Welcome to BINF 5007

Introduction to Python

Learning Objectives

1. Understand the Course: Summarize the course description, objectives, and key policies.

2. Explore Bioinformatics: Review published research and the importance of programming in bioinformatics.

3. Differentiate Roles: Distinguish between users (analysts) and programmers (developers).

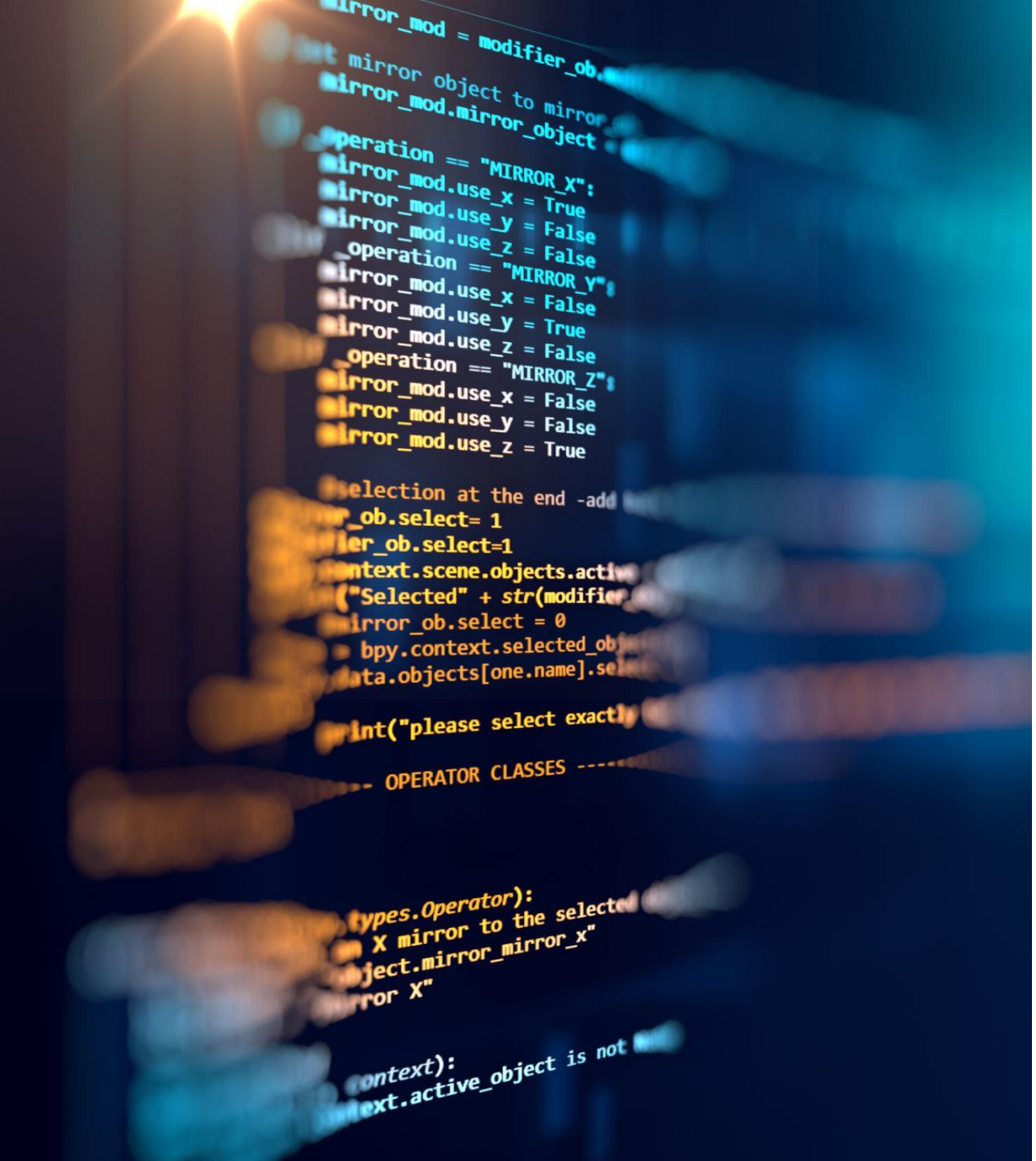
4. Learn Python Basics: Explain Python's relevance for biologists, apply basic rules, and use the interactive shell.

5. Set Up Tools: Install PyCharm and understand its use for Python programming.

Course Description

Focuses on the fundamental programming skills required in the bioinformatics industry. Python is the main programming language used. Topics include string operations, file manipulation, regular expressions,Data structures. Includes substantial out-of-classroom assignments.

In this course, we'll concentrate on the essential coding abilities vital for a career in bioinformatics. Effective program design stands out as a crucial skill. Python, being an advanced language, offers numerous advantages, especially in bioinformatics. This allows students to immediately create effective programs tailored for their biological data and research needs.



Course Objectives

- Understand Python Syntax
- Write Python Functions
- Write Abstract Python Functions and How to Refactor Python Code
- Implement Type Hinting
- Effectively Document Code
- Properly Handle Exceptions
- Leverage Advanced Data Structures Pandas
- Implement Python Modules and Packages
- Parse Text Files and Using Regular Expressions
- Manipulate a Wide Range of Data Formats (TSV, CSV, FASTA, VCF, XML, JSON)

Code Policies

We take the code policy very seriously

- Do not copy code!
  - We have ways to check for duplication
  - It is better to learn for yourself than risk getting a 0
  - We encourage working together but code should be *different*
- Programming style requirements:
  - PEP8 style Guide for Python Code
    - https://www.python.org/dev/peps/pep-0008/l
- Each assignment will specify whether
  - You may use code from other sources
  - If you can use non-standard Python Modules, i.e. do not unless you have been allowed to

Policies

- Communication Policies: For reaching out, please use **email**.
- Policies Regarding Submitting Projects:
  - Projects submitted after the deadline will lose 5% each day for up to five days (maximum penalty of 25%).
  - After five days, the work will be given a zero.
  - Please read more : https://healthsciences.humber.ca/current-students/resources/fhsw-student handbook.html?page=Policies%20%26%20Processes&tab=fhswpolicyCategory1

- Code policies – this is a good slide to introduce how you will be handling AI in the classroom

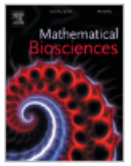
Why Python for Bioinformatics? What can it do?

Quantitative Modeling of COVID-19 Vaccines: A Bioinformatics Approach

ELSEVIER

Mathematical Biosciences

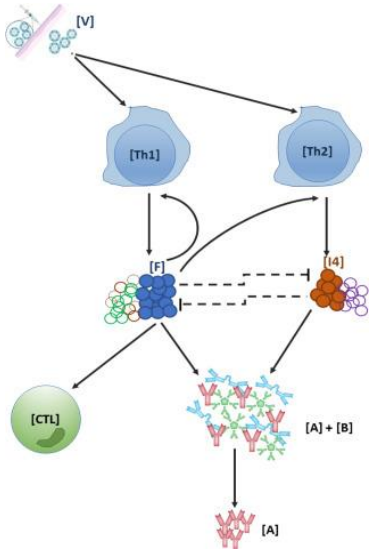
Volume 358, April 2023, 108970

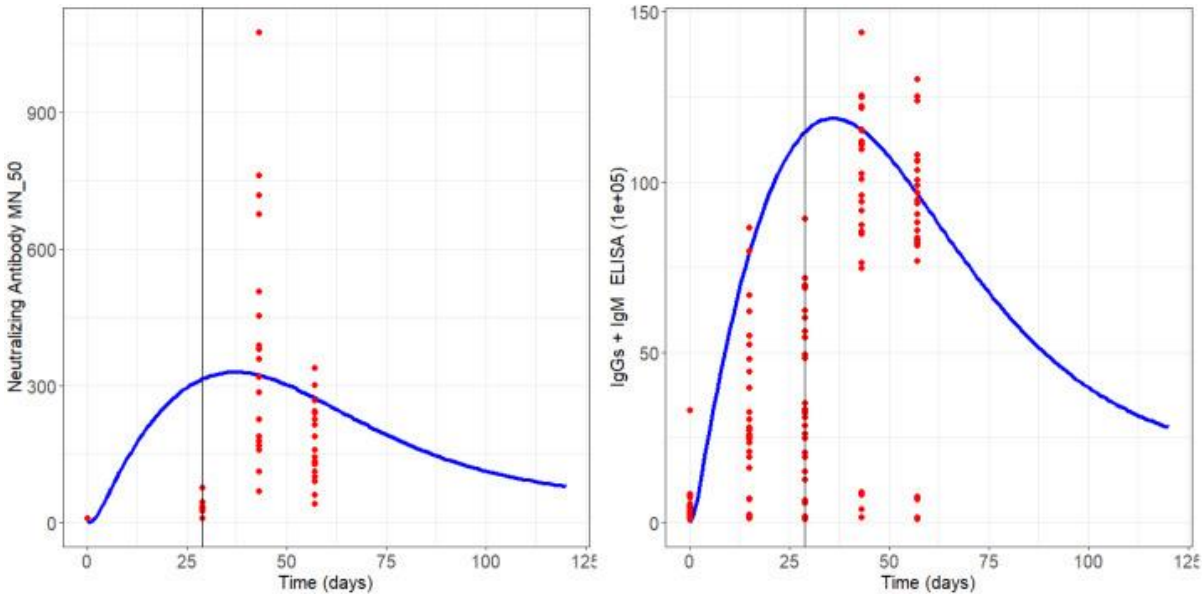


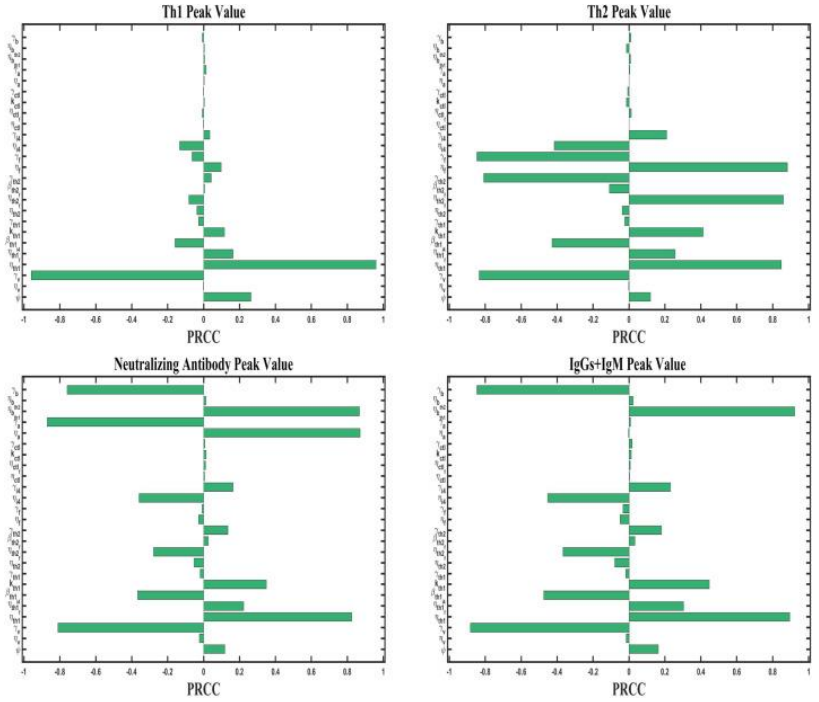
Original Research Article

A mathematical model of protein subunits COVID-19 vaccines

Samaneh Gholami [a], Chapin S. Korosec [a], Suzan Farhang-Sardroodi [b a], David W. Dick [a], Morgan Craig [c], Mohammad Sajjad Ghaemi [d], Hsu Kiang Ooi [d], Jane M. Heffernan [a]





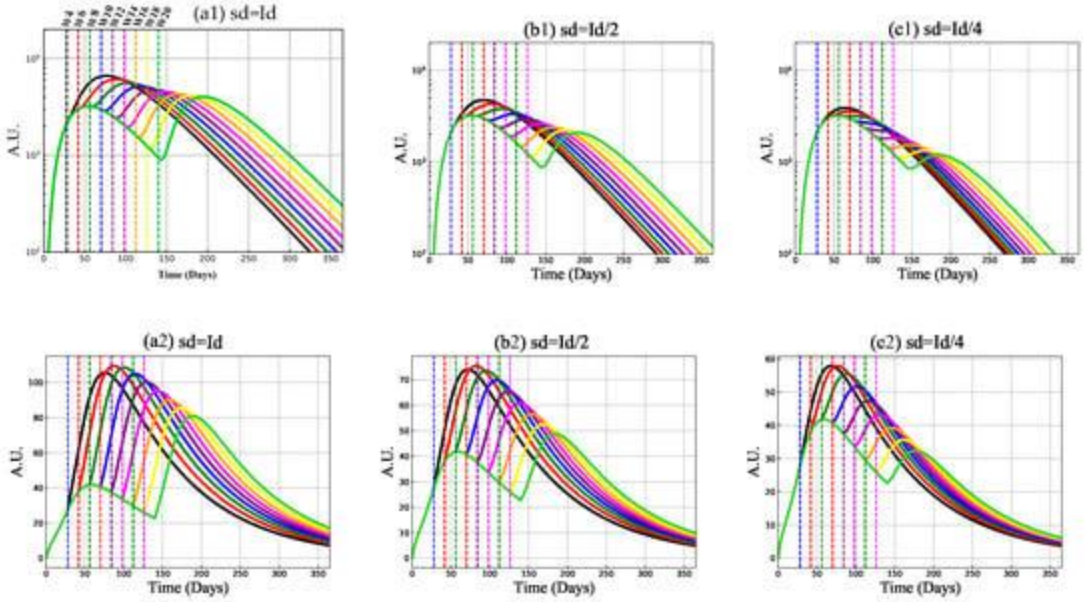


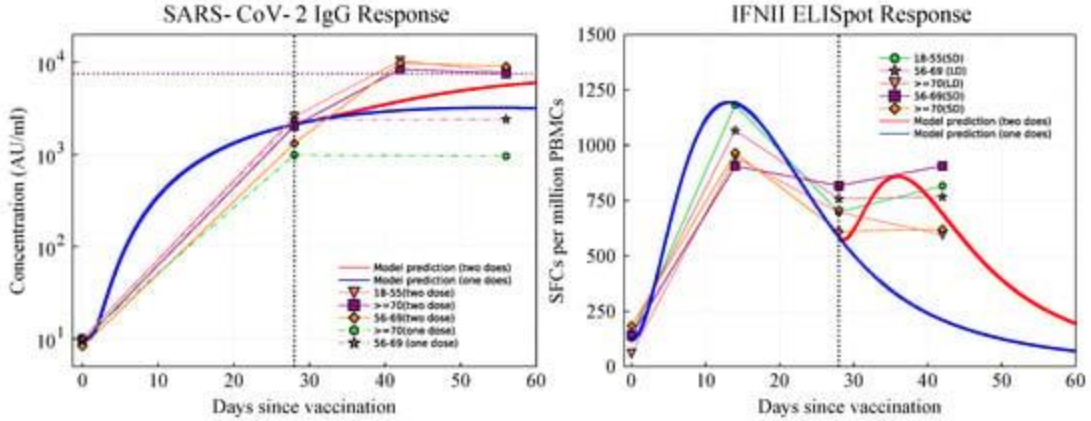
- https://www.sciencedirect.com/science/article/pii/S0025564230001111

A Bioinformatics Method for Modeling the AstraZeneca COVID-19 Vaccine

Analysis of Host Immunological Response of Adenovirus-Based COVID-19 Vaccines

by Suzan Farhang-Sardroodi [1,2,*], Chapin S. Korosec [1,2], Samaneh Gholami [1,2], Morgan Craig [3], Iain R. Moyles [2], Mohammad Sajjad Ghaemi [4], Hsu Kiang Ooi [4] and Jane M. Heffernan [1,2,*]





- https://www.mdpi.com/2076-393X/9/8/861

Bioinformatics Modeling of Moderna and Pfizer COVID-19 Vaccines

scientific reports

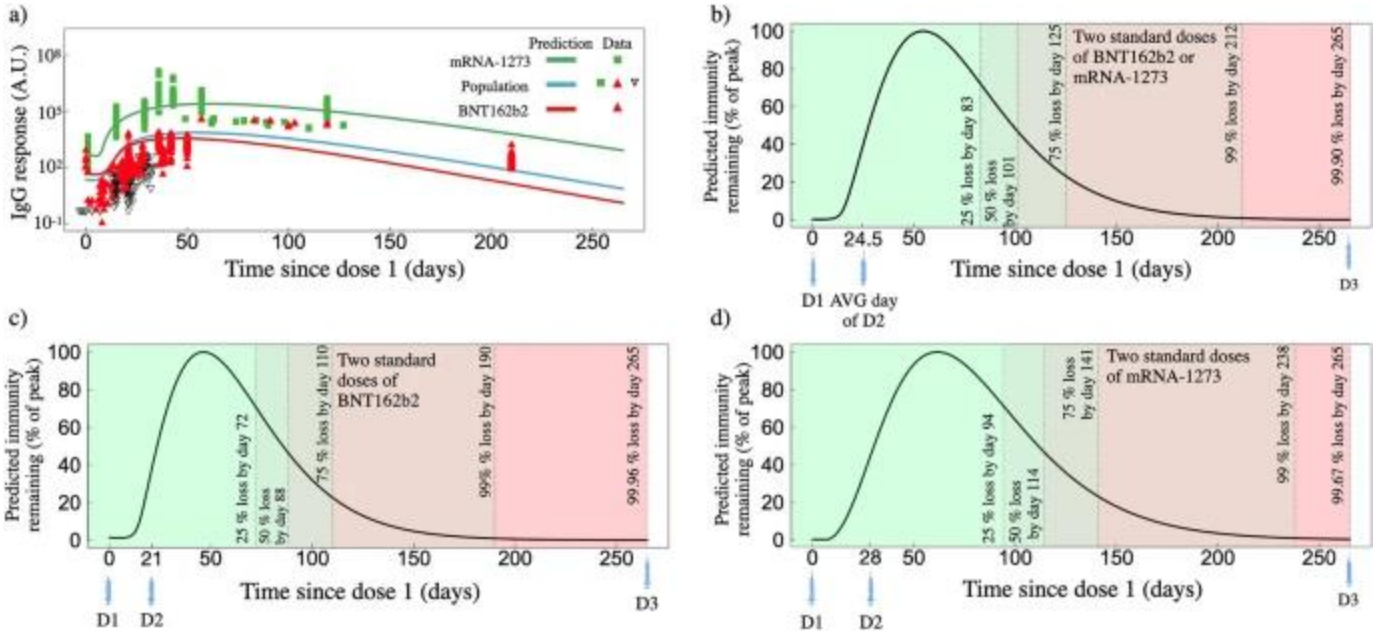
Explore content ⌄ About the journal ⌄ Publish with us ⌄

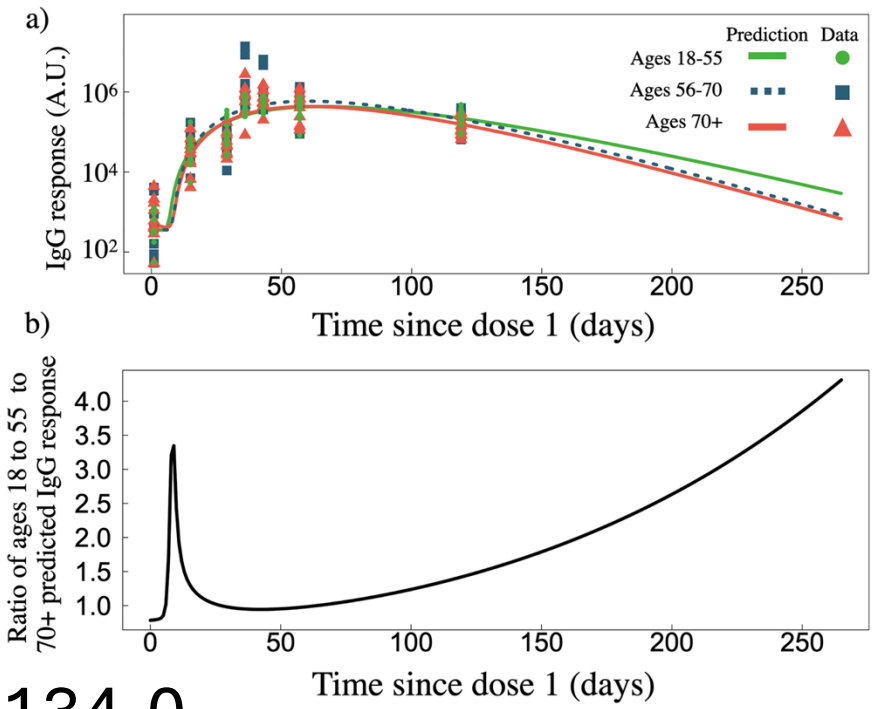
nature > scientific reports > articles > article

Article | Open access | Published: 08 December 2022

Long-term durability of immune responses to the BNT162b2 and mRNA-1273 vaccines based on dosage, age and sex

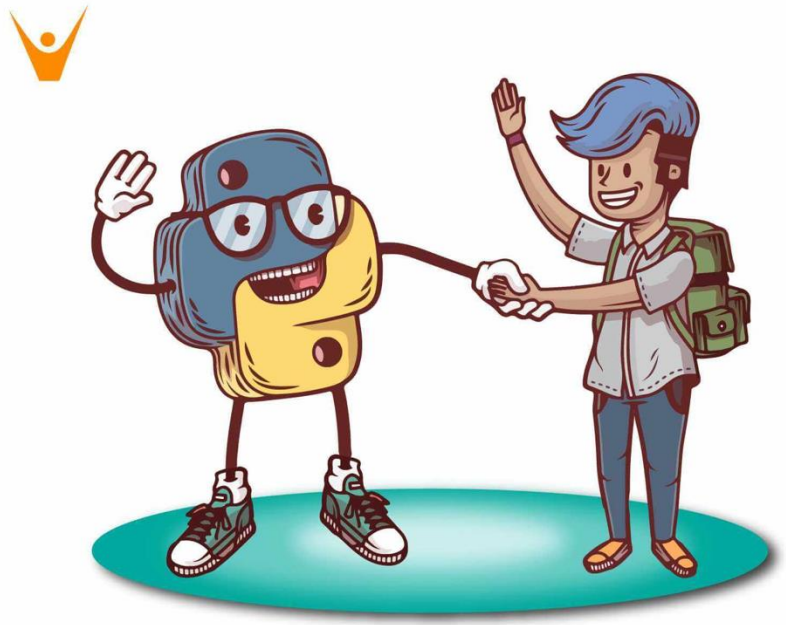
Chapin S. Korosec, Suzan Farhang-Sardroodi, David W. Dick, Sameneh Gholami, Mohammad Sajjad Ghaemi, Iain R. Moyles, Morgan Craig, Hsu Kiang Ooi & Jane M. Heffernan





- https://www.nature.com/articles/s41598-022-25134-0

Let's Get Started



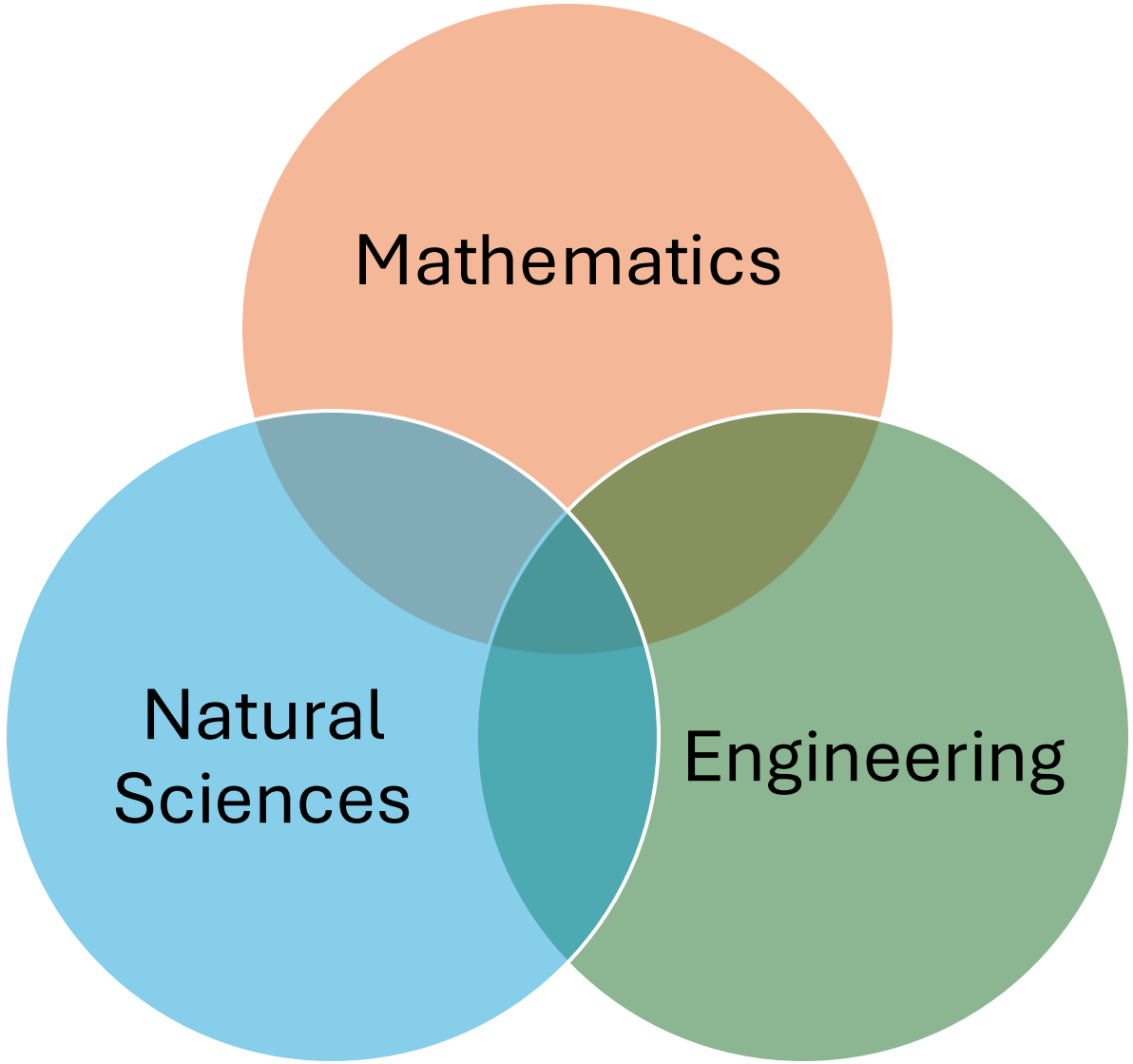
Summary

- A quick review of some aspects of Python
- We will revisit some of these concepts throughout the course. Especially the idea of Mutable / Immutable
- Remember to focus on the big picture
- Go through and implement the slides that have "Implement in PyCharm" on your own. Do this, so you can begin to practice

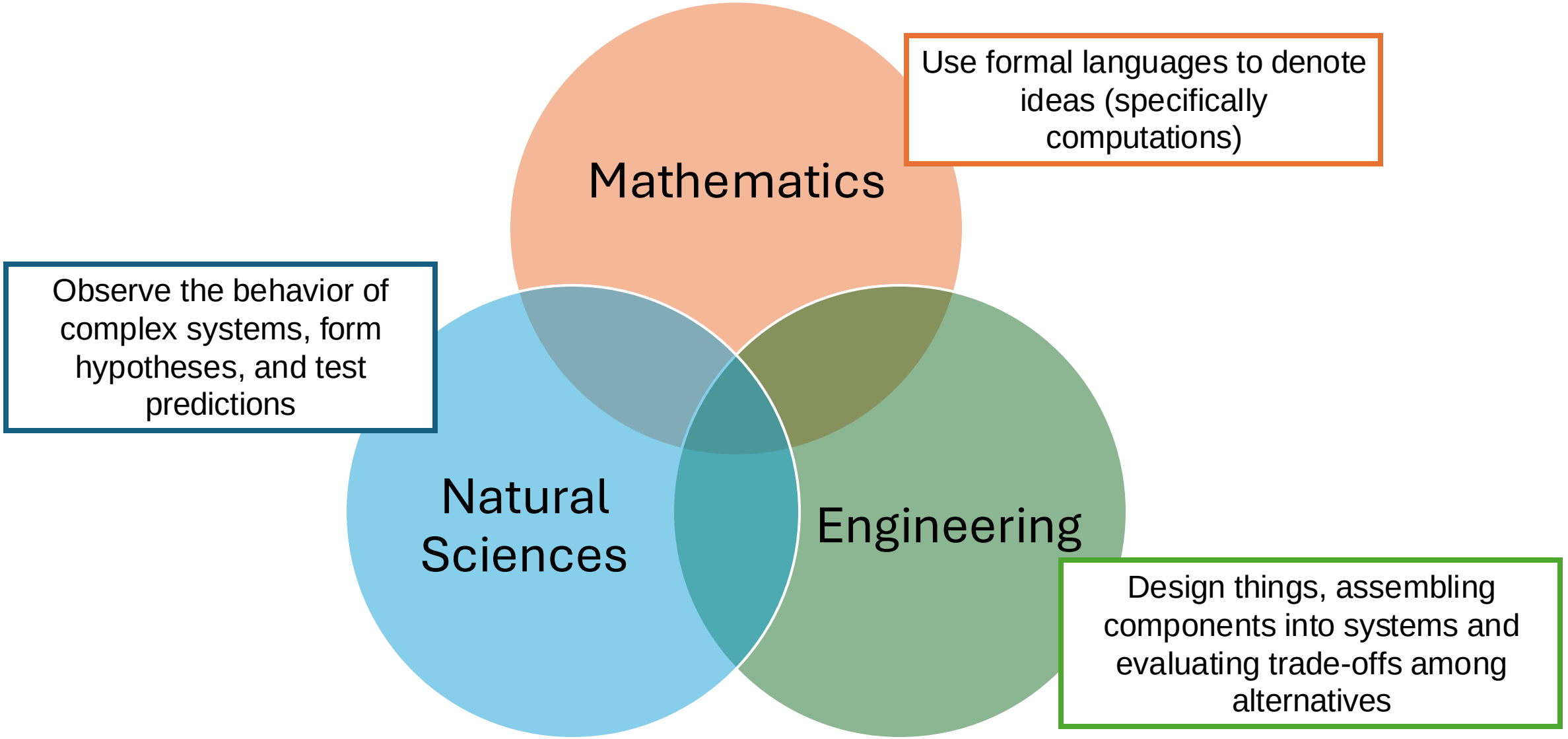
Why Learn to Program?

- I want to use programs to analyze the data I run in my lab experiments in my research
- I want to compile the data from the programs I run my lab
- I want to automate tedious tasks that I do over and over
- I found out I like the challenge of solving problems with programming
- I need to be able to communicate with other people about programming

Thinking like a Computer Scientist

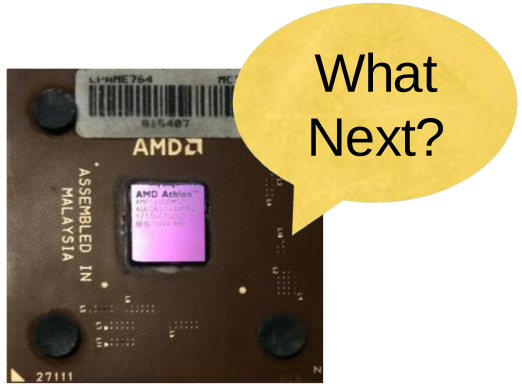


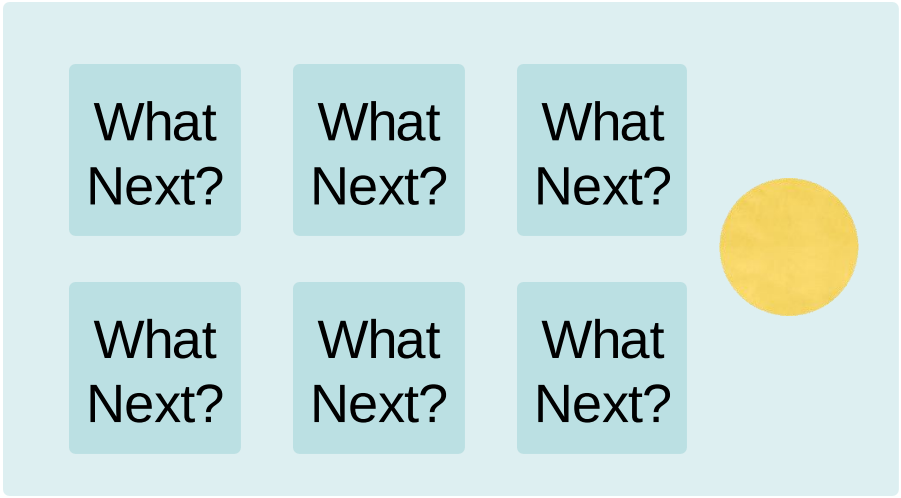
Thinking like a Computer Scientist



Computers Want to be Helpful...

- Computers are built for one purpose - to do things for us
- But we need to speak their language to describe what we want done
- Users have it easy - someone already put many different programs (instructions) into the computer and users just pick the ones they want to use

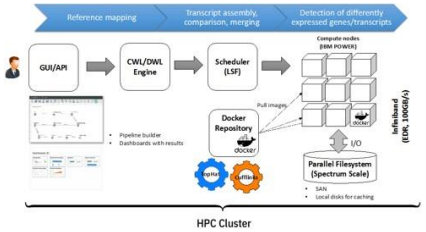




Users vs. Programmers

- Users see computers as a set of tools - word processor, spreadsheet, map, to-do list, etc.
- Programmers learn the computer "ways" and the computer language
- Programmers have some tools that allow them to build new tools
- Programmers sometimes write tools for lots of users and sometimes programmers write little "helpers" for themselves to automate a task

User



Computer
Hardware + Software

Programmers
(us)

Data

Information

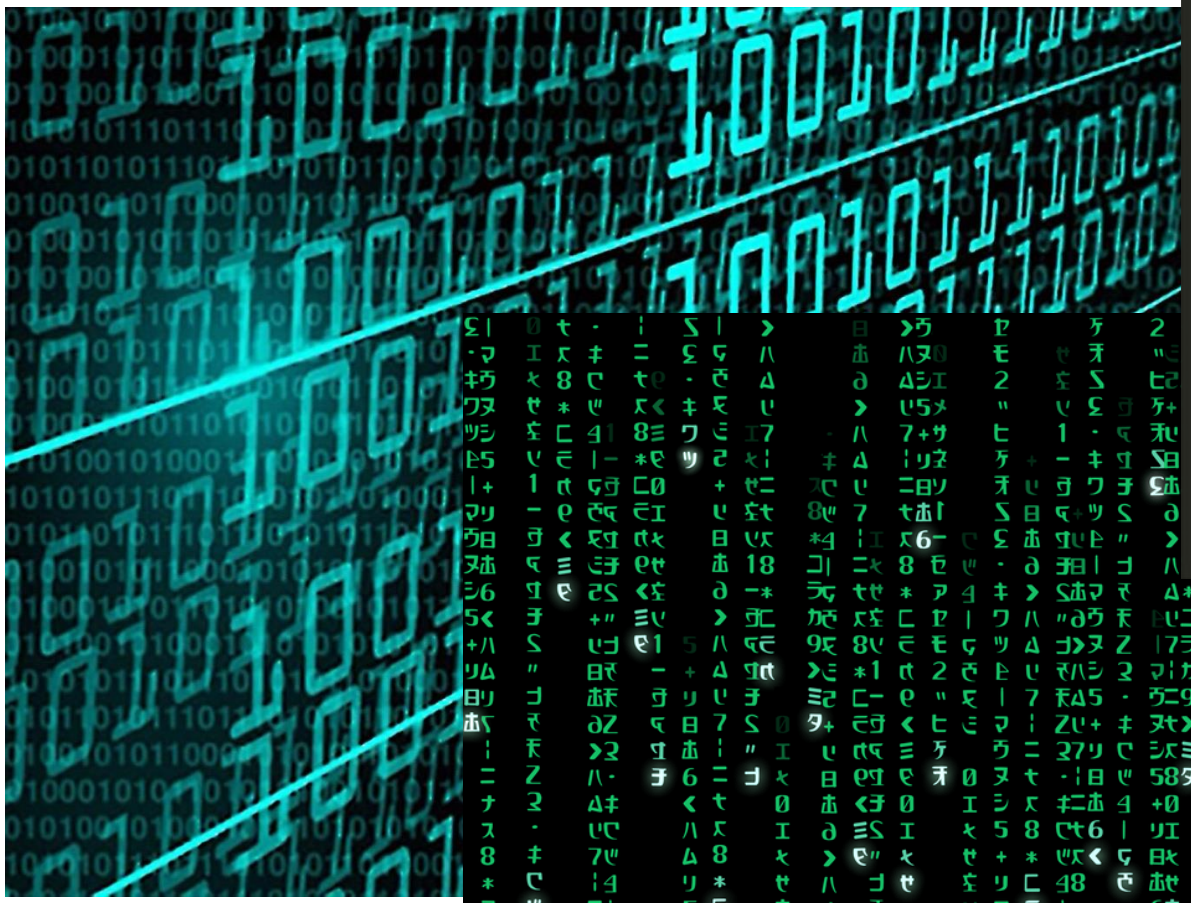
....

Networks

From a software creator's point of view, we build the software. The end users (stakeholders/actors) tell us what they would like the software to do. We the programmers, develop software and get paid in exchange for good work. However, the data, information, and networks are our problem to solve on their behalf. The hardware and software are our friends and allies in this quest.

So what is code?

To some it looks like this:



```
for i in people.data.users:
    response = client.api.statuses.user_timeline.get(screen_name=i.scre
    print 'Got', len(response.data), 'tweets from', i.screen_name
    if len(response.data) != 0:
        ltdate = response.data[0]['created_at']
        ltdate2 = datetime.strptime(ltdate,'%a %b %d %H:%M:%S +0000 %Y'
        today = datetime.now()
        howlong = (today-ltdate2).days
        if howlong < day
            print i.scre
            totaltweets
            for j in resp
                if j.ent
                    for
    else:
        print i.scre
```

What is Code?

```
name = input('Enter file: ')
handle = open(name, 'r')

counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

bigcount = None
bigword = None
for word, count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

print(bigword, bigcount)
```

```
python3 words.py
Enter file: words.txt
to 16
```

```
python3 words.py
Enter file: clown.txt
the 7
```

A lot going on here! Much to discuss!

https://www.py4e.com/code3/words.txt

https://www.py4e.com/code3/clown.txt

What is Clinical Bioinformatics?

- Clinical bioinformatics is a subfield of bioinformatics that involves analyzing and interpreting 'omics data to solve biological problems with implications in health and disease.
- We can assess risk, improve diagnostic tools, uncover biomarkers for disease, as well as inform treatment approaches through targeted drug therapies.

In Clinical Bioinformatics, we can use Python to…

Handle large clinical datasets with data from multiple samples

Pandas and NumPy - data mining, organizing and cleaning

Write scripts to filter and annotate genetic variants in order to flag variations as potentially pathogenic and linked to disease

PyVCF - Library for parsing variant call files

Develop machine learning models to predict disease risk and clinical outcomes

SciKit Learn - ML tasks

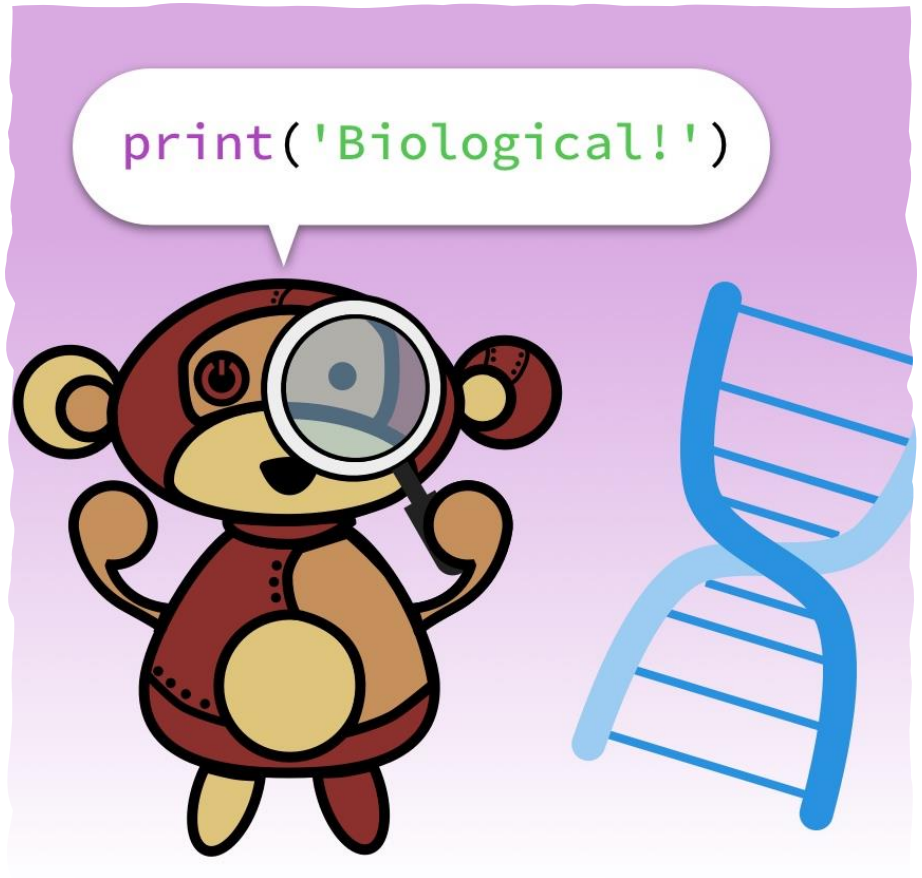
Visualize patterns of gene expression to identify differences between conditions (i.e. healthy and diseased cells) for identifying potential drug targets

Matplotlib

Seaborn

https://pandas.pydata.org/

Why Python for a Biologist?



- Versatile language that can make our work easier with tools to explore our complex (and messy!) datasets as well as automate repetitive processes
- Allows us to create a collection of small and functional scripts we can reuse for reoccurring tasks
- Useful features for easy text manipulation, pattern recognition and data integration that are helpful for working with sequencing data
- Can integrate with different software tools, process their outputs, and move data smoothly to create multi-stage analysis workflows for complex tasks

Guido van Rossum, Creator Python

- Born in the Netherlands
- Began to develop Python in 1989
- Has worked for Google, DARPA, Dropbox at present
- Named Python after *Monty Python's Flying Circus*
- Released Python 1.0 in 1994

Python Philosophy

- The Zen of Python is a collection of 19 "guiding principles" for writing computer programs
- The language's core philosophy is summarized in those documents
- Includes set of principles, like:
  - Beautiful is better than ugly
  - Simple is better than complex
  - Complex is better than complicated
  - Readability counts
  - **Explicit is better than implicit**

- Rather than having all its functionality built into its core, Python was designed to be highly extensible
- This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications

https://en.wikipedia.org/wiki/Zen_of_Python

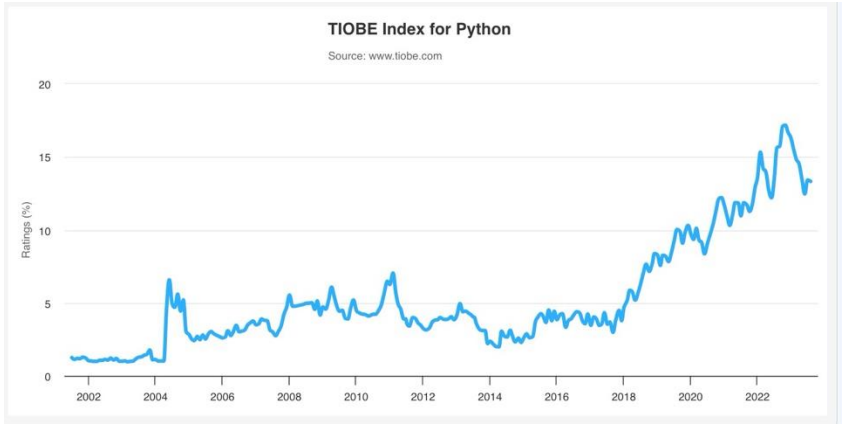
Python is meant to be an easily readable language…

- Strives for a simpler, less-cluttered syntax and grammar
- Still gives developers a choice in their coding methodology
- In contrast to Perl's "there is more than one way to do it" motto
  - Python embraces a "***there should be one—and preferably only one—obvious way to do it***" design philosophy
- Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."
- **"Pythonic: ways…" can sometime lead to hard to read code**

https://en.wikipedia.org/wiki/Python_(programming_language)#Features_and_philosophy

Python and Programming Paradigms

- Python supports several programming paradigms
- **It fully supports structured programming and object oriented**
- It also supports various concepts in **functional** programming
- Python features a **dynamic type** system and automatic memory management
- The programming paradigms and language features help you to develop large and complex software applications with Python
- **This has led to widespread adoption**
- **See the TIOBE index**



1	1		Python	13.33%	-2.30%
2	2		C	11.41%	-3.35%
3	4	↑	C++	10.63%	+0.49%
4	3	↓	Java	10.33%	-2.14%
5	5		C#	7.04%	+1.64%
6	8	↑	JavaScript	3.29%	+0.89%
7	6	↓	Visual Basic	2.63%	-2.26%
8	9	↑	SQL	1.53%	-0.14%
9	7	↓	Assembly language	1.34%	-1.41%
10	10		PHP	1.27%	-0.09%
11	21	⇈	Scratch	1.22%	+0.63%
12	15	↑	Go	1.16%	+0.20%
13	17	⇈	MATLAB	1.05%	+0.17%
14	18	⇈	Fortran	1.03%	+0.24%
15	31	⇈	COBOL	0.96%	+0.59%
16	16		R	0.92%	+0.01%
17	19	↑	Ruby	0.91%	+0.18%
18	11	⇊	Swift	0.90%	-0.35%
19	22	↑	Rust	0.89%	+0.32%
20	28	⇈	Julia	0.85%	+0.41%

The TIOBE Programming Community index is an indicator of the popularity of programming languages
Language of the Year: 2007, 2010, 2018, 2020, 2021
https://www.tiobe.com/tiobe-index/

Examples of When is Whitespace Ignored:

```
# whitespace.py

long_winded_computation = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 +
                           9 + 10 + 11 + 12 + 13 + 14 +
                           15 + 16 + 17 + 18 + 19 + 20)
# Alternatively:
long_winded_computation   = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 +  \
                          9 + 10 + 11 + 12 + 13 + 14 + \
                         15 + 16 + 17  + 18 + 19 + 20


# lists
list_of_lists = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

  easier_to_read_list_of_lists = [[1, 2,  3],
                                  [4, 5,  6],
                                  [7, 8,  9]]
```

We'll use a linter to tell us what acceptable later on…

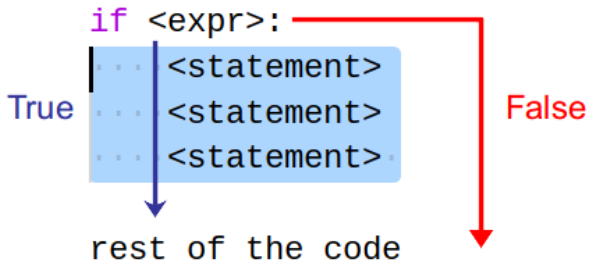
```
print(type(long_winded_computation))
print(type(list_of_lists))
print(type(easier_to_read_list_of_lists))
```

```
<class 'int'>
<class 'list'>
<class 'list'>
```

What's being printed

Indentation

- Increase indent after an `if` statement or `for` statement (after :)
- Maintain indent to indicate the block (which lines are affected by the `if`/`for`)
- Reduce indent back to the level of the `if` statement or `for` statement to indicate the end of the block
- Blank lines are ignored - they do not affect indentation
- Comments on a line by themselves are ignored with regard to indentation




```
# Python program to explain continue statement
string1 = "Stechies"

# Continue with for loop
for value in string1:

    if value == 'e':

        # If Letter is equal to 'e' next statement will skip
        continue

        # Skip statement after continue
        print('This is continue block: ',value)

    # Print the next iteration
    print("Value: ",value)

# Statement outside the for loop
print("Outside for Loop")
```

Warning: Turn Off Tabs!!

- Some editors automatically uses spaces for files with ".py" extension (nice!)
- Most text editors can turn tabs into spaces - make sure to enable this feature
- Python cares a *lot* about how far a line is indented. If you mix tabs and spaces, you may get "indentation errors" even if everything looks fine, i.e. you won't see them
- PyCharm convert tabs to spaces automatically

  https://www.youtube.com/watch?v=wYQxx2ewrbA

Early Learner: Syntax Errors

- We need to learn the Python language to communicate our instructions. We will make lots of mistakes in the beginning and speak gibberish like *someone learning a new language*
- When you make a mistake, the computer says "syntax error" – It will seem like Python is cruel
- Must remember that you are intelligent one in the relationship. The computer is simple and very fast, but cannot *learn*
- You don't need to understand every detail of Python to be productive
- The Principles we cover throughout this semester are vital to your growth as a programmer
- **But remember: Mastery comes with practice of reading and writing code**

A Simple Task to Handle with Python

- Question:
  - What 6-mer containing only A's and T's occurs mostly commonly in the *P. falciparum* genome

- Answer:
  - Scan through the 2.2 million nucleotides of the 14 chromosomes using a 6-nt window and find that ATATAT occurs most frequently

```
---------
A: 40.31 %
T: 40.30 %
C:  9.70 %
G:  9.69 %
total(ATCG only ) = 22853497
nucleotides
total(everything) = 22853764
nucleotides
---------
1       ATATAT  570577
2       TATATA  529826
3       AAAAAA  420232
4       TTTTTT  417439
5       TATTAT  147506
6       ATAATA  146948
7       TATATT  146711
8       AATATA  146695

     -- snip --

57      ATTTAA  50617
58      ATTAAT  49856
59      TTAAAT  49731
60      TAATTA  45781
61      AATTAA  45278
62      TTAATT  45212
63      AATTTA  45101
64      TAAATT  44553
```

The Process of Programming

- In scripted languages, you can write it with ANY text editor
  - **Sublime is good, but PyCharm is superior**

- As you can imagine, programming is a **process**:
  - Write code
  - Try to run the program
  - Correct the errors (debugging)
  - Look at the output
  - **Compare previous results (use your `diff` or `sdiff`)**
  - Edit the program and try again

- **If you want to program as a profession, learn to love the process. You will be required over and over and over**

- **To write better code, we'll learn more about this process over the semester**

Getting `python3`

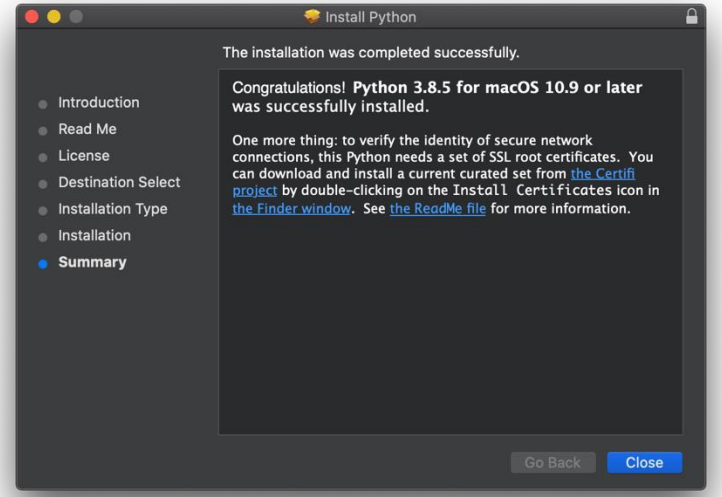
- If you need to get Python 3, it's straight forward. Download and follow instructions

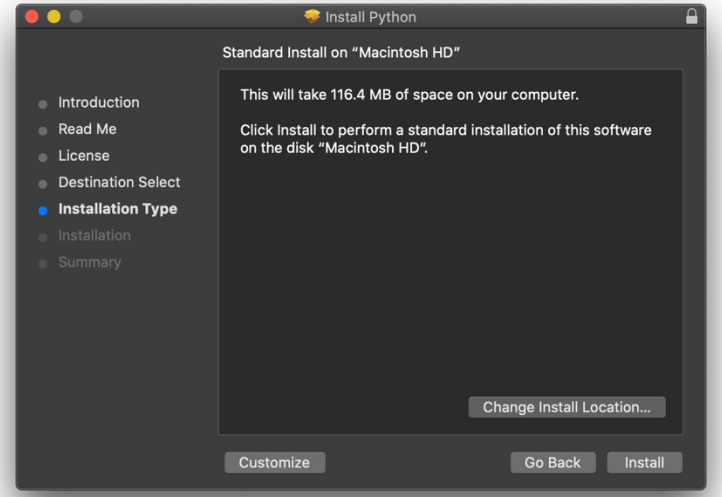
  https://www.python.org/ftp/python/3.11.5/python-3.11.5-macos11.pkg

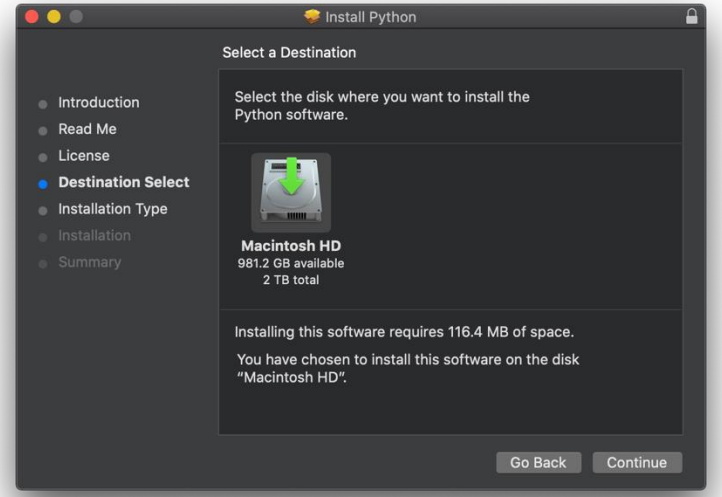
Mac Users link above

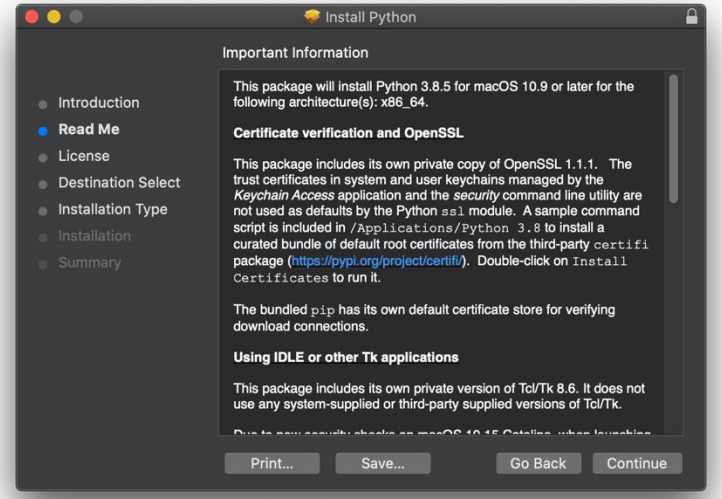
https://www.python.org/downloads/

Getting `python3`









Where Do Site Packages Go?

```
$ pip show numpy
Name: numpy
Version: 1.23.5
Summary: NumPy is the fundamental package for array computing with Python.
Home-page: https://www.numpy.org
Author: Travis E. Oliphant et al.
Author-email:
License: BSD
Location: /opt/homebrew/anaconda3/lib/python3.10/site-packages
Requires:
Required-by: astropy, bokeh, Bottleneck, contourpy, datashader, datashape, gensim, h5py,
holoviews, hvplot, ifCNV, imagecodecs, imageio, imbalanced-learn, matplotlib, numba,
numexpr, pandas, patsy, pyerfa, PyWavelets, scikit-image, scikit-learn, scipy, seaborn,
statsmodels, tables, tifffile, transformers, xarray
```

Note: I have Python 3.9 installed, and have installed **`numpy`**

Python Interactive Shell

- One of the best features of python is the REPL: is an **interactive** way to use Python
- The easiest way to open a Python interactive **shell** is to type `python3` on CLI
- To quit, **`exit()`**, **`quit()`** or press Control+D
- The REPL is useful for:
  - Experimenting with short expressions or statements to learn or test new features of the language
  - See how assignments work (not assignments you'll do in class)
  - Consulting the documentation
  - Etc.

```
[~]# python3
Python 3.9.10 (main, Jan 15 2022, 11:48:00)
[Clang 13.0.0 (clang-1300.0.29.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```


What next?

Note: if you install Python 3.9.10 yours will look like this

```
[~]# python3
Python 3.9.10 (main, Jan 15 2022, 11:48:00)
[Clang 13.0.0 (clang-1300.0.29.3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
>>> print(x)
1
>>> x = x + 1
>>> print(x)
2
>>> exit()
```

This is a good test to make sure that you have Python correctly installed. Note that `quit()` also works to end the interactive session.