

# Welcome to **BILD62**

Dr. Juavinett  
*jah-vah-nett*  
(or, Dr. J)



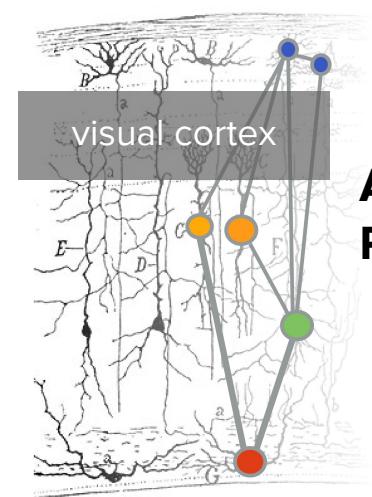
Image: [garetsworkshop/Shutterstock](#)

# Objectives for today

- Introduce the teaching staff, students, and class
  - Motivate learning how to code as a biology student
  - Discuss course logistics, expectations, & tools
  - Start coding!
-

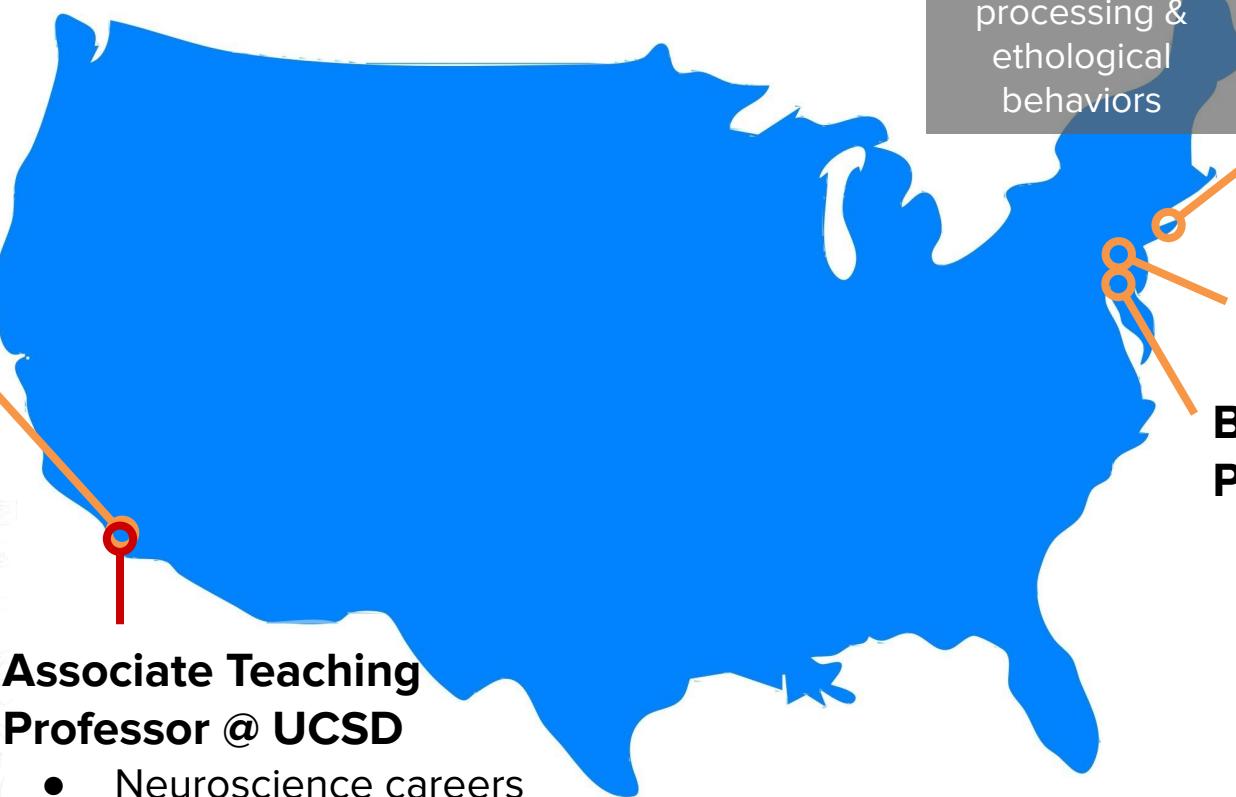


PhD in  
Neuroscience  
@ UCSD



## Associate Teaching Professor @ UCSD

- Neuroscience careers & education
- Open-source data

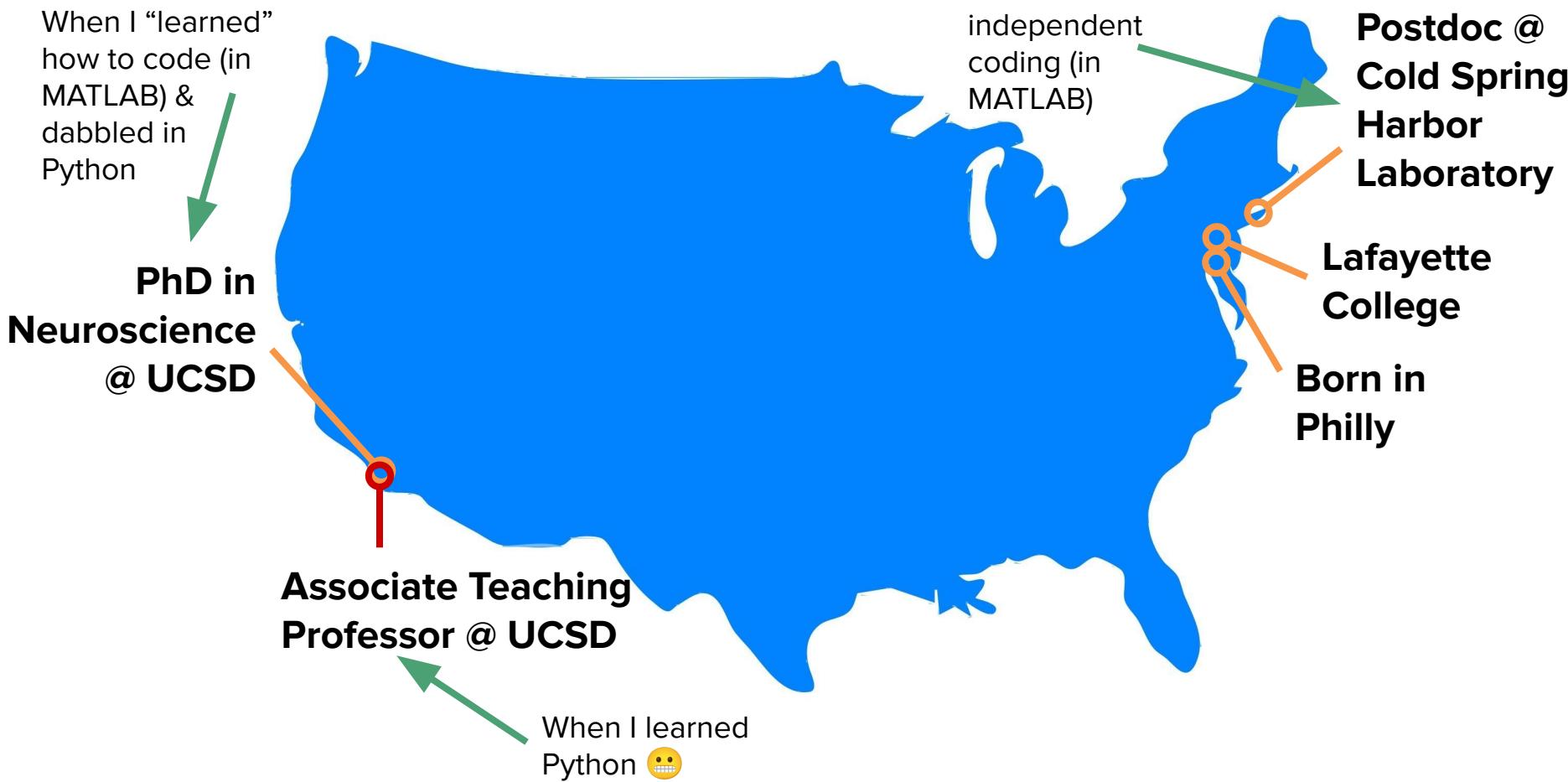


multisensory  
processing &  
ethological  
behaviors

**Postdoc @  
Cold Spring  
Harbor  
Laboratory**

Lafayette  
College

Born in  
Philly



# Introduction to our teaching staff!



Felix Taschbach  
[ftaschbach@ucsd.edu](mailto:ftaschbach@ucsd.edu)  
<https://felix.phd/>



Ber Al Jaibaji  
[bmaljaibaji@ucsd.edu](mailto:bmaljaibaji@ucsd.edu)

Let's be human,  
for just a second.

With the folks next to you,  
share:

- Your name, major, and preferred pronouns
- Something that brought you joy over the break
- Why you're taking this course



# Objectives for today

- Introduce the teaching staff, students, and class
  - **Motivate learning how to code as a biology student**
  - Discuss course logistics, expectations, & tools
-

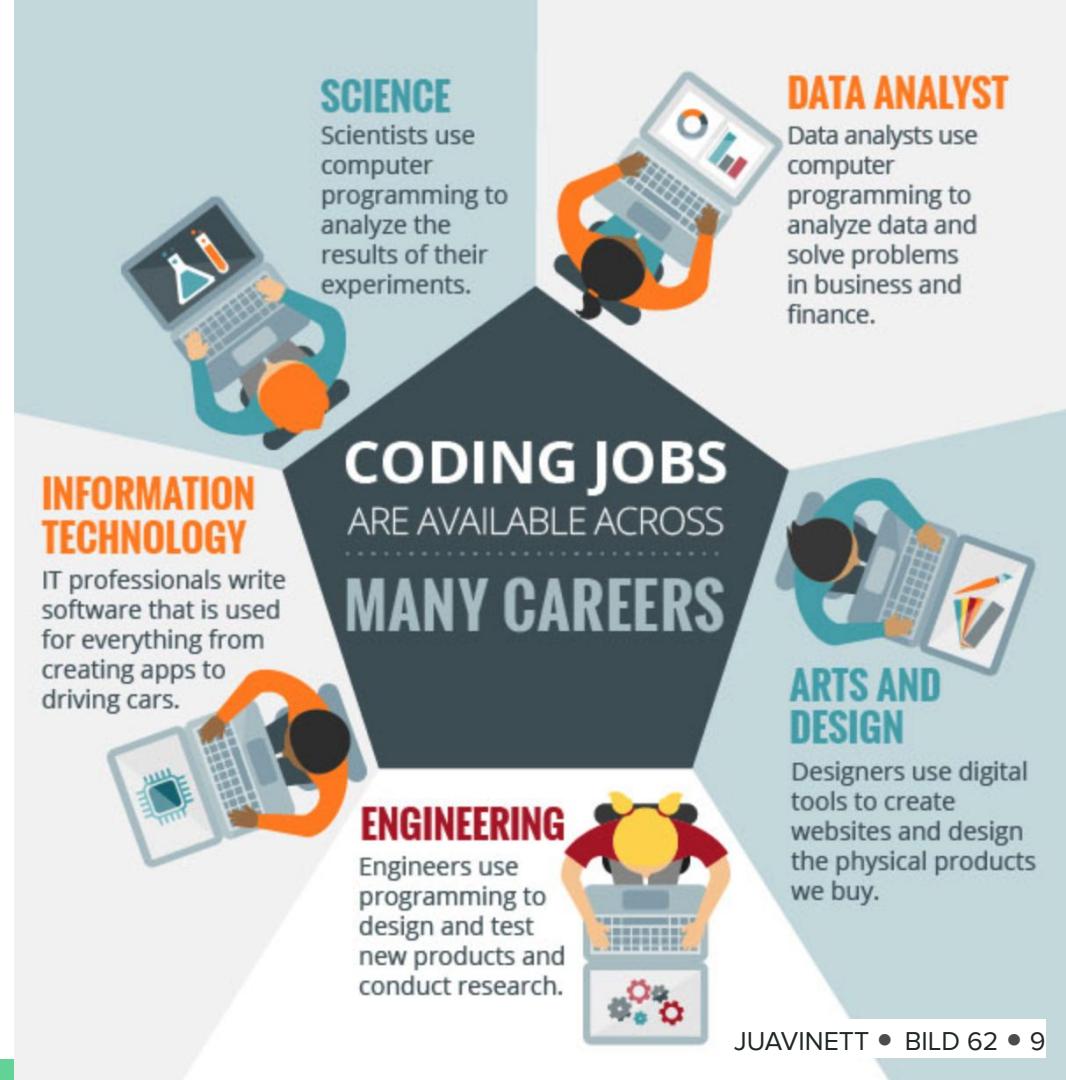
What does coding have to  
do with *biology*?

Why *you*, right now?



# Why should I learn how to code?

- Coding is useful for:
  - Data acquisition (controlling hardware, image acquisition, etc)
  - Data analysis & visualization
  - Computational modeling
- Beyond research, there are more and more jobs for software engineers, and they pay well  
(see report by Burning Glass:  
<https://www.burning-glass.com/research-project/coding-skills/>)

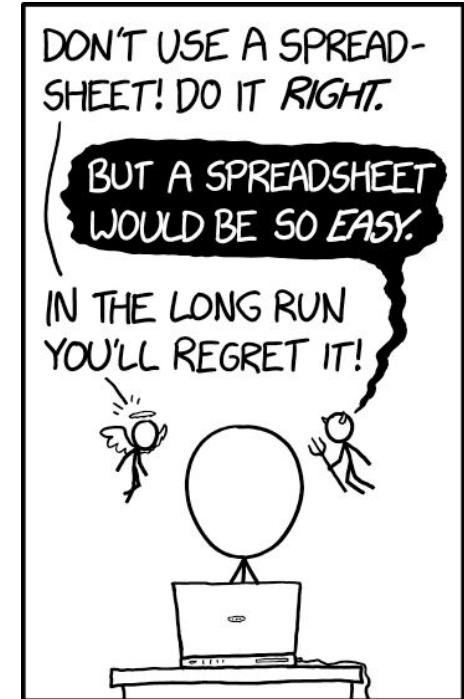


Excel can only handle datasets with **“1 million rows, and 16,000 columns** — many datasets in biology are much larger than this!

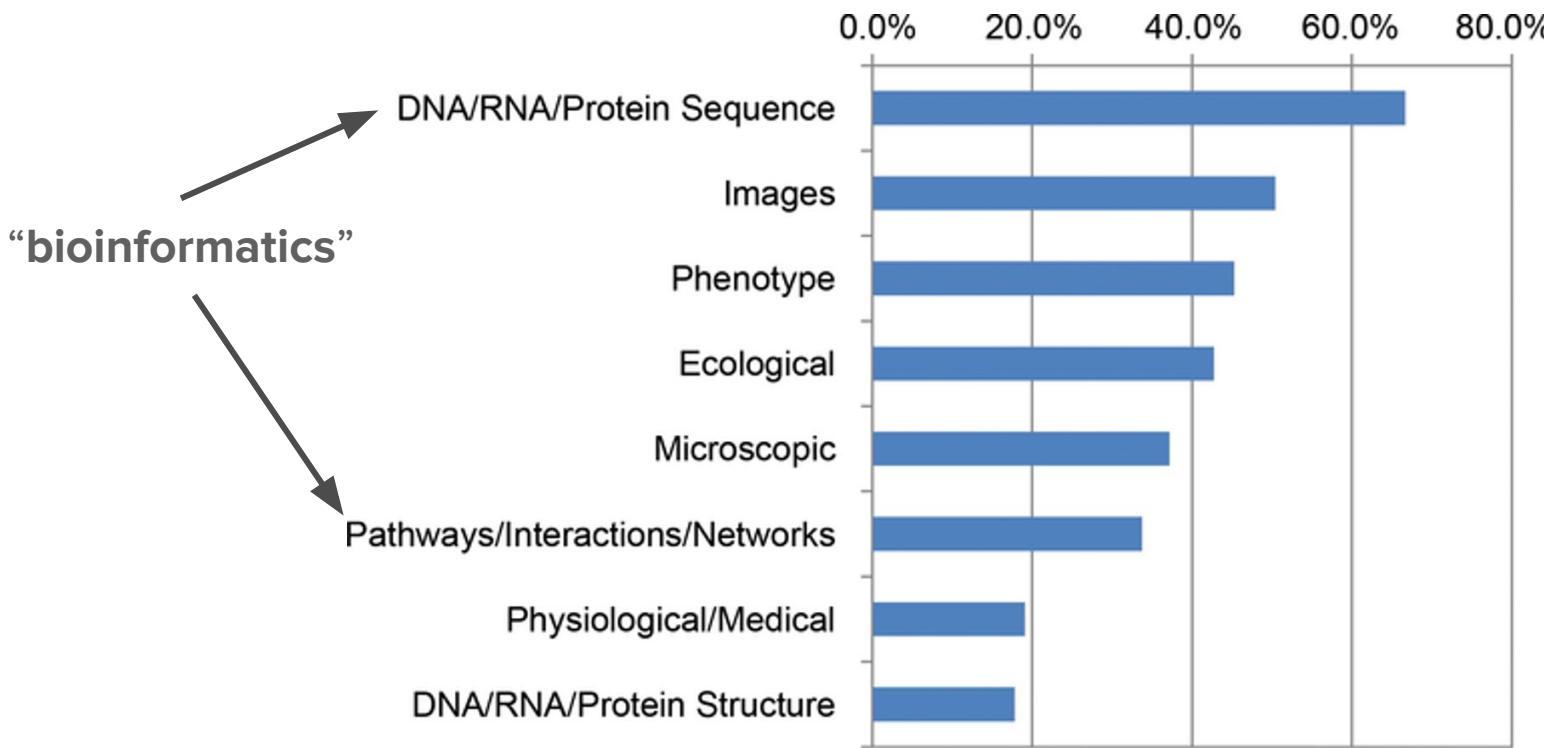
You can automate analyses in Excel, but this is quite limited.

There are also specialized biological data analysis software programs, but often these are limited in how much they can be customized.

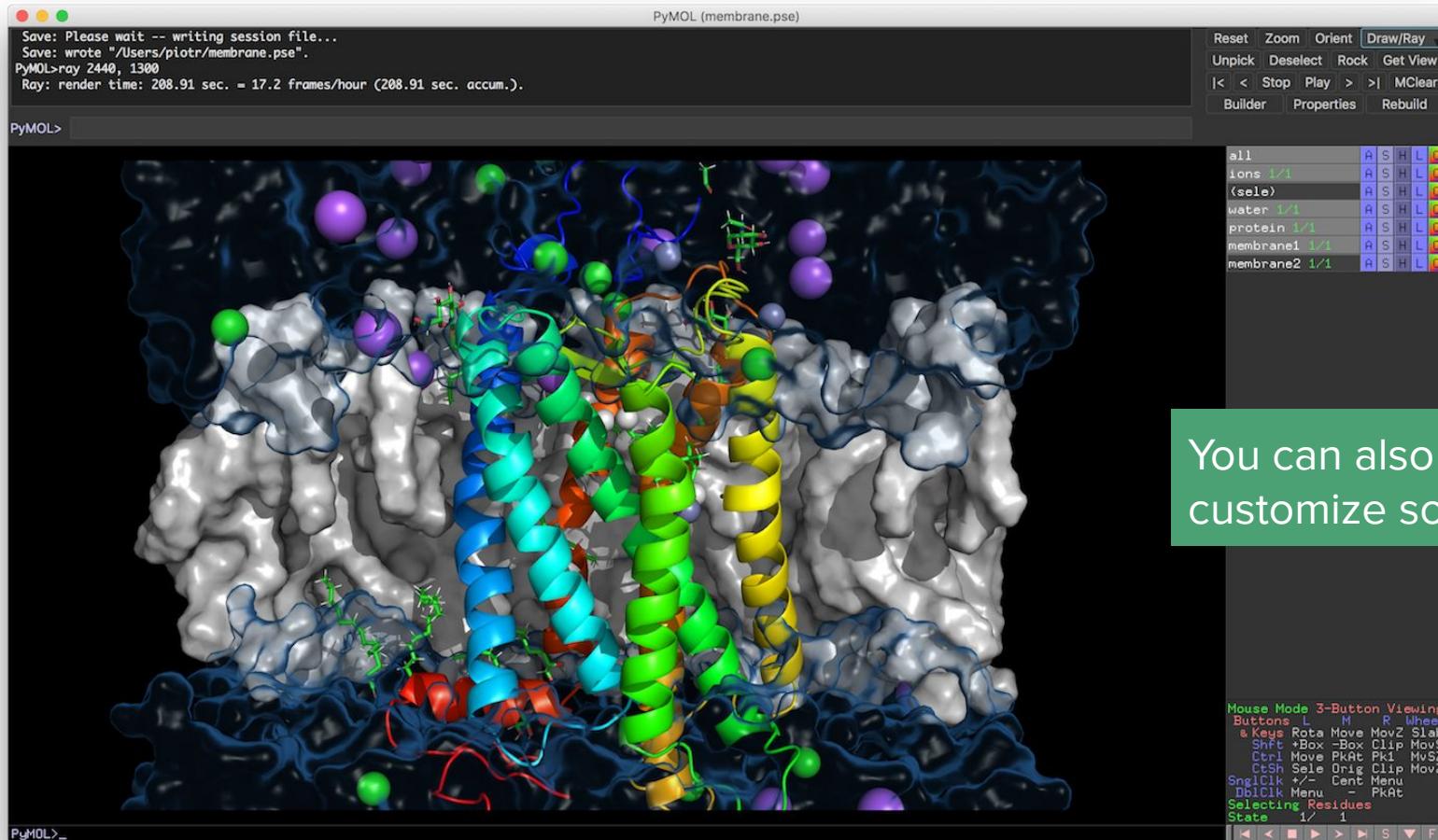
Code is *infinitely* customizable.



<https://xkcd.com/2180/>



Major data types used by National Science Foundation (NSF)  
Biological Sciences Directorate principal investigators (PIs).



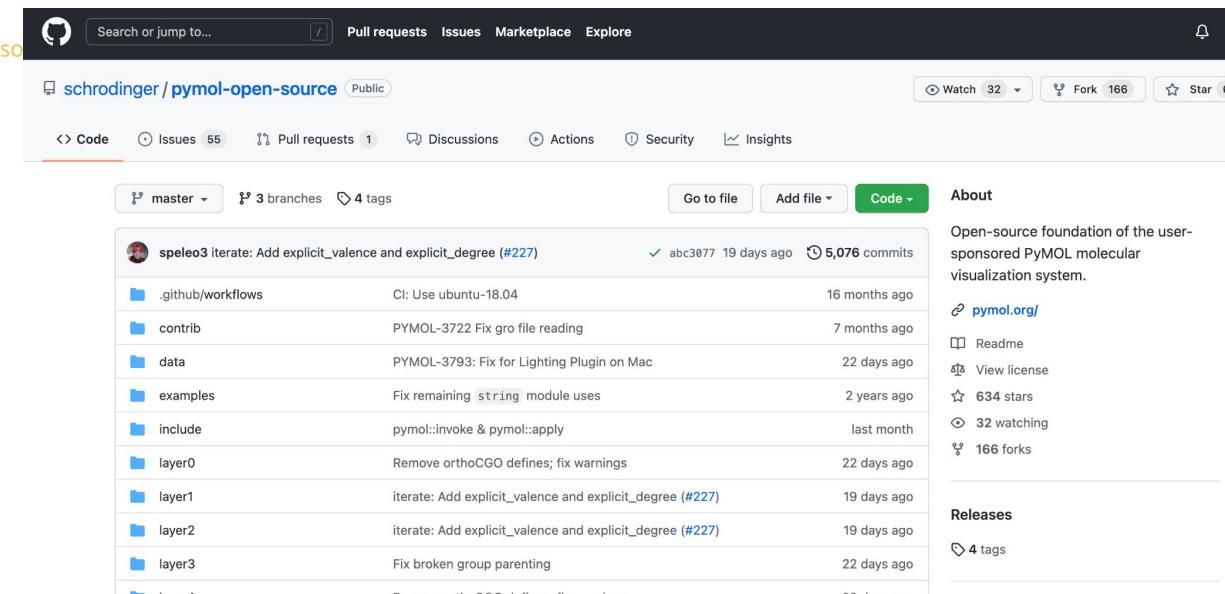
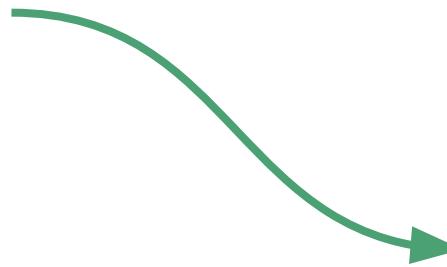
# Open-Source Philosophy

PyMOL is a commercial product, but we make most of its source code freely available under a permissive license. The open source project is maintained by [Schrödinger](#) and ultimately funded by everyone who purchases a PyMOL license.

Open source enables open science.  
This was the vision of the original PyMOL author Warren L. DeLano.

[Visit the Open-Source Project](#)

[Become a sponsor](#)



The screenshot shows the GitHub repository page for `schrodinger/pymol-open-source`. The repository is public and has 5,076 commits. The main page displays a list of recent commits, including:

- speleo3 iterate: Add explicit\_valence and explicit\_degree (#227) - 19 days ago
- .github/workflows CI: Use ubuntu-18.04 - 16 months ago
- contrib PYMOL-3722 Fix gro file reading - 7 months ago
- data PYMOL-3793: Fix for Lighting Plugin on Mac - 22 days ago
- examples Fix remaining string module uses - 2 years ago
- include pymol::invoke & pymol::apply - last month
- layer0 Remove orthoCGO defines; fix warnings - 22 days ago
- layer1 iterate: Add explicit\_valence and explicit\_degree (#227) - 19 days ago
- layer2 iterate: Add explicit\_valence and explicit\_degree (#227) - 19 days ago
- layer3 Fix broken group parenting - 22 days ago

The repository has 32 watchers, 166 forks, and 634 stars. It includes sections for About, Releases, and a list of 4 tags.

AND many software packages for biologists can be modified... if you know how to code!

**By taking this class, you're ahead of the game!**

Many researchers learn to code really informally, and relatively late in their careers



**ashley, ahem, dr. juavinett**  
@analog\_ashley



Neuroscientists of Twitter, when did you learn\* how to code?

\*Let's say, when you felt reasonably capable writing your own simple code (e.g. reading data and plotting, or communicating with an Arduino)

19% High school or earlier

30% College

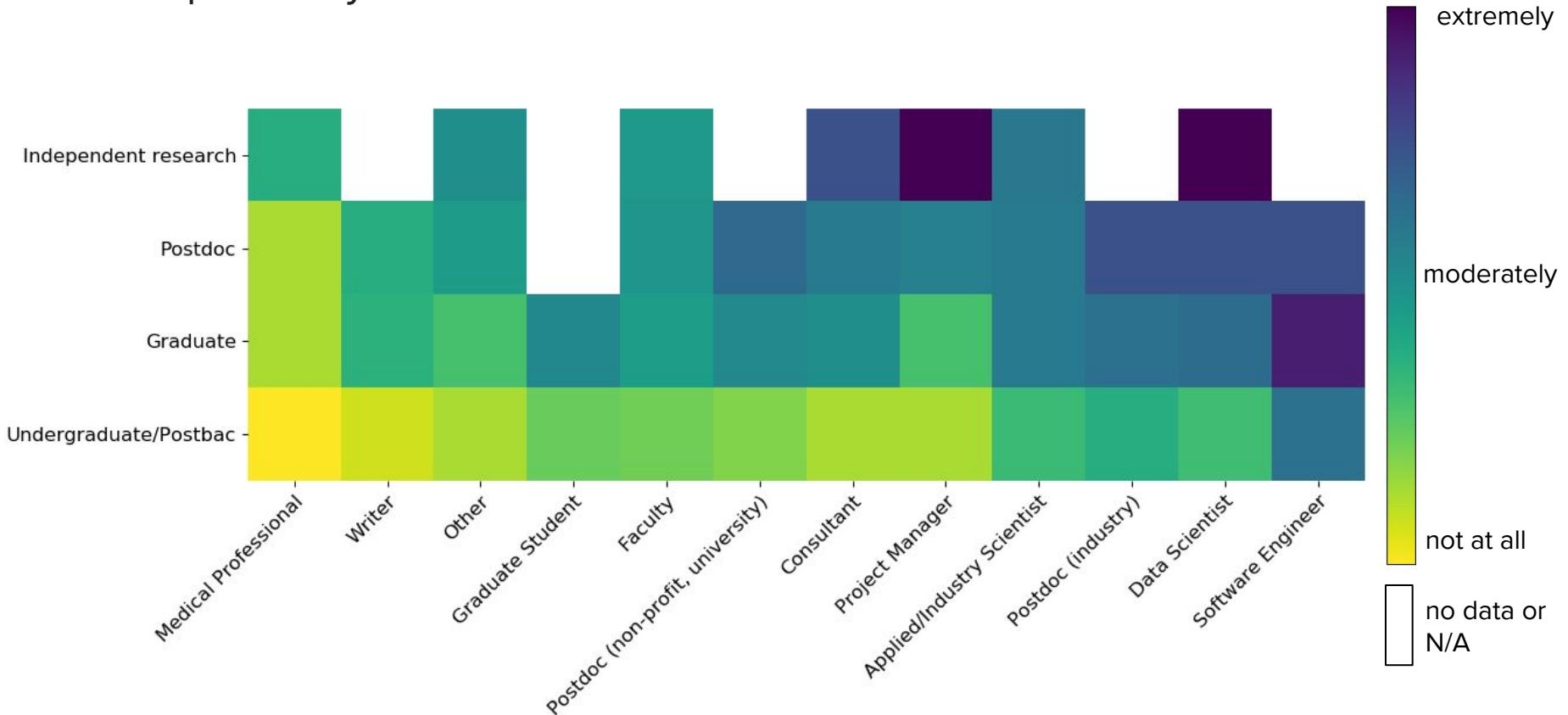
**36%** Graduate school

15% After graduate school

313 votes • Final results

+ many comments that they *still* hadn't learned how, and wanted to!

# How comfortable did/do you feel working with code at this point in your career?



# Objectives for this morning

- Introduce the teaching staff, students, and class
- Motivate learning how to code as a biology student
- **Discuss course logistics, expectations, & tools**
- Start coding!

First step: let's drop our ideas of what it means to be a *coder*.

Programming, like learning a language, *takes time*.



# Your language brain matters more for learning programming than your math brain

New research contradicts long held assumptions about coding



**Amy Nippert**

Neuroscience

University of Minnesota

May 12, 2020



2 peer comments



Christina Morillo on Wikimedia Commons.

<https://massivesci.com/articles/programming-math-language-python-women-in-science/>, summarizes this article: <https://www.nature.com/articles/s41598-020-60661-8>

Previous studies have shown that math and logic problems seem to rely mainly on the multiple demand regions in the left hemisphere, while tasks that involve spatial navigation activate the right hemisphere more than the left. The MIT team found that reading computer code appears to activate both the left and right sides of the multiple demand network, and ScratchJr activated the right side slightly more than the left. This finding goes against the hypothesis that math and coding rely on the same brain mechanisms.

<https://news.mit.edu/2020/brain-reading-computer-code-1215>  
about this study: <https://elifesciences.org/articles/58906>

# What will help you succeed in this course?

Things that predict success:

- How successful you *think* you'll be
- Completing assignments on time
- Asking questions when you have them
- Attending discussion sections & office hours

Things that **do not** predict success:

- Gender
- Age
- Personality
- Math ability



29A



@StuxnetStudios · 14h



New programming student:

"I'm not very good at this. When I type out the code, I have to fix lots of errors. And I have to look up how to do most of it."

Instructor:

"You're doing it right."

29

275

1.4K



# Historical sidenote: why is it called a bug?

In 1947, computer scientist & legend **Grace Hopper** found a *literal* bug in their computer, causing it to produce many errors.



Interview with Grace Hopper:  
<https://www.youtube.com/watch?v=QA33wW5LaNY>

Photo # NH 96566-KN (Color) First Computer "Bug", 1947

92	
9/9	
0 800	Anton started
1 000	stopped - Anton ✓
	13' UC (032) MP - MC
	(033) PRO 2
	cosack
	{ 1.2700 9.037 847 025 9.037 846 995 cosack
	2.130476415 (23) 4.615925059 (-)
	2.130476415
	Relays 6-2 in 033 failed several speed test
	in relay                          " 10.00 test .
1 100	Relays changed
1 525	Started Cosine Tape (Sine check) Started Mult + Adder Test.
1 545	Relay #70 Panel F (moth) in relay.
1 600	First actual case of bug being found.
1 700	Closed down .

<https://www.nationalgeographic.org>thisday/sep9/worlds-first-computer-bug/>

# What is programming, anyway?

- Programming is the way humans communicate with **computers**
  - It's a language!



# Wait, what's a computer?

**Hardware:** the physical parts of the computer (CPU, hard drive, etc.)

## Memory:

*Primary:* fast, temporary storage

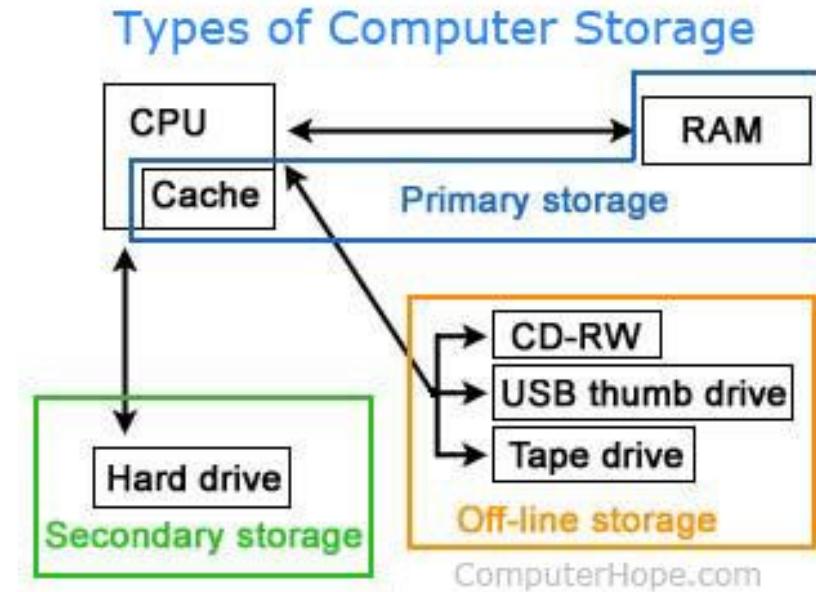
*Secondary:* slower, long-term storage

*Tertiary:* cloud storage

All computer memory is stored in **binary**.

[More information about memory](#)

[Great explainer about binary memory](#)



# Wait, what's a computer? (continued)

**Interface:** software, the operating system,  
what you see

## File structure

**cats = files**

Each has their own name.

**boxes = folders/directories**

Where you store the cats.

You can put a box into another box,  
but you can't put a box into a cat.

This is a copy  
of a cat, *not*  
an actual cat



# What is programming, anyway?

- Programming is the way humans communicate with computers
  - It's a language!
- The instructions we give the computer are taken **literally** and **sequentially**

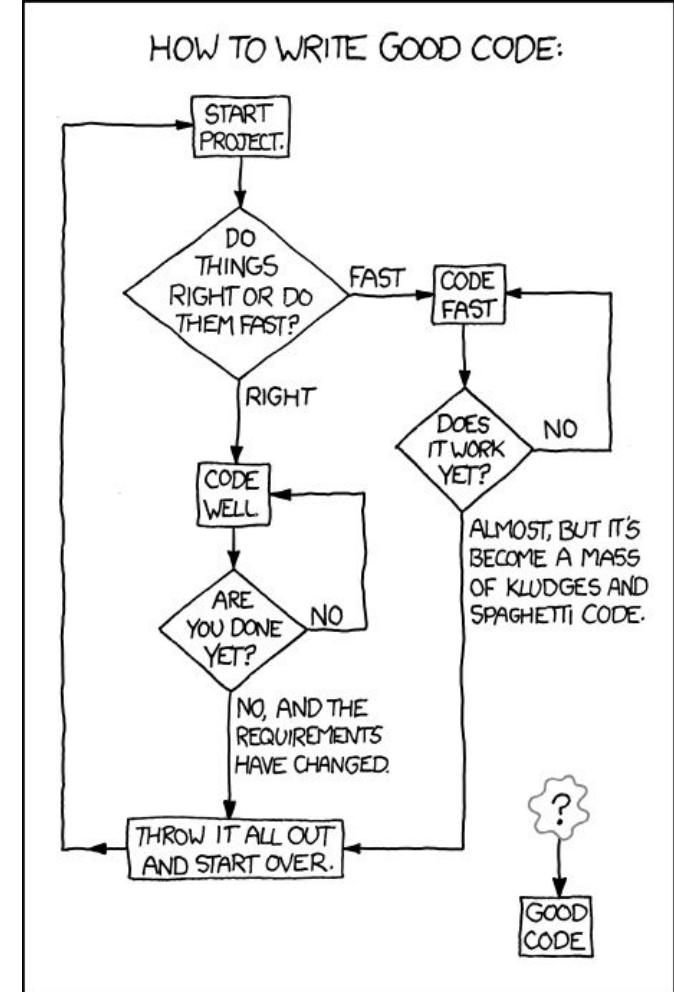
Capitalization matters:  
`print()` ≠ `Print()`

`b = a * 2`  
`a = 2`

computer: what is a?

# The path to writing good, efficient code

1. Make it **work**
2. Make it **right**
3. Make it **fast**



XKCD, <https://xkcd.com/844/>

# The path to writing good, efficient code

1. Make it **work**
2. Make it **right**
3. Make it **fast**

Our goal is to get to this step

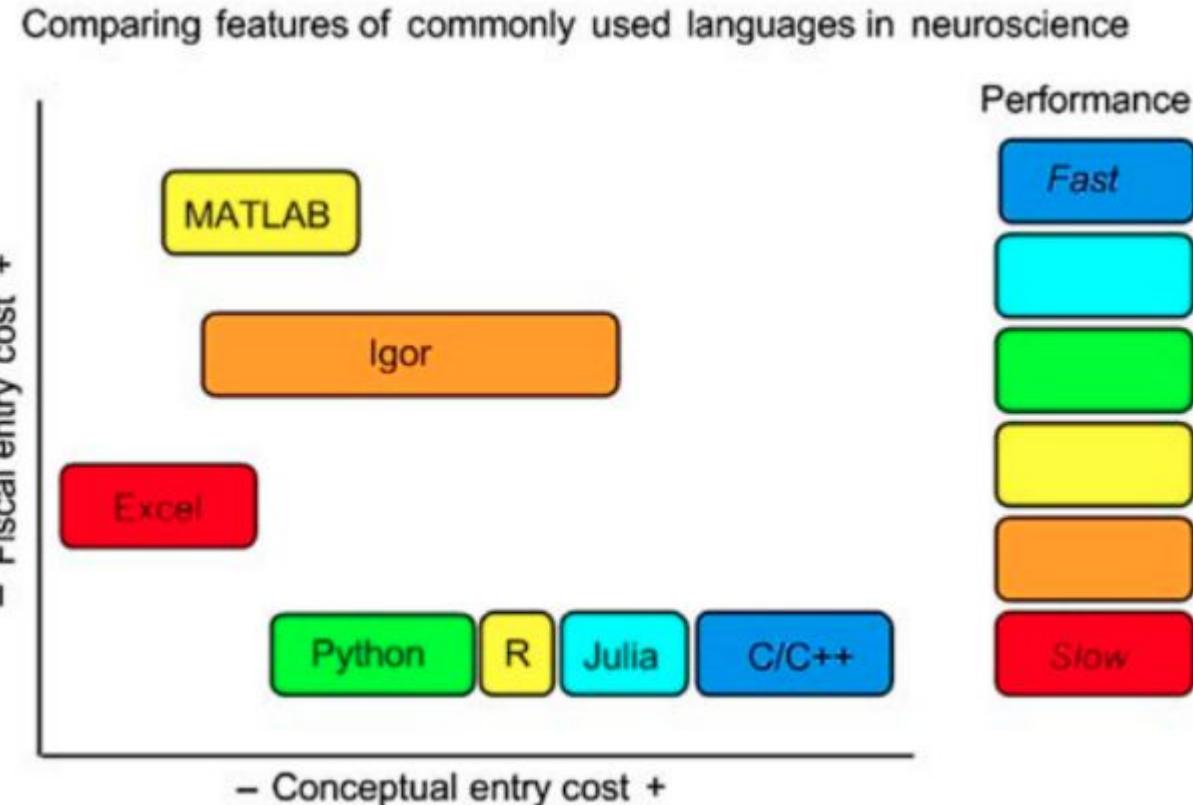
If you ultimately became a back-end programmer, you'd care about step 3.

For most problems biologists face, step 3 isn't paramount.



# Considerations for choosing a programming language

- Fiscal & conceptual entry
- Usage in particular field or profession



From Wallisch (2017)

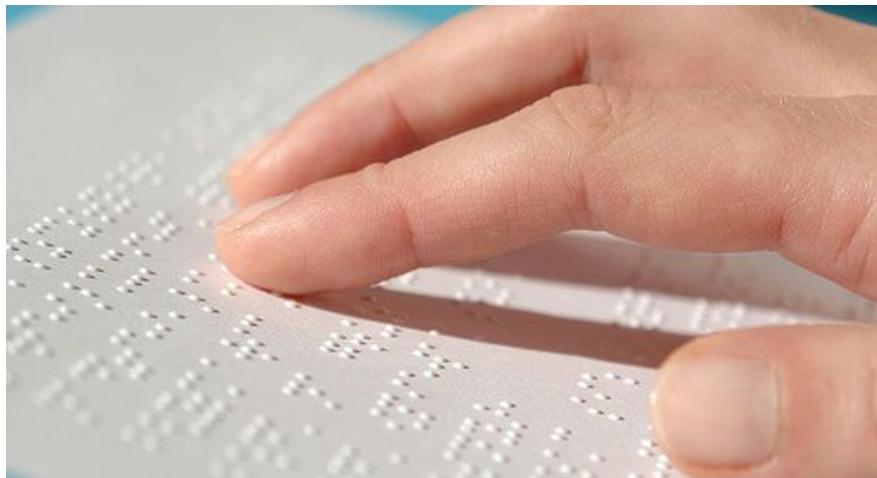
**All coding languages eventually need to talk to the computer in binary:**

01001000 01100101 01101100 01101100 01101111 00100001

**(hello)**

[Learn How To Write Your Name In Binary Code](#)

# There are many types of binary code, beyond computers



Braille

<https://www.afb.org/blindness-and-low-vision/braille/what-braille>

A • -	J • - - -	S • • •
B - • • •	K - • -	T -
C - • - •	L • - • •	U • • -
D - • •	M --	V • • • -
E •	N - •	W • - -
F • • - •	O - - -	X - • • -
G - - •	P • - - •	Y - • - -
H • • • •	Q - - • -	Z - - • •
I • •	R • - -	

Morse code

[https://www.discoveryworld.org/about/blog/discover\\_at\\_home/morse-code/](https://www.discoveryworld.org/about/blog/discover_at_home/morse-code/)

# In this class, we'll use Python

- Programming language, development led by Python Software Foundation ([www.python.org](http://www.python.org))
- Uses concise structure & wording similar to human language
- A “high-level language”



# Assembly language vs. high-level language

```
section .text
global _start
_start:
    mov ecx, 10
    mov eax, '0'
    l1:
    mov [num], eax
    mov eax, 4
    mov ebx, 1
    push ecx
    mov ecx, num
    mov edx, 1
    int 0x80
    mov eax, [num]
    inc eax
    pop ecx
    loop l1
    mov eax, 1
    int 0x80
section .bss
    num resb 1
```

```
for num in range(10):
    print(num)
```

Inspired by Porter & Zingaro,

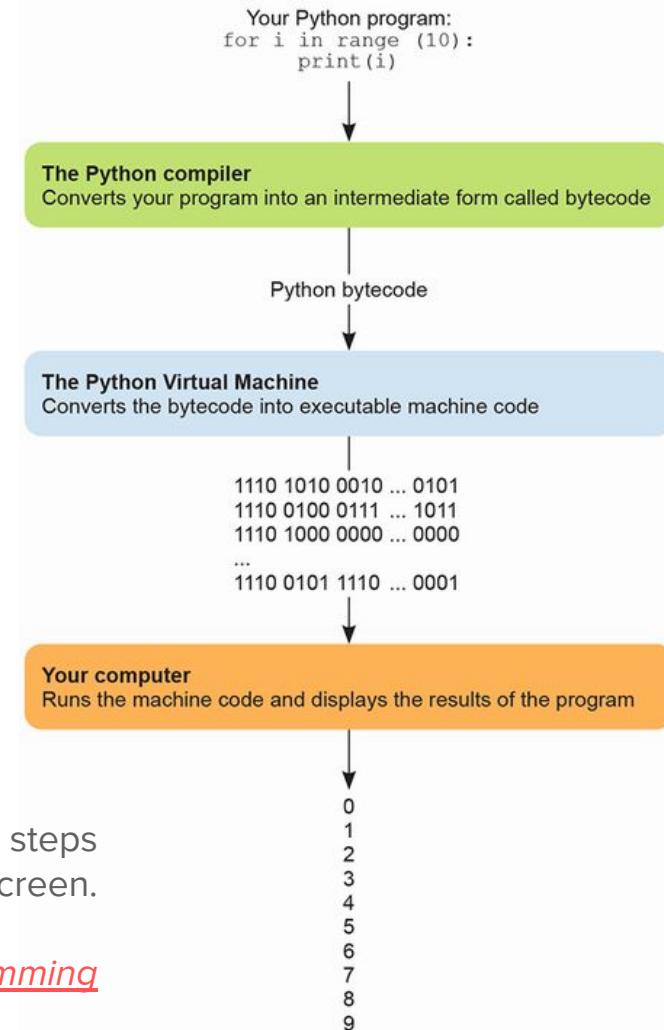
[Learn AI-Assisted Python Programming](#)

# In this class, we'll use Python

- Python can be used for many purposes, from web programming, to creating games, to analyzing & visualizing data
  - Extension: '.py'
- We'll also work in **Jupyter Notebooks**
  - Extension '.ipynb'

Your Python program goes through several steps before you see the output on your screen.

From Porter & Zingaro, [Learn AI-Assisted Python Programming](#)



# Course Objectives

- Read and run basic Python programs, recognizing the structures used and explaining how they work
- Manipulate and create objects in Python
- Write, edit, and execute Python code in Jupyter Notebooks as well as the command line
- Visualize and run hypothesis-testing on simple datasets in Python
- Implement common algorithms for analyzing biological data and determine when such computations are appropriate

And, we're going to do all of this in a world with AI assistants (i.e. chatGPT):

*It's still important to be able to read Python code to gain an overall understanding of what it does.*

# Grading breakdown

---

**In-class work & participation (10%)**

**Assignments (50%)**

**Midterm (15%)**

**Final project (25%)**

Assignments & project components

**lose 10% each day they are late.**

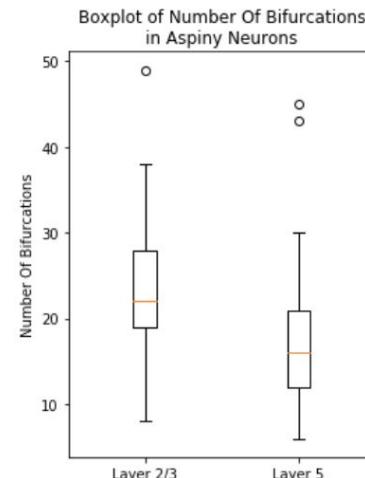
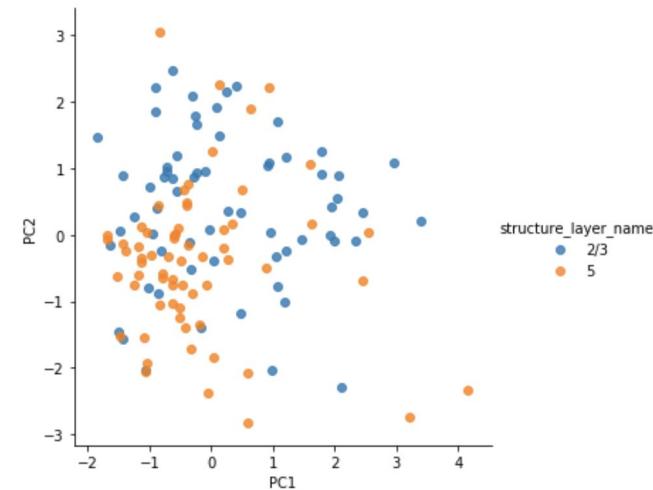


# Assignments

- Due **every Wednesday at 5 pm**
- Worth 2.5-10% each
- Completed individually
- Programmatically graded (via Datahub/NBGrader)
- Lowest assignment grade dropped.
- In discussion, you'll cover how to submit these. There are also written instructions on Canvas.

# Project, groups of 2-3

- Includes the project proposal, code, and presentation.
- Your final project will either:
  - Write a program to complete a task
  - Analyze & visualize data of your choosing
- We will discuss possibilities for your project as we move through the course.



END OF YEAR SALE - SAVE 50%

0 1  
Days0 6  
Hours1 1  
Minutes0 6  
Seconds[VIEW PLANS](#)

DATAQUEST

[COURSES](#)[STUDENT STORIES](#)[WE'RE HIRING](#)[BLOG](#)[START LEARNING](#)[LOG IN](#)

# Learn Data Science

Whether you're new to the field or looking to take a step up in your career, Dataquest can teach you the data skills you'll

Take a FREE course!

 Email Password

You can also sign up for **Stepik** (<https://stepik.org/course/56730/>) or **DataQuest** (free!) & complete lessons in parallel with our course.

Python Basics for Data Analysis (Skill Path) or Data Scientist in Python (Career Path)

# Office hours

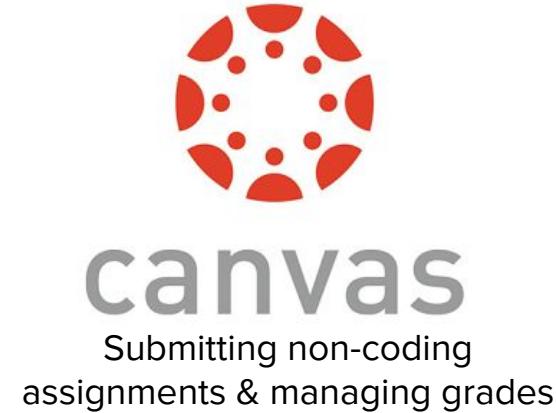
*On Zoom, where it's easier to troubleshoot!*

## **Why should you come to office hours?**

- You have clarifying questions about the course or its content
- You have concerns about the course and your progress
- You'd like to talk about career paths in biology or neuroscience



# Course Tools



“Tutoring” and project help;  
more on this later.



Sharing public course materials  
<https://github.com/BILD62>

# Interacting with course materials



You can find all of our course materials on either Canvas or the course GitHub: [https://github.com/BILD62/BILD62\\_WI24](https://github.com/BILD62/BILD62_WI24)

## Lectures

In other words, PDF slides shown during class.

Hosted on GitHub in the Lectures folder

If I use both a PDF and a Jupyter Notebook during lecture, these numbers will match

## Materials

Jupyter Notebooks

You can pull these locally or to DataHub, or look at them online via GitHub or Colab/ Binder



## Assignments

Jupyter Notebooks, submitted through **Assignments** tab

Answers posted in the Assignments folder on Github





THE MAGIC LINK FOR THIS  
COURSE:

[https://datahub.ucsd.edu/hub/user-redirect/  
git-sync?repo=https://github.com/BILD62/BILD62\\_WI24](https://datahub.ucsd.edu/hub/user-redirect/git-sync?repo=https://github.com/BILD62/BILD62_WI24)



# THE MAGIC LINK FOR THIS COURSE:

*Sync with your datahub:*



[https://datahub.ucsd.edu/hub/user-redirect/  
git-sync?repo=https://github.com/BILD62/BILD62\\_WI24](https://datahub.ucsd.edu/hub/user-redirect/git-sync?repo=https://github.com/BILD62/BILD62_WI24)



*Where our course  
content lives*

# To clone Materials to DataHub:

1. Click on the magic link.
2. Log in to DataHub as prompted.
3. You'll be in our course folder now!
4. Save your own copy by adding your initials to the end of the file name. **DO NOT DO THIS FOR ASSIGNMENTS!**
5. Next time you click the link, you'll have a fresh copy, plus your copy.

# To interact with Jupyter Notebooks on your computer

## OPTIONAL

1. Install Anaconda with Python 3.7 for your operating system.
2. If you're using Windows, [download git](http://www.github.com/BILD62/BILD62_WI24.git).
3. In Terminal (Mac) or the Anaconda Prompt (Windows), clone the repository by running the following command:  
`git clone http://www.github.com/BILD62/BILD62\_WI24.git`
4. Open Jupyter Notebook. There are two ways to open:
  - o In Terminal (Mac) or the Anaconda Prompt (Windows), type **jupyter notebook**
  - o Open Anaconda Navigator and launch jupyter notebook
5. On the Jupyter landing page, navigate to the notebook and open it.
  - o It will open in a browser but is *not* using an internet connection.

We'll also be trying to learn  
about how this is going with  
some *education research*.



## **Exempt Information Sheet**

You are being invited to participate in a research study titled Coding Attitudes of Biology Students Coding Attitudes of Biology Students. This study is being done by Ashley Juavinett, PhD from UC San Diego. You were selected to participate in this study because you are a student in a class that is being studied.

The purpose of this research study is to create knowledge that has the potential to improve the learning and educational experience of students at UC San Diego and beyond, and specifically to understand student attitudes towards coding

Your participation in this research should last approximately 15 minutes. If you agree to take part in this study, you will be asked to take pre- and post-course surveys containing about 20 multiple choice items and several open-ended questions.

Your participation in this study is completely voluntary and you can withdraw at any time. Choosing not to participate or withdrawing will result in no penalty or loss of benefits to which you are entitled, and will not adversely impact your standing with the instructional staff of your course. You are free to skip any question that you choose.

Complete both pre- and post-surveys on Qualtrics, gain 1% extra credit

# Introduction to the UCSD DataHub & Jupyter Notebooks

A screenshot of a Jupyter Notebook interface. At the top, there's a toolbar with icons for file operations, cell types (Code, Markdown, etc.), and execution (Run, Kernel Restart). Below the toolbar is a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. To the right of the menu bar are buttons for Not Trusted, Python 3, Logout, and Control Panel. The status bar at the bottom shows the title "jupyter ProgrammingFundamentals", the last checkpoint date "11/05/2019 (autosaved)", and a Python logo icon.

# About Jupyter Notebooks

- Jupyter is a loose acronym for Julia, Python, and R
- Run in a web browser but it's not *necessarily* online (it is when we use the DataHub)
- Usefully, it will show plots directly in the notebook as you work your way through, performing analyses in real-time
- Two main components:
  - **Kernel:** the engine that runs the code
  - **Dashboard:** landing page where you can see the notebooks you've created



# Using Jupyter Notebooks

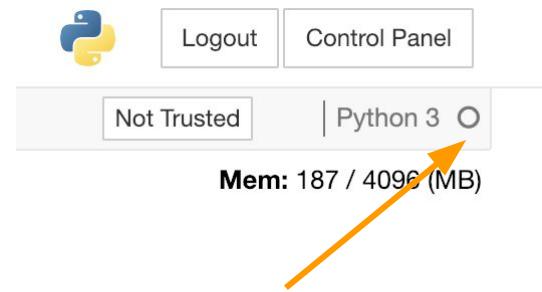
- **Cell:** the main organizational structure of the notebook
  - Use **Shift+Enter** to run a cell (or press Run)
  - You can run cells out of order, and move cells around!
  - Cells can be **code** (the default) or **markdown** (descriptive text or images)
    - Code cells have **In[ ]** :
      - If there is a star (**In[ ]\* :**), that means your cell is running
    - Change between code & markdown using dropdown menu (or keyboard shortcuts)
    - Turns **green** in edit mode

# Using Jupyter Notebooks

- You can clear your **namespace** and get a fresh start by restarting the kernel



- If you change anything in the cell, you need to re-run it.
- For help:
  - Help > User Interface tour
  - Help > Keyboard Shortcuts



You can tell if the kernel is busy by whether or not the circle next to Python 3 (upper right corner) is filled or not. (filled = busy)

**Expressions** describe  
how to combine pieces of  
data (e.g., add them!)

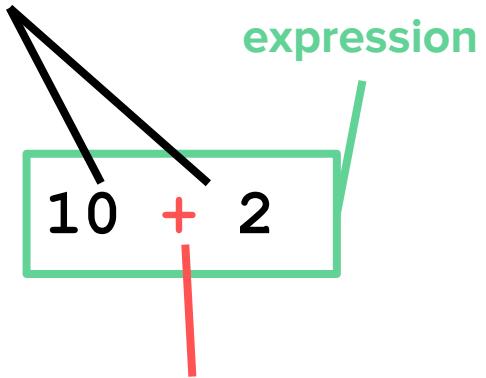
## SYMBOLS YOU WILL ENCOUNTER IN THIS COURSE

Symbol	Name	Sample Usage
=	Equal sign	Assign variable
#	Pound sign; hashtag	Line comments
[ ]	Brackets	Indexing & Slicing
( )	Parentheses	Using functions
{ }	Curly Brackets	Defining dictionary
' '	Single quotes	Creating string
" "	Double quotes	Creating string
_	Underscore	In variable names
!	Explanation point	To test not equal (!=)
\	Back slash	Delineate line break
:	Colon	Indexing

# Basic arithmetic operators in Python

Symbol	Operation	Usage
+	Addition	$10+2$
-	Subtraction	$10-2$
*	Multiplication	$10*2$
/	Division	$10/2$
**	Exponent	$10**2$
%	Modulo	$10\%2$

inputs



operand

If you want a whole number (floor division), use // instead.

Let's get into a Jupyter notebook!  
Use the [magic link](#) to sync up  
your DataHub with our folder, and  
open notebook 01.

# Before next class...

- Take the computing attitudes survey
- Access Canvas ([canvas.ucsd.edu](https://canvas.ucsd.edu)) & DataHub ([datahub.ucsd.edu](https://datahub.ucsd.edu))
- (Optional) Sign up for Stepik and/or DataQuest
- (Optional) Install Python 3.7 (via the Anaconda distribution) on your computer (<https://www.anaconda.com/distribution/>)

You only *really* need access to the DataHub, but having the ability to use Python & Jupyter Notebooks on your local computer *may* be useful (especially for final projects)!