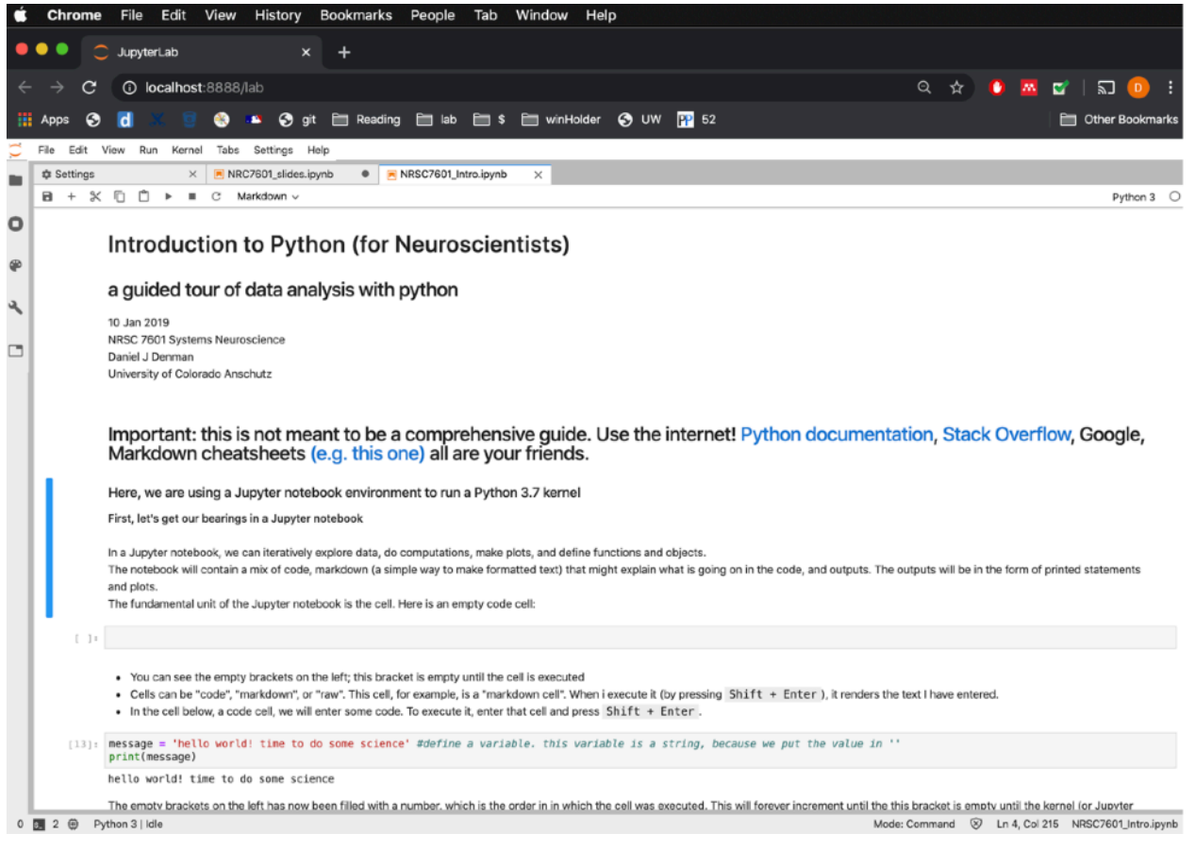
Scripts

```
Users > danieljdenman > github > mouse_tunnel > mouse_tunnel_auto_CUtest.py
1   from direct.showbase.ShowBase import ShowBase
2   from direct.task import Task
3   # from direct.gui.OnscreenText import OnscreenText
4   # from direct.showbase.DirectObject import DirectObject
5   from direct.interval.MetaInterval import Sequence
6   from direct.interval.LerpInterval import LerpFunc
7   from direct.interval.FunctionInterval import Func
8   from panda3d.core import Mat4, WindowProperties, CardMaker, NodePath, TextureStage, MovieTexture, MovieVideo
9
10  import sys,glob,time,datetime,os
11  from math import pi, sin, cos
12  from numpy.random import randint, exponential
13  from numpy import arange,concatenate
14  import numpy as np
15  from pyglet.window import key
16
17  try:
18      from toolbox.toolbox.IO.nidaq import DigitalInput,DigitalOutput, AnalogInput, AnalogOutput
19      have_nidaq=True
20  except:# Exception, e:
21      print("could not import iodaq.")
22      have_nidaq=False
23
24
25  MOUSE_ID = 'test'
26
27  #this is used to change whether the mouse's running and licking control the rewards.
```

Packages



Notebooks (IPython, Jupyter, Jupyter Lab)



why is it good for doing neuroscience?

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)

packages!

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work

packages!

Stack Overflow <— not cheating!

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

Data science

scikit-learn
Pandas
TensorFlow

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

Data science

scikit-learn
Pandas
TensorFlow

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

Data science

scikit-learn
Pandas
TensorFlow

Specialized Tools

Image anlysis: PIL / OpenCV
Ca2+ analysis: Suite2P, AQUA
Movement Tracking: DeepLabCut
Expression Analysis: scanpy
…

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

Data science

scikit-learn
Pandas
TensorFlow

Specialized Tools

Image anlysis: PIL / OpenCV
Ca2+ analysis: Suite2P, AQUA
Movement Tracking: DeepLabCut
Expression Analysis: scanpy
…

Sharing

Docker
Google Colab
Jupyter
[it is free!]

why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implelement yourself (i.e., ML)
- automate boring stuff / use other people's hard work
- make your science, and science., better

packages!

Stack Overflow <— not cheating!

also it is free —> democratizing science

in this realm, cloud resources (data, compute) also open science to a wider group that aren't collecting their own data and running their own super computers

Hardware control

RasberryPi
Arduino
PyDAQMX
PsychoPy
…many APIs…

Data science

scikit-learn
Pandas
TensorFlow

Specialized Tools

Image anlysis: PIL / OpenCV
Ca2+ analysis: Suite2P, AQUA
Movement Tracking: DeepLabCut
Expression Analysis: scanpy
…

Sharing

Docker
Google Colab
Jupyter
[it is free!]

Guided example:

Variables: definitions, types
Functions
Some important packages
Making plots

git

…and GitHub, are **version control**

- This is important. And not intuitive. It will likely make you frustrated and/or confused at some point.
- Version control is not optional; if you don't use git for version control, you are going to use something else (e.g,: analysis_script_v1.py, analysis_script_v2.py, analysis_script_v2_20210622.py, analysis_script_v3_07142021.py, analysis_script_final.py,analysis_script_final2.py,…, analysis_script_final2_for.py)
- Making git a part of your workflow can simplify and provide redundancy and flexibility; more advanced features also makes sharing simpler. Evaluation.
- git has to be installed, which we will use Anaconda to do so
- GitHub Desktop https://desktop.github.com/ is by far the easiest way; git bash (command line) is another option
- We're going to go over some git interactively to get course materials today.