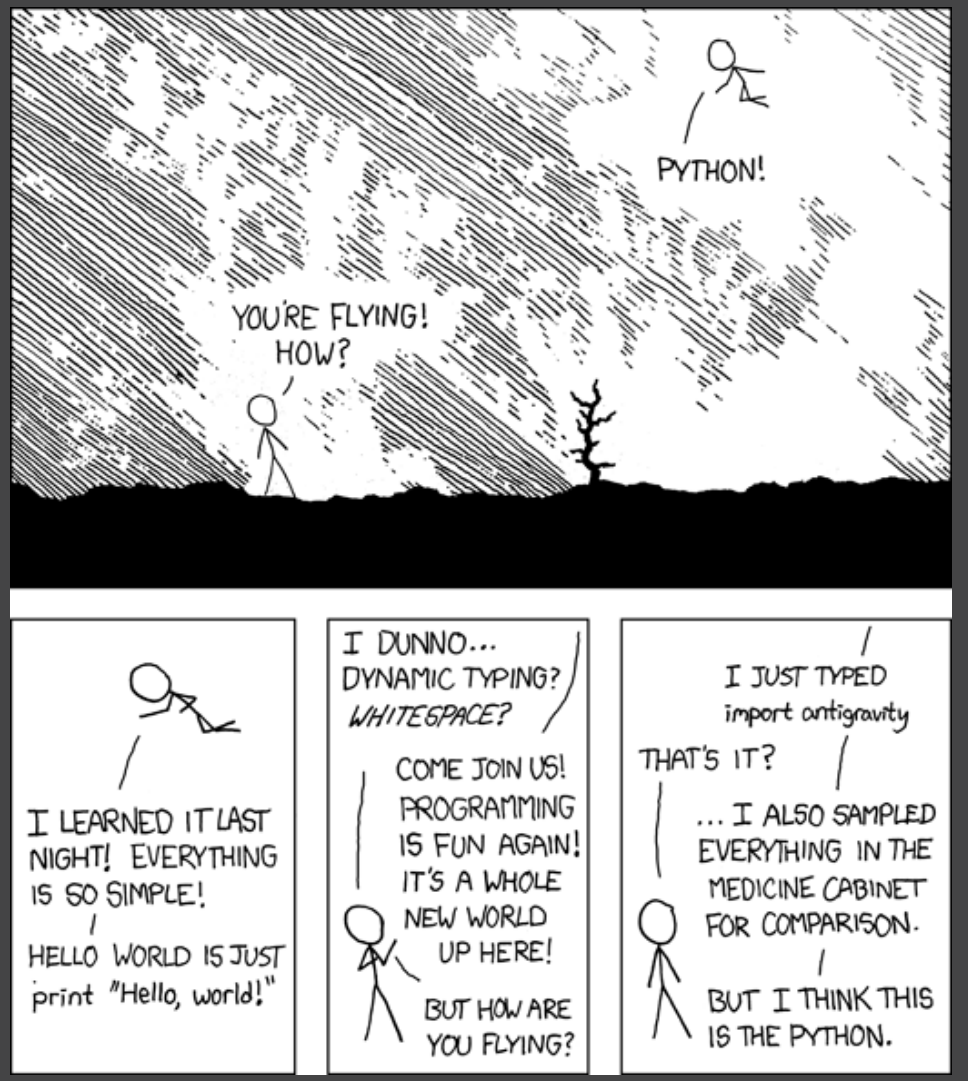
APS106

UNIVERSITY OF TORONTO

The Coding Toolbox.

Week 1 | Lecture 2 (1.2)



Upcoming:

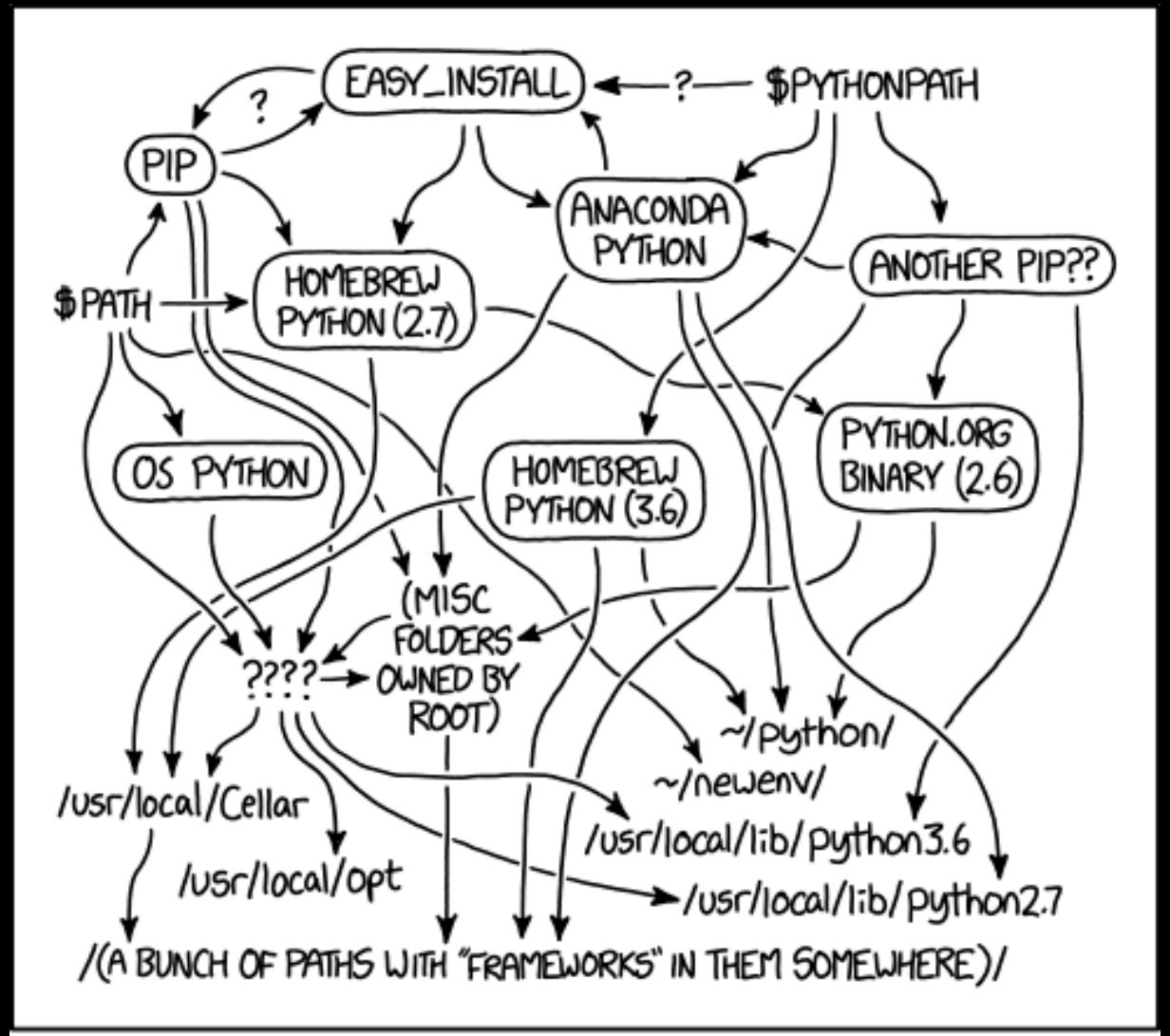
- Lab 0 released Thursday at 6 PM
- Reflection 1 released Friday at 6 PM

- First PRA section – Friday 3-5 PM

if nothing else, write #cleancode

This Week's Content

- Lecture 1.1
  - Introduction
- **Lecture 1.2**
  - **The Coding Toolbox**
- Lecture 1.3
  - Variables, Operators, and Expressions



What you will get out of today's lecture:

- Everything you need to begin coding in APS106
- Learn about the core building blocks of a computer
- Develop computational language and vocabulary for the files and programs used in APS106
- Learn about the key differences in the software and tools used in APS106
- Get experience opening and using the most important tools
- Execute your first files of APS106!

What do you need to code?

- A Computer
  - Input/Output devices
  - Storage
  - Memory
  - Processor
- An Integrated Development Environment (IDE)
  - VSCode
  - Jupyter vs JupyterHub
  - Installing with Anaconda
- A Programming Language
  - Why we use languages
  - Compiling from high level to low level languages
- A File with appropriate format
  - File formats and extensions
  - Python programming files (.py vs .ipynb)

Outside Your Computer

Input Devices
- Keyboard
- Mouse
- Trackpad
- Touch screen
- Webcam
- Gesture trackers
- Eye trackers

Output Devices
- Monitors
- Printers
- Headphones
- Speakers
- Video/sound cards
- Braille readers

Eye Tracking Glasses

Braille Reader

Inside Your Computer

- Storage
  - Disks, hard drives, thumb drives, etc.
  - Stores long-term data (files, programs, movies, etc.)
  - Non-volatile: maintains contents when powered off
- Memory
  - Random Access Memory (RAM)
  - Stores short-term data required for open programs and processes
  - Volatile: loses contents when powered off

Software programs are usually stored and executed from RAM.

Later this term, we will learn how to read from and write to long-term storage.

Inside Your Computer

- Central Processing Unit (CPU)
  - Runs the programs, including the operating system
  - Reads and writes the data between storage and memory
  - Composed of billions of miniature electronic switches called transistors
  - Popular examples include Intel's Core i7 and AMD's Ryzen 7
- Clock
  - Controls the rate processors execute instructions (recall: programming is just a series of instructions)
  - Modern computers have clocks that beat around 4 GHz (4 billion times per second!)

World records for number of transistors on a chip are held by U of T researchers in ECE!

Feeling Confused?

- Modules -> Course Resources -> How Does That Work?

How Does That Work? (HDTW)

How Does That Work (HDTW) is a series of short instructional videos focused on the tools we use in this course.

HDTW - Installing Python & Anaconda

HDTW - Anaconda Navigator

HDTW - Jupyter Notebooks

HDTW - UofT JupyterHub

HDTW - IDEs & VSCode

HDTW - Debugging (VSCode)

HDTW - Environments & Installing Packages

Using an Integrated Development Environment (IDE)

- IDEs are programs that provides tools and features to programmers in a unified environment
- IDEs often include:
  - A code editor
    - A place to type and edit code, usually with colour-coded syntax highlighting to improve readability
  - Code compilers or interpreters
    - Turns the readable Python code into something the machine can understand
  - Debuggers
    - Pause the code at pre-determined locations and go line-by-line through your code
  - Version Control (i.e., git)
  - Code navigation tools
  - A Built-in terminal
  - Integrated documentation (access to help resources from within the IDE)

Which should IDE to use?

- IDLE is the official IDE included with Python that provides a basic environment for editing and running programs
- Other popular IDEs differ in features and supported programming languages
- Popular IDEs include:
  - Visual Studio / VS Code (Microsoft)
  - Xcode (Apple)
  - Android Studio (Google)
  - PyCharm
  - Jupyter (web-based)
  - NetBeans
  - Eclipse

To write a document, you use a "Word Processor" such as Microsoft Word, Google Docs, or Apple Pages

Similarly, to write a software program, you use an IDE such as IDLE, PyCharm, or Jupyter

- This course will both VSCode and Jupyter.
  - Jupyter will be used in lectures, and VSCode is the supported IDE in later labs.

Friendly Warning: Visual Studio (VS) Code ≠ Visual Studio

Visual Studio Code

- Windows, Mac, Linux
- Faster for laptops & older computers
- Simple interface, quick to set up
- Flexible extensions as required

Visual Studio

- Primarily Windows
- For large enterprise projects
- Requires more system resources
- Complex interface for experts

- VSCode is good enough (and recommended) for APS106!
- You may use other IDEs if you are comfortable
- VSCode is the supported IDE in labs

A Quick Note on File Formats and Extensions

- Files store data in different formats based on specific definitions
- An "extension" is the file name's suffix, or the set of characters (usually 2-4) at the end of a file name, separated by a period (.)

File Types	Common Extension Examples
Plain text file	.txt
Word processor files	.doc, .docx, .pdf
Image files	.jpg, .gif, .png
Compressed/Archive files	.zip, .rar, .7z
Video files	.mpg, .mp4, .avi
Executable files	.exe, .bat, .bin
Python file	.py
Jupyter Notebook file	.ipynb
Comma Separated Value file	.csv

File extensions help indicate:
- The file format
- How the data is structured
- Which programs can open it

Anaconda Navigator

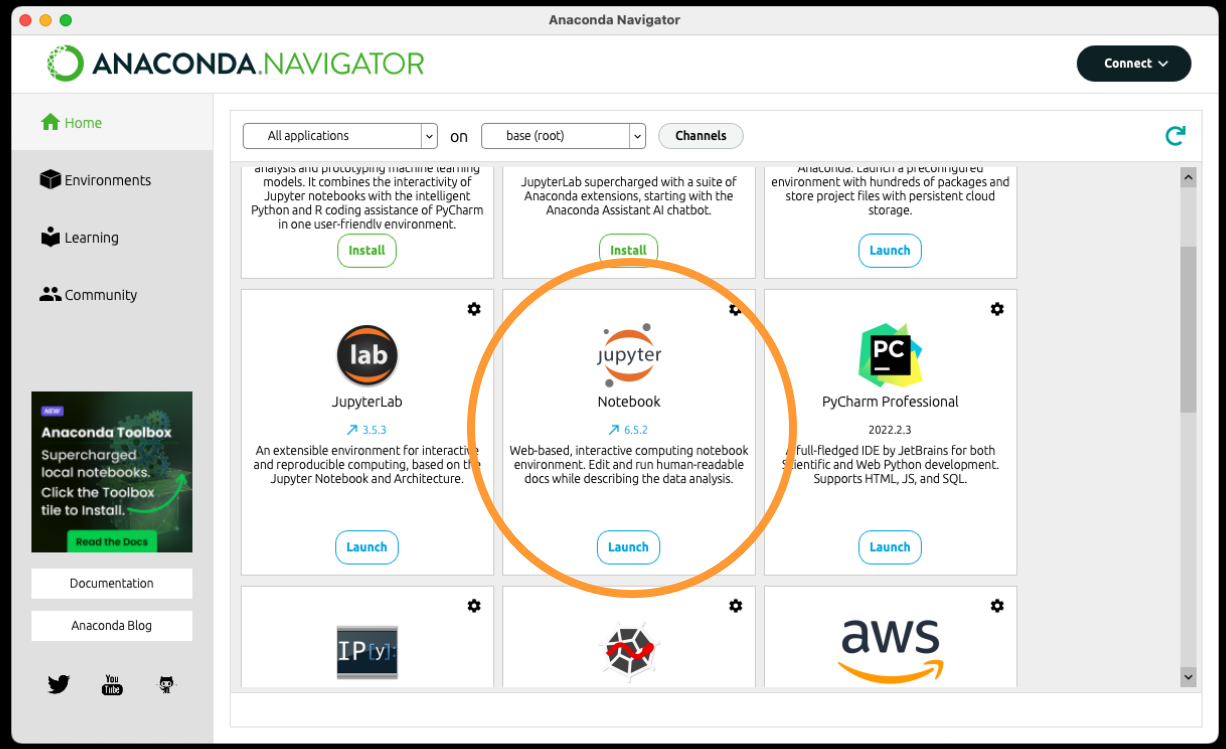
- Anaconda is a distribution of Python that includes tools and packages geared towards scientific computing (such as data science and machine learning)
- Anaconda Navigator is the graphical user interface (GUI) allowing users to install and manage their programming environment **without command line (terminal) prompts**

Anaconda Navigator

- Your first core use of Navigator is to open Jupyter Notebook (not JupyterLab)

More advanced (optional):

- In the Environments tab on the left you can manage the packages installed for different "environments"
- Installing a specific version or package for one project shouldn't necessarily affect all projects
- "Environments" allow developers to isolate project workspaces (with specific Python versions and installed packages



What is Jupyter?

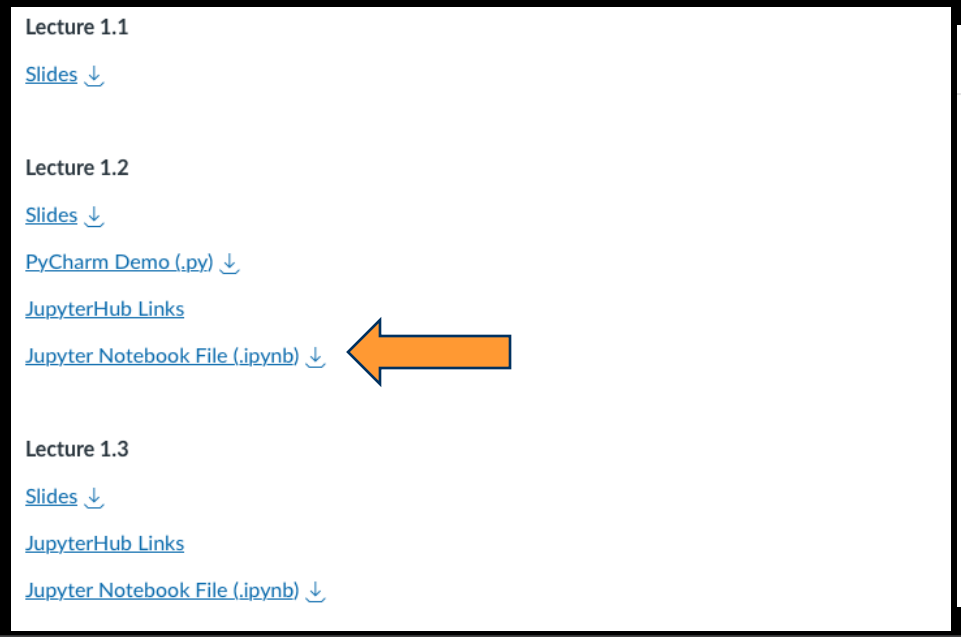
- Jupyter Notebook 

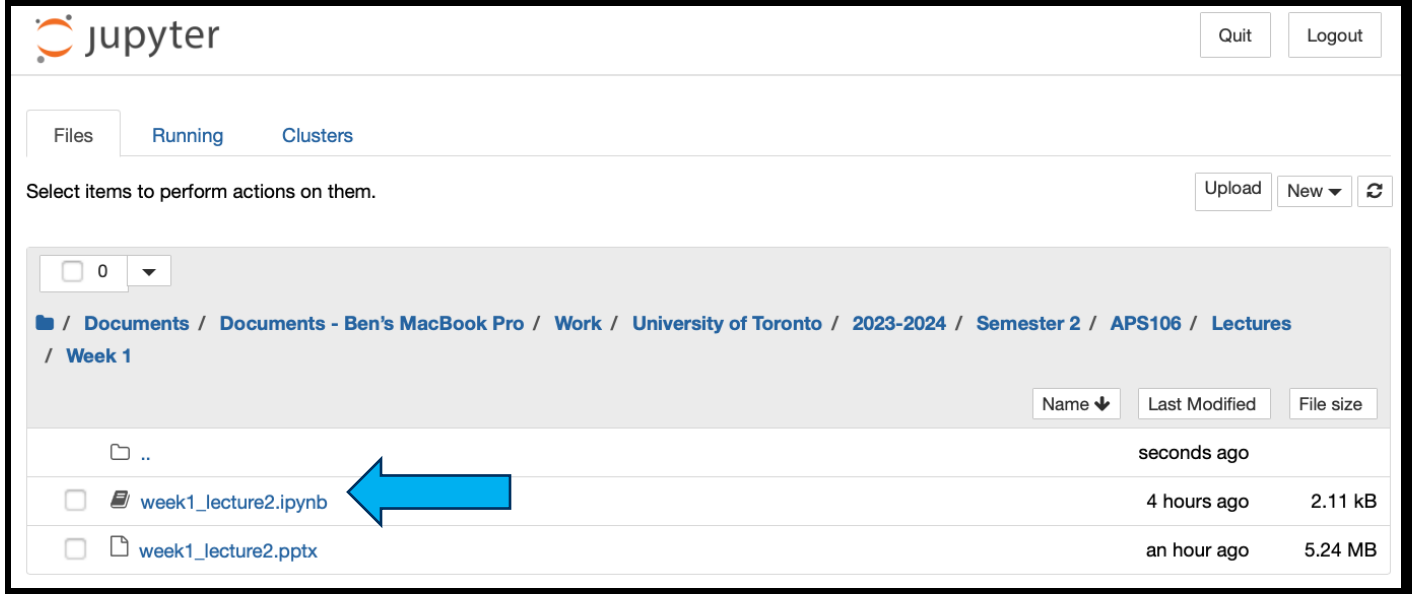
  - Interactive and web-based environment
  - Creates "notebooks" (.ipynb files) that can combine live code, visualisations, and narrative text
  - Code can be divided into individually-executable "cells"
  - Cells can include either executable code, or formatted text and images
  - Can export notebooks to HTML or PDF
- JupyterHub 

  - A cloud-based server for running Jupyter notebooks
  - Does NOT run locally on your computer (think: Microsoft Word vs the cloud-based Google Docs)
  - Allows you to "clone" lecture notebooks and work on notebooks in the cloud
  - A good solution if your local Jupyter environment suddenly stops working in lecture

Opening Your First Notebook (.ipynb) File

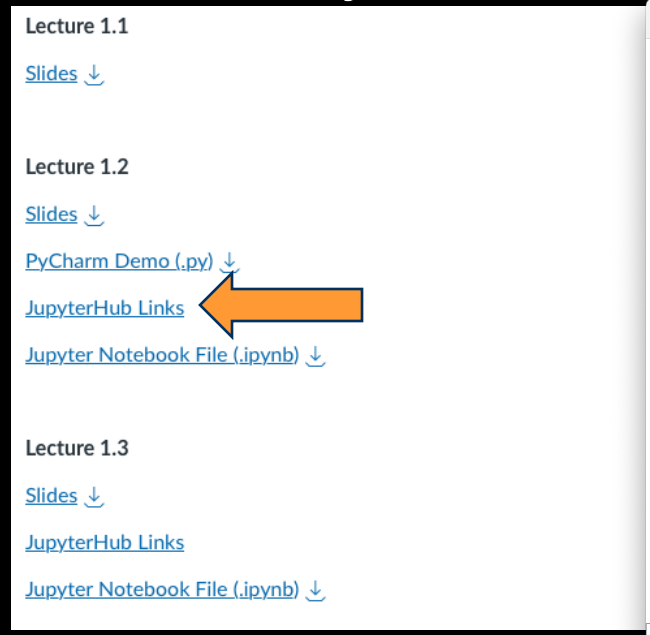
- Download the Notebook (.ipynb) file from Quercus
- Open Jupyter Notebook through Anaconda Navigator
- Jupyter Notebook will open in your web browser, and you will see the directory (folder) system for your computer
- Navigate to the folder where you downloaded the Notebook file (such as "Downloads" or your APS106 folder) and open the .ipynb file

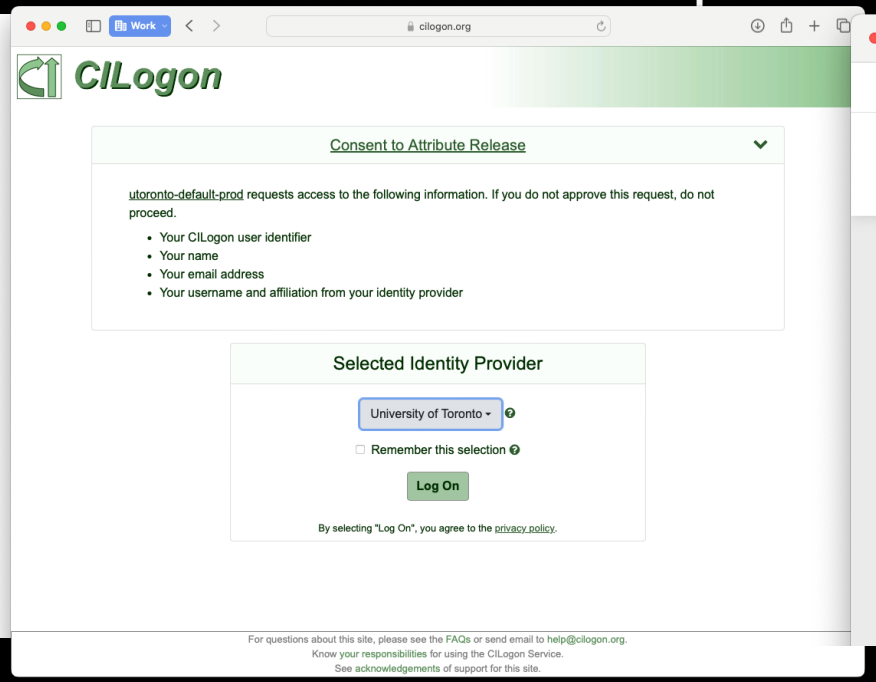


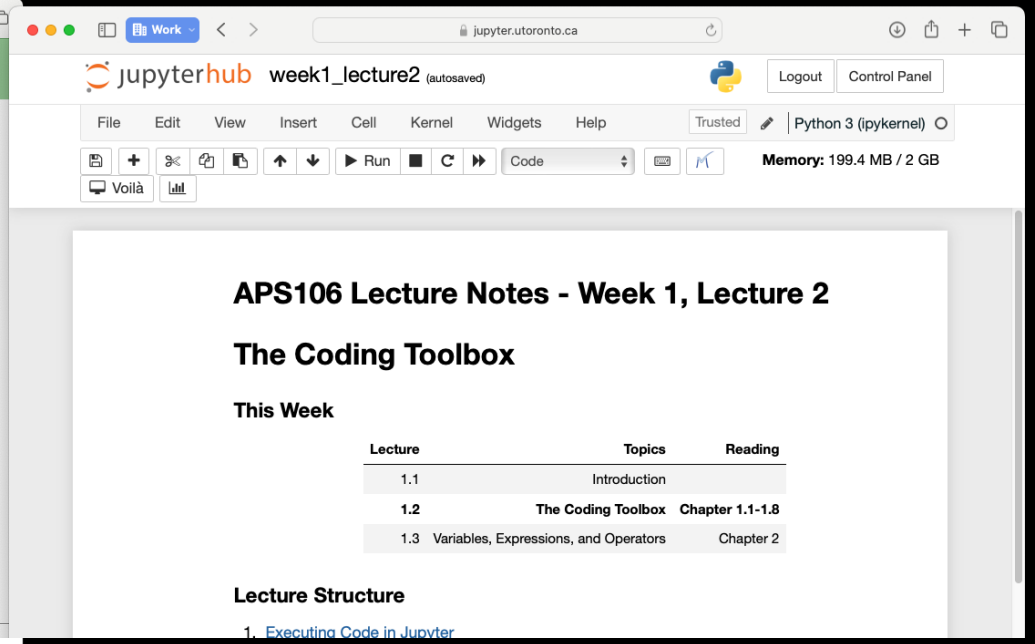


Opening From JupyterHub

- Follow Quercus to this lecture's JupyterHub link (Week 1, Lecture 2)
- The first time using JupyterHub, you may see a "CILogon" screen
- This is normal, continue logging in with your U of T account
- Today's lecture Notebook should open

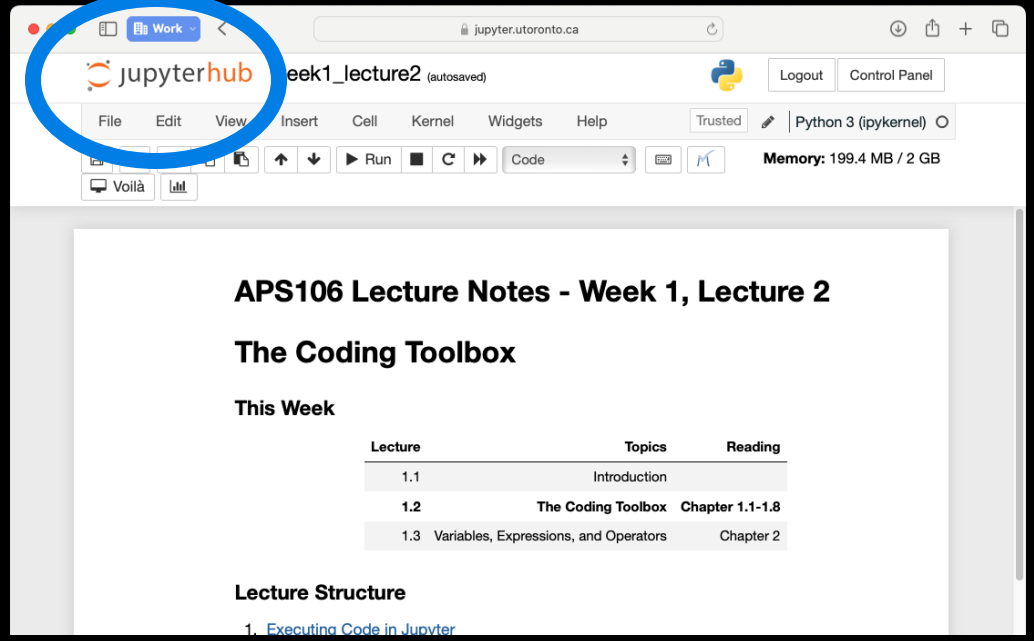


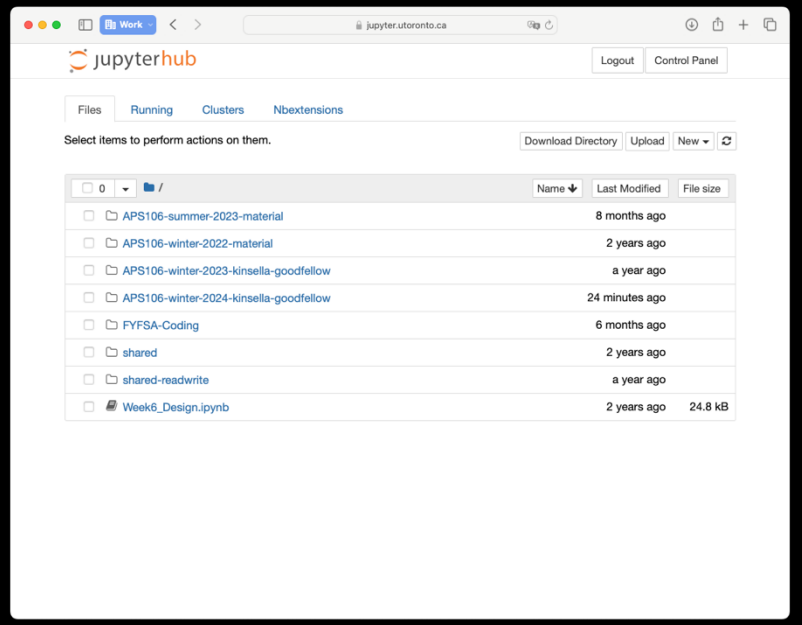




Navigating to your JupyterHub files

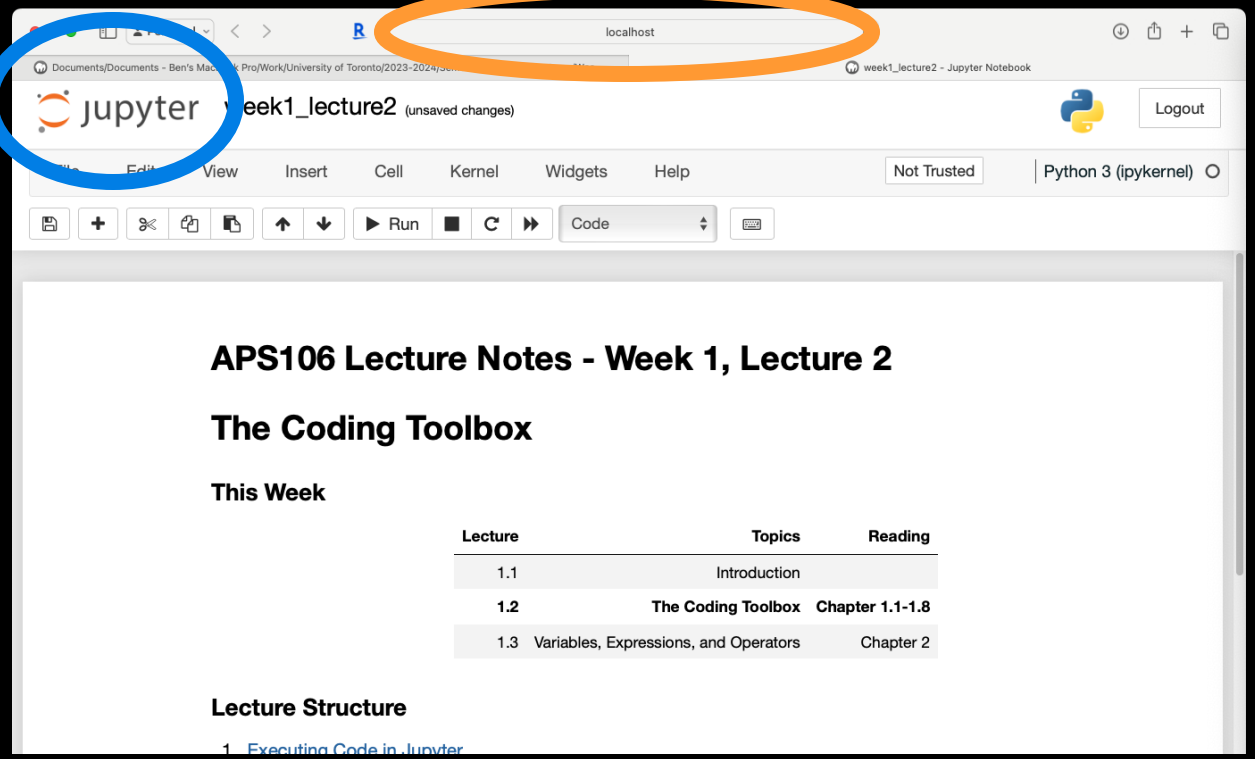
- By clicking the link on Quercus, you have "cloned" (or copied) the lecture slides into your JupyterHub account
- You can click the JupyterHub logo on the top left to open the directory to your JupyterHub account and access any other lectures you have previously cloned
- It is always safer to store files on your computer's local storage (i.e., not on JupyterHub)!

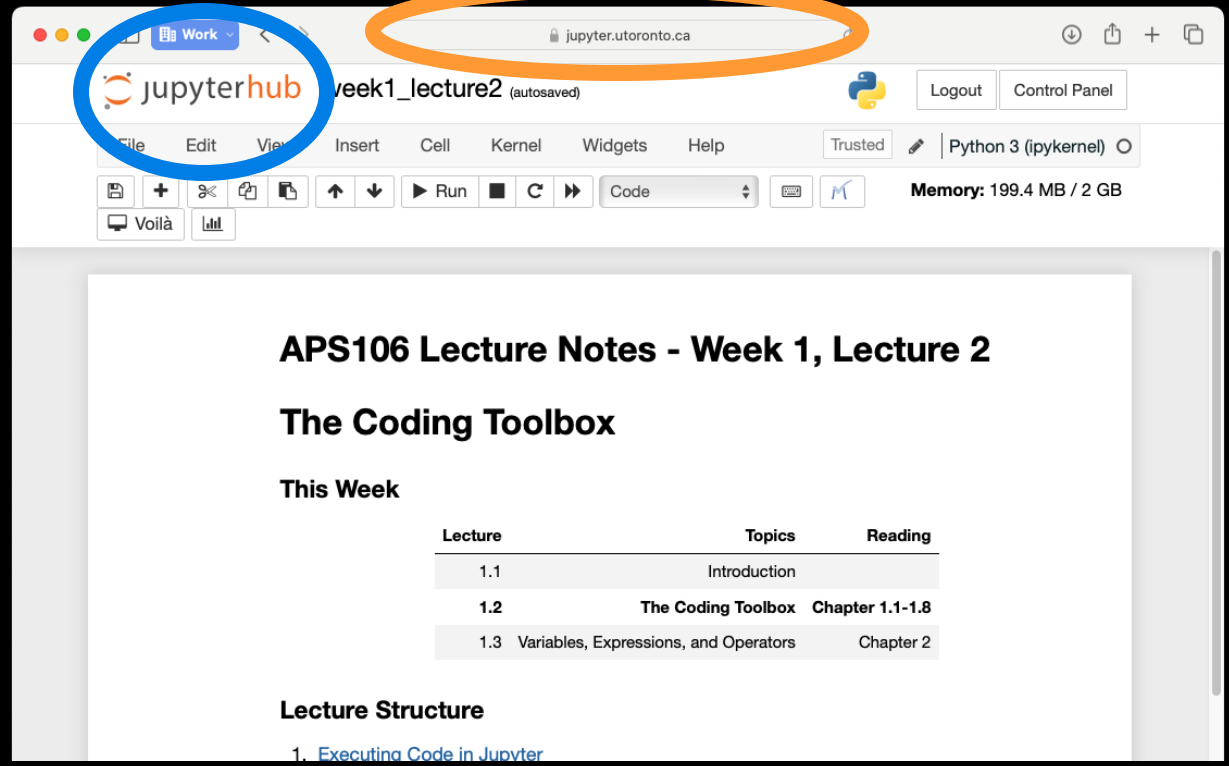




Jupyter vs. JupyterHub

- Compare the difference in "URLs" on your computer
- You'll see Jupyter points to a local file on your computer, whereas JupyterHub is on an external website





Let's Quickly Explore Jupyter

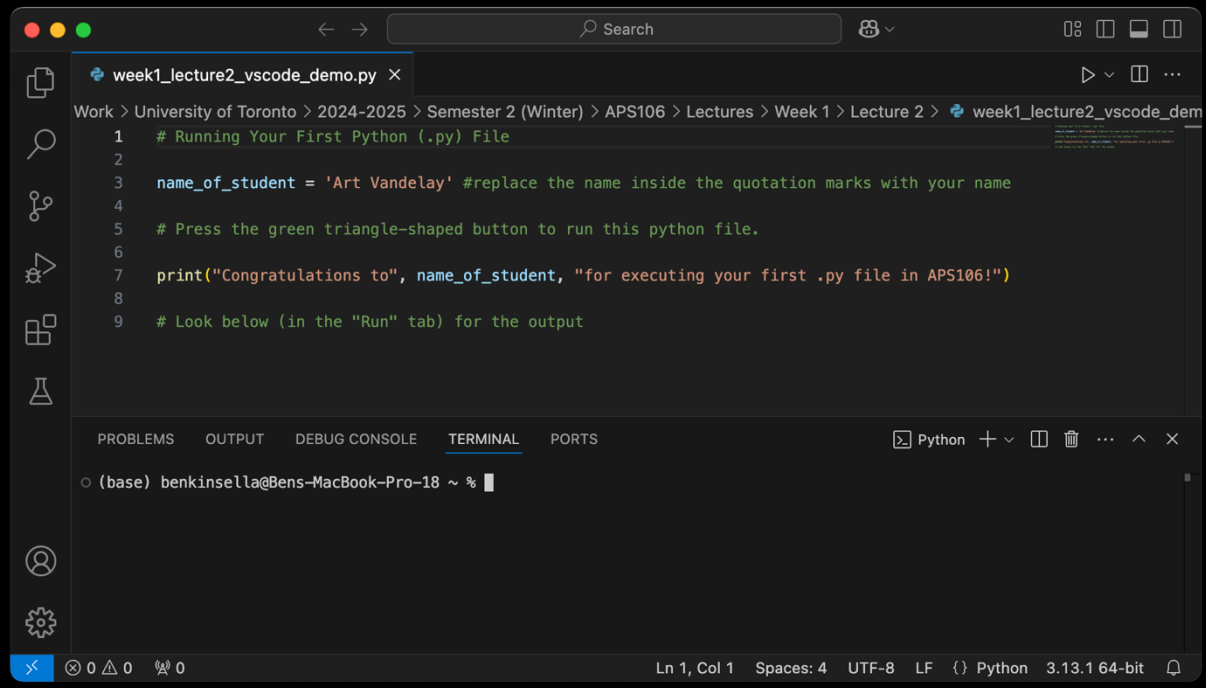
- Explore different cell types
- Execute Python code in Jupyter
- Other helpful Jupyter tips
- You can use the downloaded .ipynb file or JupyterHub links to follow along

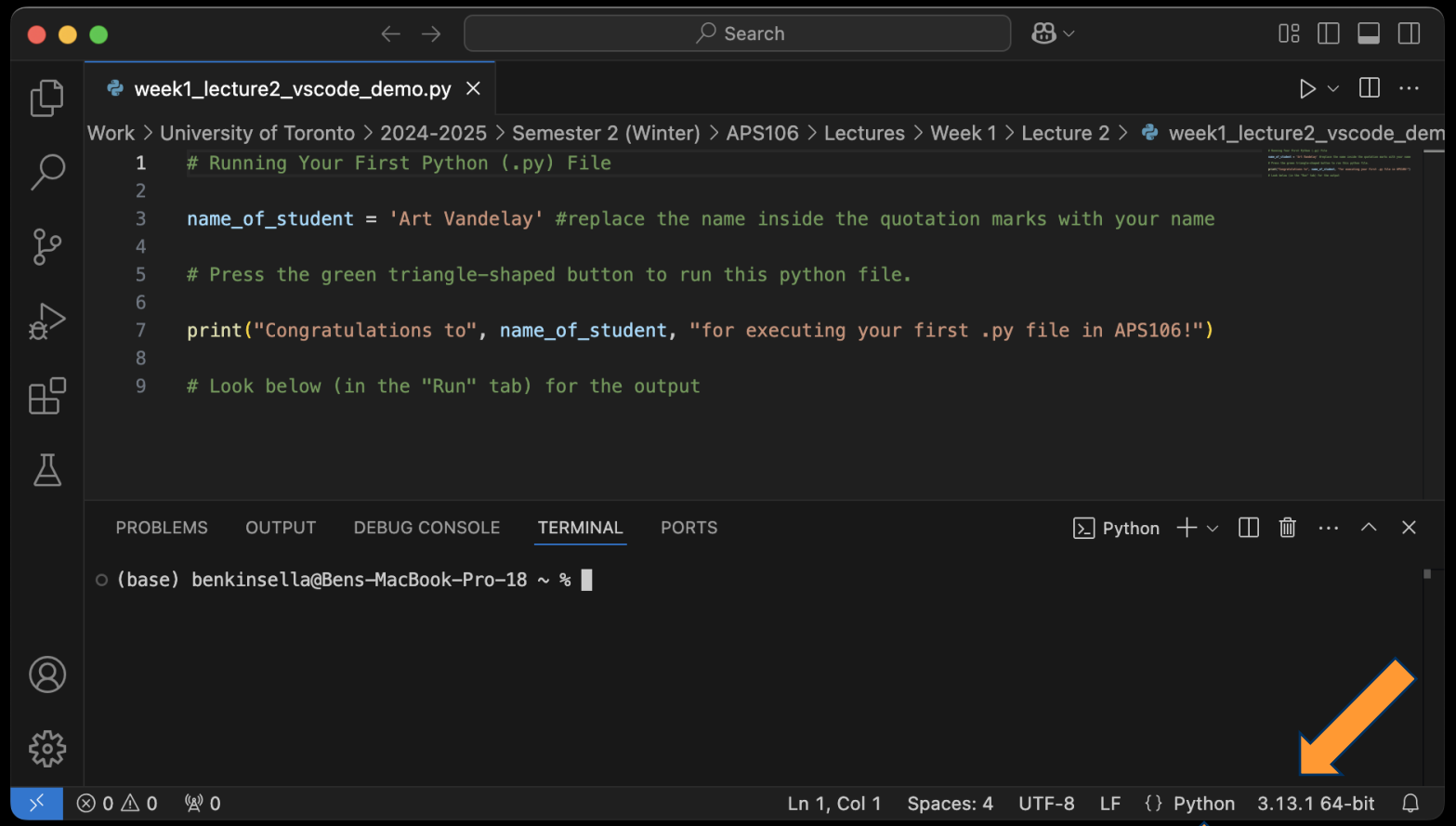
Open your notebook

Click Link:
1. Exploring Cell Types in Jupyter

Running Your First Python (.py) File

- Download this .py file from Quercus
- Open VSCode on your Computer
- In VSCode go to File -> Open -> find and open the .py file where you downloaded it

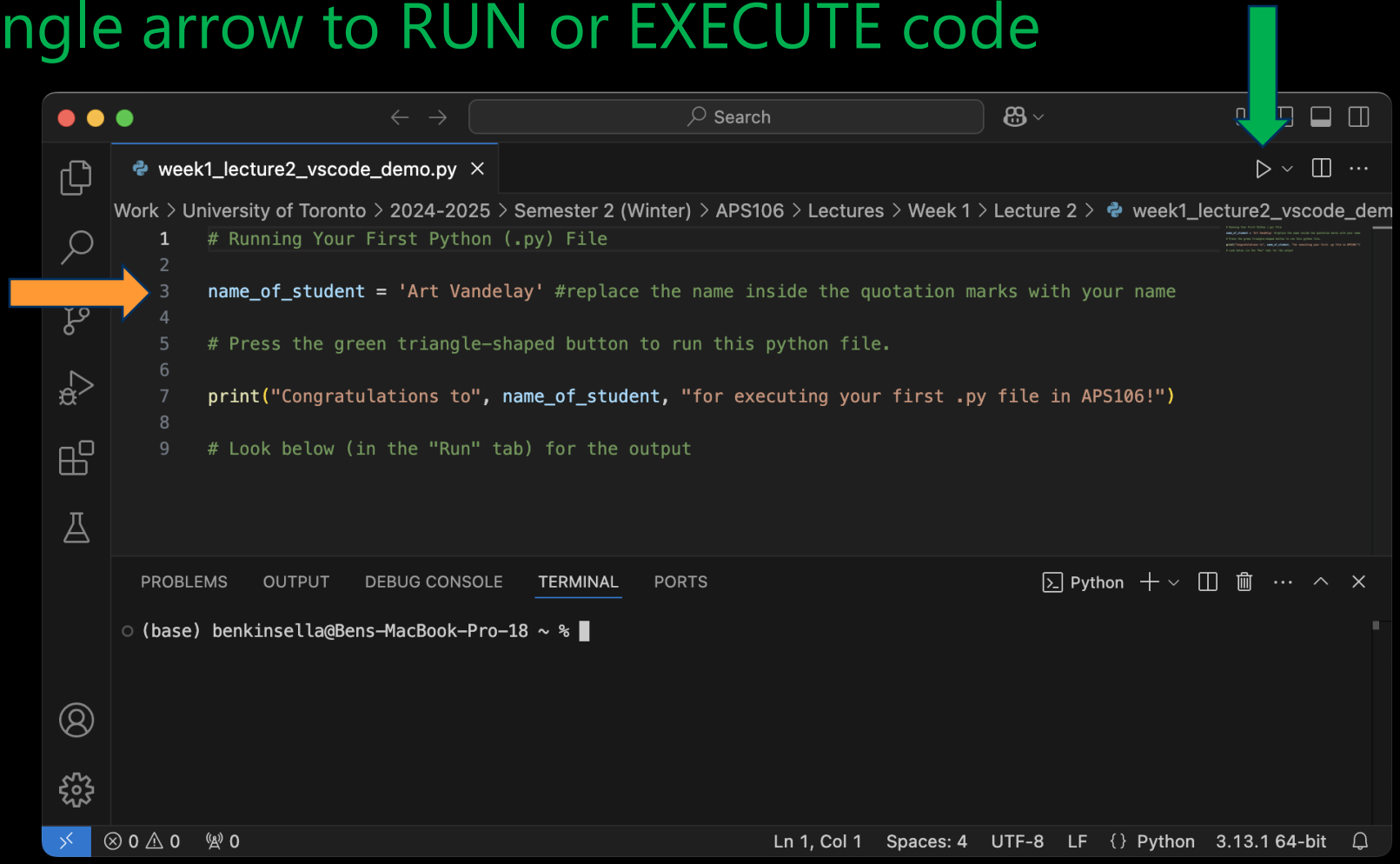




Click here to change the Python version and interpreter being used.

Click here to change the programming language. It should say Python for APS106.

- Take some time opening this on your personal computer
- On Line 3, replace Art Vandelay with your name
- Hit the triangle arrow to RUN or EXECUTE code



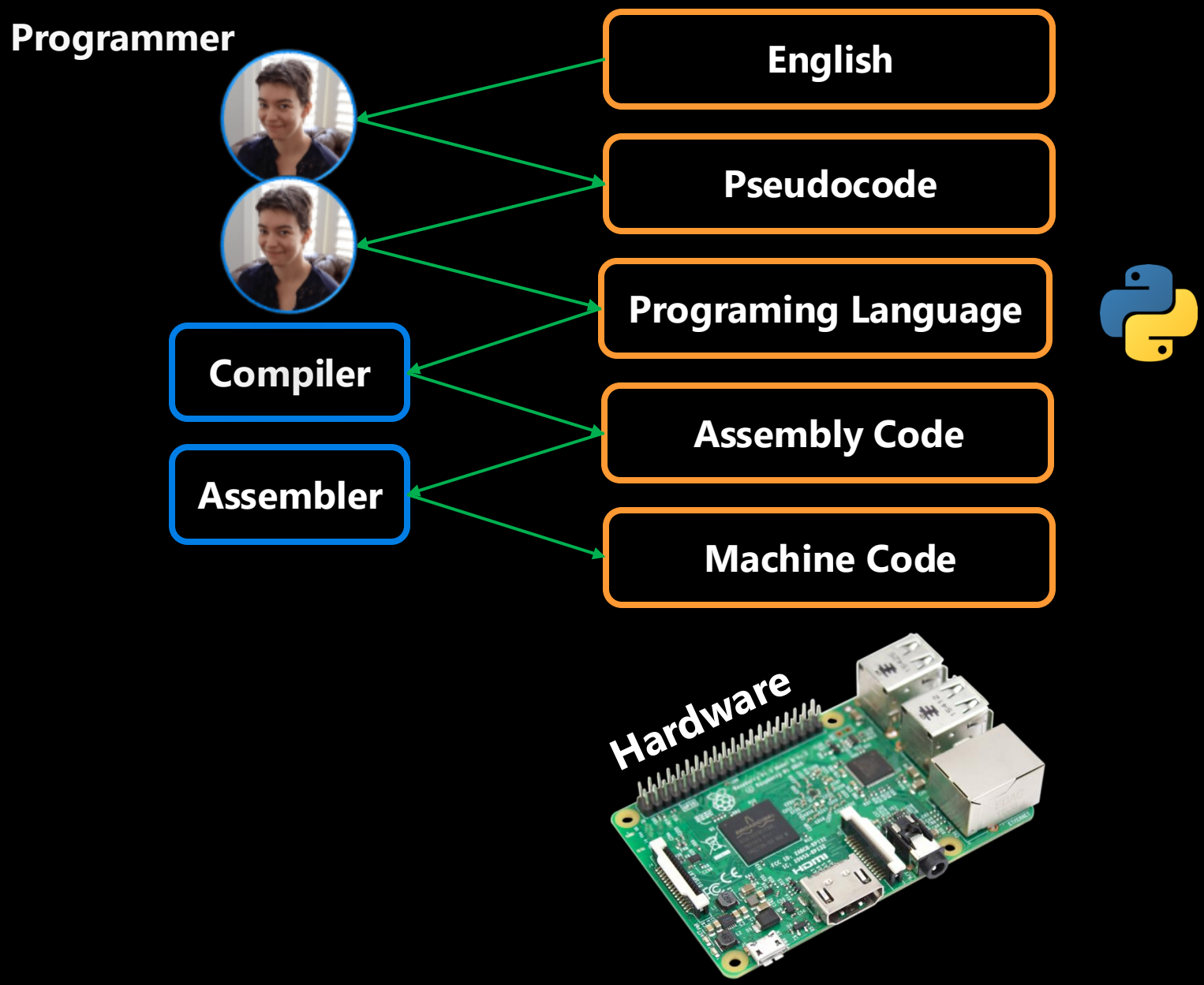
What is Programming?

- A way of telling a computer what to do.
- A computer can't infer (…yet).
  - Need to tell a computer every single step (or "instruction") it needs to do in a language it can understand.
  - How would you request an egg for breakfast to a chef and to a computer/robot?

- **To a Chef**
  1. Sunny-side up, please!

- **To a Computer**
  1. "Turn on stove"
  2. "Take out pan"
  3. "Take one egg out of fridge"
  4. "Crack egg"
  5. "Pour egg into pan"
  6. "Wait 5 minutes"

How to Program a Computer.

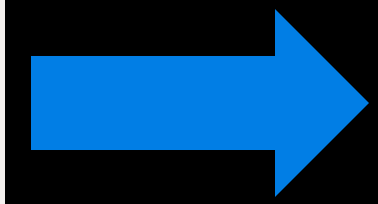


The power of programming languages

- `if x > 10:`

  `print("x is greater than 10")`

```
section .text
    global _start

_start:
    ; compare x with 10
    mov eax, [x]
    cmp eax, 10
    jle else_block ; jump to else_block if x <= 10

    ; if x > 10, print "x is greater than 10"
    mov edx, len_msg
    mov ecx, msg
    mov ebx, 1
    mov eax, 4
    int 0x80
    jmp end_if

else_block:
    ; code for else block goes here

end_if:
    ; code after if-else block goes here

section .data
    msg db "x is greater than 10", 0xa
    len_msg equ $ - msg
```



Packaging instructions for our computer

(1) Python converts to machine code

- Python takes human-readable instructions and converts this to machine code (binary: 1s and 0s)

Most of this is outside the scope of APS106.

If we can provide the correct Python instructions in **Step (1)**, the computer will do what we tell it to!

(2) Machine code determines transistor voltages

- This sequence of 1s and 0s determines which transistors in the CPU are ON and OFF, representing one specific instruction for the machine

(3) Instructions given sequentially on each clock cycle

- Each instruction is given in sequence and executed according to the processors clock speed (number of instructions per second)

- Next class we will learn how to assign variables to memory and begin using programming as a tool to solve problems!

APS106

UNIVERSITY OF TORONTO

The Coding Toolbox.

Week 1 | Lecture 2 (1.2)

Upcoming:

- Lab 0 released Thursday at 6 PM
- Reflection 1 released Friday at 6 PM

- First PRA section – Friday 3-5 PM

if nothing else, write #cleancode