

# **Week 1: overview / review**

**NRSC 7657 Workshop in Advanced Programming for  
Neuroscientists**

# Course outline

- Python and MATLAB
- Didactic and practical
  - Language specifics
  - General concepts in computing (version control, debugging, unit testing, scaling)
- Independent project
  - Work with real data; preferably your own (whatever it is!) or choose a public data set (see next slide)
  - Develop an idea; **\*\*Schedule a meeting with Dan this week\*\***

<https://doodle.com/meeting/participate/id/azmXJmre>

June 13	Week 1 – overview / review	Course overview: theory of computing, landscape of computing options. Basic usage in python and MATLAB; basic data types; environments Style guidelines ( <a href="#">ten simple rules</a> ); git and version control
June 20	Week 2 – language fundamentals	Functions; Objects and Classes; Workspaces Typical data formats: working with tabular data, images, and time series. NeurodataWithoutBorders format
June 27	Week 3 – workflow management and outputs	Importing and exporting Plotting and visualization - from bar charts to 3D animation
July 11	Week 4 – usability	Troubleshooting and debugging; unit testing
July 18	Week 5 – scaling	Iteration and code profiling; parallel computing. Code quality-of-life topics
July 25	Week 6 – collaboration	Cloud-based tools: AWS, GCC, Colab, jupyterhub, deepnote. Overview of some available SaaS tools, python focused. Group programming time
August 1	Week 7 – applications/flex topic	Applications: image processing (ES) Group programming time
August 8	Week 8 – applications/flex topic	Applications: spike sorting (DD) Group programming time
August 15	Week 9 – applications/flex topic	Applications: DeepLabCut and cloud ML Group programming time
August 22	Week 10 – Final presentations and code review	Final pres. and code review Final pres. and code review

# Project datasets

- Yours!
- Someone from your lab

# Public datasets

(Suggestions, incomplete list)

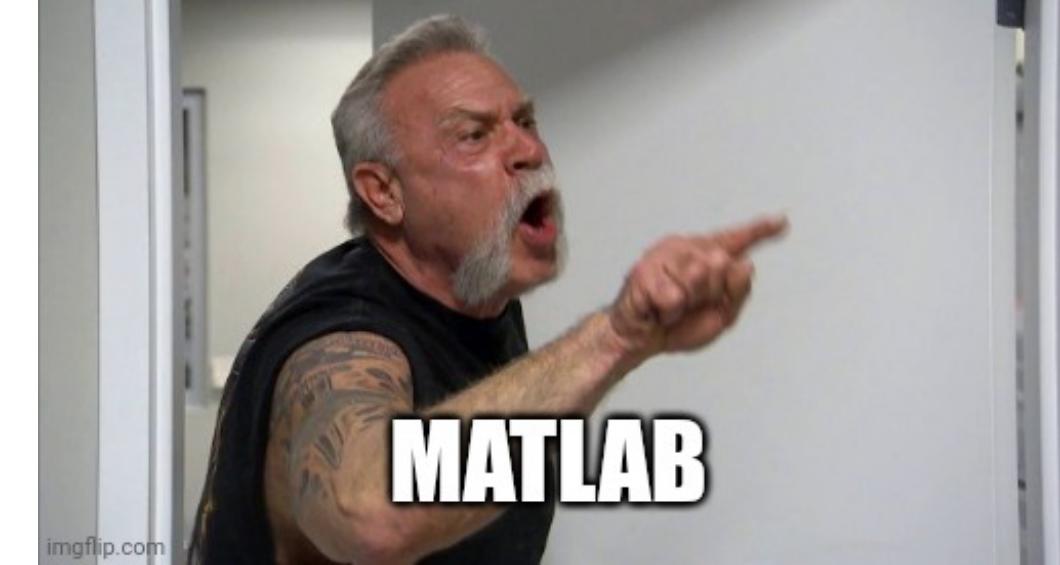
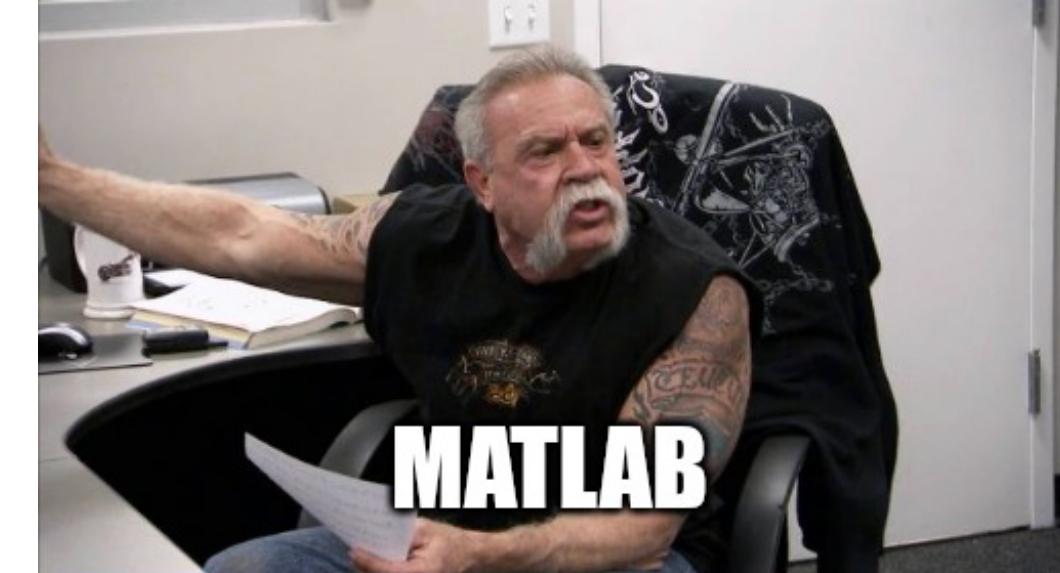
- Allen Institute for Brain Science [brain-map.org](https://brain-map.org)
- DABI Data Archive BRAIN Initiative <https://dabi.loni.usc.edu/home>

The screenshot shows a grid of project datasets. The first row contains three items: 'Transcriptional Landscape of the Brain' (with an icon of a brain grid), 'Behavioral Circuits & Sensory Processing' (with an icon of brain waves), and 'Connectivity Matrices' (with an icon of a brain network). The second row contains two items: 'Computational Modeling & Theory' (with an icon of a brain model) and 'Cell Taxonomies' (with an icon of a brain cell). Each item has a brief description and a blue 'Explore' button.

The screenshot shows a 'MENU' bar with options like 'Explore Studies' and 'Explore by Interest'. Below it is a 'Explore Data' section with a search bar labeled 'Search Interests'. A list of studies is shown, including 'Adaptive Neurostimulation to Restore...' by University of Colorado (Thompson). A call-to-action button 'Open' is visible. On the right, there are navigation icons for 'Build Cohorts' and 'Explore by study by areas of interest'. Numbered circles (1, 2, 3) are overlaid on the bottom right corner.

# What this course is \*\*not\*\*

- Computational neuroscience
- Mathematics
- Computer science



# **What this course is**

## **Each session:**

- Some slides (0 - 60 minutes)
- Working through a notebook or interactive coding together
- Independent coding. We're going to do your science, with a computer. Maybe your computer, or maybe a cloud computer. Definitely your science.

# What this course is

## Each session:

- Some slides (0 - 60 minutes)
- Working through a notebook or **interactive coding** together
- Independent coding. We're going to do your science, with a computer. Maybe your computer, or maybe a cloud computer. Definitely your science.

**Ask questions!**



# Goals of this course

- Exposure to differing approaches to computing in neuroscience
- Develop confidence in independent coding skills
- Complete a project using data relevant to your thesis work

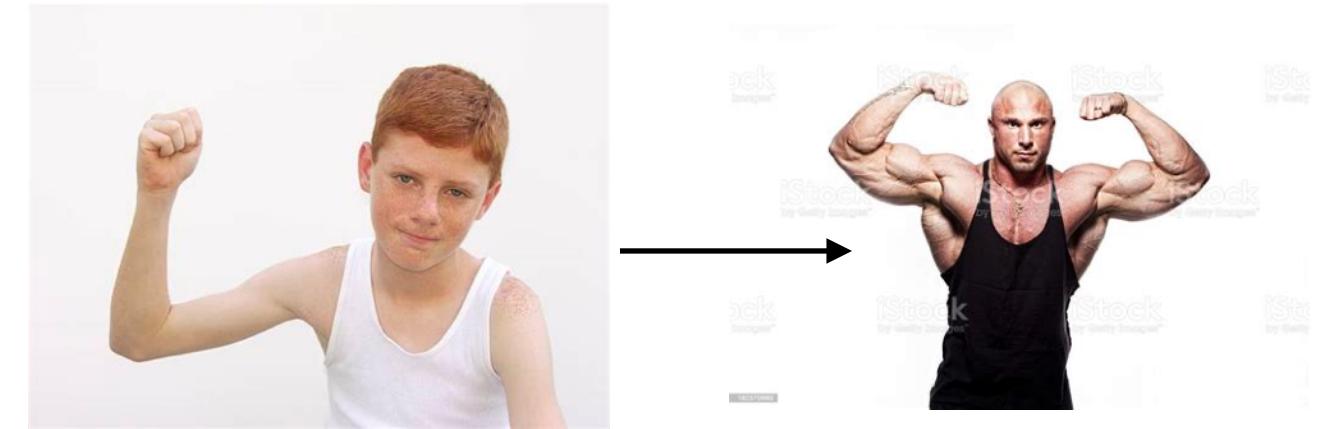
# Goals of this course

- Exposure to differing approaches to computing in neuroscience
- Develop confidence in independent coding skills
- Complete a project using data relevant to your thesis work



# Goals of this course

- Exposure to differing approaches to computing in neuroscience
- Develop confidence in independent coding skills
- Complete a project using data relevant to your thesis work

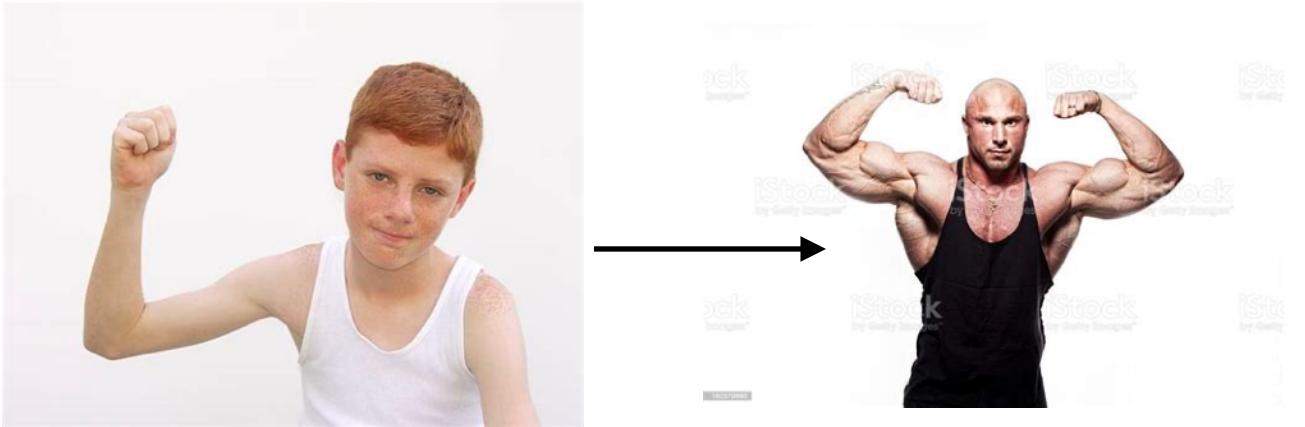


# Goals of this course

- Exposure to differing approaches to computing in neuroscience



- Develop confidence in independent coding skills

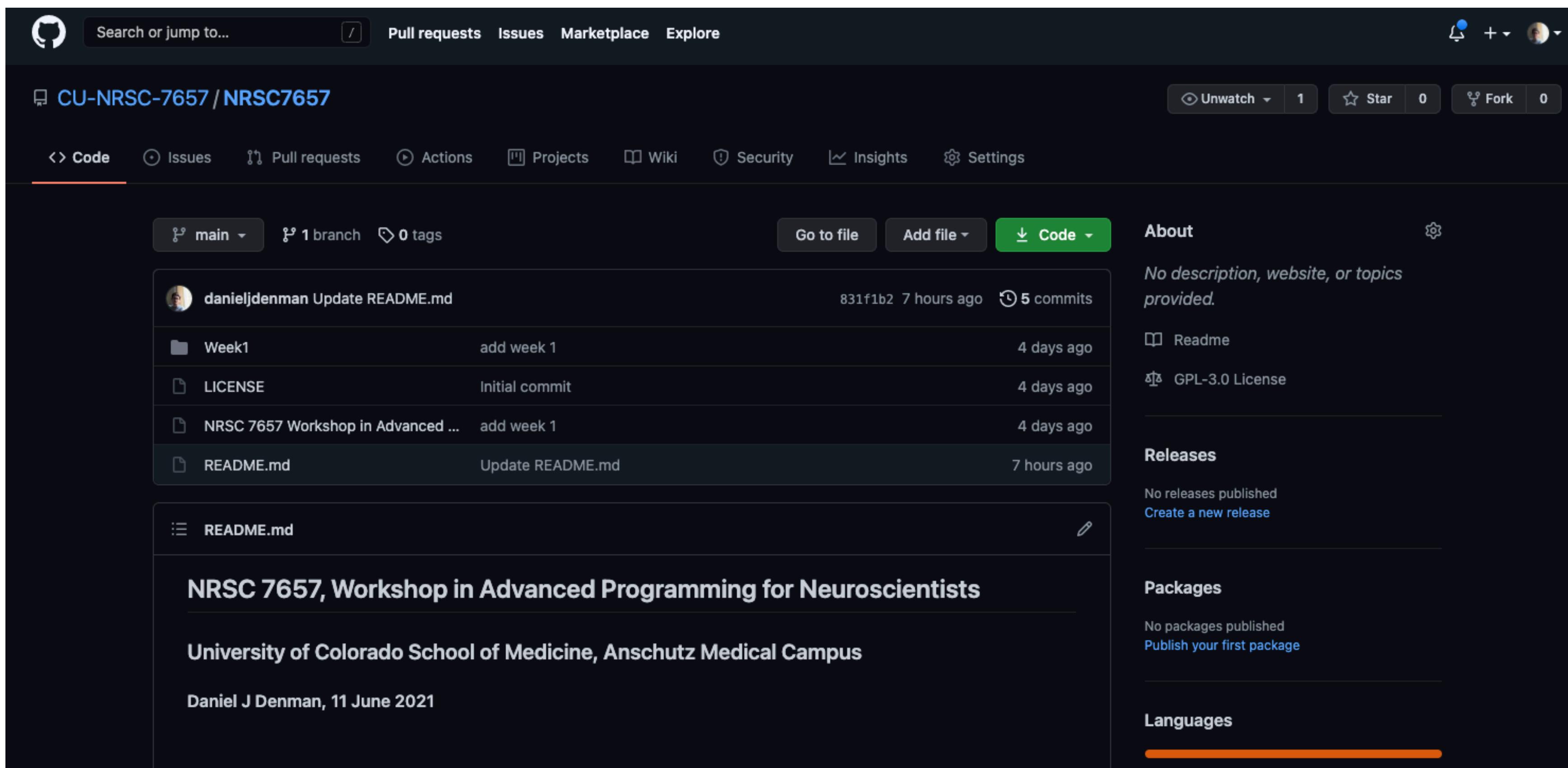


- Complete a project using data relevant to your thesis work



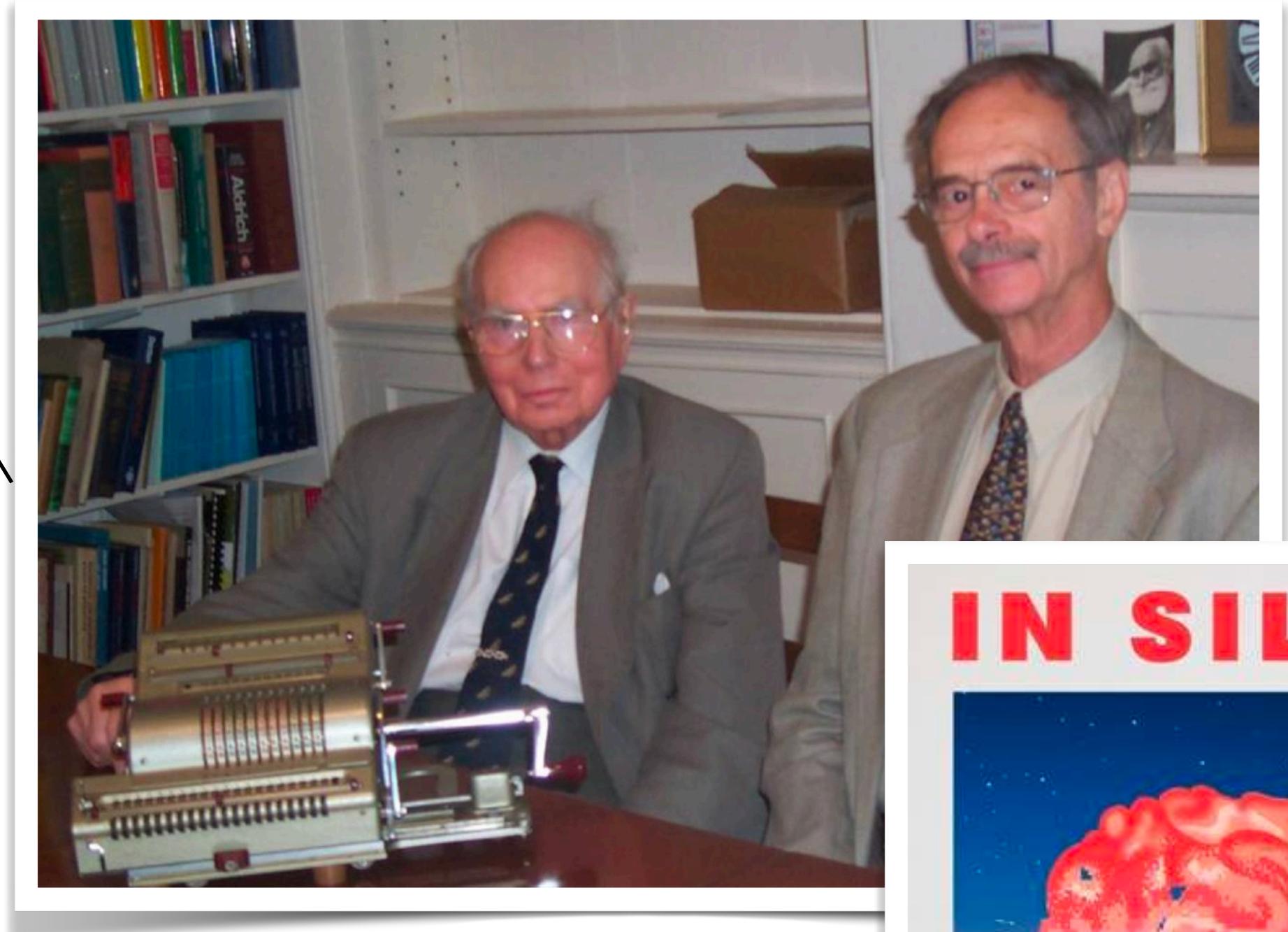
# Final practicalities

- GitHub: <https://github.com/CU-NRSC-7657/NRSC7657>

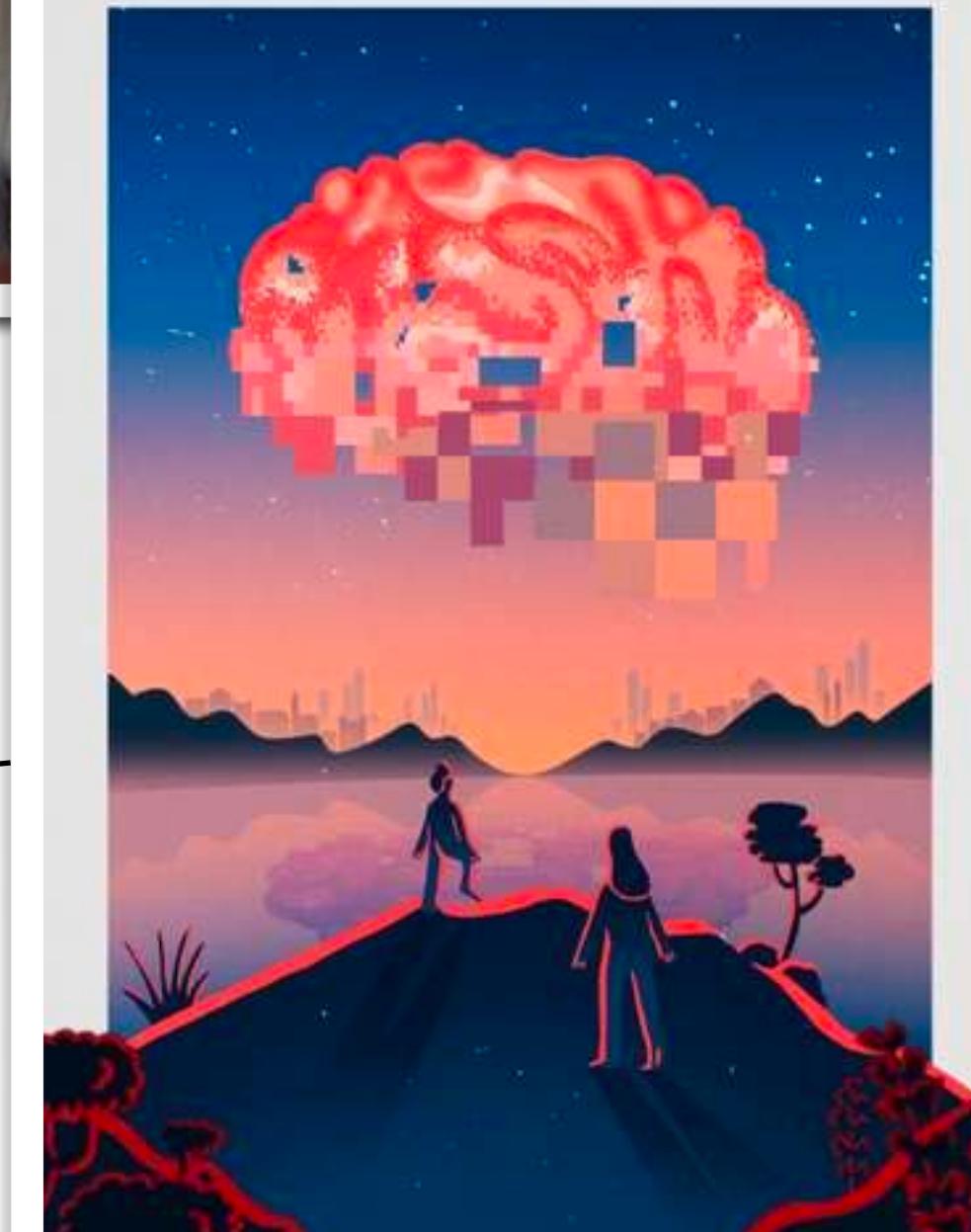


# Computing in neuroscience

- Hodgkin-Huxley's mechanical computer
- Computer based analysis / replacing chart measurements
- NEURON simulations; single neuron modeling
- ...
- “data science”, open data
- Blue brain project, *In silico*



**IN SILICO**



# Computing in neuroscience

## An example: code written by neuroscientists for one experiment

### planning

Probe trajectory: MATLAB  
Model prediction: python - jupyter notebook

### experimental control

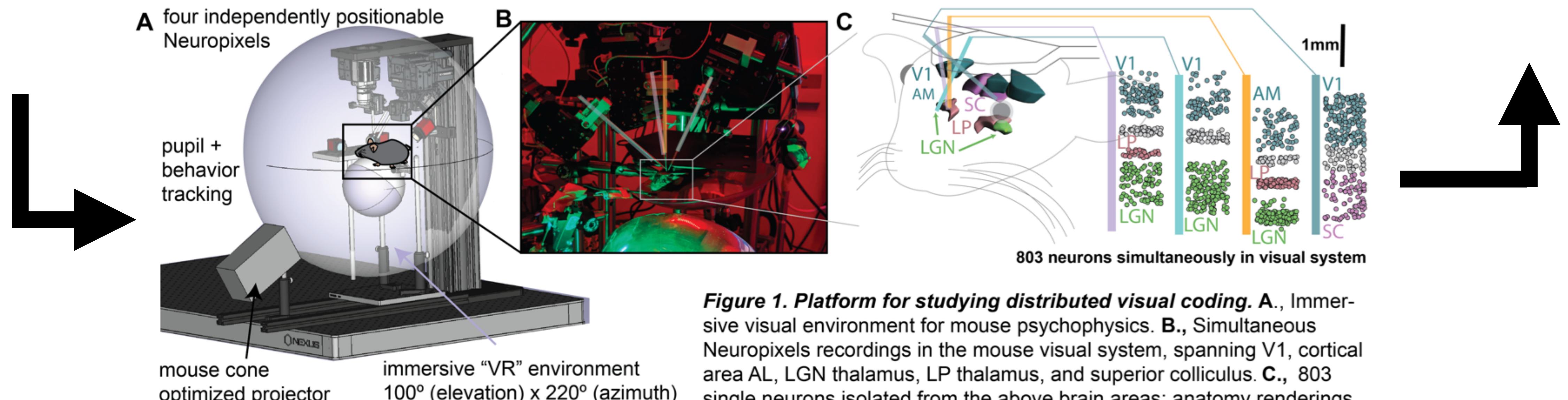
Acquisition hardware: FPGA gate array programming  
Acquisition software: C++; python plugin; [Julia]  
Visual stimuli: python script; embedded python in React  
Video monitoring: python script

### “pre-processing”

Spike sorting: MATLAB  
Unit quality: python - jupyter notebook  
Depth: python - jupyter notebook

### analysis

Histology registration to 3D brain: python  
Stimulus responses: python  
Population statistics: MATLAB and python



**Figure 1. Platform for studying distributed visual coding.** **A.**, Immersive visual environment for mouse psychophysics. **B.**, Simultaneous Neuropixels recordings in the mouse visual system, spanning V1, cortical area AL, LGN thalamus, LP thalamus, and superior colliculus. **C.**, 803 single neurons isolated from the above brain areas; anatomy renderings from the Allen Reference Atlas ([brain-map.org](http://brain-map.org)).

# Computing in neuroscience

## An example: code written by neuroscientists for one experiment

planning

Probe trajectory: **MATLAB**

Model prediction: python - jupyter notebook

experimental control

Acquisition hardware: **FPGA gate array programming**

Acquisition software: **C++; python plugin; [Julia]**

Visual stimuli: python script; embedded python in **React**

Video monitoring: python script

“pre-processing”

Spike sorting: **MATLAB**

Unit quality: python - jupyter notebook

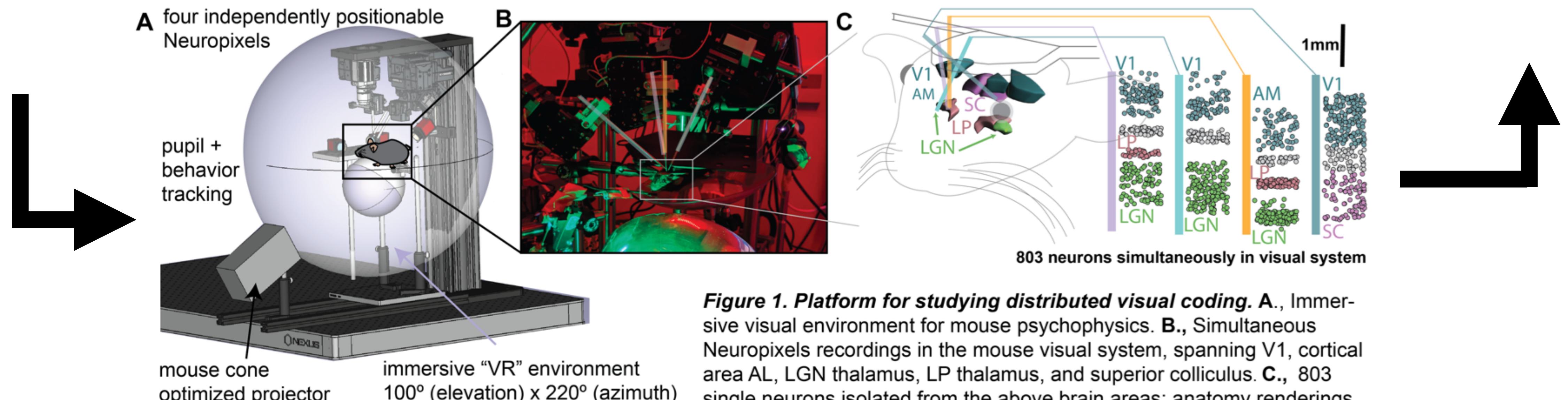
Depth: python - jupyter notebook

analysis

Histology registration to 3D brain: python

Stimulus responses: python

Population statistics: **MATLAB and python**



**Figure 1. Platform for studying distributed visual coding.** A., Immersive visual environment for mouse psychophysics. B., Simultaneous Neuropixels recordings in the mouse visual system, spanning V1, cortical area AL, LGN thalamus, LP thalamus, and superior colliculus. C., 803 single neurons isolated from the above brain areas; anatomy renderings from the Allen Reference Atlas ([brain-map.org](http://brain-map.org)).

# Computing in neuroscience

## An example: code written by neuroscientists for one experiment

planning

Probe trajectory: **MATLAB**

Model prediction: python - jupyter notebook

experimental control

Acquisition hardware: **FPGA gate array programming**

Acquisition software: **C++; python plugin; [Julia]**

Visual stimuli: python script; embedded python in **React**

**Video monitoring: python script**

“pre-processing”

Spike sorting: **MATLAB**

Unit quality: python - jupyter notebook

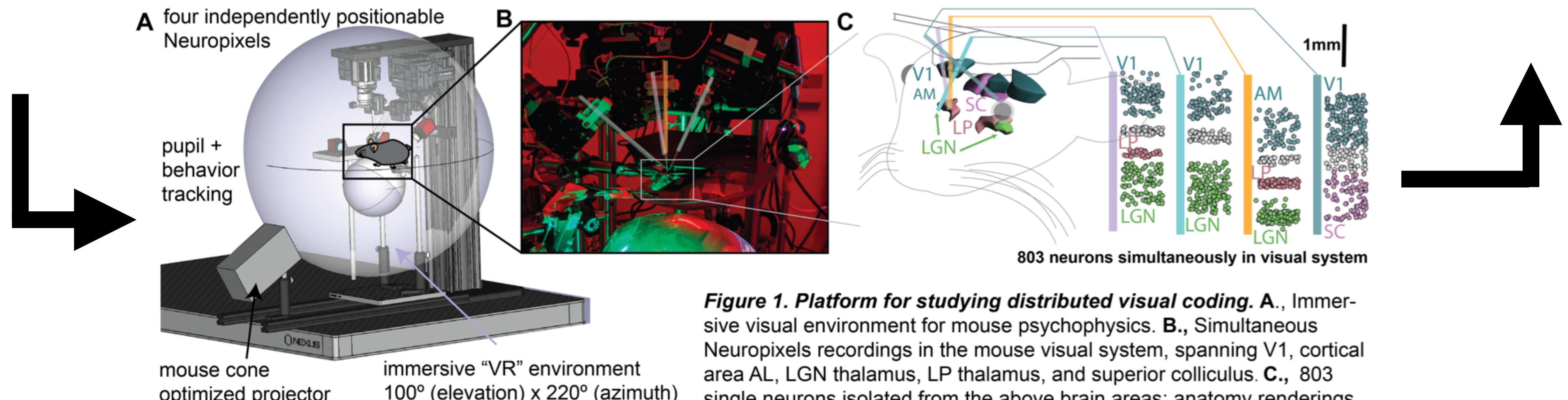
Depth: python - jupyter notebook

analysis

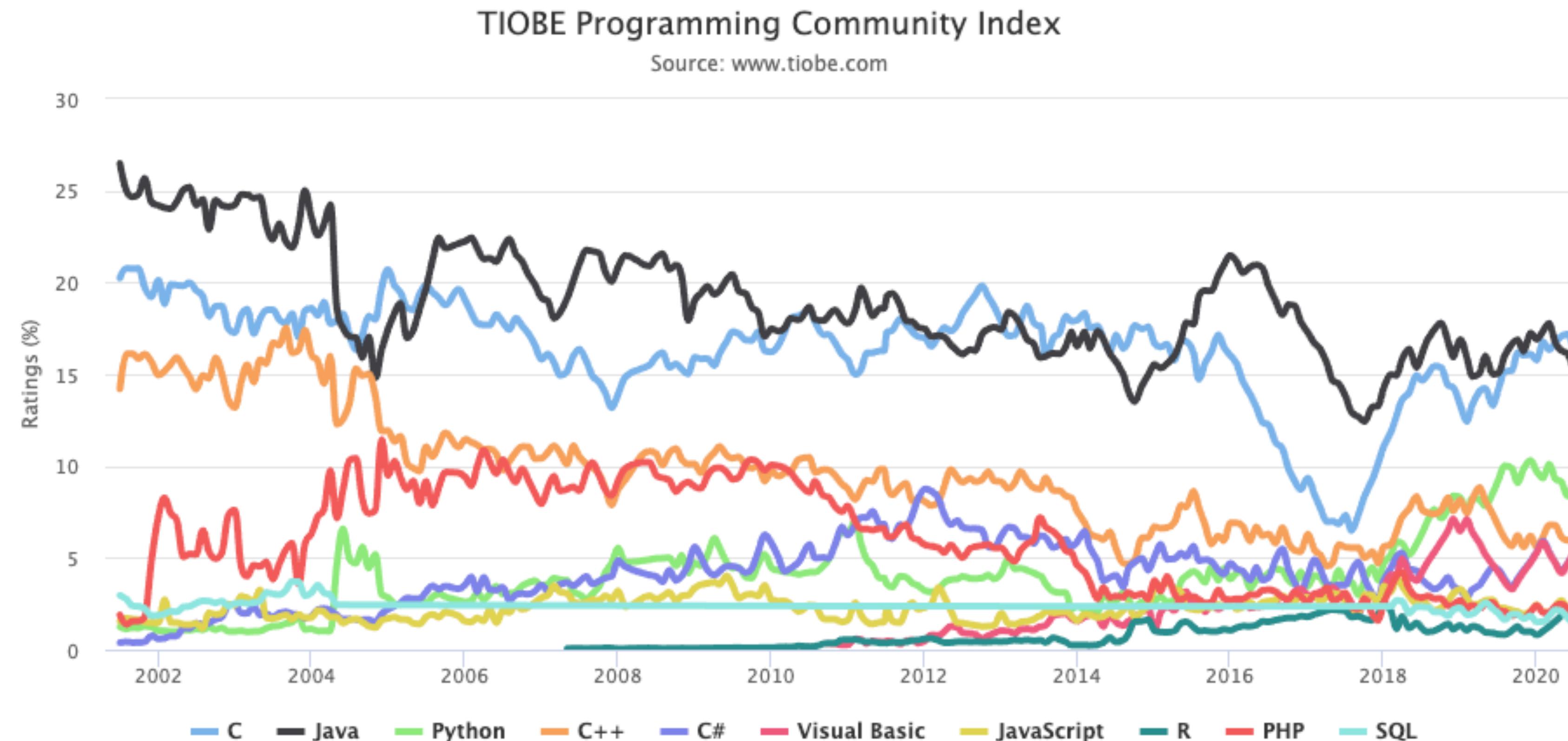
Histology registration to 3D brain: python

Stimulus responses: python

Population statistics: **MATLAB and python**



# history: languages used for neuroscience



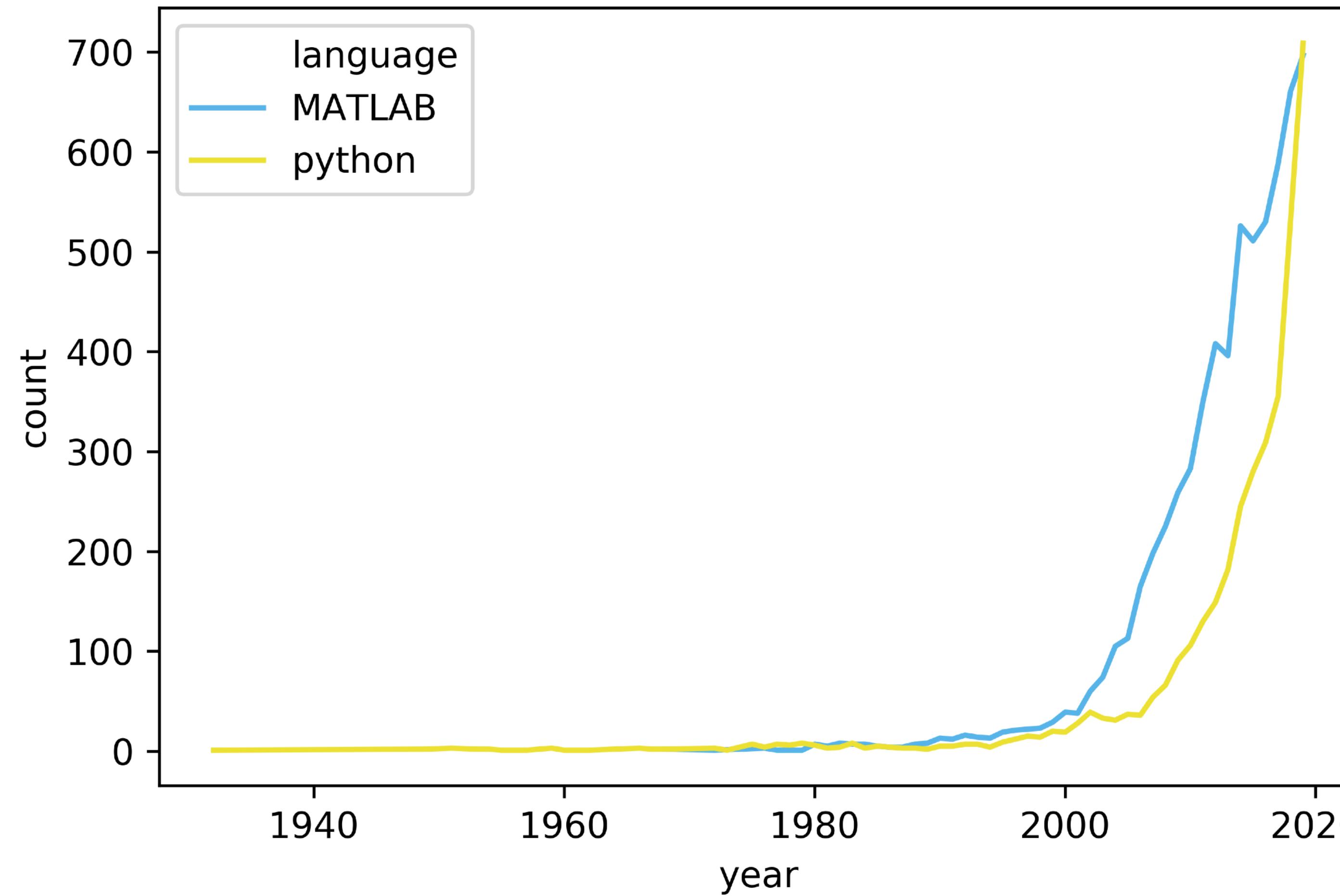
# history: languages used for neuroscience

# 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%

# history: languages used for neuroscience

# history: languages used for neuroscience



# history: languages used for neuroscience

Volume 16 Issue 12, December 2019



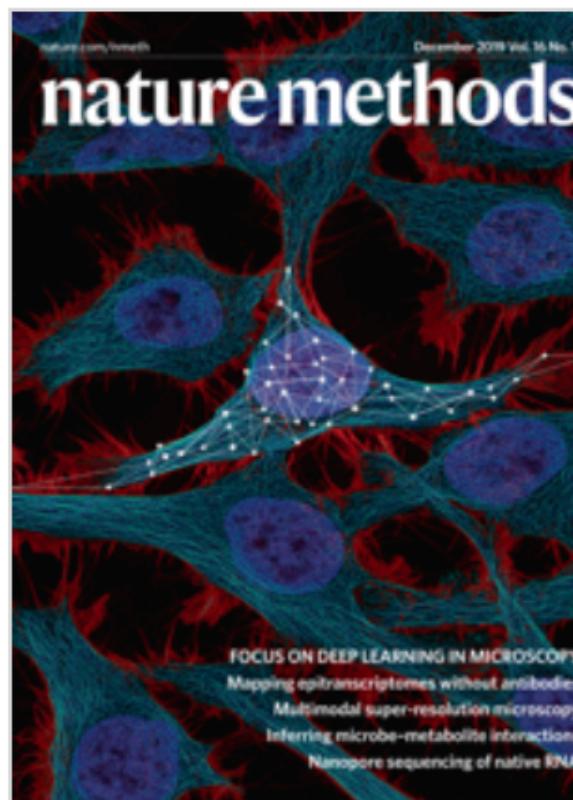
## Focus on Deep Learning in Microscopy

Artwork representing the application of deep learning methods in microscopy.

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

# 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 Haghghi, CherKeng Heng, Tim Becker, Minh Doan, Claire McQuin, Mohammad Rohban, Shantanu Singh & Anne E. Carpenter 

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

3257 Accesses | 1 Citations | 41 Altmetric | [Metrics](#)

2 out top 3 entries used python

# 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

# 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”).

# 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”).

If there is something you are doing over and over again, especially in your analysis, script it!

# overview

**Some plusses**

**Some minuses**

# overview

## Some plusses

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

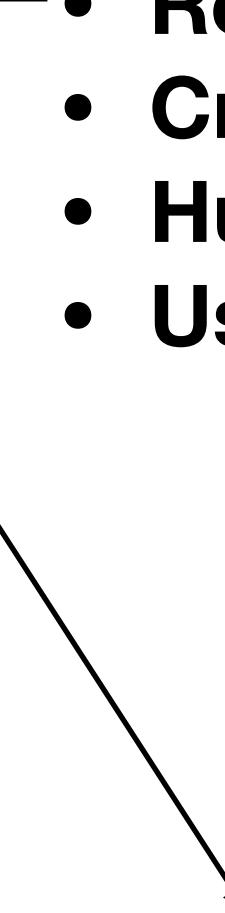
## Some minuses

# overview

## Some plusses

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

## Some minuses



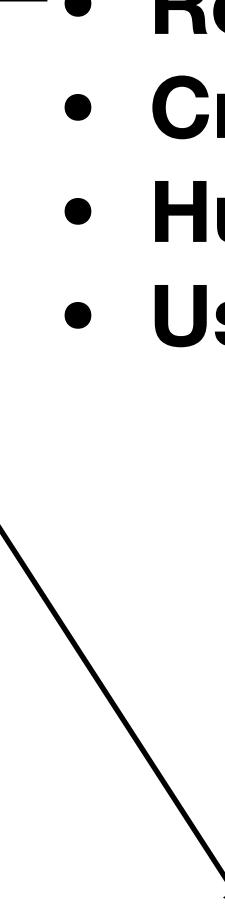
```
a = 1
b = 2
c = 2
if a is not b:
    print('a is not b')
if b is c:
    print('b and c are the same')
```

# 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

```
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
```

## Some minuses

amazon



aws



NETFLIX



You Tube

# 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
```

amazon



aws



NETFLIX



YouTube

# overview

## Some plusses

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

Week 1

```
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
```

## 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



# overview: packages

Find, install and publish Python packages  
with the Python Package Index

Search projects



Or [browse projects](#)

381,490 projects

3,536,531 releases

6,207,207 files

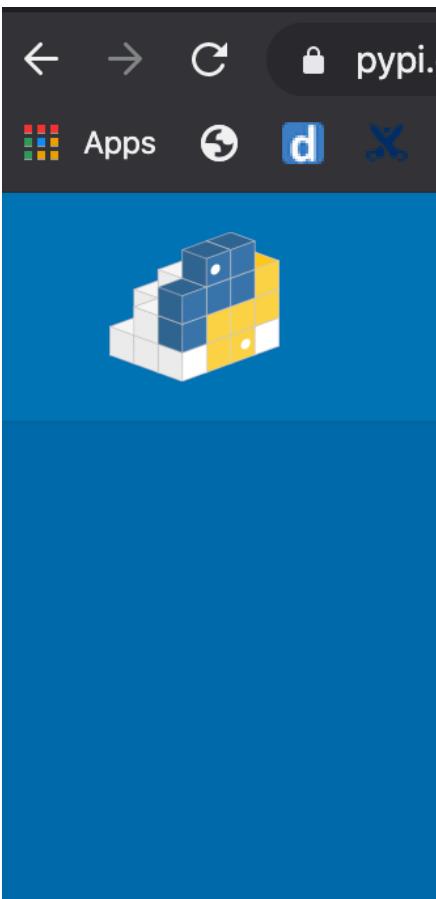
600,160 users

## 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

Find, install and publish Python packages  
with the Python Package Index

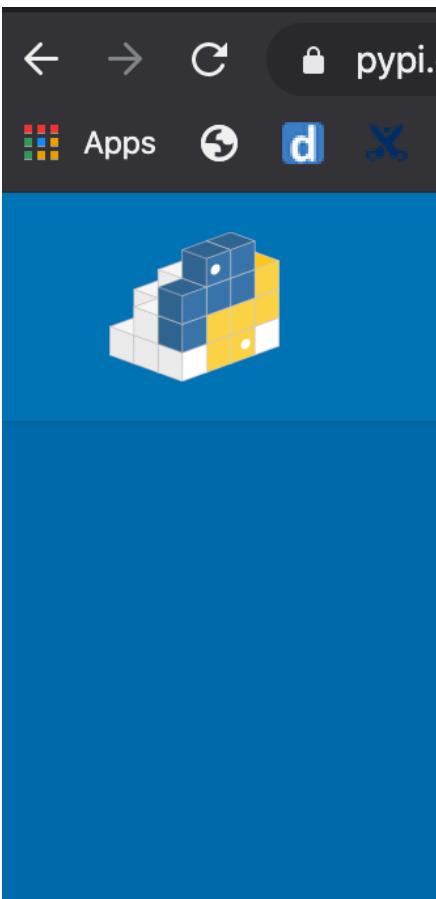
A screenshot of the Python Package Index (PyPI) homepage. At the top, there is a search bar with the placeholder "Search projects" and a magnifying glass icon. Below the search bar, the text "Or [browse projects](#)" is displayed. At the bottom of the page, there is a summary of statistics: "381,490 projects" (which is highlighted with an orange border), "3,536,531 releases", "6,207,207 files", and "600,160 users".

## 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

Week 3

Find, install and publish Python packages with the Python Package Index

Search projects 

Or [browse projects](#)

381,490 projects  3,536,531 releases 6,207,207 files 600,160 users

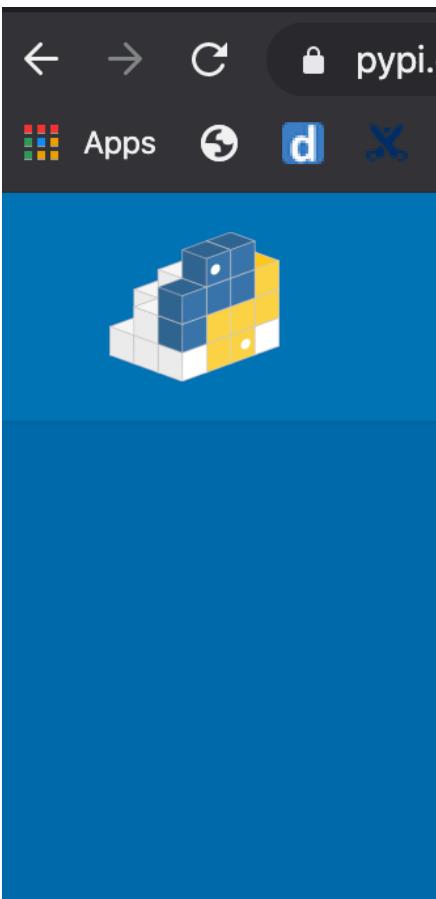


## Packages

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**



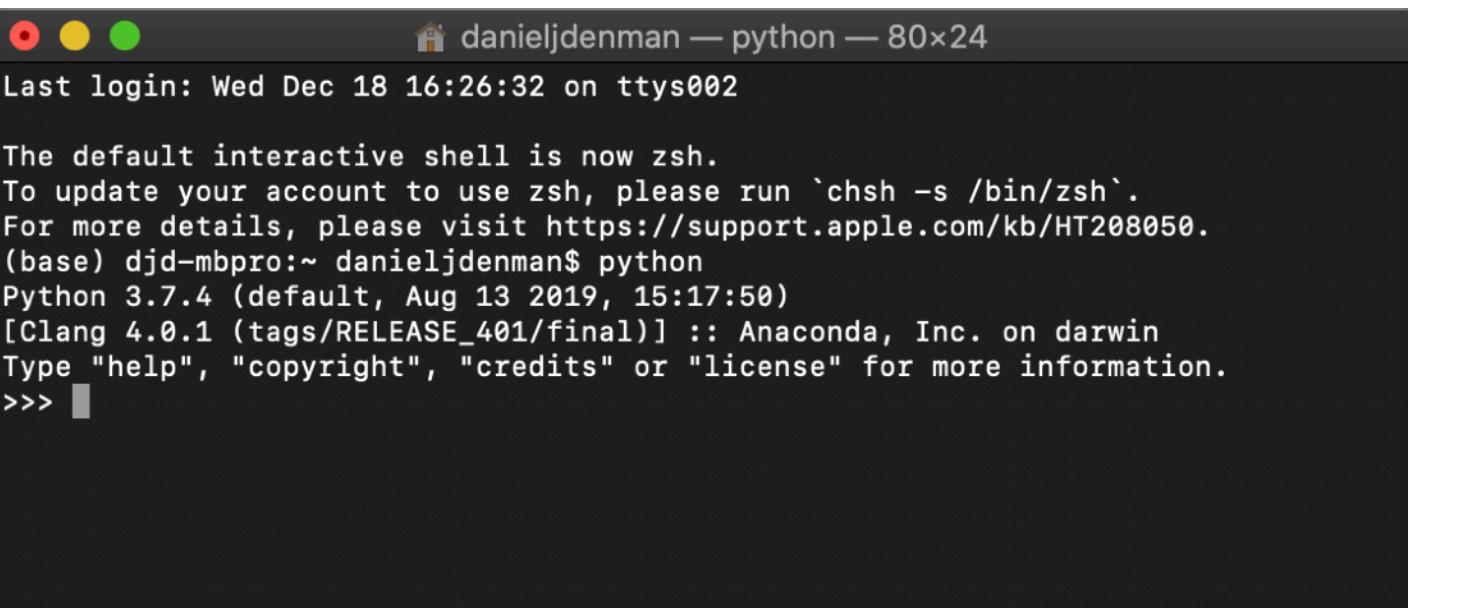


# overview: levels



# overview: levels

## System



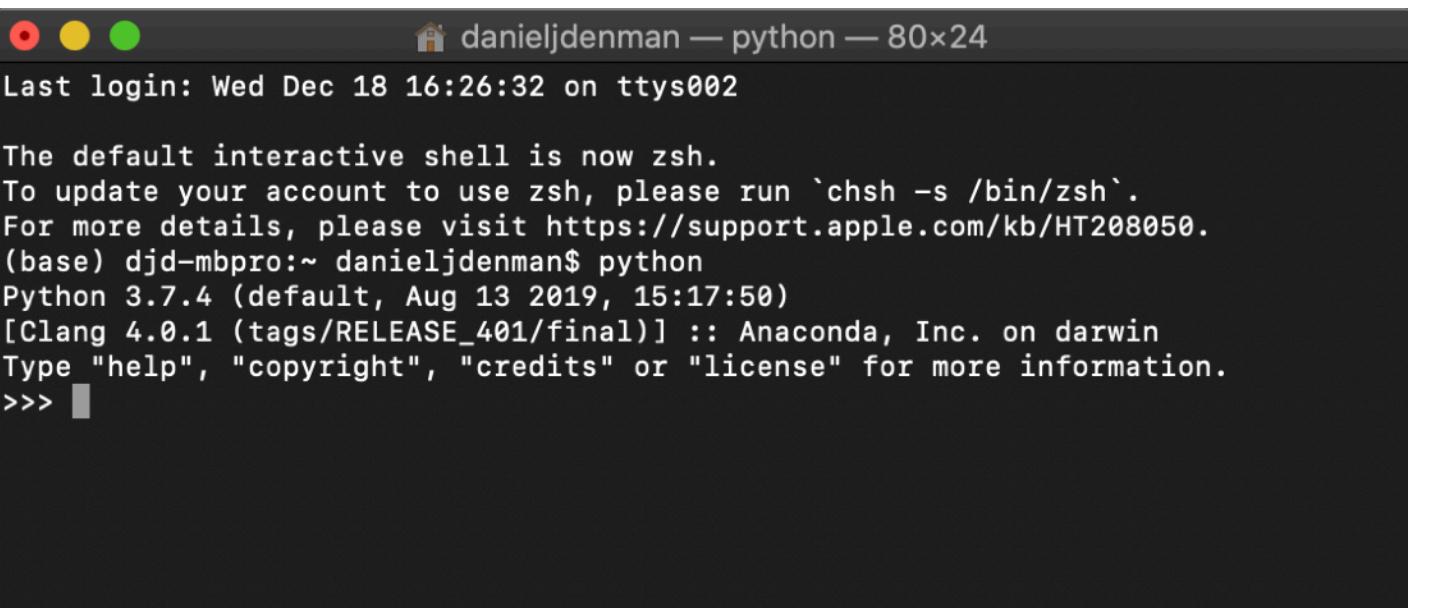
```
>Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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



# overview: levels

## System



A screenshot of a Mac OS X terminal window titled "danieljdenman — python — 80x24". The window shows the following text:

```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

## Package Managers Environments





# overview: levels

## System

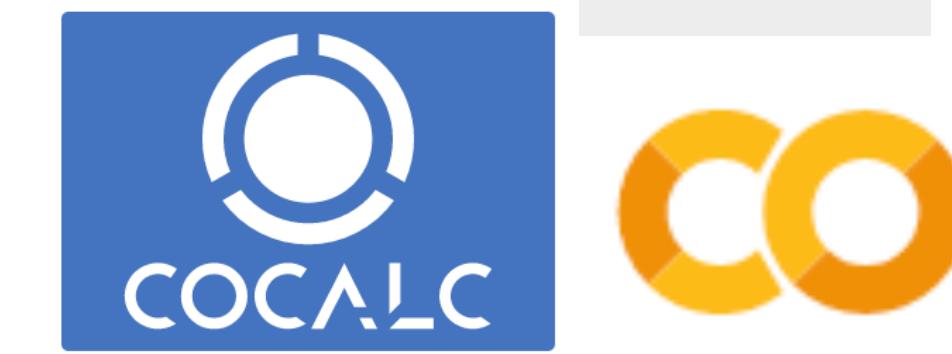
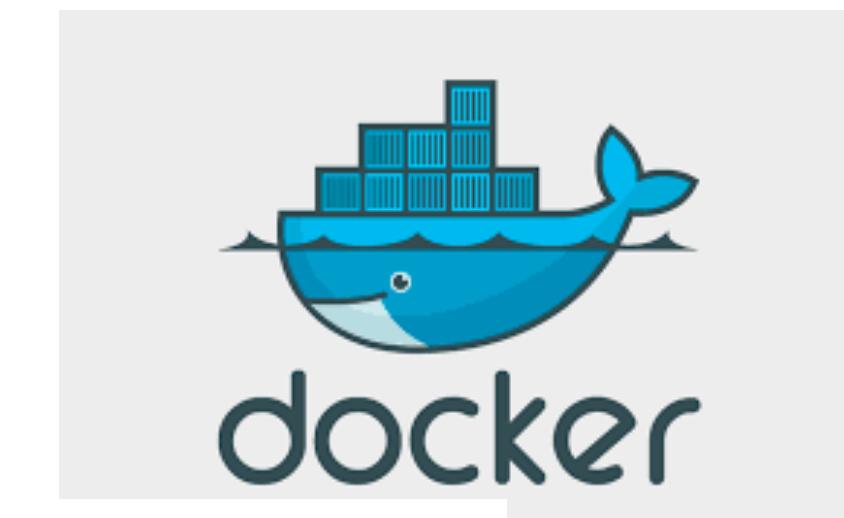
```
>Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

## Package Managers Environments



## Containerized





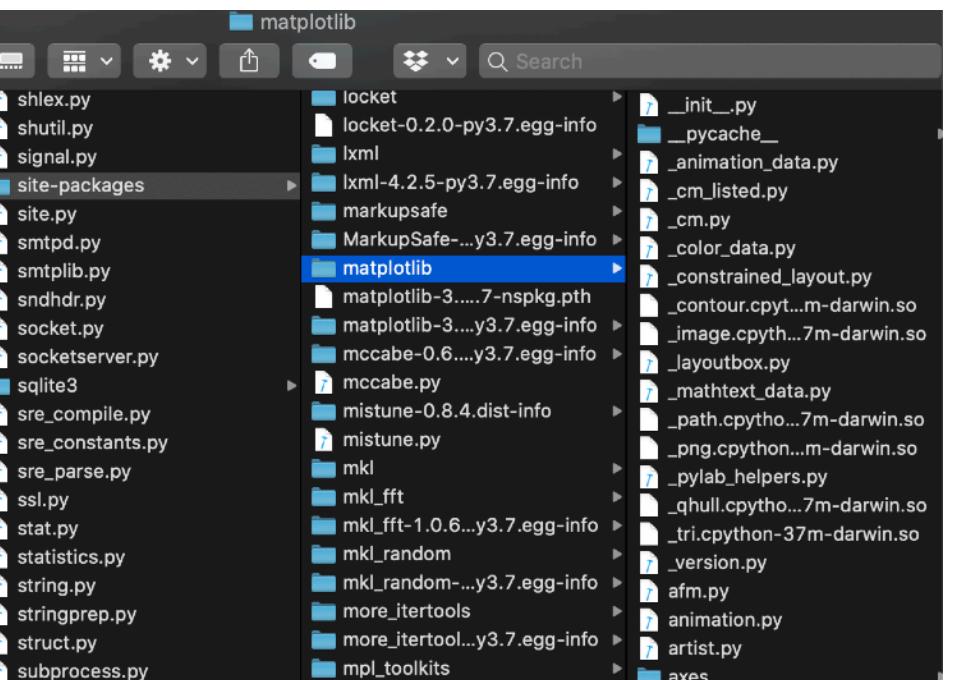
# overview: levels

## System

```
Last login: Wed Dec 18 16:26:32 on ttys002
danieljdenman — python — 80x24
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

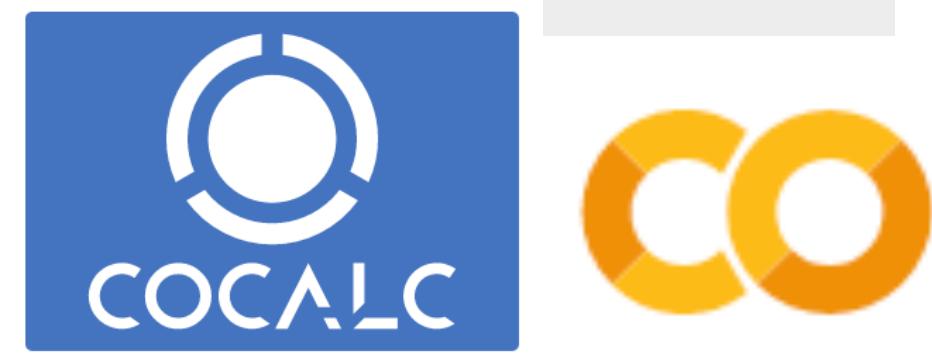
## Packages



## Package Managers Environments



## Containerized





## System

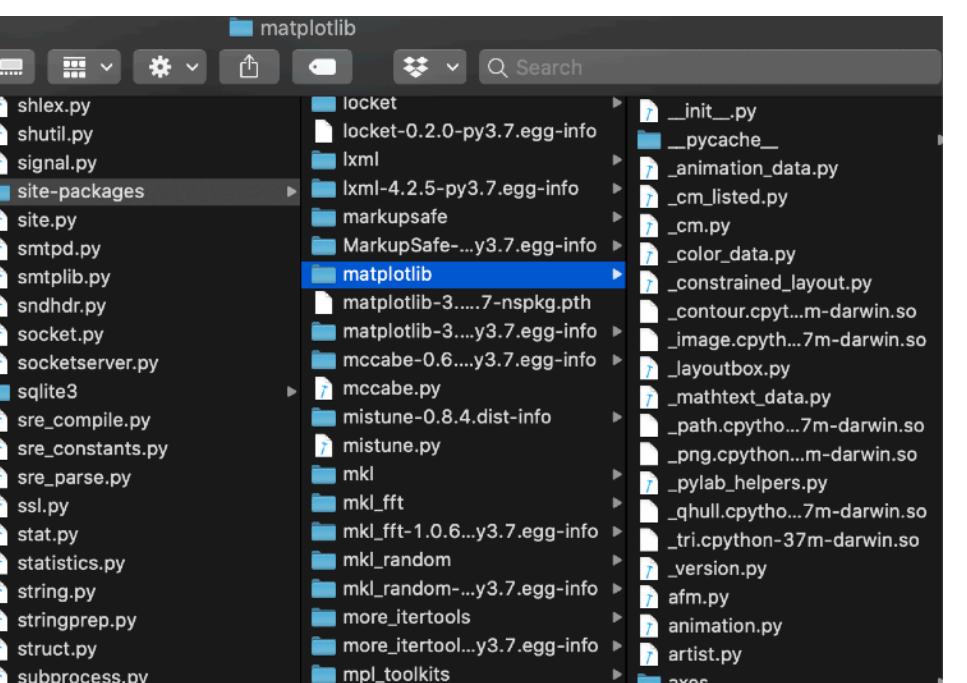
```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

## 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

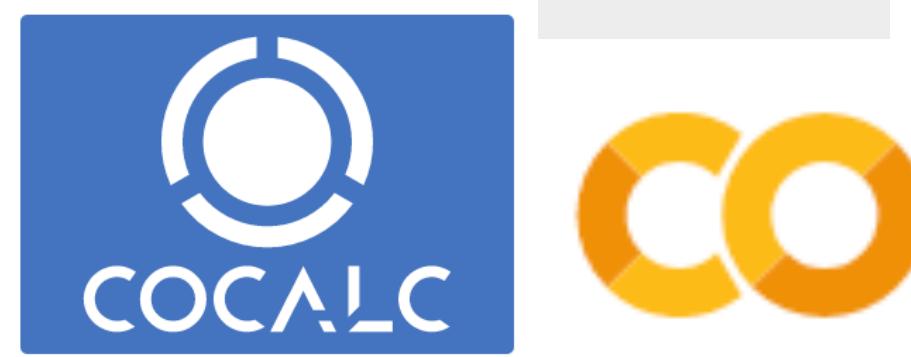


# overview: levels

## Package Managers Environments



## Containerized





## System

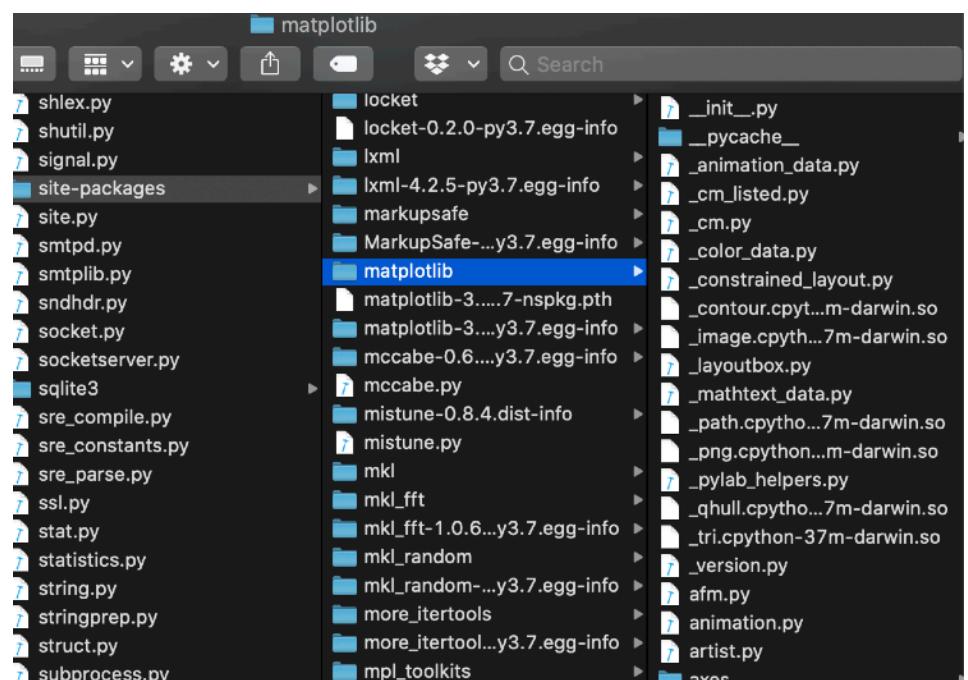
```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

## 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



# overview: levels

## Containerized



## Package Managers Environments



## Notebooks (IPython, Jupyter, Jupyter Lab)

A screenshot of a JupyterLab notebook interface. The browser window title is "JupyterLab" and the URL is "localhost:8888/lab". The notebook tab is "NRC7601\_intro.ipynb". The content of the notebook includes:

- Introduction to Python (for Neuroscientists)**
- a guided tour of data analysis with python**
- 10 Jan 2019**
- NRSC 7601 Systems Neuroscience**
- Daniel J Denman**
- University of Colorado Anschutz**
- Important:** this is not meant to be a comprehensive guide. Use the internet! [Python documentation](#), [Stack Overflow](#), [Google](#), [Markdown cheatsheets \(e.g. this one\)](#) all are your friends.
- Here, we are using a Jupyter notebook environment to run a Python 3.7 kernel
- First, let's get our bearings in a Jupyter notebook
- In a Jupyter notebook, we can iteratively explore data, do computations, make plots, and define functions and objects. The notebook will contain a mix of code, markdown (a simple way to make formatted text) that might explain what is going on in the code, and outputs. The outputs will be in the form of printed statements and plots.
- The fundamental unit of the Jupyter notebook is the cell. Here is an empty code cell:

```
[1]:
```

- You can see the empty brackets on the left; this bracket is empty until the cell is executed
- Cells can be "code", "markdown", or "raw". This cell, for example, is a "markdown cell". When I execute it (by pressing Shift + Enter), it renders the text I have entered.
- In the cell below, a code cell, we will enter some code. To execute it, enter that cell and press Shift + Enter.

```
[13]: message = 'Hello world! Time to do some science' #define a variable. this variable is a string, because we put the value in ''
```

```
print(message)
```

```
Hello world! Time to do some science
```

The empty brackets on the left have now been filled with a number, which is the order in which the cell was executed. This will forever increment until the this bracket is empty until the kernel for Jupyter



## System

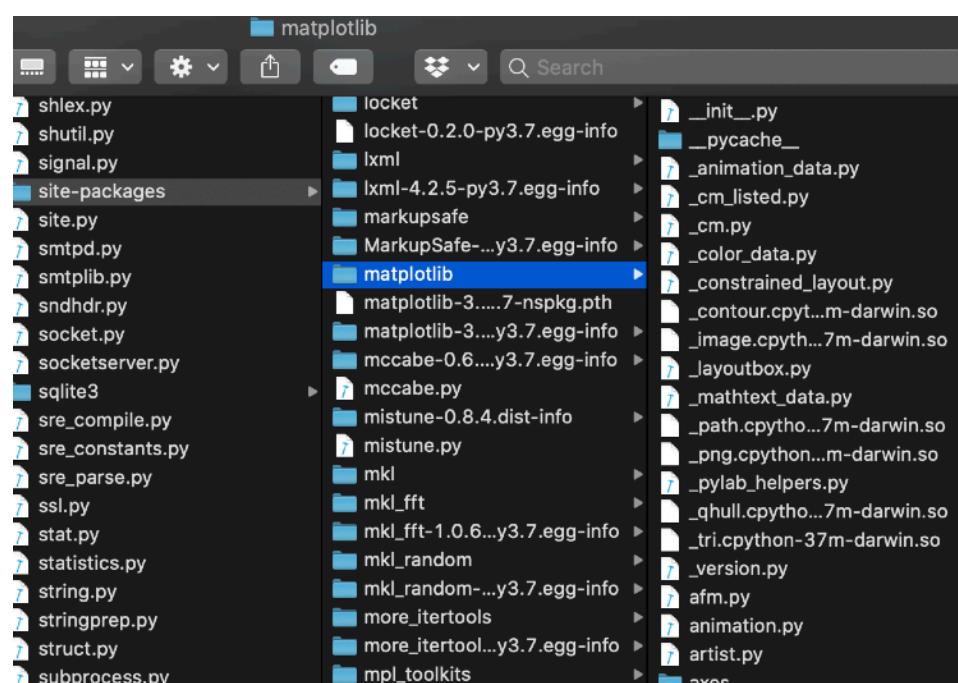
```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

## 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

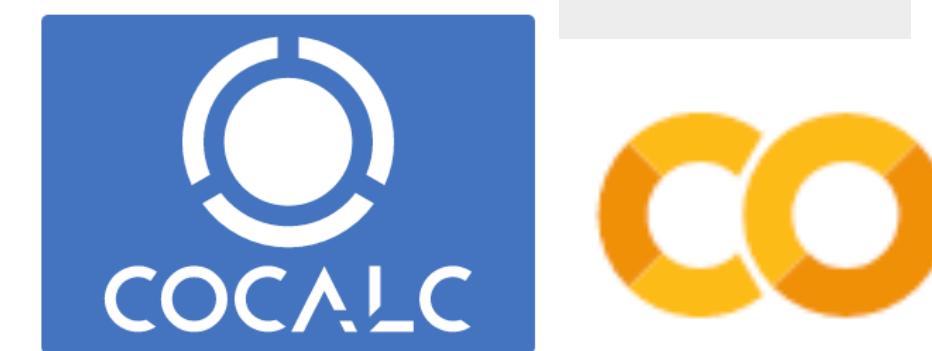


# overview: levels

## Containerized



## Package Managers Environments



**Week 1**

## Notebooks (IPython, Jupyter, Jupyter Lab)

Introduction to Python (for Neuroscientists)  
a guided tour of data analysis with python  
10 Jan 2019  
NRSC 7601 Systems Neuroscience  
Daniel J Denman  
University of Colorado Anschutz

Important: this is not meant to be a comprehensive guide. Use the internet! [Python documentation](#), [Stack Overflow](#), Google, [Markdown cheatsheets](#) (e.g. [this one](#)) all are your friends.

Here, we are using a Jupyter notebook environment to run a Python 3.7 kernel  
First, let's get our bearings in a Jupyter notebook  
In a Jupyter notebook, we can iteratively explore data, do computations, make plots, and define functions and objects. The notebook will contain a mix of code, markdown (a simple way to make formatted text) that might explain what is going on in the code, and outputs. The outputs will be in the form of printed statements and plots.  
The fundamental unit of the Jupyter notebook is the cell. Here is an empty code cell:

```
[1]:
```

- You can see the empty brackets on the left; this bracket is empty until the cell is executed
- Cells can be "code", "markdown", or "raw". This cell, for example, is a "markdown cell". When I execute it (by pressing Shift + Enter), it renders the text I have entered.
- In the cell below, a code cell, we will enter some code. To execute it, enter that cell and press Shift + Enter.

```
[13]: message = 'Hello world! Time to do some science' #define a variable. this variable is a string, because we put the value in ''  
print(message)  
Hello world! Time to do some science
```

The empty brackets on the left has now been filled with a number, which is the order in which the cell was executed. This will forever increment until the this bracket is empty until the kernel for Jupyter



## System



```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

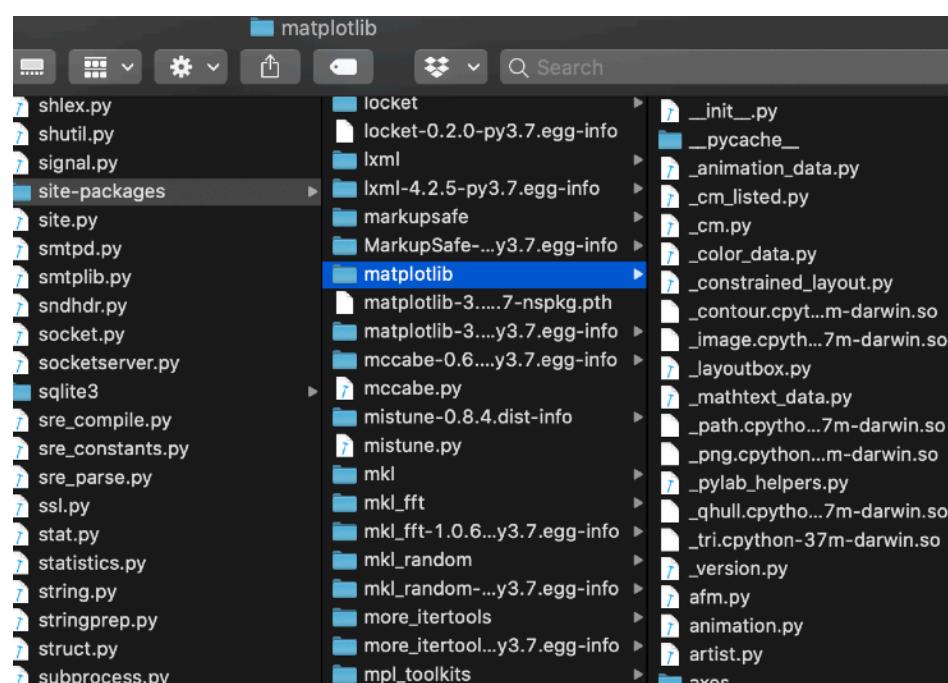
Native in Mac OS X, Linux; in Windows store (free)

## 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



# overview: levels

## Package Managers Environments



## Containerized



Week 6

## Notebooks (IPython, Jupyter, Jupyter Lab)

Introduction to Python (for Neuroscientists)  
a guided tour of data analysis with python  
10 Jan 2019  
NRSC 7601 Systems Neuroscience  
Daniel J Denman  
University of Colorado Anschutz

Important: this is not meant to be a comprehensive guide. Use the internet! [Python documentation](#), [Stack Overflow](#), Google, [Markdown cheatsheets](#) (e.g. [this one](#)) all are your friends.

Here, we are using a Jupyter notebook environment to run a Python 3.7 kernel  
First, let's get our bearings in a Jupyter notebook  
In a Jupyter notebook, we can iteratively explore data, do computations, make plots, and define functions and objects. The notebook will contain a mix of code, markdown (a simple way to make formatted text) that might explain what is going on in the code, and outputs. The outputs will be in the form of printed statements and plots.  
The fundamental unit of the Jupyter notebook is the cell. Here is an empty code cell:

```
[1]:
```

- You can see the empty brackets on the left; this bracket is empty until the cell is executed
- Cells can be "code", "markdown", or "raw". This cell, for example, is a "markdown cell". When I execute it (by pressing Shift + Enter), it renders the text I have entered.
- In the cell below, a code cell, we will enter some code. To execute it, enter that cell and press Shift + Enter.

```
[13]: message = 'Hello world! Time to do some science' #define a variable. this variable is a string, because we put the value in '
print(message)
Hello world! Time to do some science
```

The empty brackets on the left has now been filled with a number, which is the order in which the cell was executed. This will forever increment until the this bracket is empty until the kernel for Jupyter



## System



```
Last login: Wed Dec 18 16:26:32 on ttys002
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) djd-mbpro:~ danieljdenman$ python
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

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

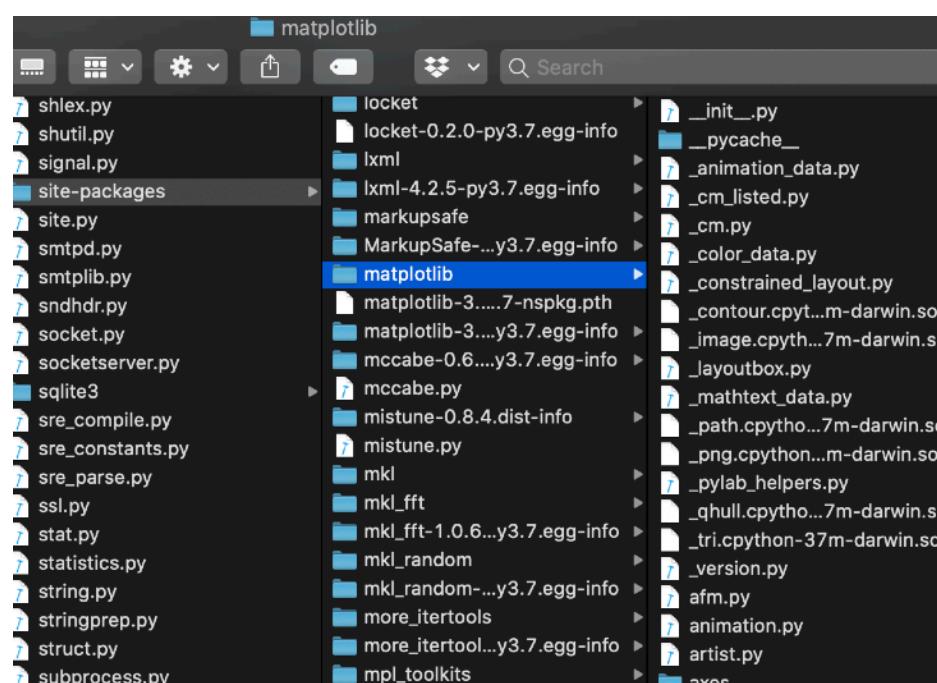
## Scripts



```
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.
```

**Week 3**

## Packages



# overview: levels

## Package Managers Environments



## Containerized



**Week 6**

## Notebooks (IPython, Jupyter, Jupyter Lab)

Introduction to Python (for Neuroscientists)  
a guided tour of data analysis with python  
10 Jan 2019  
NRSC 7601 Systems Neuroscience  
Daniel J Denman  
University of Colorado Anschutz

Important: this is not meant to be a comprehensive guide. Use the internet! [Python documentation](#), [Stack Overflow](#), Google, [Markdown cheatsheets](#) (e.g. [this one](#)) all are your friends.

Here, we are using a Jupyter notebook environment to run a Python 3.7 kernel  
First, let's get our bearings in a Jupyter notebook  
In a Jupyter notebook, we can iteratively explore data, do computations, make plots, and define functions and objects. The notebook will contain a mix of code, markdown (a simple way to make formatted text) that might explain what is going on in the code, and outputs. The outputs will be in the form of printed statements and plots.  
The fundamental unit of the Jupyter notebook is the cell. Here is an empty code cell:

```
[1]:
```

- You can see the empty brackets on the left; this bracket is empty until the cell is executed
- Cells can be "code", "markdown", or "raw". This cell, for example, is a "markdown cell". When I execute it (by pressing Shift + Enter), it renders the text I have entered.
- In the cell below, a code cell, we will enter some code. To execute it, enter that cell and press Shift + Enter.

```
[1]: message = 'Hello world! Time to do some science' #define a variable. this variable is a string, because we put the value in ''  
print(message)  
Hello world! Time to do some science
```

The empty brackets on the left has now been filled with a number, which is the order in which the cell was executed. This will forever increment until the this bracket is empty until the kernel for Jupyter

# why is it good for doing neuroscience?

# why is it good for doing neuroscience?

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

# why is it good for doing neuroscience?

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

packages!

**Stack Overflow <— not cheating!**

# why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implement 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 implement 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

RaspberryPi

Arduino

PyDAQMX

PsychoPy

...many APIs...

# why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implement 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

RaspberryPi  
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 implement 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

RaspberryPi  
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 implement 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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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



## Data science

scikit-learn  
Pandas  
TensorFlow

# why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implement 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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Data science

scikit-learn  
Pandas  
TensorFlow



## Specialized Tools

Image analysis: 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 implement 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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

## Data science

scikit-learn  
Pandas  
TensorFlow



## Sharing

Docker  
Google Colab  
Jupyter  
[it is free!]



# why is it good for doing neuroscience?

- Do analyses that would be a whole PhD to implement 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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

...

## Data science

scikit-learn  
Pandas  
TensorFlow



**Week 2**

## Sharing

Docker  
Google Colab  
Jupyter  
[it is free!]



# why is it good for doing neuroscience?

Week 2  
packages

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

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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

## Data science

scikit-learn  
Pandas  
TensorFlow



Week 2

## Sharing

Docker  
Google Colab  
Jupyter  
[it is free!]



# why is it good for doing neuroscience?

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

Week 2  
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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

## Data science

scikit-learn  
Pandas  
TensorFlow



Week 2

Sharing  
Docker  
Google Colab  
Jupyter  
[it is free!]



Week 5

# why is it good for doing neuroscience?

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

Week 2  
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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

## Data science

scikit-learn  
Pandas  
TensorFlow



Week 2

Sharing  
Docker  
Google Colab  
Jupyter  
[it is free!]



Week 5  
Week 6

# why is it good for doing neuroscience?

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

Week 2  
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

RaspberryPi  
Arduino  
PyDAQMX  
PsychoPy  
...many APIs...



## Specialized Tools

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

...

Weeks 7-9

## Data science

scikit-learn  
Pandas  
TensorFlow



Week 2

## Sharing

Docker  
Google Colab  
Jupyter  
[it is free!]



Week 5  
Week 6

# MATLAB

- At this time, some understanding of both Python and MATLAB is extremely useful. We use both in the Denman Lab, but my primary expertise is in Python
- John T going to be MATLAB point person for the course

# MATLAB

- At this time, some understanding of both Python and MATLAB is extremely useful. We use both in the Denman Lab, but my primary expertise is in Python
- John T going to be MATLAB point person for the course

## Week 1

- **Anaconda environments and where python is on your computer**
- **Command line: git, starting a jupyter notebook, running a script**
- **Basics and syntax**

# Style and philosophy

- **Don't:** overoptimize; get it done first
- **Do:** document as you go
- Pay attention to variable names
- There is a slippery slope from the minimal functional code to unusable/unshareable code
- Read this PLoS paper this week!

PLOS COMPUTATIONAL BIOLOGY

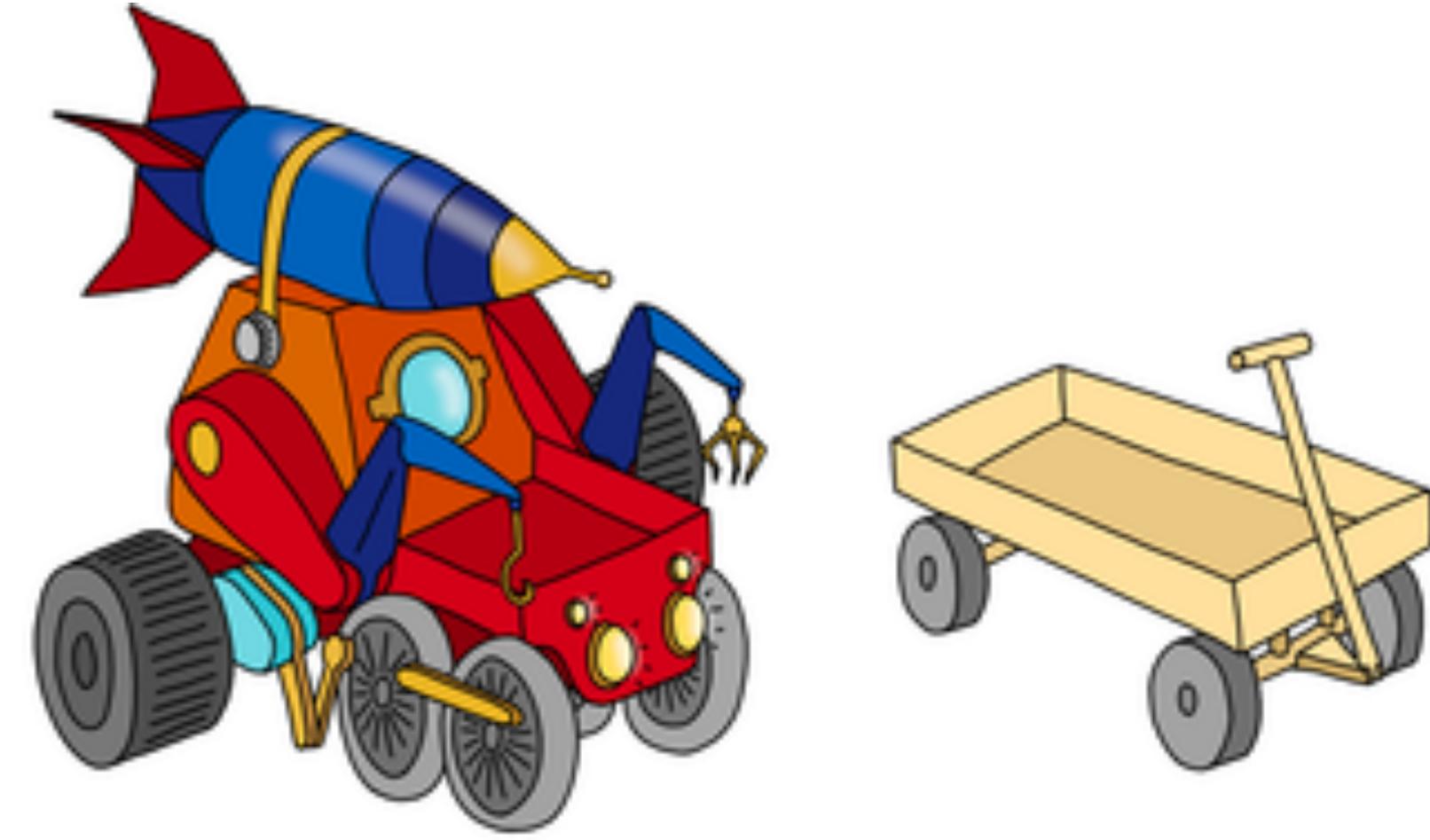
OPEN ACCESS

EDITORIAL

Ten simple rules for quick and dirty scientific programming

Gabriel Balaban, Ivar Grytten, Knut Dagestad Rand, Lonneke Scheffer, Geir Kjetil Sandve 

Published: March 11, 2021 • <https://doi.org/10.1371/journal.pcbi.1008549>



[https://journals.plos.org/ploscompbiol/  
article?id=10.1371/journal.pcbi.1008549](https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008549)

# 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 can use Anaconda to do (even if you are using MATLAB only for your project)
- Command-line is great, but if you are new to git we recommend using [GitHub Desktop](#)
- **We're going to go over some git interactively to get course materials today.**