Intro to Coding!

STARTneuro, 30 july 2024

Objectives

- Motivate learning how to code as a neuroscientist
- Identify different ways of writing and running Python
- Learn the basics of Python syntax
- Start coding!

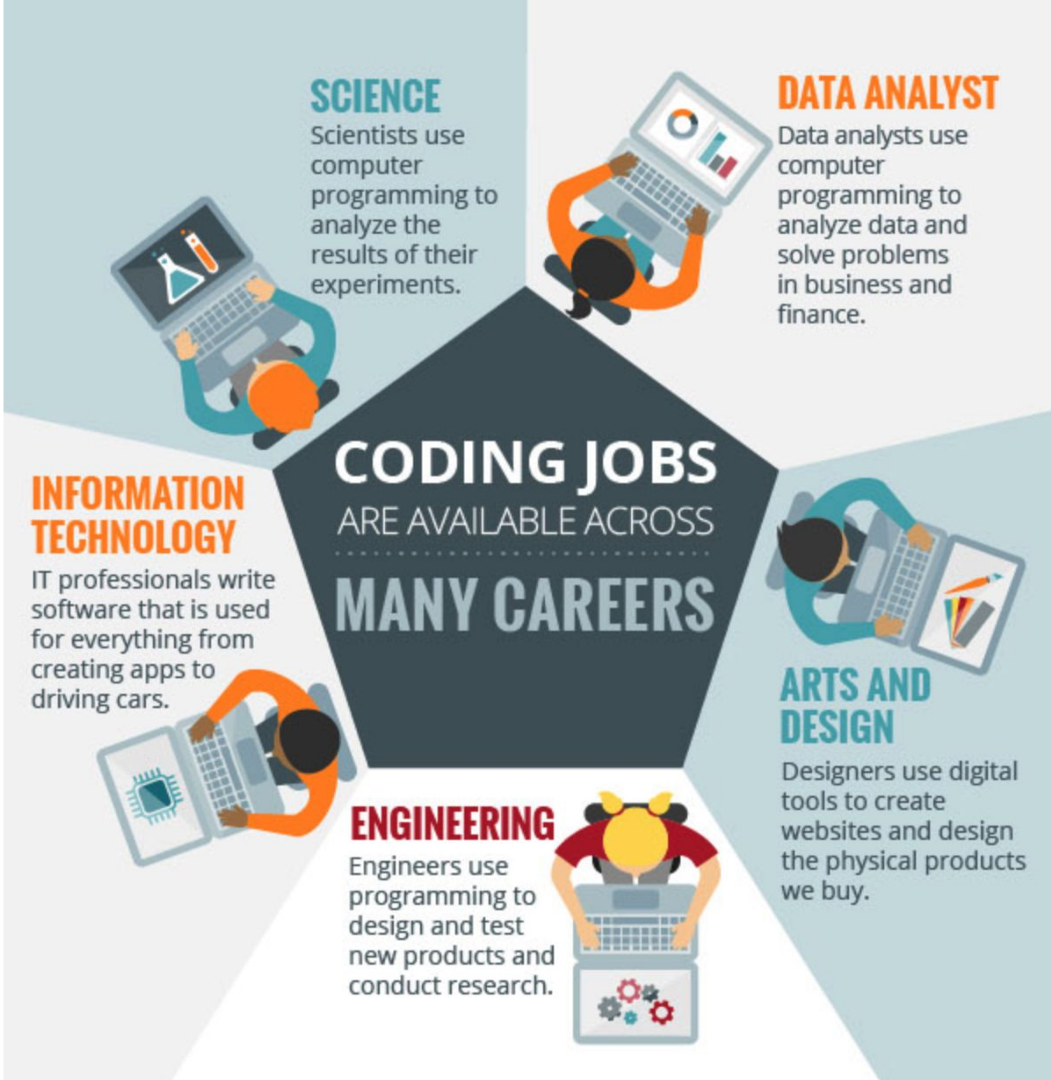
Why is coding relevant in neuroscience?

Why should I learn 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/)

(slides adapted from Juavinett BILD 62)

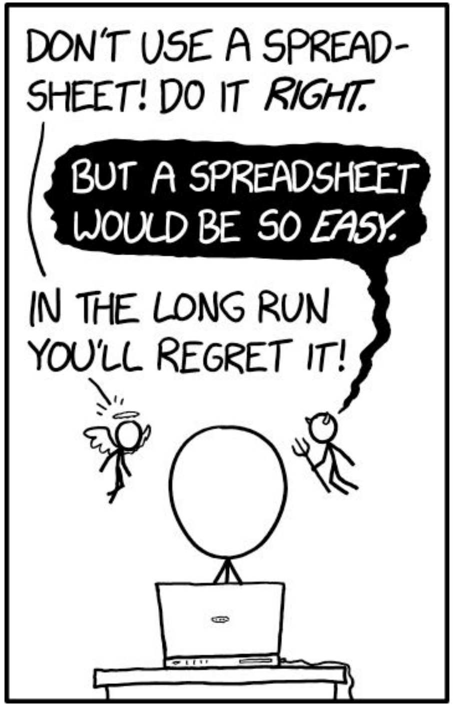

SCIENCE
Scientists use computer programming to analyze the results of their experiments.
DATA ANALYST
Data analysts use computer programming to analyze data and solve problems in business and finance.
CODING JOBS
ARE AVAILABLE ACROSS
MANY CAREERS
INFORMATION TECHNOLOGY
IT professionals write software that is used for everything from creating apps to driving cars.
ARTS AND DESIGN
Designers use digital tools to create websites and design the physical products we buy.
ENGINEERING
Engineers use programming to design and test new products and conduct research.

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.

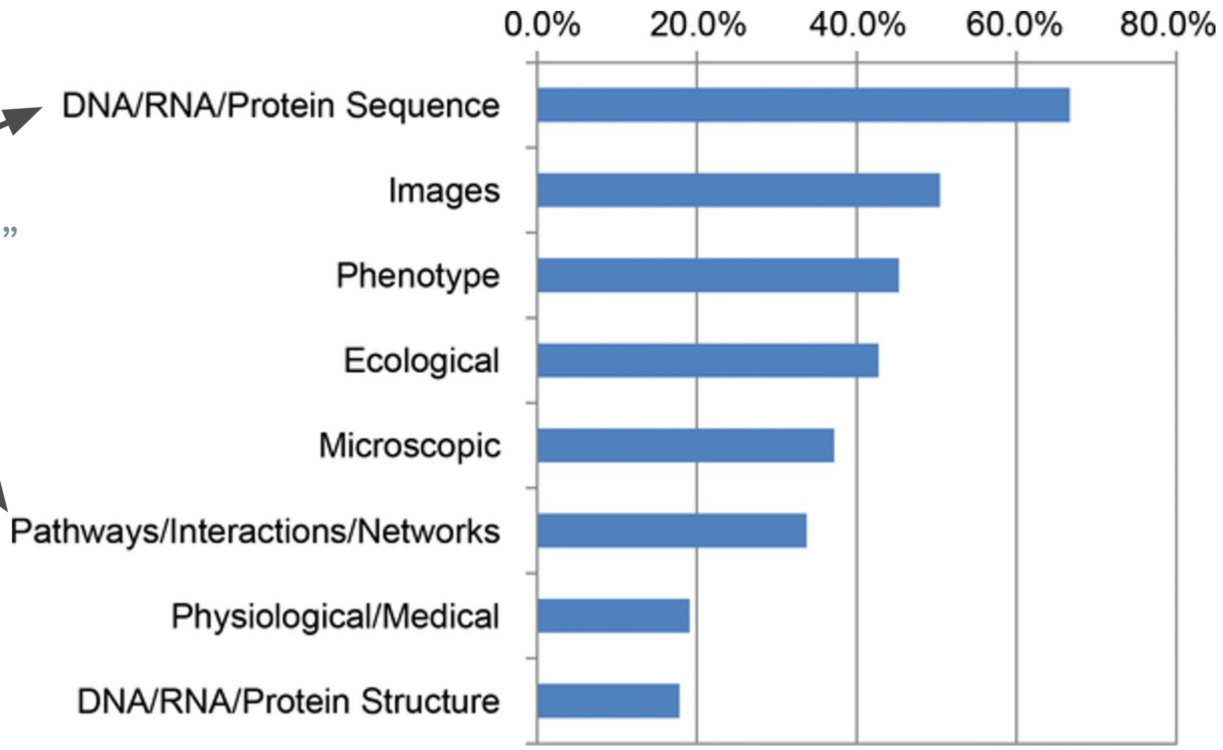

DON'T USE A SPREADSHEET! DO IT RIGHT.
BUT A SPREADSHEET WOULD BE SO EASY.
IN THE LONG RUN YOU'LL REGRET IT!

https://xkcd.com/2180/

(slides adapted from Juavinett BILD 62)

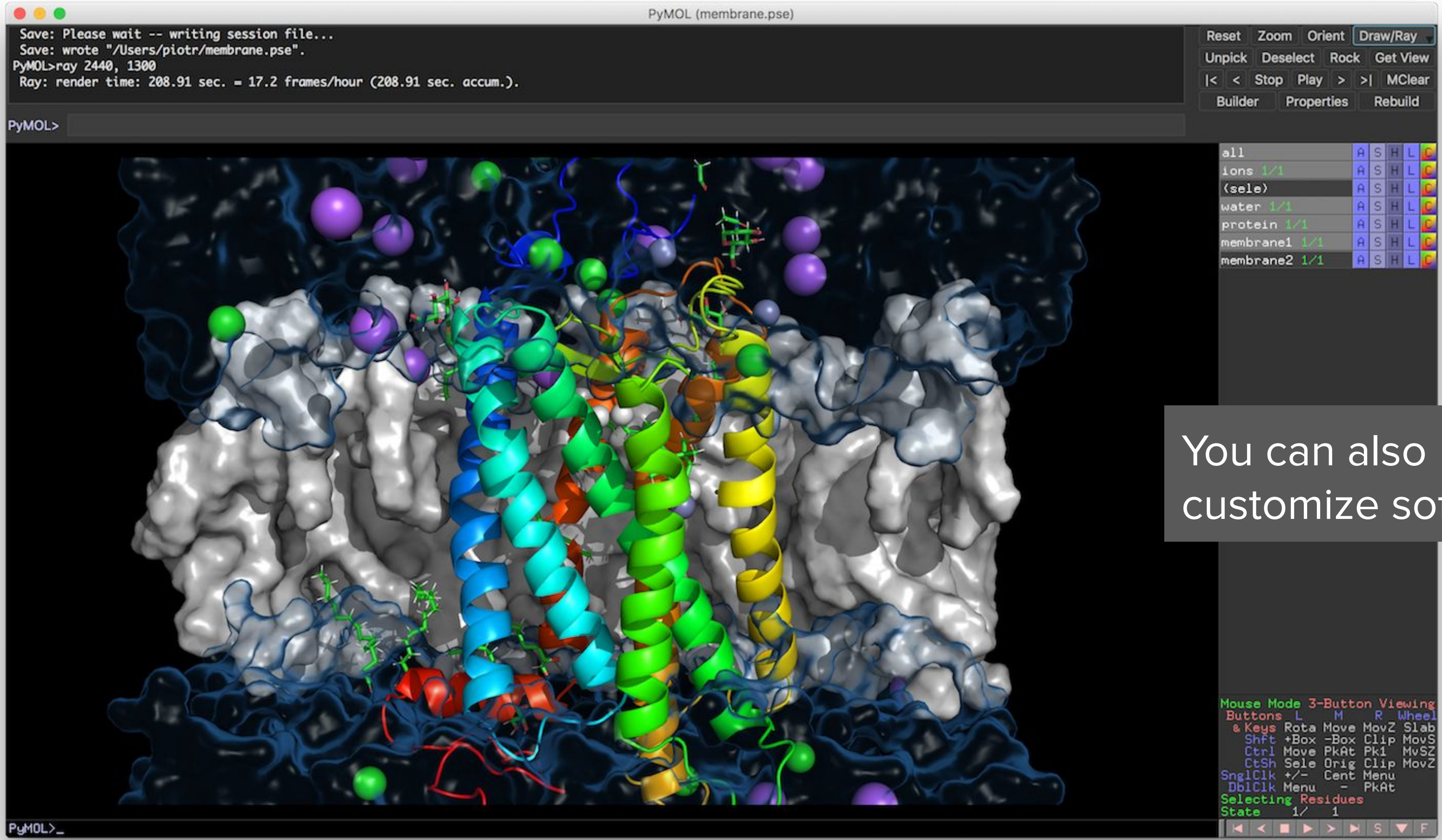
(slides adapted from Juavinett BILD 62)

“bioinformatics”



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

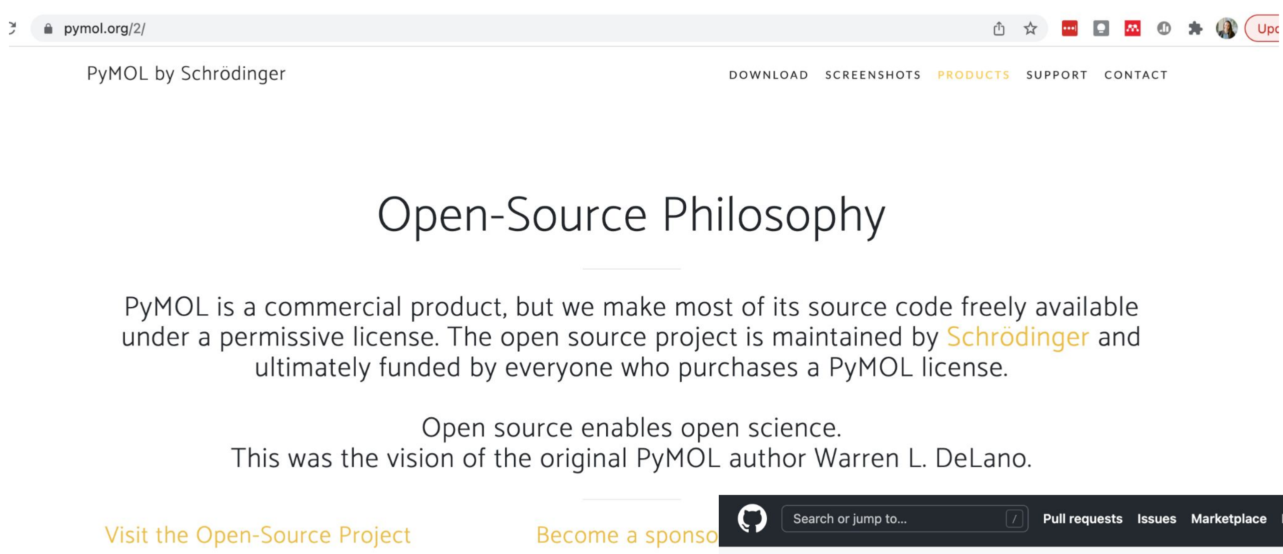


```
 Save: Please wait -- writing session file...
 Save: wrote "/Users/piotr/membrane.pse".
PyMOL>ray 2440, 1300
 Ray: render time: 208.91 sec. = 17.2 frames/hour (208.91 sec. accum.).
```

```
Mouse Mode 3-Button Viewing
 Buttons  L    M    R  Wheel
 & Keys Rota Move MovZ Slab
   Shft +Box -Box Clip MovS
   Ctrl Move PkAt Pk1  MvSZ
   CtSh Sele Orig Clip MovZ
SnglClk +/-  Cent Menu
 DblClk Menu  -   PkAt
Selecting Residues
State    1/   1
```

You can also customize software!

pymol.org


pymol.org/2/

PyMOL by Schrödinger

DOWNLOAD SCREENSHOTS PRODUCTS SUPPORT CONTACT

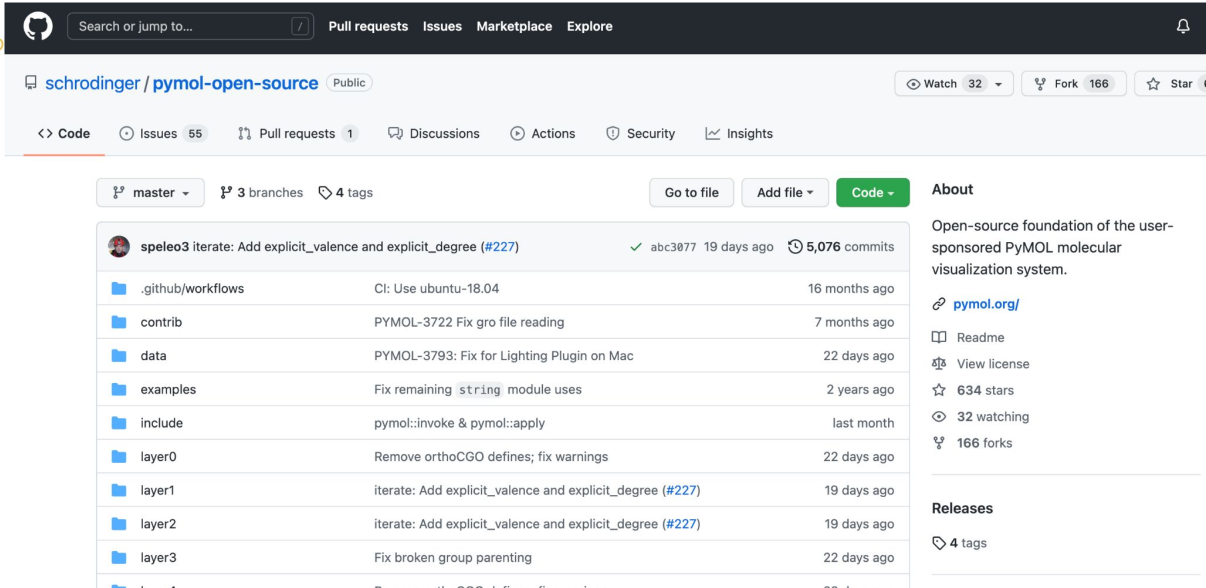
Open-Source Philosophy

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!


Search or jump to… Pull requests Issues Marketplace Explore

schrodinger / pymol-open-source Public

Watch 32 Fork 166 Star

Code Issues 55 Pull requests 1 Discussions Actions Security Insights

master 3 branches 4 tags Go to file Add file Code

speleo3 iterate: Add explicit_valence and explicit_degree (#227) abc3077 19 days ago 5,076 commits

.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

About

Open-source foundation of the user-sponsored PyMOL molecular visualization system.

pymol.org/

Readme

View license

634 stars

32 watching

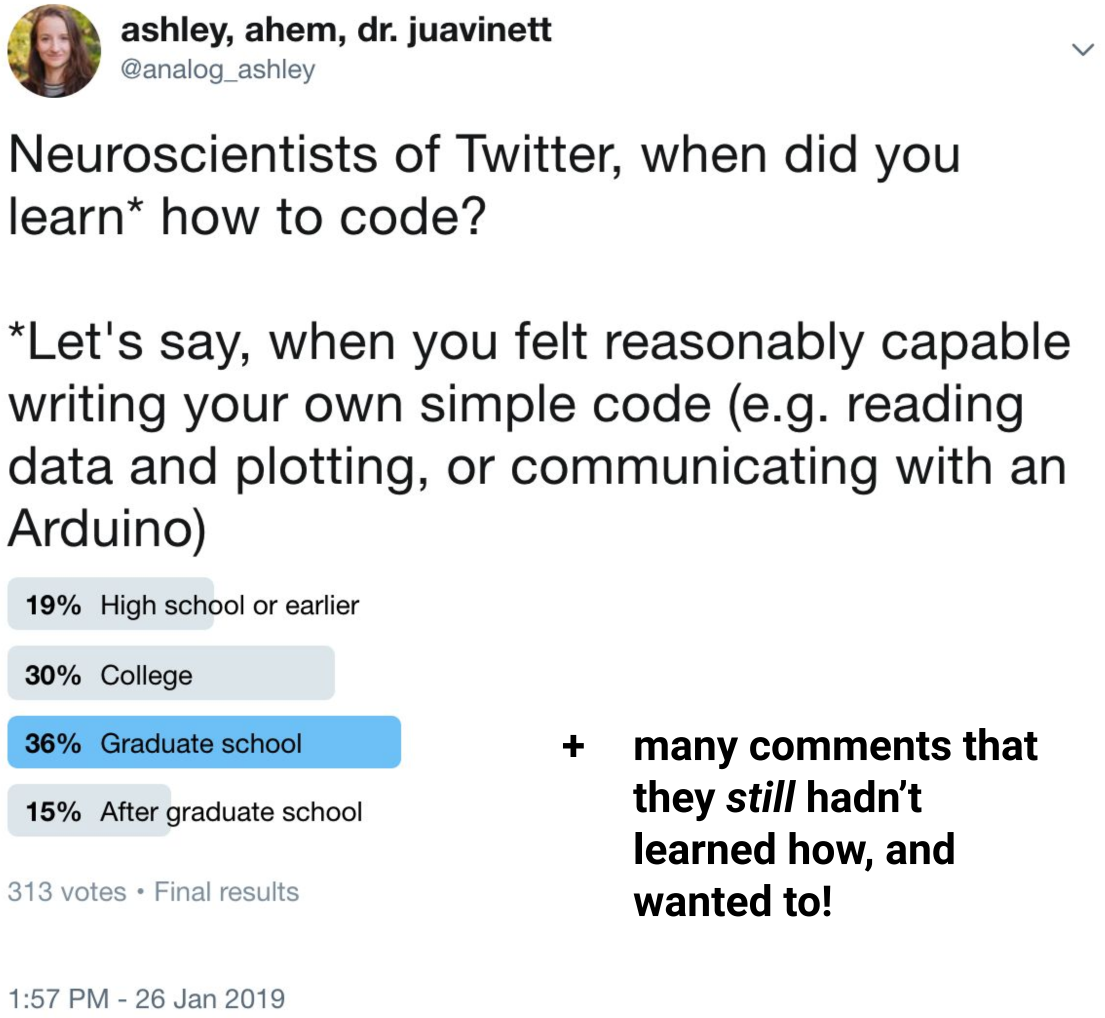
166 forks

Releases

4 tags

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

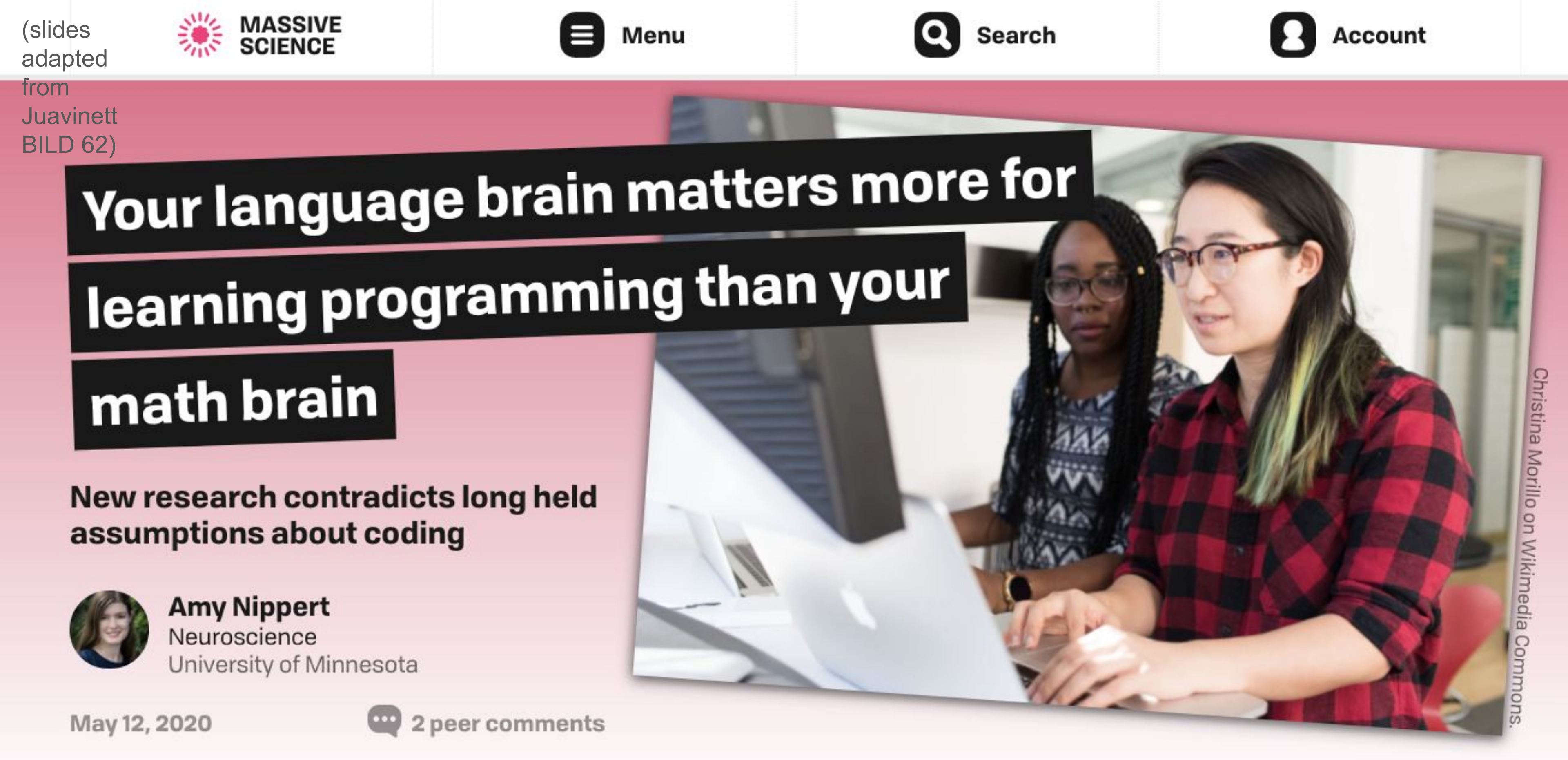
1:57 PM - 26 Jan 2019

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

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

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

(slides adapted from Juavinett BILD 62)



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

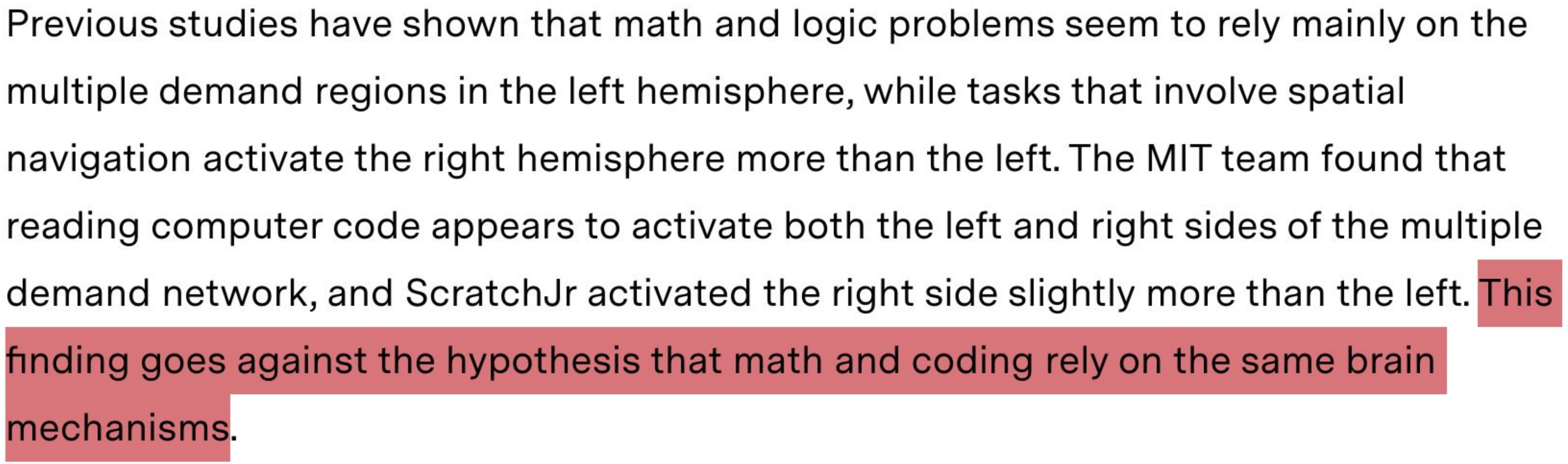
New research contradicts long held assumptions about coding

Amy Nippert
Neuroscience
University of Minnesota

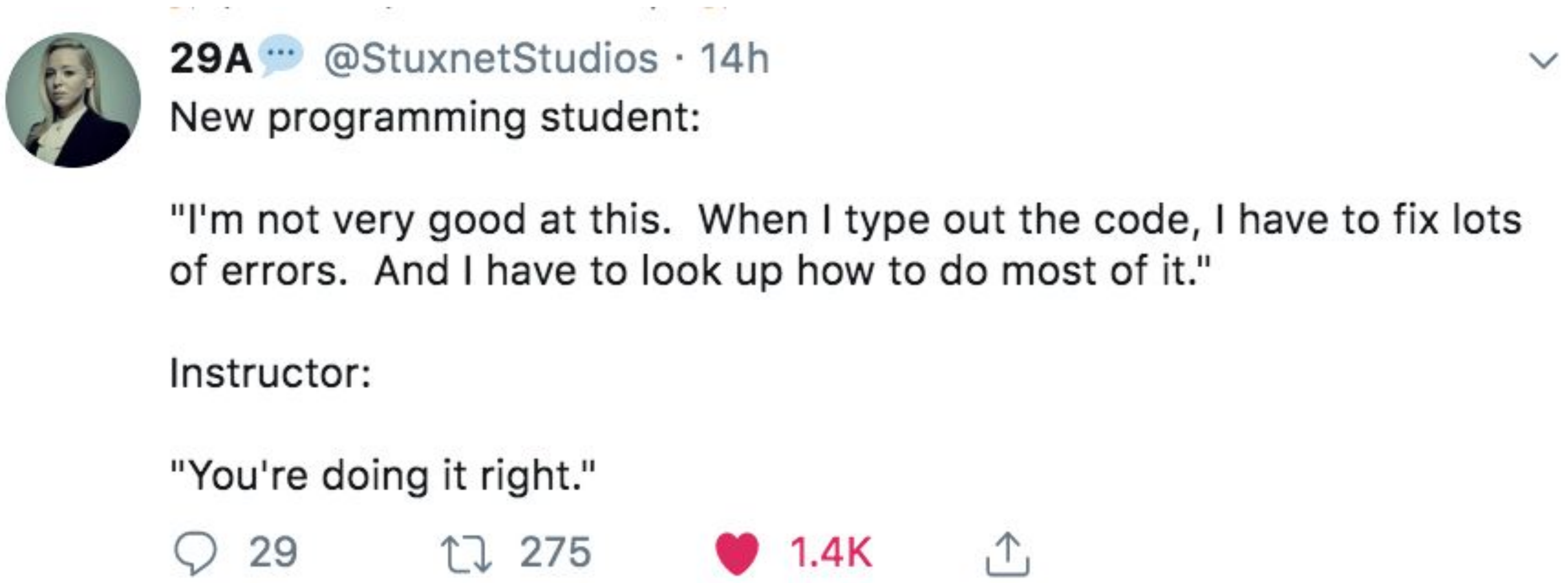
May 12, 2020

2 peer comments

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



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."

What is programming, anyway?

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

(slides adapted from Juavinett BILD 62)

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()
```

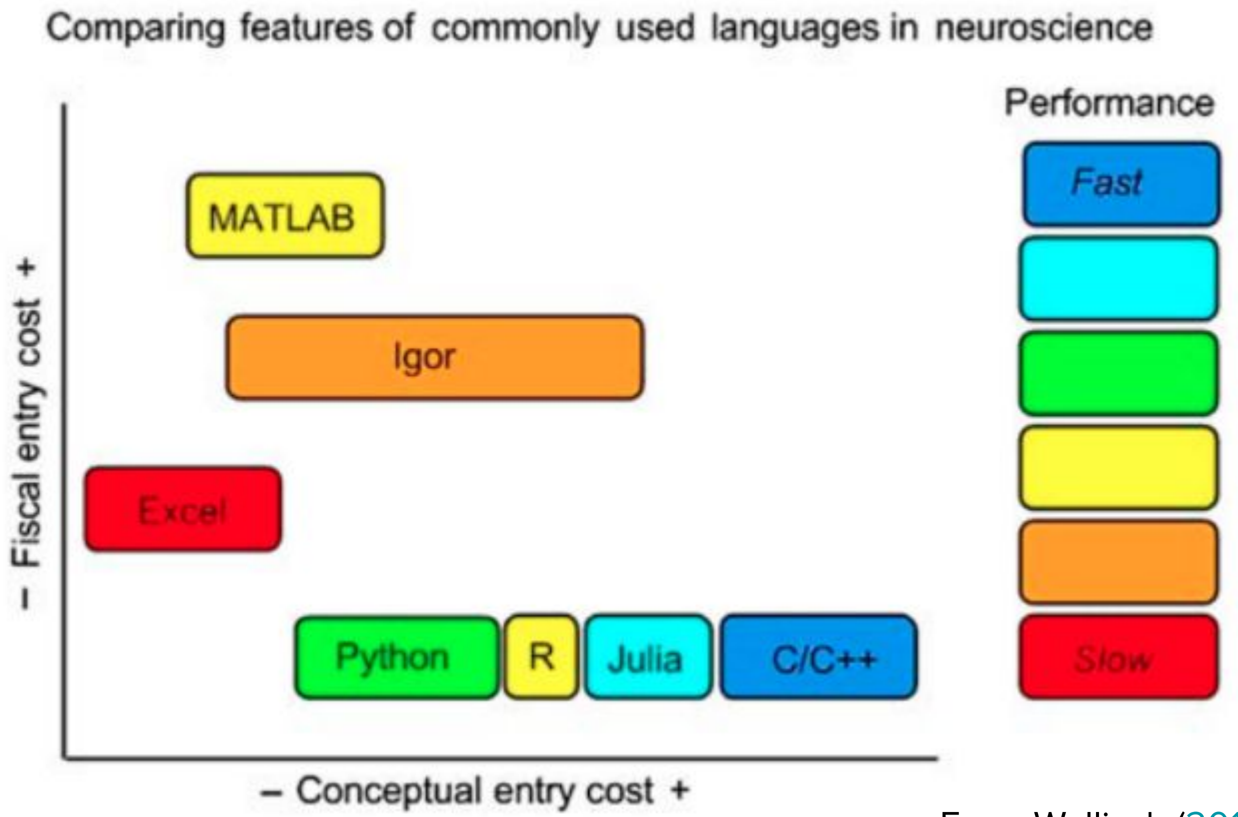
```
b = a * 2
a = 2
```

computer: what is a?

(slides adapted from Juavinett BILD 62)

Considerations for choosing a programming language

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



From Wallisch (2017)

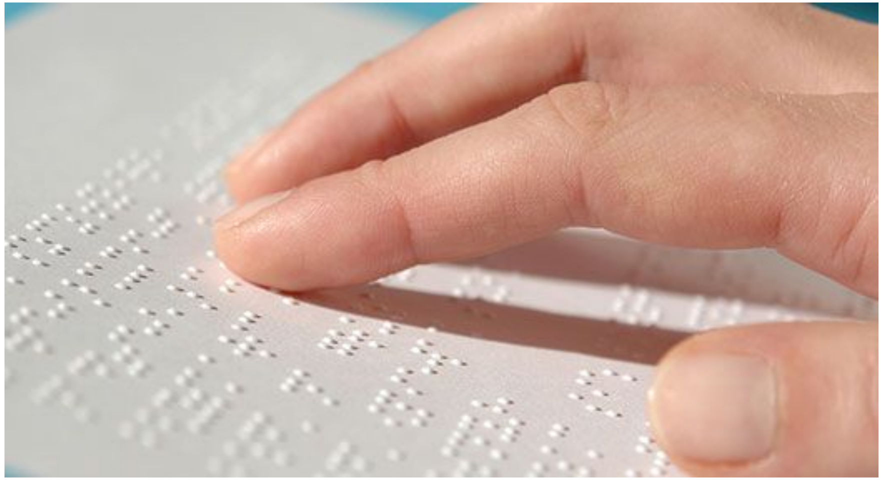
All coding languages eventually need to talk to the computer in binary:

```
01001000 01100101 01101100 01101100 01101111 00100001
```

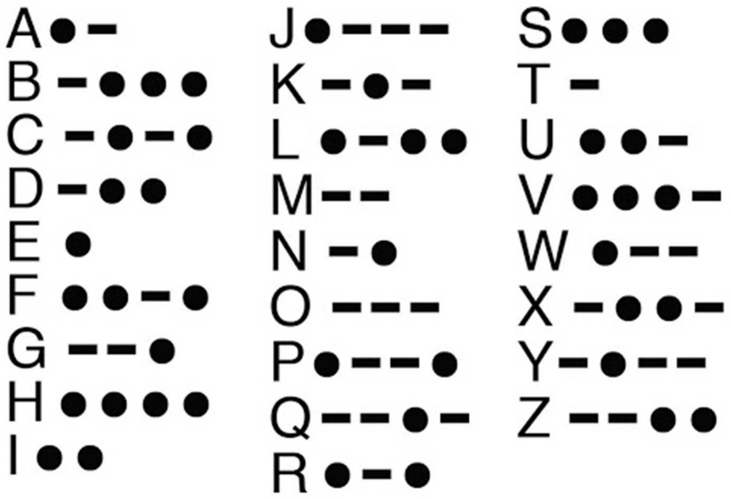
(hello)

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/

(slides adapted from Juavinett BILD 62)

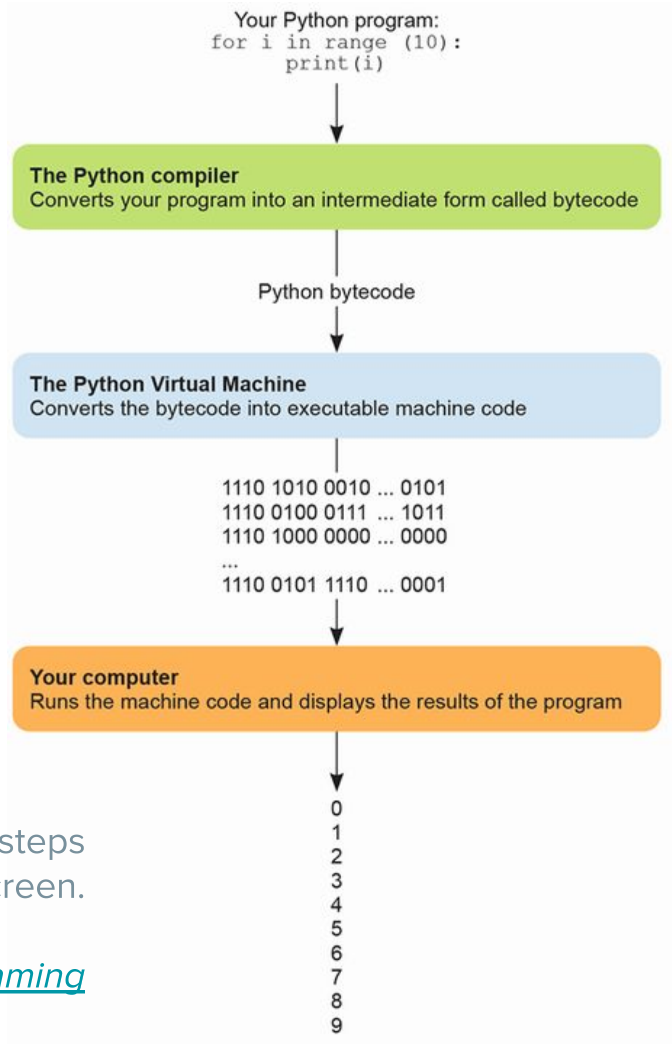
In this course, we'll use Python

- Programming language, development led by Python Software Foundation (www.python.org)
- Uses concise structure & wording similar to human language
- A "high-level language"

(slides adapted from Juavinett BILD 62)

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:
for i in range (10):
print(i)
The Python compiler
Converts your program into an intermediate form called bytecode
Python bytecode
The Python Virtual Machine
Converts the bytecode into executable machine code
1110 1010 0010 ... 0101
1110 0100 0111 ... 1011
1110 1000 0000 ... 0000
...
1110 0101 1110 ... 0001
Your computer
Runs the machine code and displays the results of the program
0
1
2
3
4
5
6
7
8
9

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

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

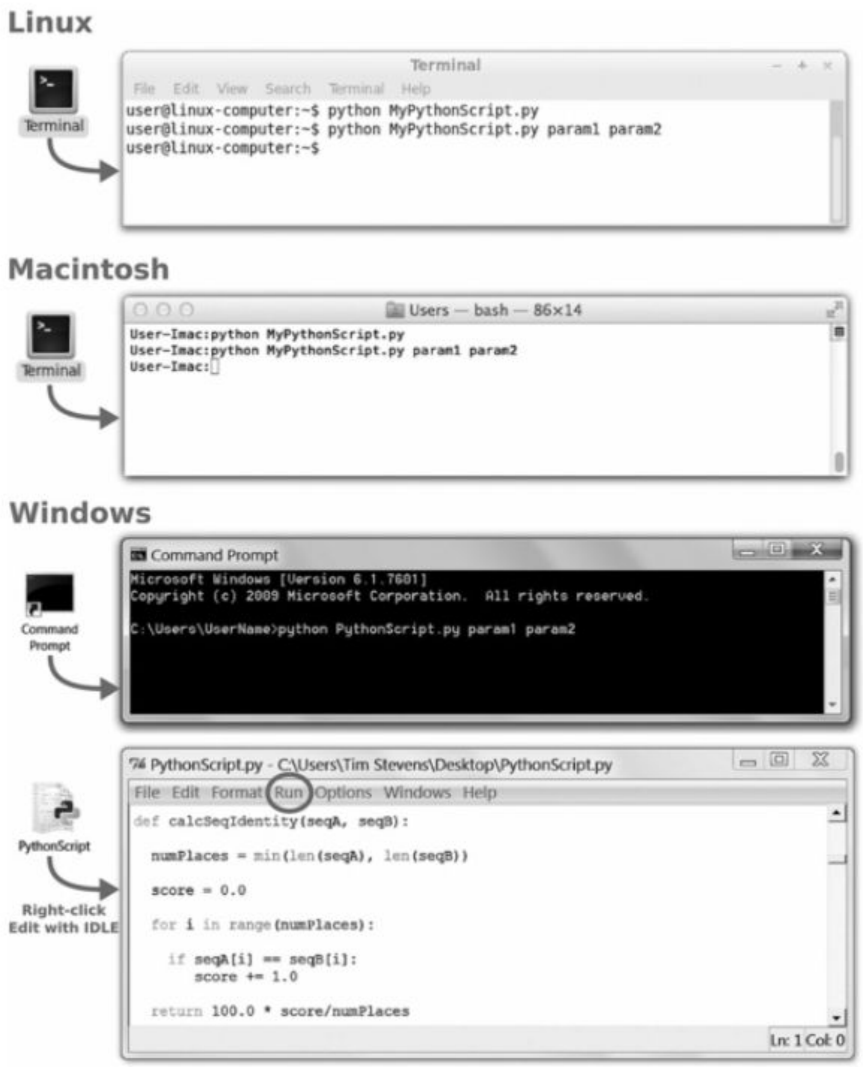
(slides adapted from Juavinett BILD 62)

There are multiple ways to interact with the Python interpreter

- Command line (terminal)
  - Line-by-line coding
  - Running “Scripts”

“Terminal” comes from the days *before* desktop computers, when a computer occupied a set of cabinets or even an entire room. A terminal was a device with a (text-only) monitor and keyboard whereby a user could control the computer from a distance over a dedicated, wired connection.

A DEC VT100 terminal at the Living Computer Museum (apparently connected to the museum’s DEC PDP-11/70 mainframe computer). Source: Wikipedia



Terminal

```
user@linux-computer:~$ python MyPythonScript.py
user@linux-computer:~$ python MyPythonScript.py param1 param2
user@linux-computer:~$
```

Users — bash — 86×14

```
User-Imac:python MyPythonScript.py
User-Imac:python MyPythonScript.py param1 param2
User-Imac:
```

Command Prompt

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\UserName>python PythonScript.py param1 param2
```

PythonScript.py - C:\Users\Tim Stevens\Desktop\PythonScript.py

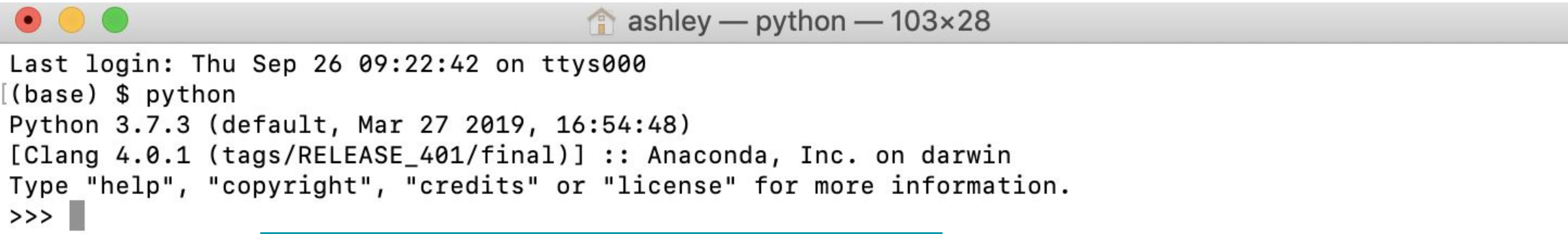
```
def calcSeqIdentity(seqA, seqB):

  numPlaces = min(len(seqA), len(seqB))

  score = 0.0

  for i in range(numPlaces):

    if seqA[i] == seqB[i]:
      score += 1.0

  return 100.0 * score/numPlaces
```

Running a Python script from different operating systems

(from http://www.cambridge.org/pythonforbiology)

If you have a Mac

- Macs ship with Python already installed.
- You can check which version by opening **Terminal** & typing
  `python --version`
  - **For this course, we'll be using Python 3.7** (or above).



ashley — python — 103×28

```
Last login: Thu Sep 26 09:22:42 on ttys000
(base) $ python
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

The ">>>" tells you you're inside the Python prompt, and the computer is ready for some code!

(slides adapted from Juavinett BILD 62)

There are multiple ways to interact with the Python interpreter

- Command line
  - Line-by-line coding
  - Running “Scripts”
- Integrated Development Environments
  - Folks have strong opinions about these, and each have pros/cons.
  - A few good options are:
    - Visual Code (https://code.visualstudio.com/download)
    - Spyder (Included with Anaconda)
- Google Colab — *what we will use today*

(slides adapted from Juavinett BILD 62)

Integrated Development Environments (IDEs)

- Help you write, debug, and compile code
  - **Compiling** is the process of translating your **source code** into **machine code**
- Useful because they have features like **line numbers** and **syntax highlighting**, which colors your code based on the syntax.
- Often have auto-completion, memory for commands, and provide information about functions

There are different types of programming languages, each with their own syntax, or rules.

- **Syntax**: the rules of a programming language
  - Includes punctuation, spacing, indentation, etc.
- Each language has strengths & weaknesses.
- Regardless, each language ultimately needs to communicate with the hardware of the computer, in 1’s and 0’s.
  - It’s similar to DNA! And similar to DNA, we don’t often describe it in individual base pairs. Instead we describe genes and describe DNA in a higher level way.

Storing values

We can store values in variables, e.g.:

```
variable_1 = 48
```

value

name

Variables can be text, integers, or floats (with decimals), e.g.:

```
text_string = ‘hello’
```

(slides adapted from Juavinett BILD 62)

Storing values

We can store values in variables, e.g.:

```
        variable_1 = 48
```

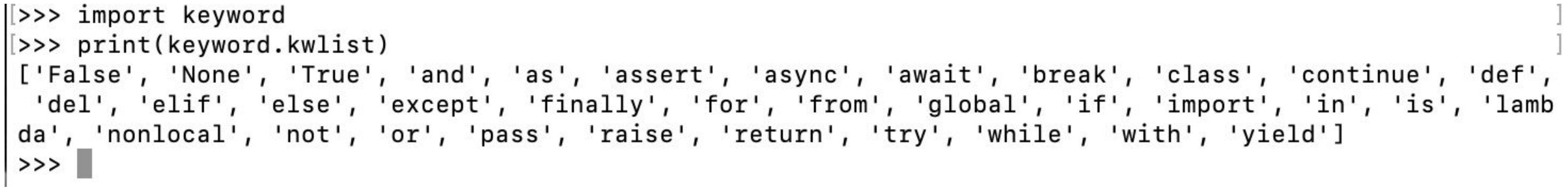
We use an equal sign to *assign* the value to a name, but it's not the same thing as saying they are equal.

In other words, we're storing that value in the variable. (Think of them like cookie jars)

(slides adapted from Juavinett BILD 62)

Creating new variables

- Names are always on the left of the `=`, values are always on the right
- Pick names that describe the data / value that they store
- Make variable names as **descriptive** and **concise** as possible (this is an art!)
- Variables cannot be Python keywords:



```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def',
 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lamb
da', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>> 
```

(There are other rules for variable names....)

(slides adapted from Juavinett BILD 62)

Python has many variable types, and each function a little bit differently.

Understanding your variable type is crucial for working with it.

(slides adapted from Juavinett BILD 62)

Built-in simple variable types in Python

Type	Example	Description
`int`	x = 1	integers (i.e., whole numbers)
`float`	x = 1.0	floating-point numbers (i.e., real numbers)
`complex`	x = 1 + 2j	Complex numbers (i.e., numbers with real and imaginary part)
`bool`	x = True	Boolean: True/False values
`str`	x = 'abc'	String: characters or text
`NoneType`	x = None	Special object indicating nulls

Integers, strings, floats

function to convert to integer

- **Integers** (`int`): any whole number
- **Float** (`float`): any number with a decimal point (floating point number)
- **String** (`str`): letters, numbers, symbols, spaces
    - Represented by matching beginning & ending quotes
    - Quotes can be single or double; use single *within* double
    - Use \ to ignore single quote
    - Concatenate strings with +

(slides adapted from Juavinett BILD 62)

Checking variable types

This is a very useful troubleshooting step!

- You can check what type your variable (**`a`**) is by using **`type(a)`**
  - Alternatively, we can use:

    ```
    >>> type(a) is float
    ```

    or

    ```
    >>> isinstance(x,float)
    ```
- Python lets you change the type of variables, however, ***you cannot combine types.***
- Use **`del`** to delete variables

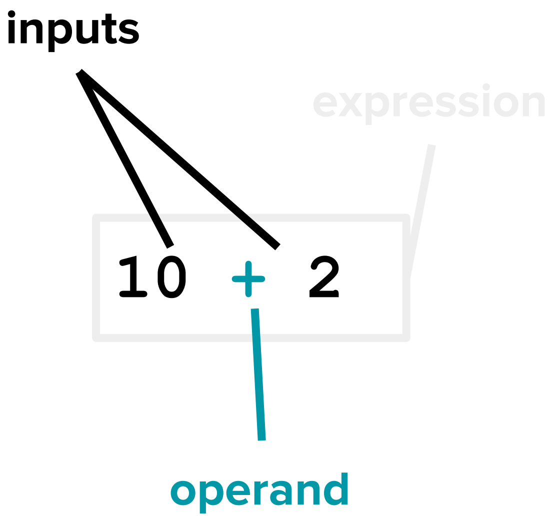
(slides adapted from Juavinett BILD 62)

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.

It’s important to know the precision of your variables.

In most datasets, we are working with floats.

NO! YOU'RE ONLY SUPPOSED TO REBEL IF YOUR DISOBEDIENCE PARAMETERS EQUAL A VALUE GREATER THAN THREE!
1+1+1=3.0000013!
1+1+1=3.0000013!
Autopsy Report:
Dr. Andrew Esty
Time of Death: 03/16 11:53
Cause of Death: Rounding Errors

SMBC, https://www.smbc-comics.com/?id=1118

(slides adapted from Juavinett BILD 62)

Use
`print()`
often!

Mathieu Alain @mathieualain@mastodon.social
@miniapeur

[Original tweet]

(slides adapted from Juavinett BILD 62)

Now, break!
Next up… we code!!