

An overview of the Python programming ecosystem

Terence Parr
MSDS program
University of San Francisco

Executing programs

- Computers don't inherently understand Python or any other programming language
- A programmer has to create a program in some language X, called an interpreter, that understands statements in language Y (Python's interpreter is written in the C language)
- We'll use three interfaces to access a Python interpreter:
 - pythontutor.com (no setup and visualizes Python executions)
 - The UNIX commandline to run Python .py files and to interact with OS
 - Jupyter notebooks, a browser interface for code and notes
- We'll even learn to execute code on a remote cloud computer

Python tutor

<http://pythontutor.com/>

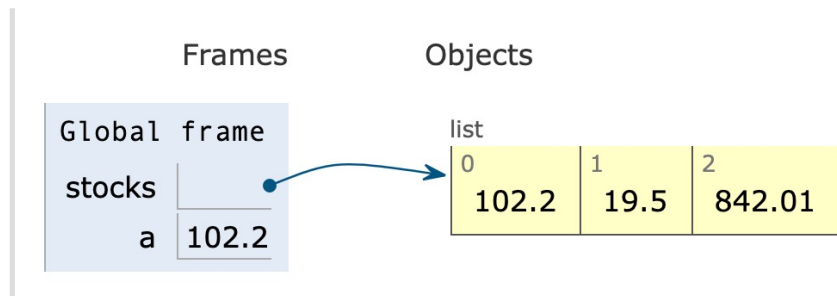
- Type/execute code without installing software on your laptop
- Visualizes the state of the Python interpreter
- Really nice interactive tool for learning Python (and other langs)

Python 3.6
([known limitations](#))

```
1 stocks = [102.2, 19.5, 842.01]
→ 2 a = stocks[0]
```

[Edit this code](#)

executed



Python from the commandline

- The commandline is built in to Macs / unix machines
- Python interpreter installed with [Anaconda](#)

Interactive

```
beast:~ $ python
Python 3.8.8 (default, Apr 13 2021, 12:59:45)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>> print("Hello, World")
Hello, World
>>> stocks = [102.2, 19.5, 842.01]
>>> a = stocks[0]
>>> print(a)
102.2
>>> █
```

batch

```
beast:~ $ cat t.py
stocks = [102.2, 19.5, 842.01]
a = stocks[0]
print(a)
beast:~ $ python t.py
102.2
beast:~ $ █
```

Python within a Jupyter notebook

- Interactive execution for code, notes, data, visualizations
- Installed automatically with [Anaconda](#)
- The Python interpreter keeps running in the background so we can interactively try different code snippets
- This will likely be your primary data science development environment

Simple notebook

A demonstration of notebooks

```
[1]: stocks = [102.2, 19.5, 842.01]
a = stocks[0]
```

Show **a** by referencing variable:

```
[2]: a
```

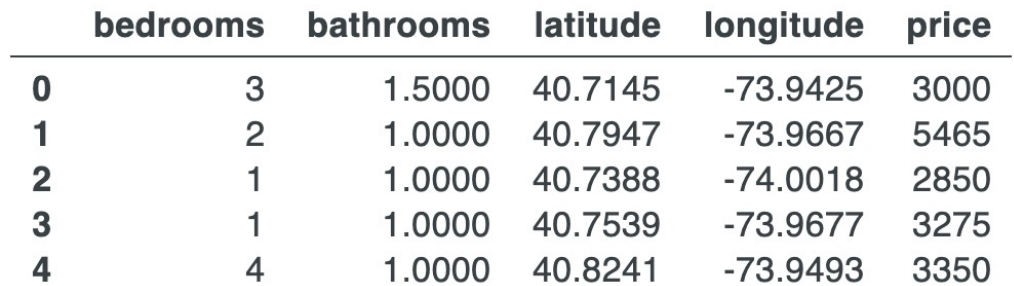
```
[2]: 102.2
```

```
[3]: stocks
```

```
[3]: [102.2, 19.5, 842.01]
```

Python libraries make us more productive

- Libraries provide repertoire of existing functionality we can leverage to boost productivity
- You'll use **matplotlib**, **numpy**, **pandas**, and **scikit-learn** extensively throughout the MSDS program
- The application programmers interface (API) is huge for these libraries and it takes a while to learn them, but they are very powerful
- Google / stack overflow will be your friends here as you learn
- Never ask what the parameters are to a library function; you must be independently functional so look it up yourself

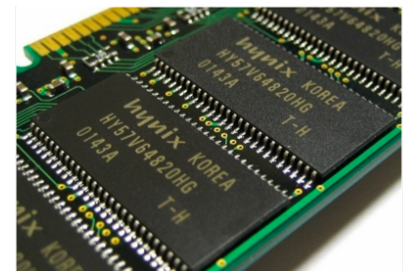


Getting to know your computer

A programmer's perspective

Computer components

- Processor (*CPU*)
- Memory (*RAM*) code+data
- Nonvolatile Storage (*disk*)
- *Network*
- CPU executes code and operates on data in RAM, saving results on disk
- Can send and receive data across the network



Processor

iMac (Retina 5K, 27-inch, Late 2014)

Processor 4 GHz Quad-Core Intel Core i7

Memory 32 GB 1600 MHz DDR3

Mac mini (M1, 2020)

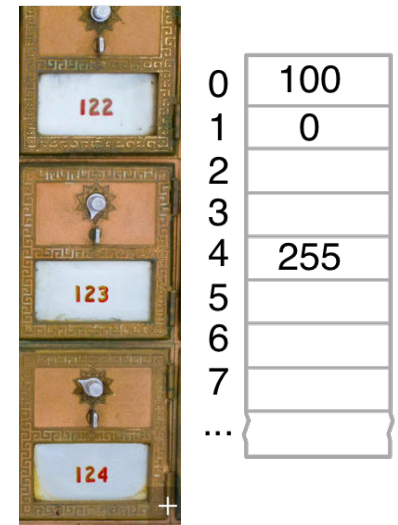
Chip Apple M1

Memory 16 GB

- Performs five principal operations:
 - load small chunks of data from memory into the CPU
 - perform arithmetic computations on data in the CPU
 - store small chunks of data back to memory
 - jump to a new location (this is how we loop)
 - jump to a new location if condition is true
- Each instruction does a tiny amount of work but CPU can execute billions of them per second
- E.g., **total = cost + tax** might take 2 loads, 1 add, and 1 store

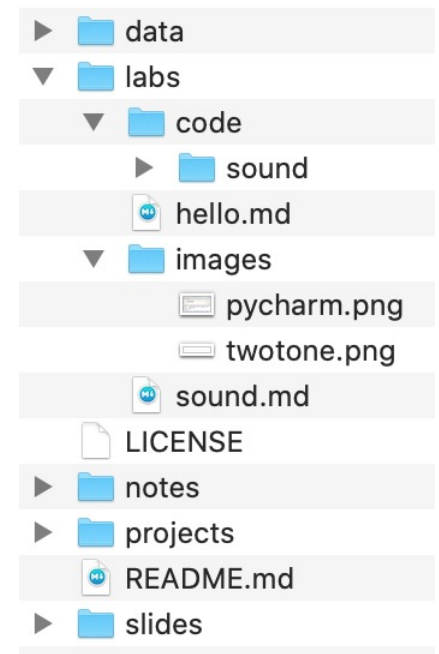
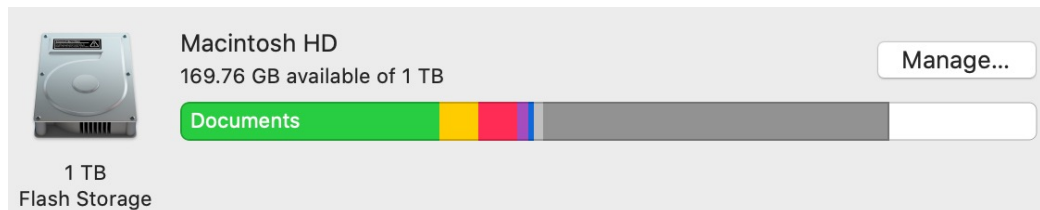
Memory

- RAM is much faster but usually much smaller than the disk and all RAM data is lost when the computer powers off
- Think of memory as your working or scratch space and the disk as your permanent storage
- Memory is broken up into discrete cells of a fixed size called a *byte* that can hold a number between 0 and 255 (8 bits)
- Cells have integer addresses just like mailboxes
- CPUs can read/write data at specific memory locations
- Everything from actual numbers to music to videos is broken down and stored as a sequence of bytes



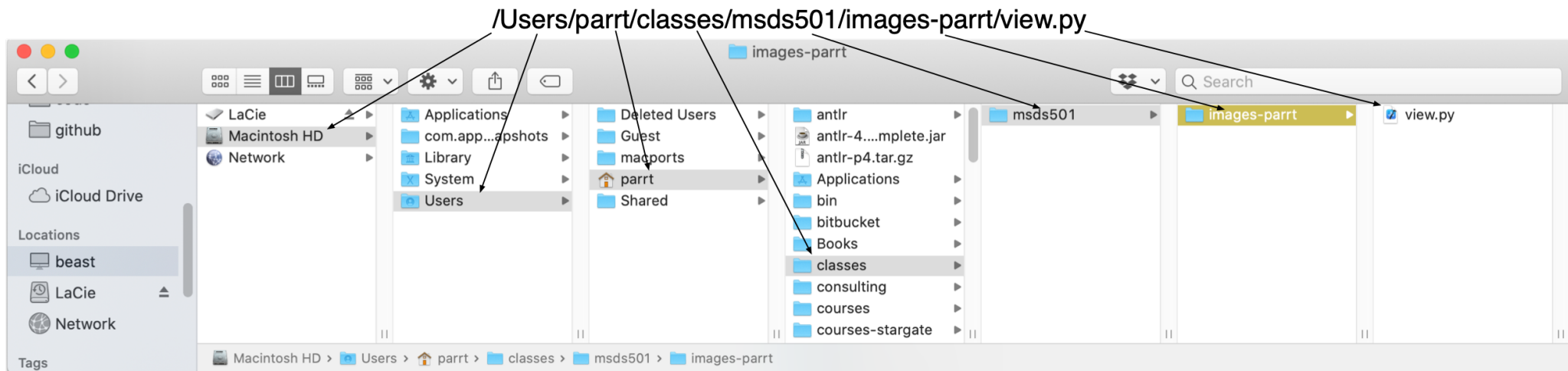
Disk/SSD (nonvolatile) storage

- *Files* hold data meant to be treated as a unit such as **ladygaga.mp3** or **sales.csv**
- *Directories* (folders) group files and other (sub-)directories
- Disk is a tree of folders with files in the leaves
- Executing programs have a *current working directory* and file specs (paths) are relative to that



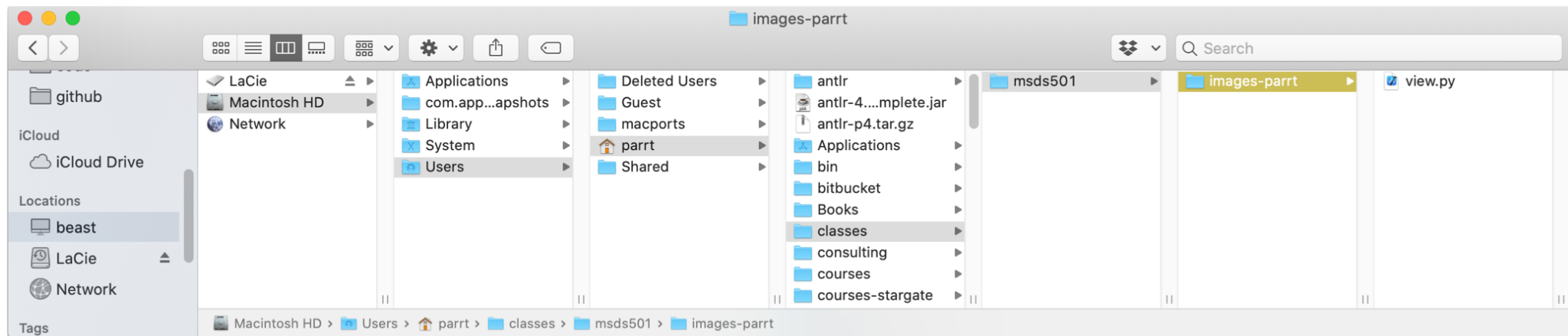
Paths to directories and files

- A *Path* is slash-separated sequence of dir names, optionally followed by a file name
 - The *root* is a single slash "/" and absolute paths start with "/"
 - Shorthand for your user home directory is "~" such as **/Users/parrt**
- *Relative paths* do not start with "/" or "~" and are relative to current working directory



Path examples

- The **parrt** dir you see is **/Users/parrt** or **~parrt** or just **~**
- If current working directory is **parrt** then **msds501** dir is **classes/msds501** relative to that
- If current working directory is **images-parrt**, path to **view.py** is just **view.py**



The shell / terminal / command line

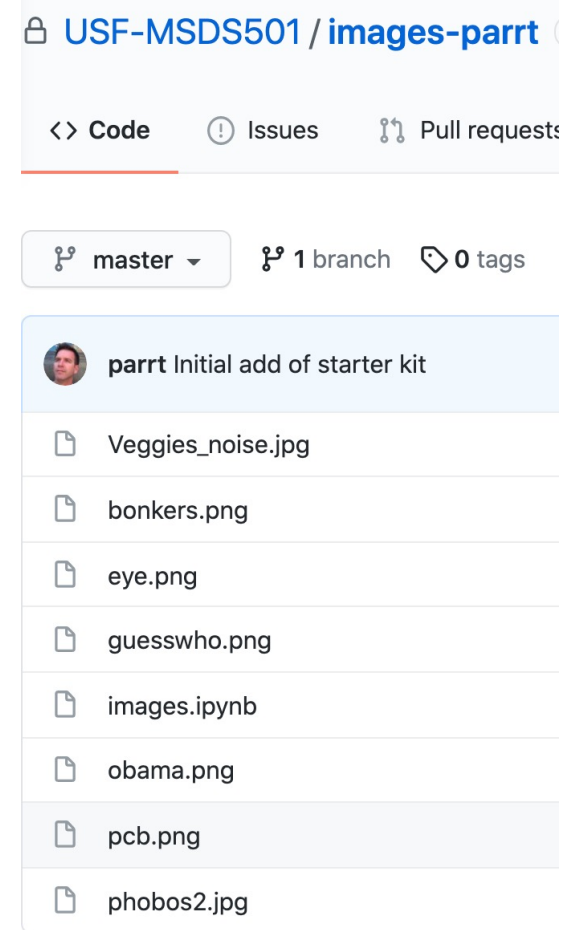
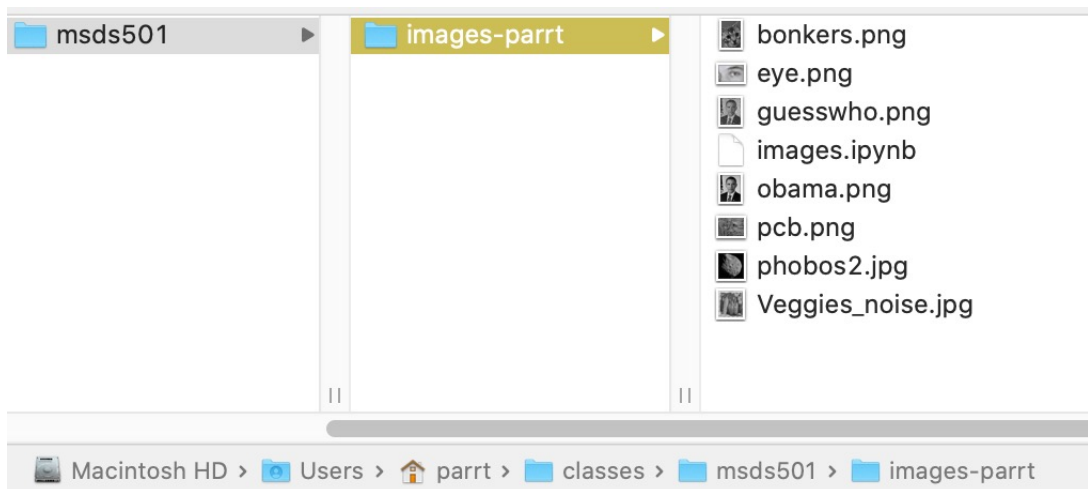
- Before GUIs, the terminal was all we had and what we used to examine files, check the state of the machine, execute programs, transmit files to other computers, etc.
- The terminal is running a "shell", a command interpreter that is another simple language, but one designed with commands to control your computer
- **echo** is like **print**
- **cd** changes directory
- **ls** lists files in directory



```
/Users/parrt/classes/msds501/images-parrt — -bash — 53x9
/Users/parrt/classes/msds501/images-parrt — -bash
[beast:~ $ echo "Hello, World"
Hello, World
[beast:~ $ cd classes/msds501/images-parrt
[beast:~/classes/msds501/images-parrt $ ls
view.py
beast:master:~/classes/msds501/images-parrt $
```

Git / github

- Git = version control for a *repository*, which is represented by a directory with your code, data
- Git tracks changes made to files in repo dir
- Github = website that hosts git repositories
- We push/pull from laptop to github to share code



Summary

- Programming in Python is about way more than just typing in grammatically correct code
- It's about conjuring up sequences of instructions that perform a specific task then coding that up in Python but...
- There's an entire ecosystem we have to deal with:
 - Machine CPU/RAM capabilities
 - Libraries
 - Commandline
 - Git
 - Github
 - Notebooks, script files
 - Cloud computing / distributed computing