***Backgounrd in Pytnhon + Unix***

*Python*

* Jupyter 🡪 originally the combination of 3 languages, Julia, Python and R.
* These days, Jupyter supports over 40 programming languages.
* NumPy and Pandas libraries are incredibly valuable for doing data science in Python.
* Python is powerful + surprisingly fast for **interpretive language,** which aregenerally slower b/c they'd run on top of an **interpreter** rather than being compiled directly from the machine on which they're running.
* Think of this as having a middle man between the code + the machine, which isn't there for a language like C or C++.
* Python works well with other languages + is often used as a **blue language**, running in between components written in other languages.
* Python plays well with other languages in Jupyter notebooks as well.
* Python, because it's interpreted, can run everywhere w/ Python installed
* B/c Python is open source, you can install it anywhere w/out worrying about licenses.
* Python has a number of features, like **dynamic typing** (no explicit types needed for variable **declaration**) + automatic memory management, which make it both easy to learn + read
* Everything in Python is an **object**
* “x = 3” causes a PyIntObject to be created, which hold the value of the object, in this case 3, along w/ other details for Python to work w/ under the hood (including the type of object, # of references to the object, etc.)
* When you say x = 3, Python creates a PyIntObject with a value of 3 + have your variable x point directly to that object.
* x is created on the **stack** (holds local values + is managed by the program) + the PyIntObject is created on the **heap** (holds dynamically-created data + is managed by the OS)
* If we then say x = 4.5, The **garbage collector** in Python automatically frees the space associated w/ that 1st PyIntObject b/c nothing is pointed to it anymore, and now we have a PyFloatObject
* If ever curious to know if 2 variables are pointing to the same object, use the **is** command (False if x points to a PyIntObject and y points to a PyFloatObject)
* To test for *numeric* equality, use the **==** operator.
* x.lower() 🡪 lower **method**
* Strings CANNOT be changed 🡪 using x.lower on a string returns a new value, but the x variable still holds the original value
* To change it, re-declare x w/ the new value 🡪 x = x.lower() 🡪 x now points to new lowercase object