# Welcome to BILD62

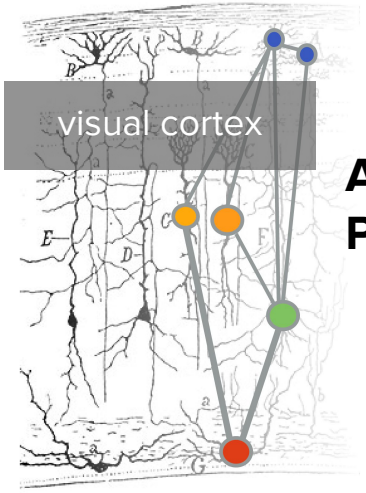
Dr. Juavinett
*jah-vah-nett*
(or, Dr. J)

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**




multisensory processing & ethological behaviors

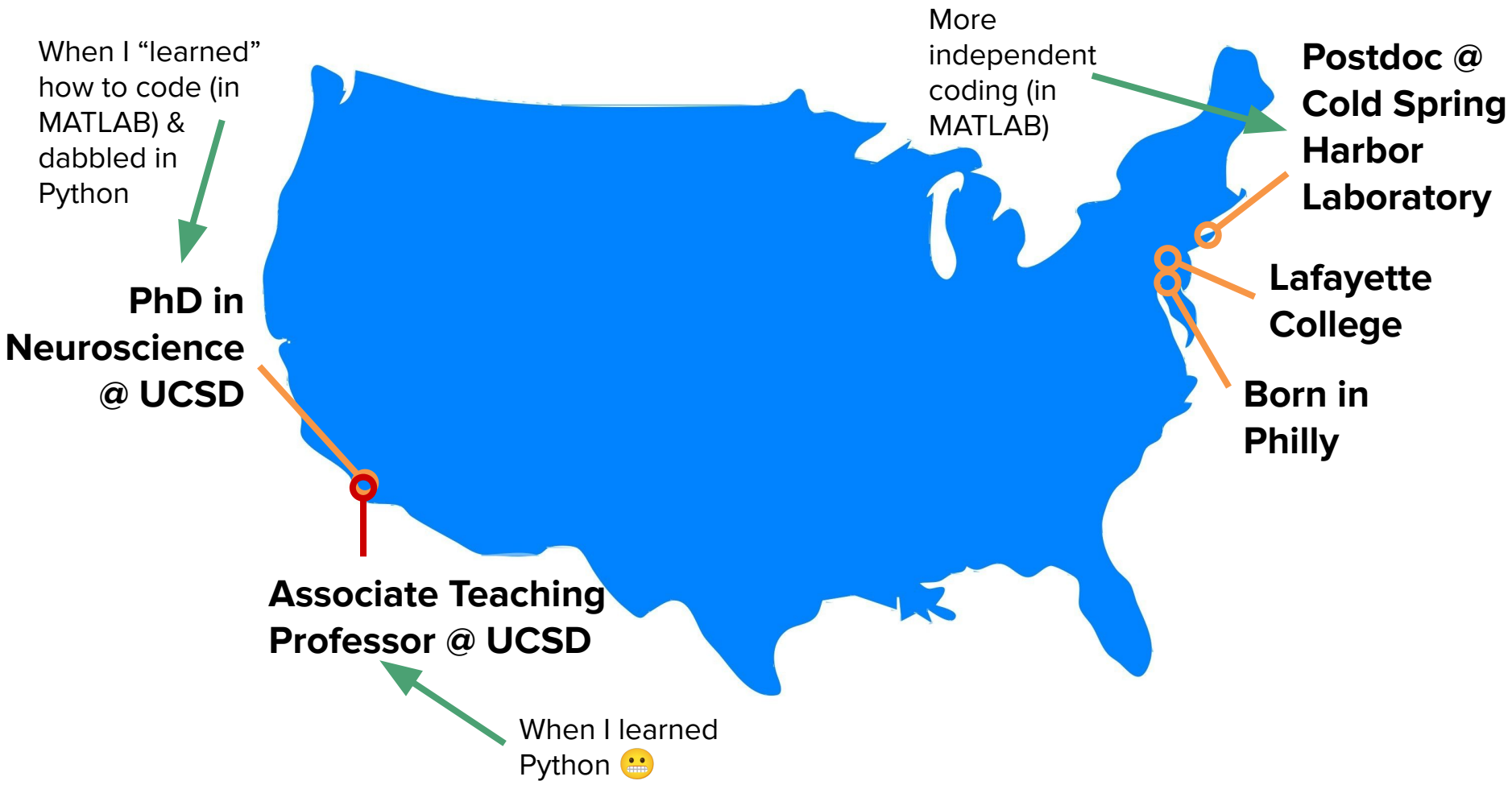
**Postdoc @ Cold Spring Harbor Laboratory**

**Lafayette College**

**Born in Philly**

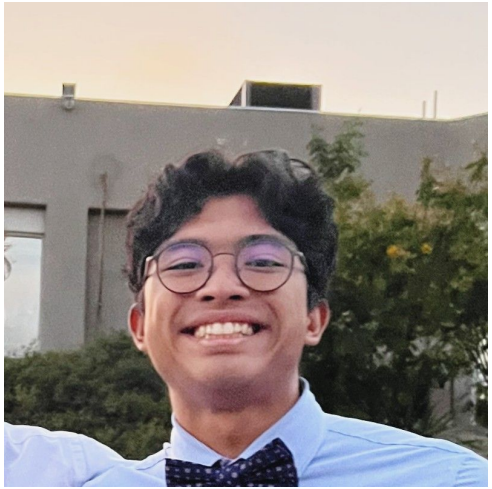
**Associate Teaching Professor @ UCSD**

- Neuroscience careers & education
- Open-source data


When I “learned” how to code (in MATLAB) & dabbled in Python
PhD in Neuroscience @ UCSD
Associate Teaching Professor @ UCSD
When I learned Python 😬
More independent coding (in MATLAB)
Postdoc @ Cold Spring Harbor Laboratory
Lafayette College
Born in Philly

# Introduction to our teaching staff!

**Dante Fischer**
Teaching Assistant (TA)
d2fisher@health.ucsd.edu



**Miguel Ybanez**
Pedagogy and Learning Apprentice (PLA)
mybanez@ucsd.edu

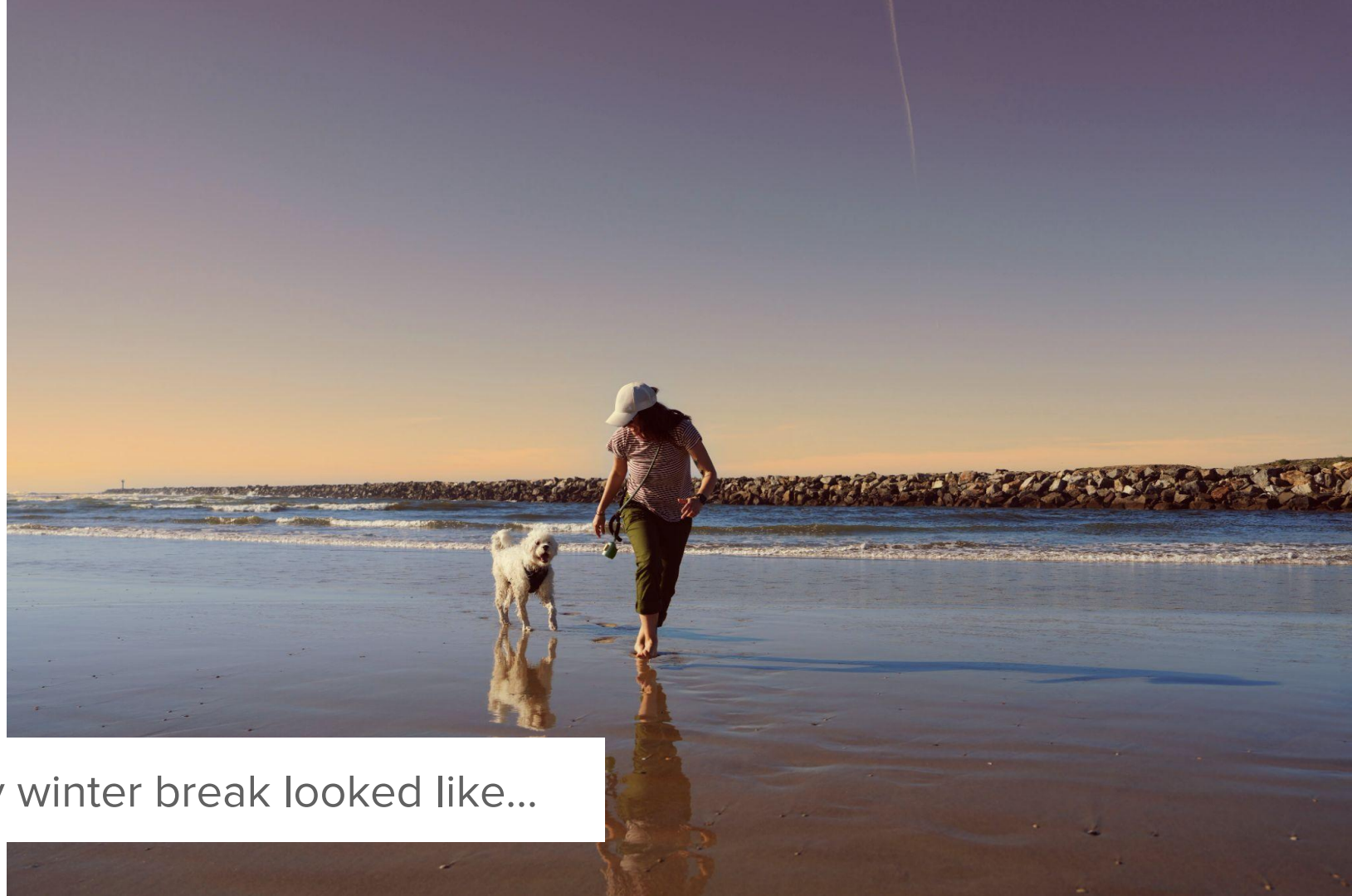
*Readers:*
Jia Bhavnani
jbhavnani@ucsd.edu

Grisha Tamazyan
gtamazyan@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



What my winter break looked like...

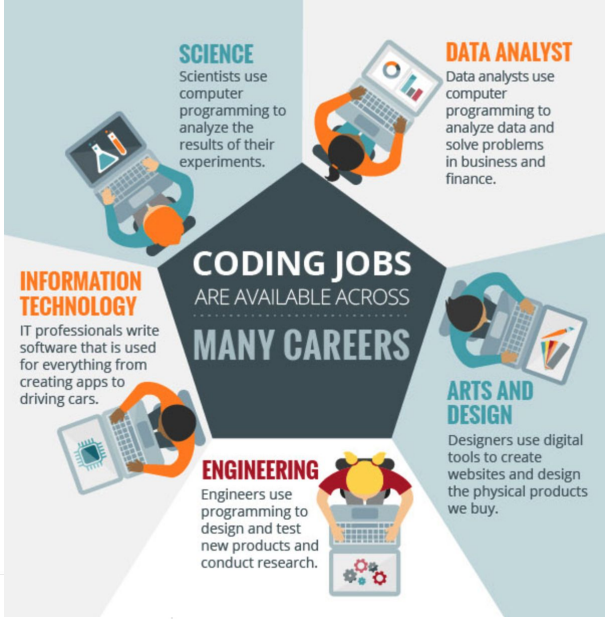
# Objectives for today

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

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, especially those with specialized knowledge




nature

Explore content | About the journal

nature › career q&a › article

CAREER Q&A | 31 May 2022

## Why science needs more research software engineers

**Ten years after their profession got its name, research software engineers seek to swell their ranks.**

By Chris Woolston

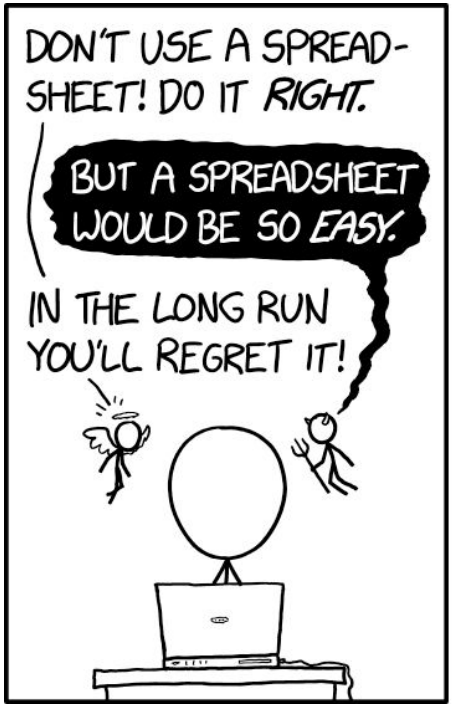
https://www.nature.com/articles/d41586-022-01516-2

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

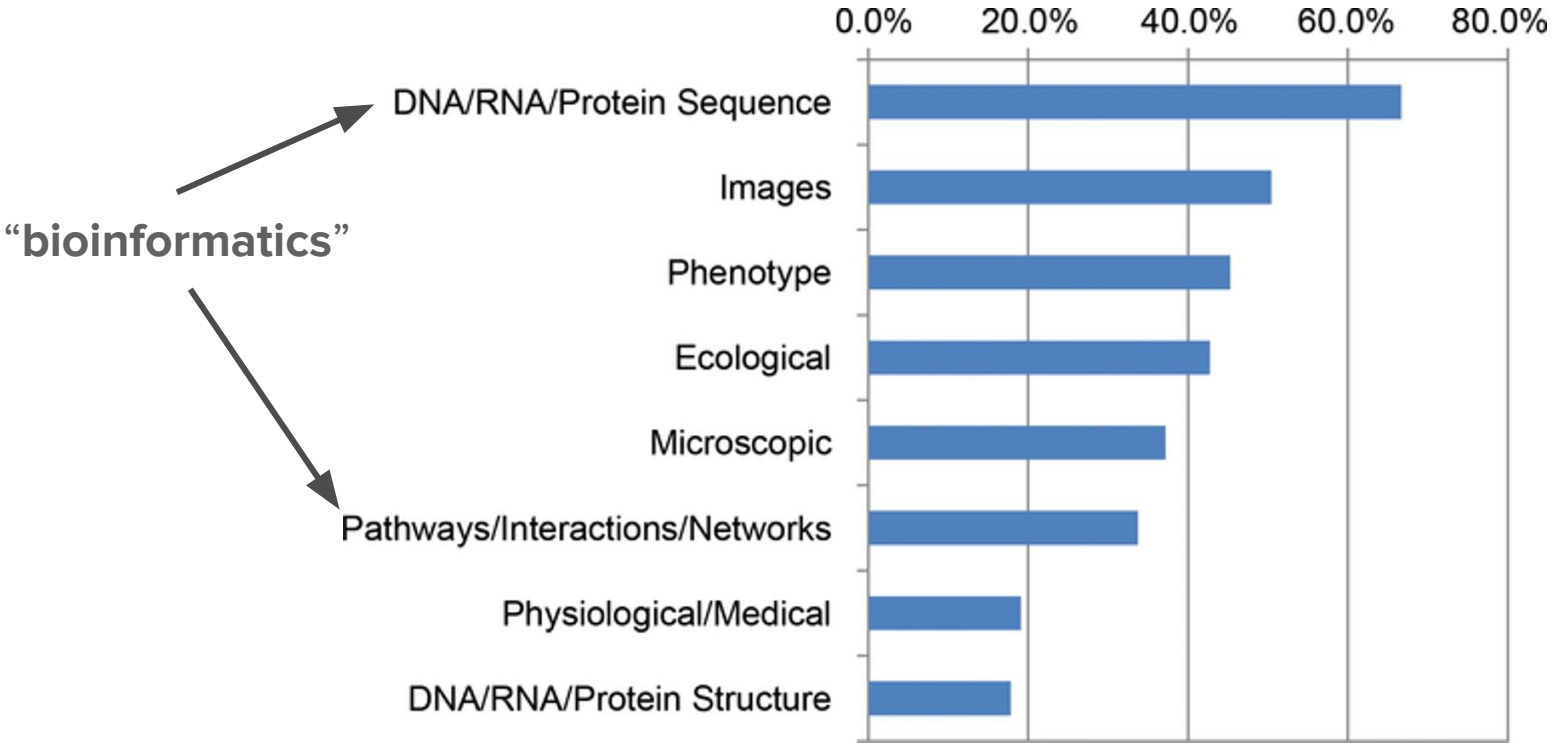
You can automate analyses in Excel, but this is quite limited.

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.

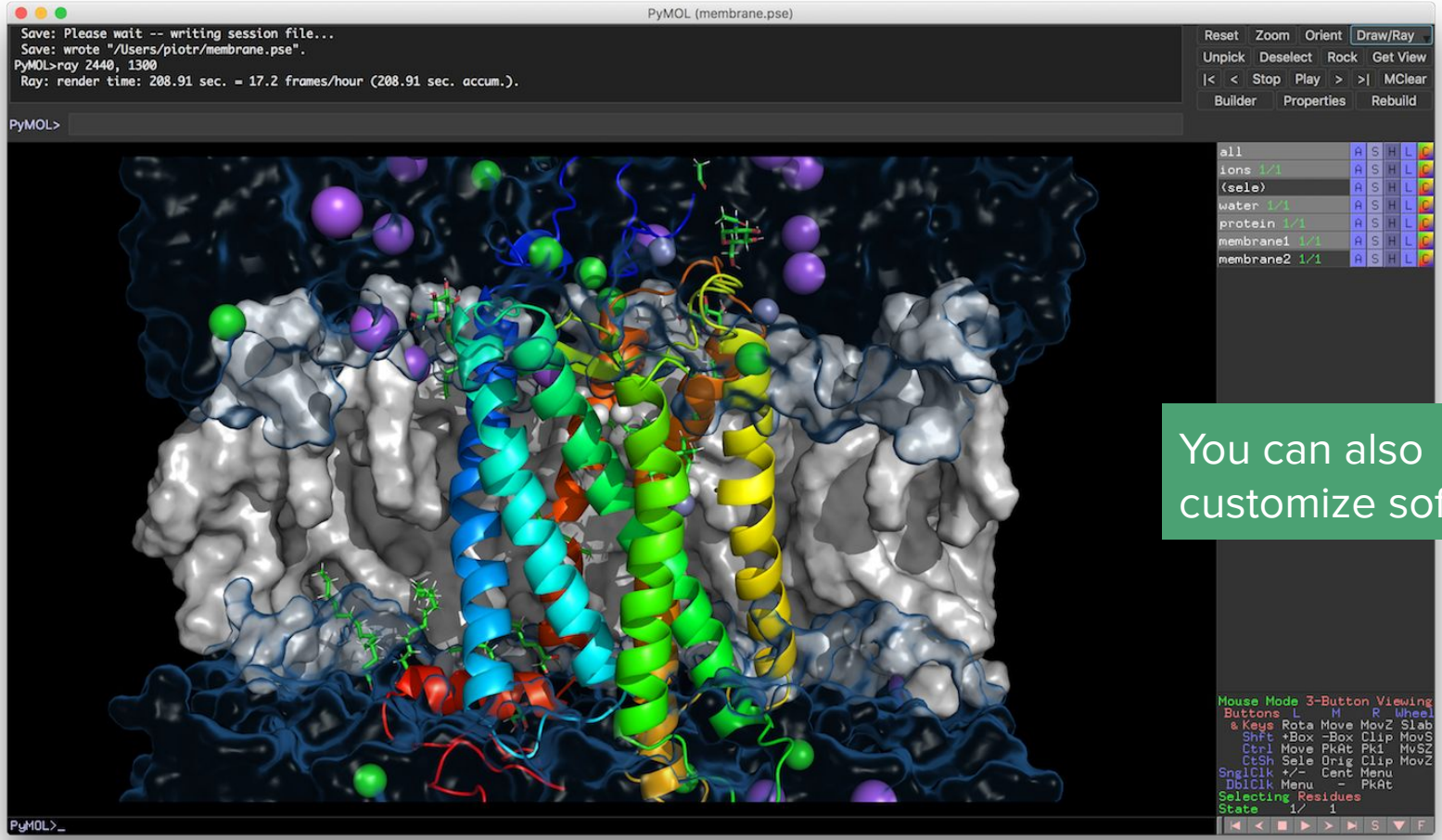



https://xkcd.com/2180/




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

Barone L, Williams J, Micklos D (2017) Unmet needs for analyzing biological big data: A survey of 704 NSF principal investigators. PLOS Computational Biology https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005755




You can also customize software!

pymol.org

pymol.org/2/

PyMOL by Schrödinger

DOWNLOAD SCREENSHOTS PRODUCTS SUPPORT CONTACT

# Open-Source Philosophy

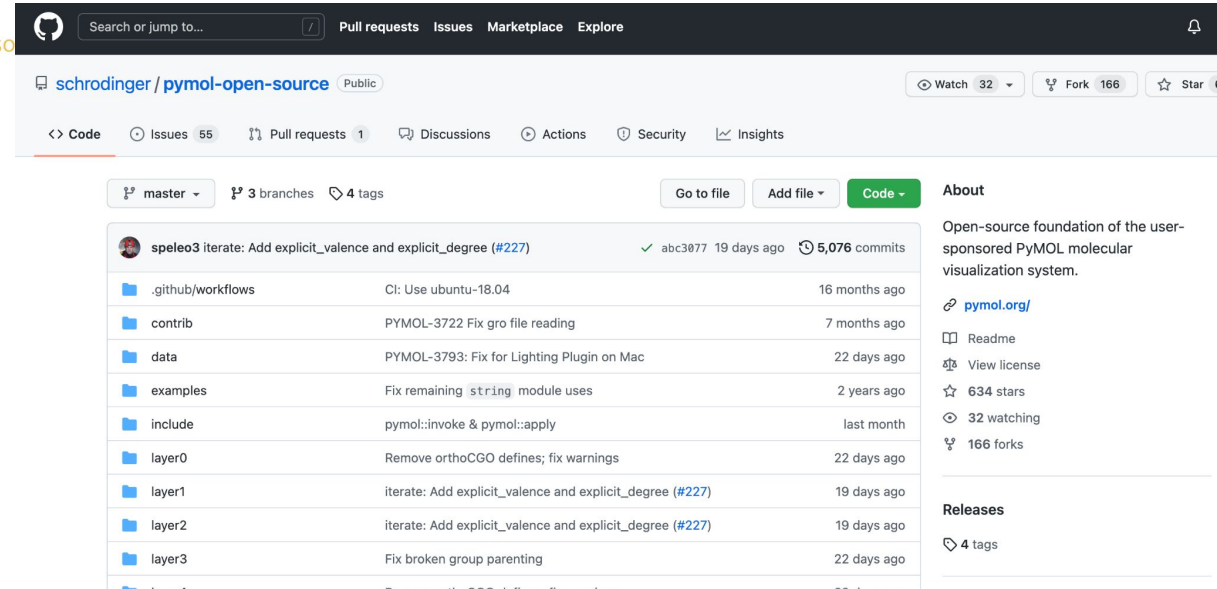
PyMOL is a commercial product, but we make most of its source code freely available under a permissive license. The open source project is maintained by Schrödinger and ultimately funded by everyone who purchases a PyMOL license.

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

Visit the Open-Source Project

Become a sponso

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



# 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*.**

MASSIVE SCIENCE | Menu | Search | Account

# 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:

- Your belief that you *can* learn how to code
- Completing assignments on time and on your own
- Asking questions when you have them
- Attending discussion sections & office hours
- Testing your knowledge in new situations

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:

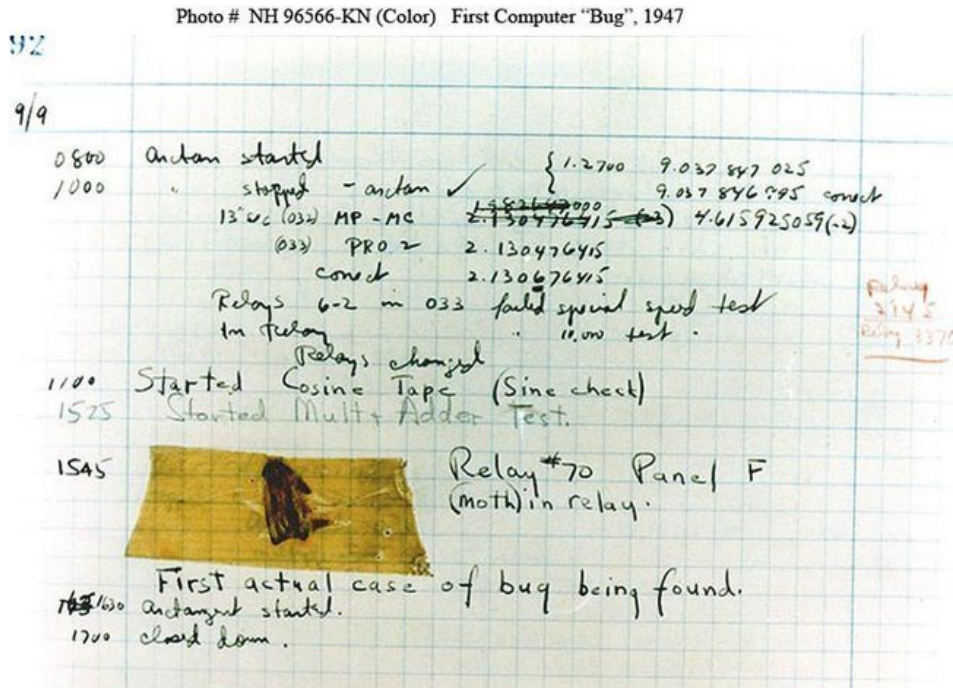
"You're doing it right."

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




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

*Historical sidenote:*

# Women were the first computers!

**Doris Baron**, a human "computer" works with tape from the air pressure measuring machines in 1955

The Gendered History of Human Computers

# 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

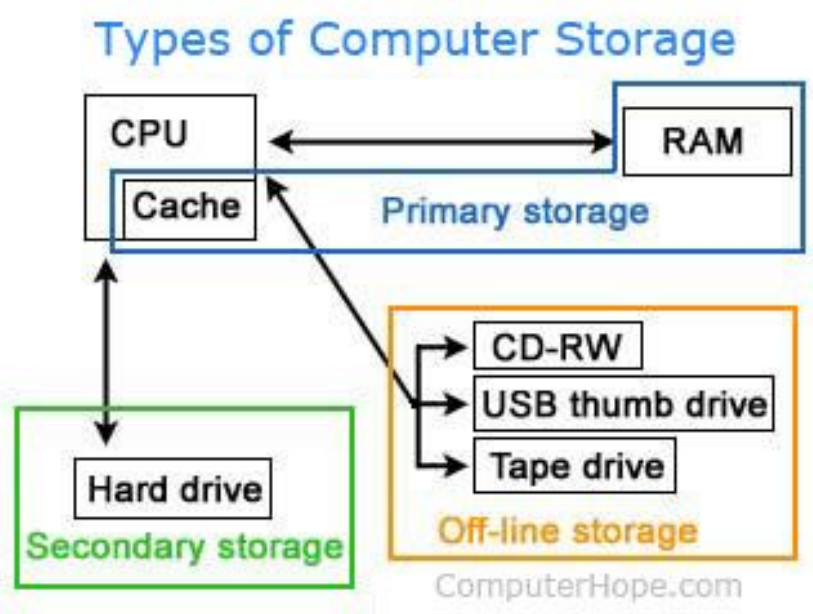
*Secondary*: slower, long-term storage

*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**

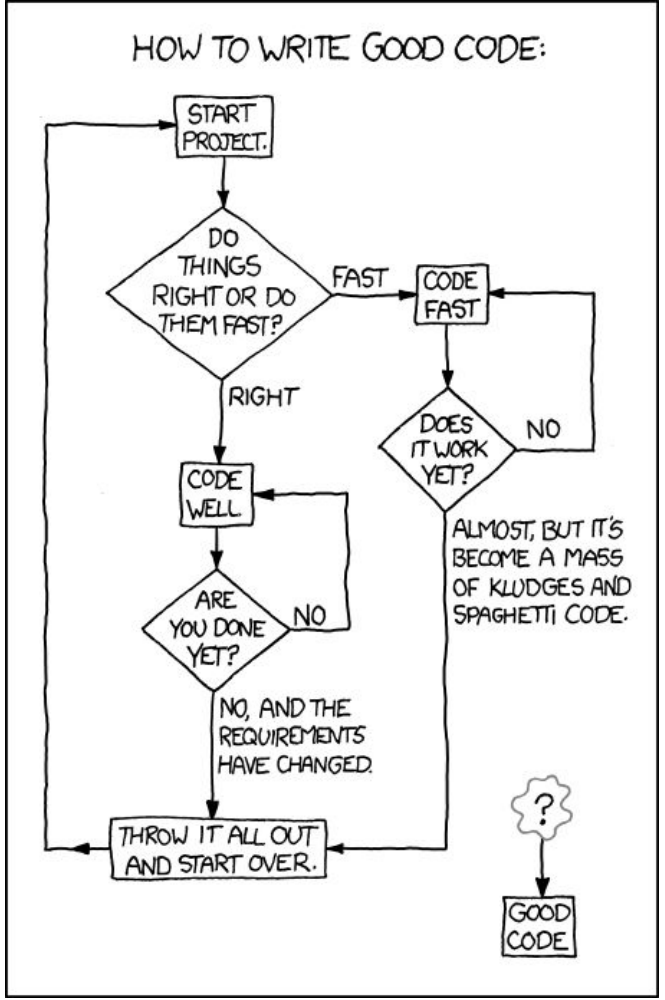
Capitalization matters:
`print() ≠ Print()`

```
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**

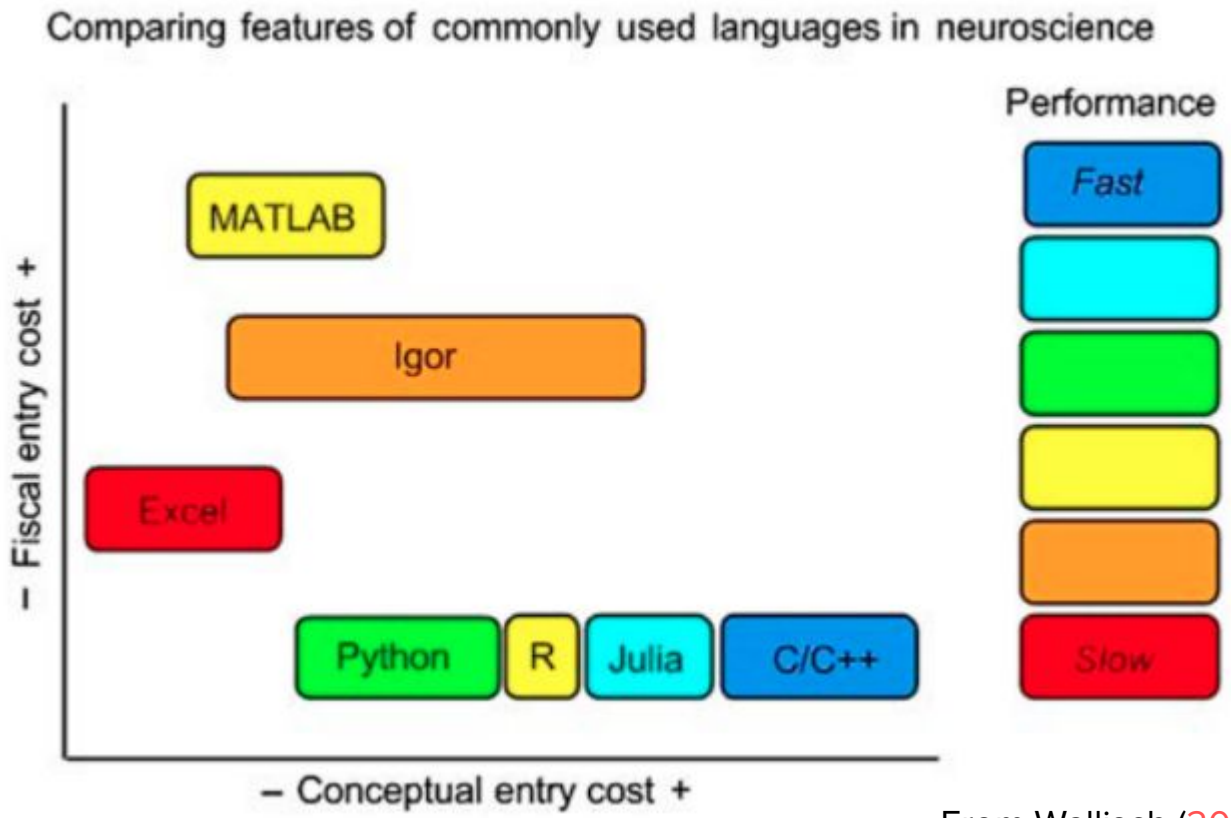
Our goal is to get to this step

If you ultimately became a ***full-time, professional developer***, 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)

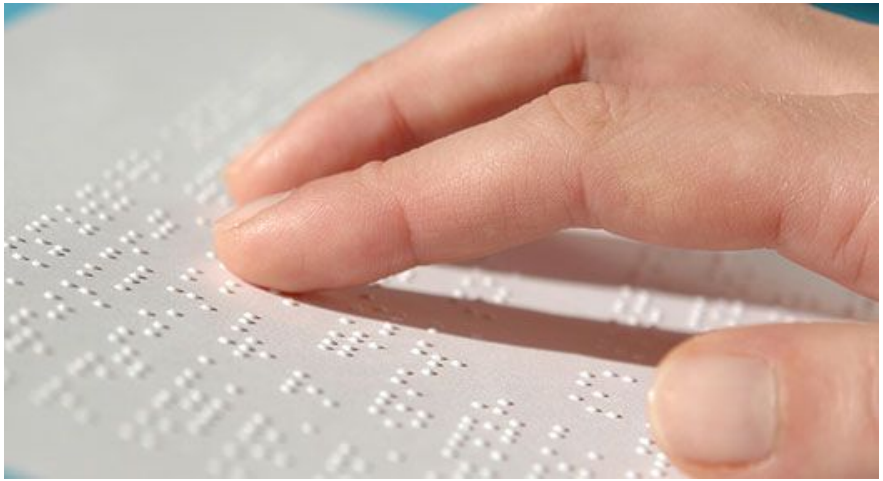
**All coding languages eventually need to talk to the computer in binary:**

`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/

# In this class, we'll use Python

- Programming language, development led by Python Software Foundation (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
```

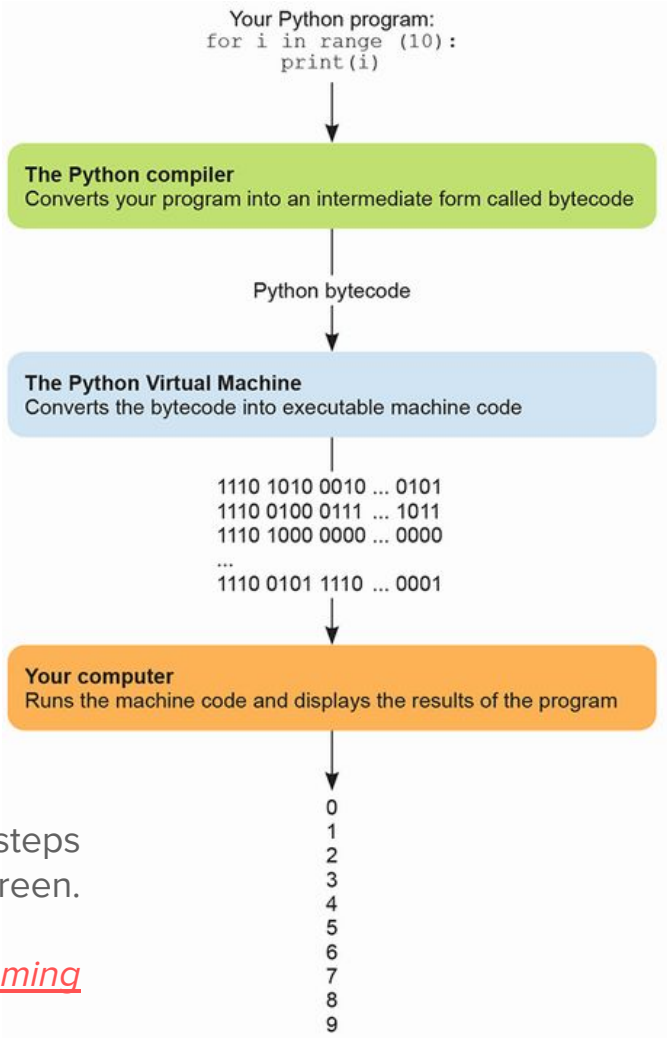
```
for num in range(10):

    print(num)
```

Inspired by Porter & Zingaro,
*Learn AI-Assisted Python Programming*

# In this class, we'll use Python

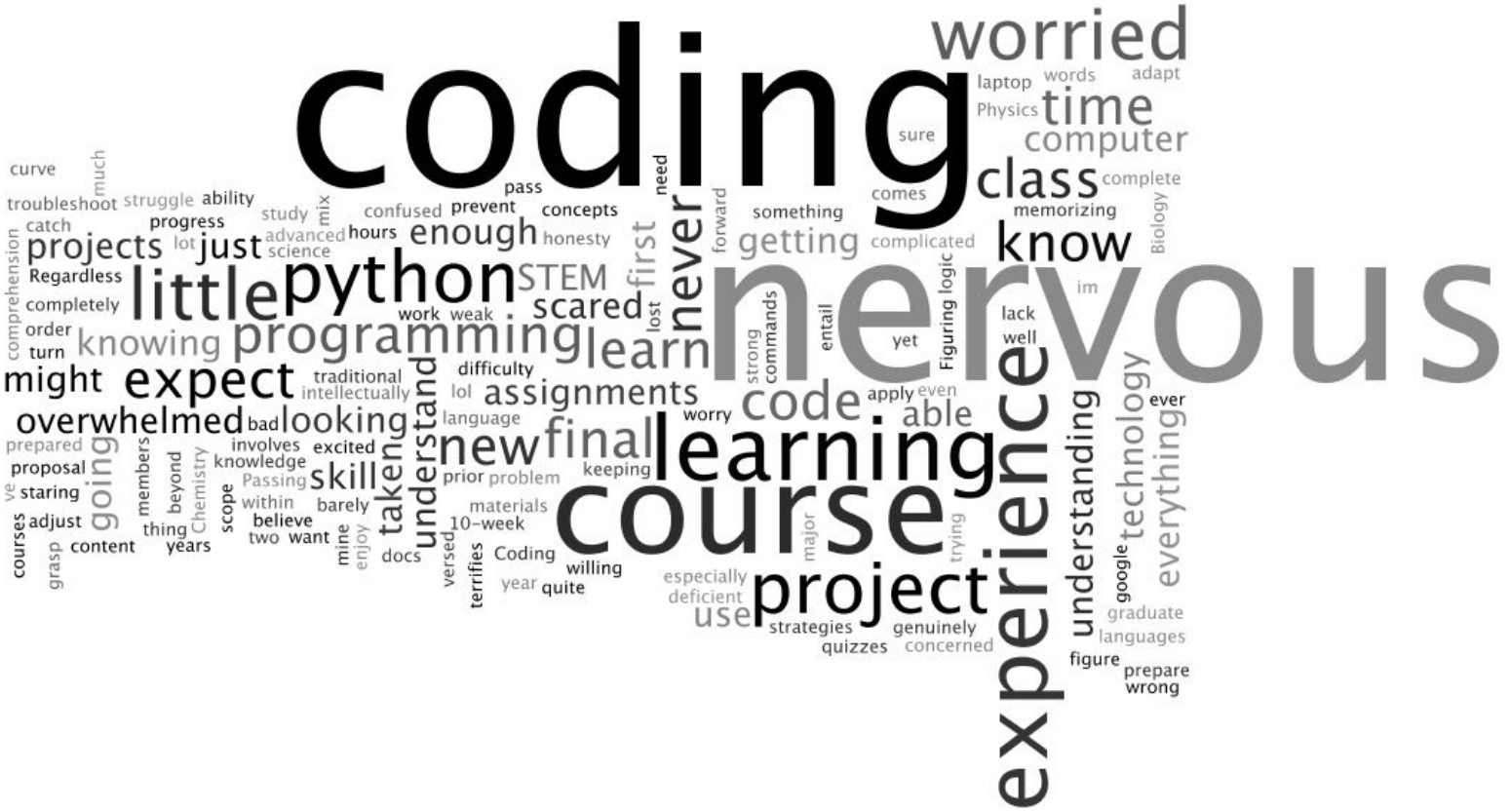
- Python can be used for many purposes, from web programming, to creating games, to analyzing & visualizing data
  - Extension: '.py'
- We'll primarily work in **Jupyter Notebooks**
  - Extension '.ipynb'



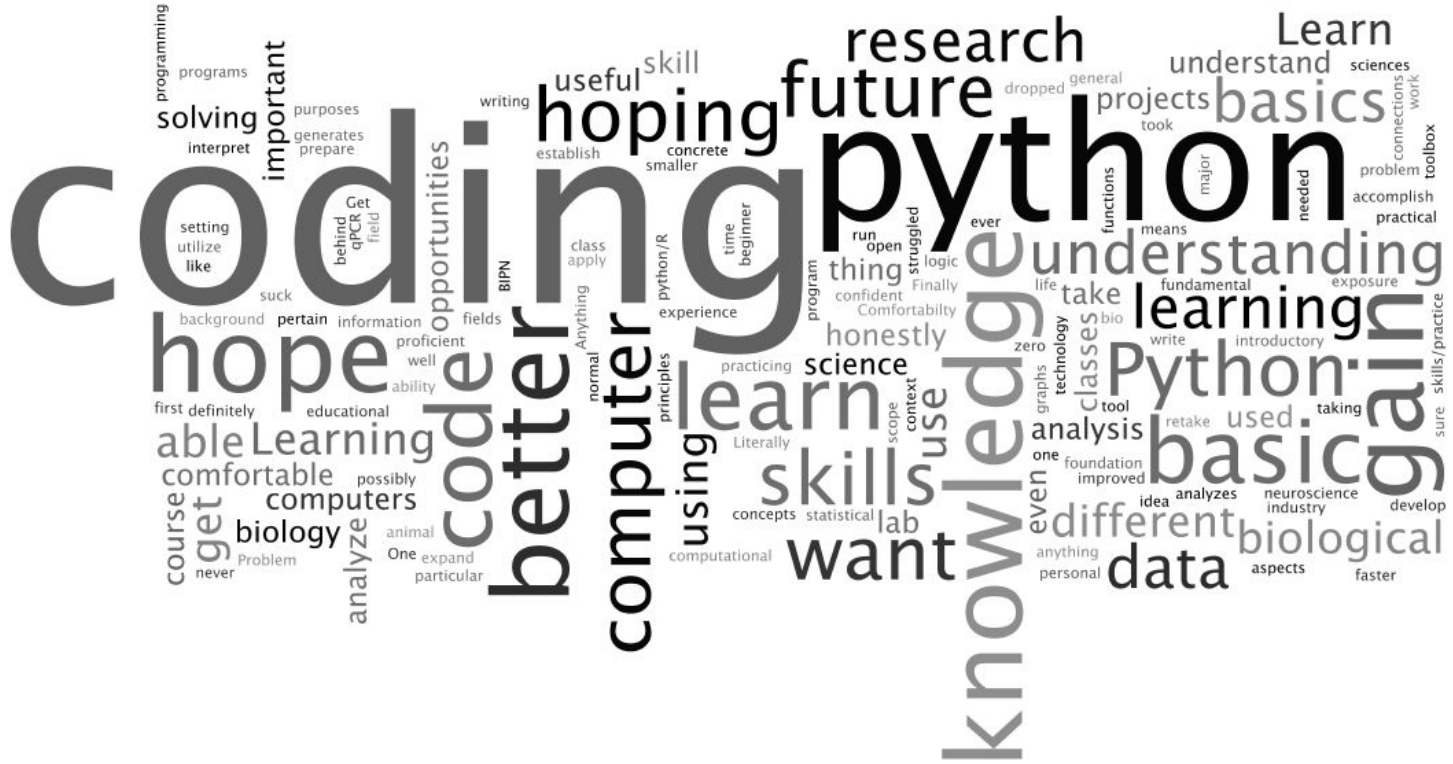

Your Python program goes through several steps before you see the output on your screen.

From Porter & Zingaro, *Learn AI-Assisted Python Programming*

# What are you worried about?



# What do you hope to gain?



# 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 ensure it is *really* doing what you think it is.

# Grading breakdown

**In-class work & participation (15%)**

**Assignments (20%)**

**Quizzes & Final Exam (40%)**

**Final project (25%)**

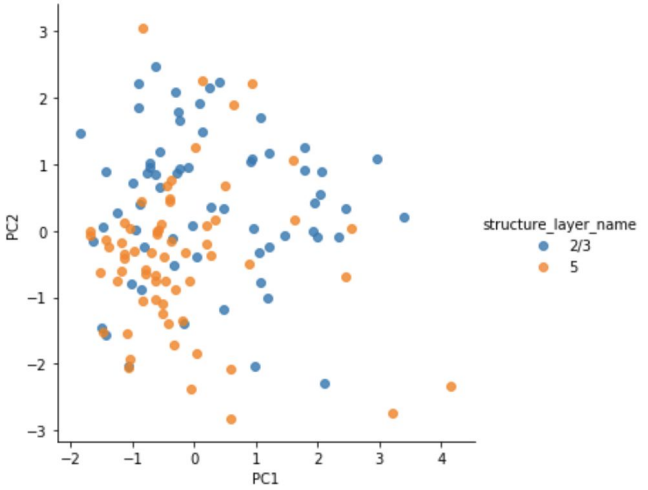
Assignments & project components **lose 10% each day they are late.**

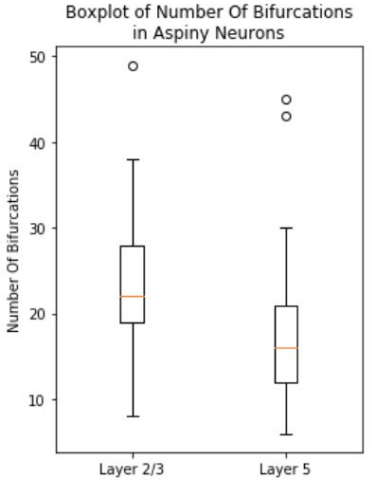
# Assignments

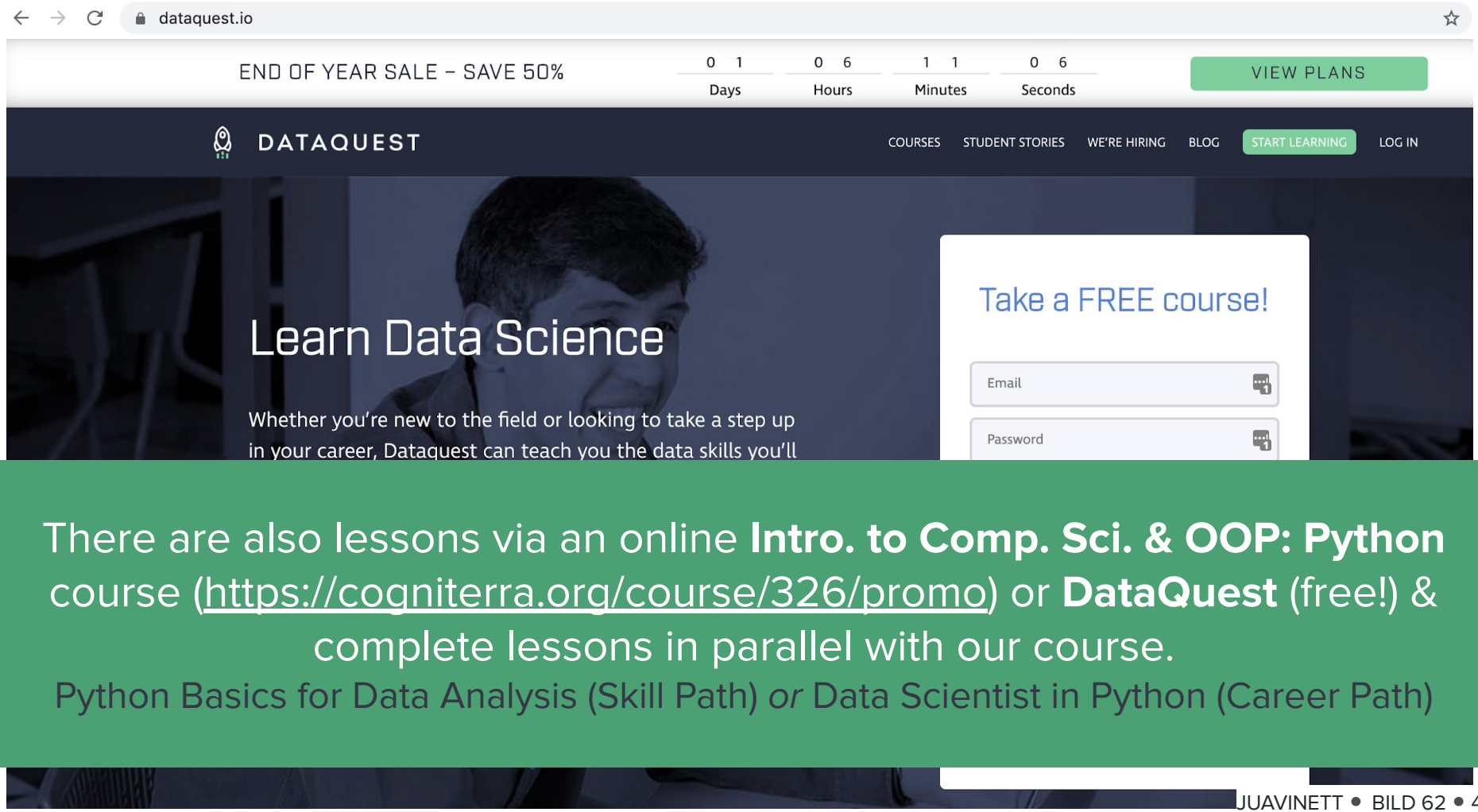
- Due **every Friday at 5 pm**
- Worth 2-4% each
- Completed individually
- Programmatically graded (via Datahub/NBGrader)
- In discussion, you'll cover how to submit these. There are also written instructions on Canvas.

# Final Project

- Completed in groups of 3
  - Your group will be automatically assigned based on the discussion section *that you regularly attend*.
- 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.






There are also lessons via an online **Intro. to Comp. Sci. & OOP: Python** course (https://cogniterra.org/course/326/promo) 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

*Vote on the entry survey!*

**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

canvas

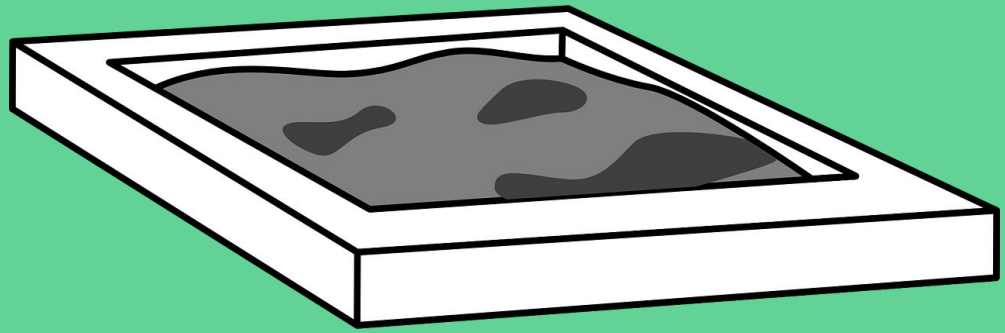
Submitting non-coding assignments & managing grades

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


Coding exercises & assignments


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_WI26

## Lectures

In other words, PDF slides shown during class.

Hosted on GitHub in the Lectures folder

## 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

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

# 💫 THE MAGIC LINK FOR THIS COURSE:

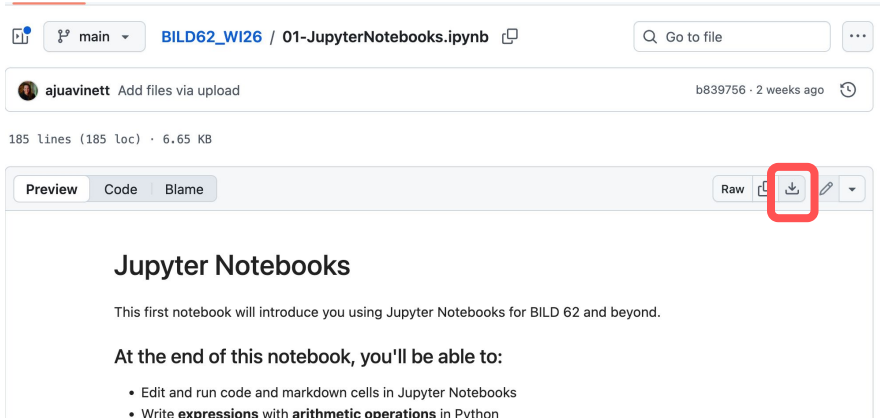
*Sync with your datahub:*

https://datahub.ucsd.edu/hub/user-redirect/git-sync?repo=https://github.com/BILD62/BILD62_WI26

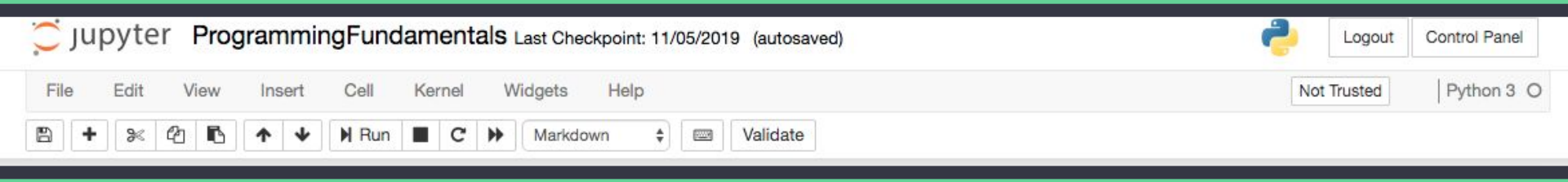
*Where our course content lives*

# To clone Materials to DataHub:

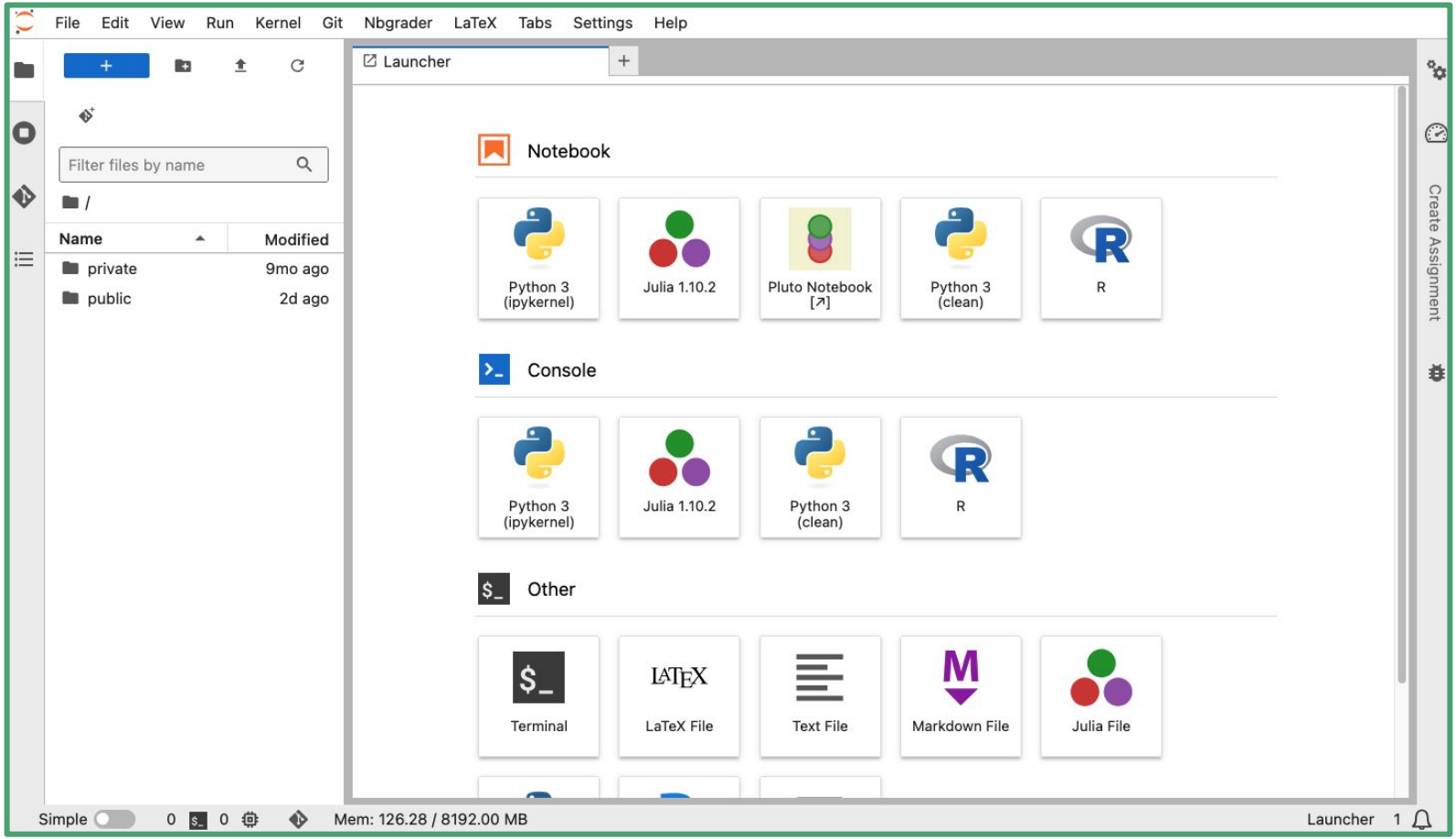
1. Click on the magic link.
2. Log in to DataHub as prompted.
3. You'll see the newest version of the course materials in your DataHub!
    - If you don't see the course materials, click the magic link one more time.
4. If you want a fresh copy of the materials,
   you can manually download it
   from our Github
   & upload to DataHub.



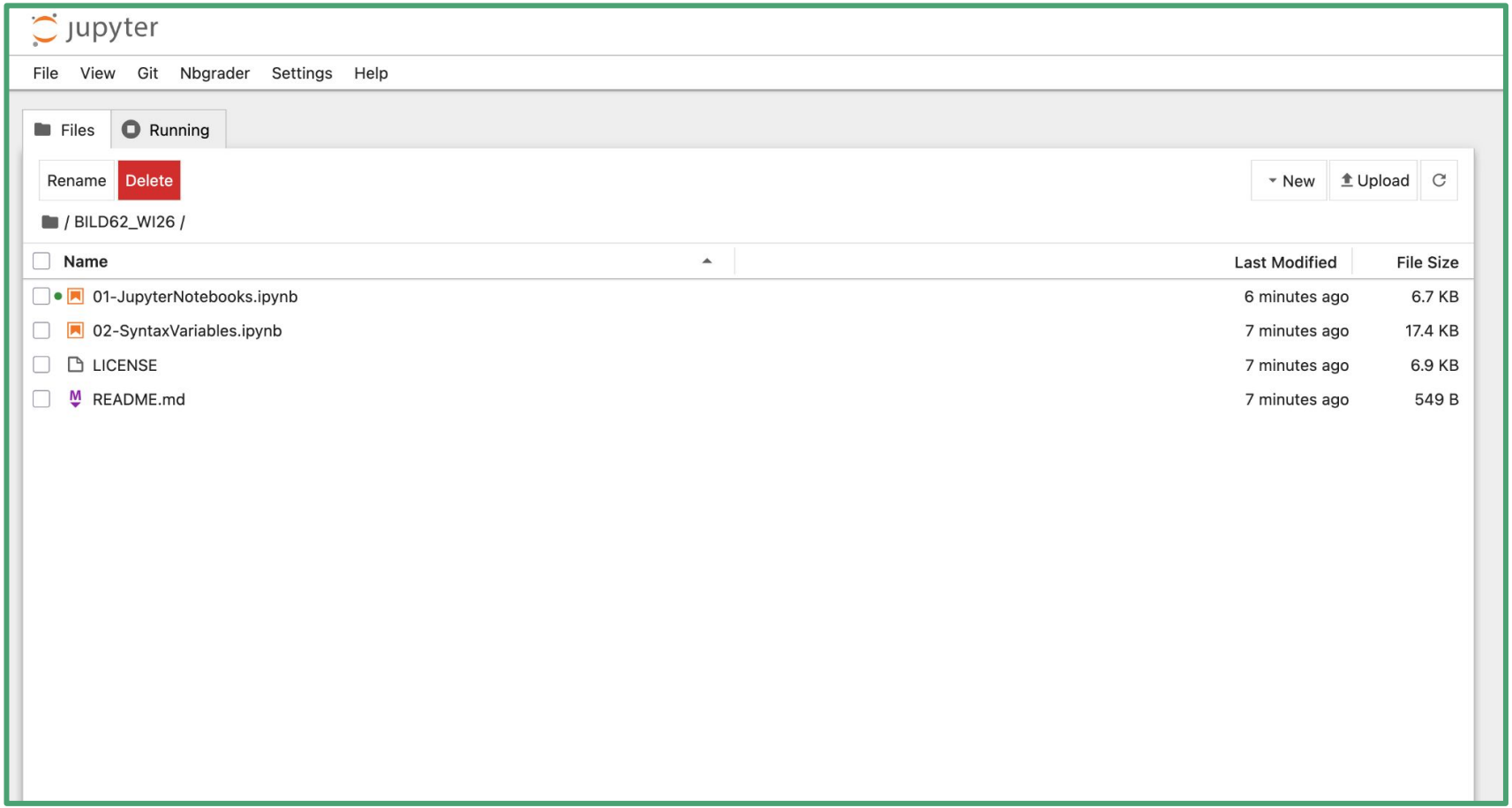
# Introduction to the UCSD DataHub & Jupyter Notebooks



# The **launcher**: what you'll see if you launch from datahub.ucsd.edu



**file browser view**: what you'll see if you launch from the magic link

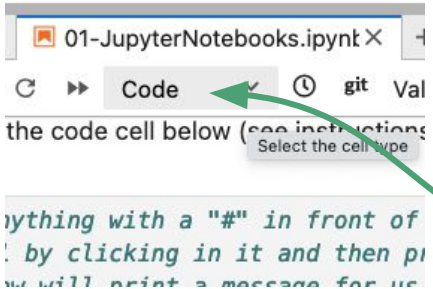


# About Jupyter Notebooks

- Jupyter is a loose acronym for Julia, Python, and R
- Run in a web browser but it's not *necessarily* online (it is online 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 (this is why it is used by many scientists!)
- **If you change anything in the cell, you need to re-run it.**

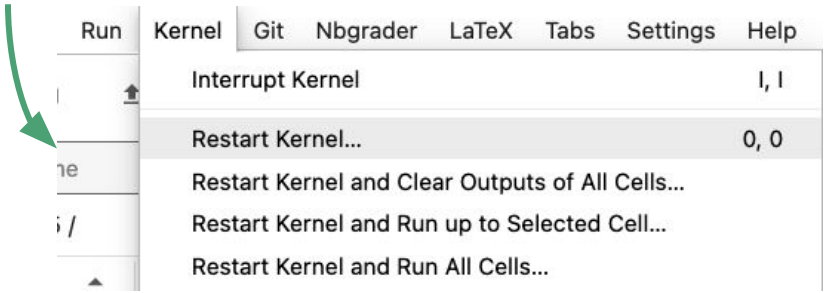
# Using Jupyter Notebooks

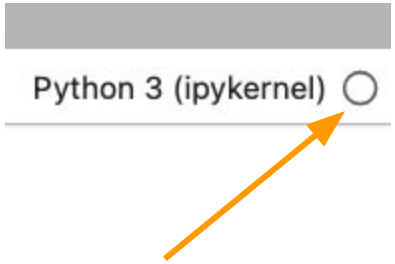
- **Cell**: the main organizational structure of the notebook
  - Use **Shift+Enter** to run a cell (or press Run)
  - You can run cells out of order, and move cells around!
  - Cells can be **code** (the default) or **markdown** (descriptive text or images)
    - Code cells have `[ ]`:
      - If there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



# **Kernel**: the engine that runs the code

- You can clear your **namespace** and get a fresh start by restarting the kernel
- Use **Kernel** menu to interrupt and/or restart if it gets stuck!



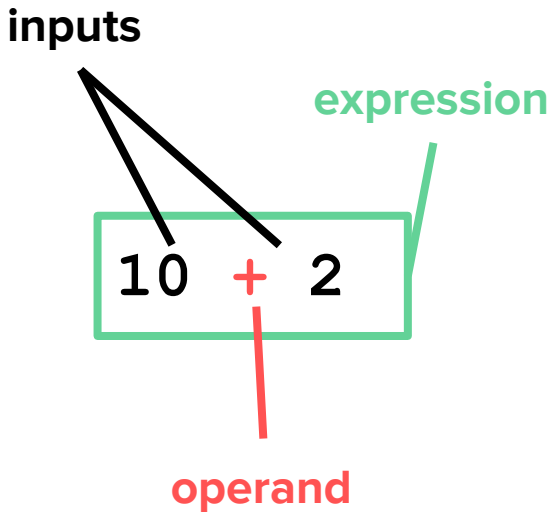

You can tell if the kernel is busy by whether or not the circle next to Python 3 (upper right corner) is filled or not. (filled = busy)

**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` |

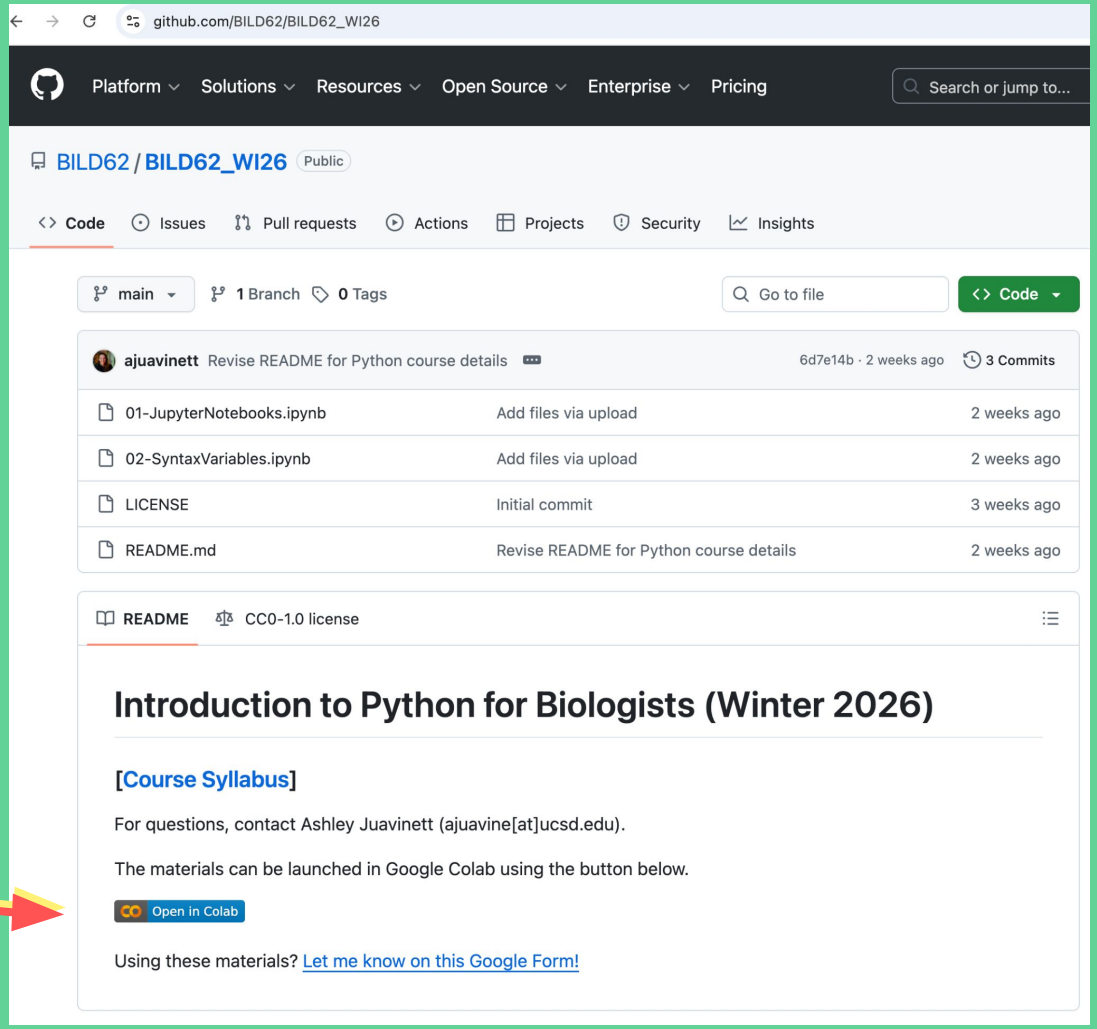



If you want a whole number (floor division), use // instead.

# *Let's code!*

Use the magic link (on Canvas and these slides) to sync up your DataHub with our folder, and open notebook 01.

If DataHub isn't working, use the "Open in Colab" link on our GitHub



# Before next class...

- Access Canvas (canvas.ucsd.edu) & DataHub (datahub.ucsd.edu)
- Fetch your first assignment — instructions in discussion section
- (Optional) Sign up for Cogniterra and/or DataQuest
- (Super Duper Optional) Install VS Code *or* Anaconda

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)!

# To interact with Jupyter Notebooks on your computer using VS Code *OPTIONAL*

1. Install VS Code for your operating system.
2. If you're using Windows, download 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_WI25.git`
4. File > Open Folder > Find the BILD62_WI25 folder you just created
5. Install ipykernel when prompted

# To interact with Jupyter Notebooks on your computer using Anaconda *OPTIONAL*

1. Install Anaconda with Python 3.7 for your operating system.
2. If you're using Windows, download 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_WI26.git`**
4. Open Jupyter Notebook. There are two ways to open:
   - In Terminal (Mac) or the Anaconda Prompt (Windows), type **`jupyter notebook`**
   - Open Anaconda Navigator and launch jupyter notebook
5. On the Jupyter landing page, navigate to the notebook and open it.
   - It will open in a browser but is *not* using an internet connection.