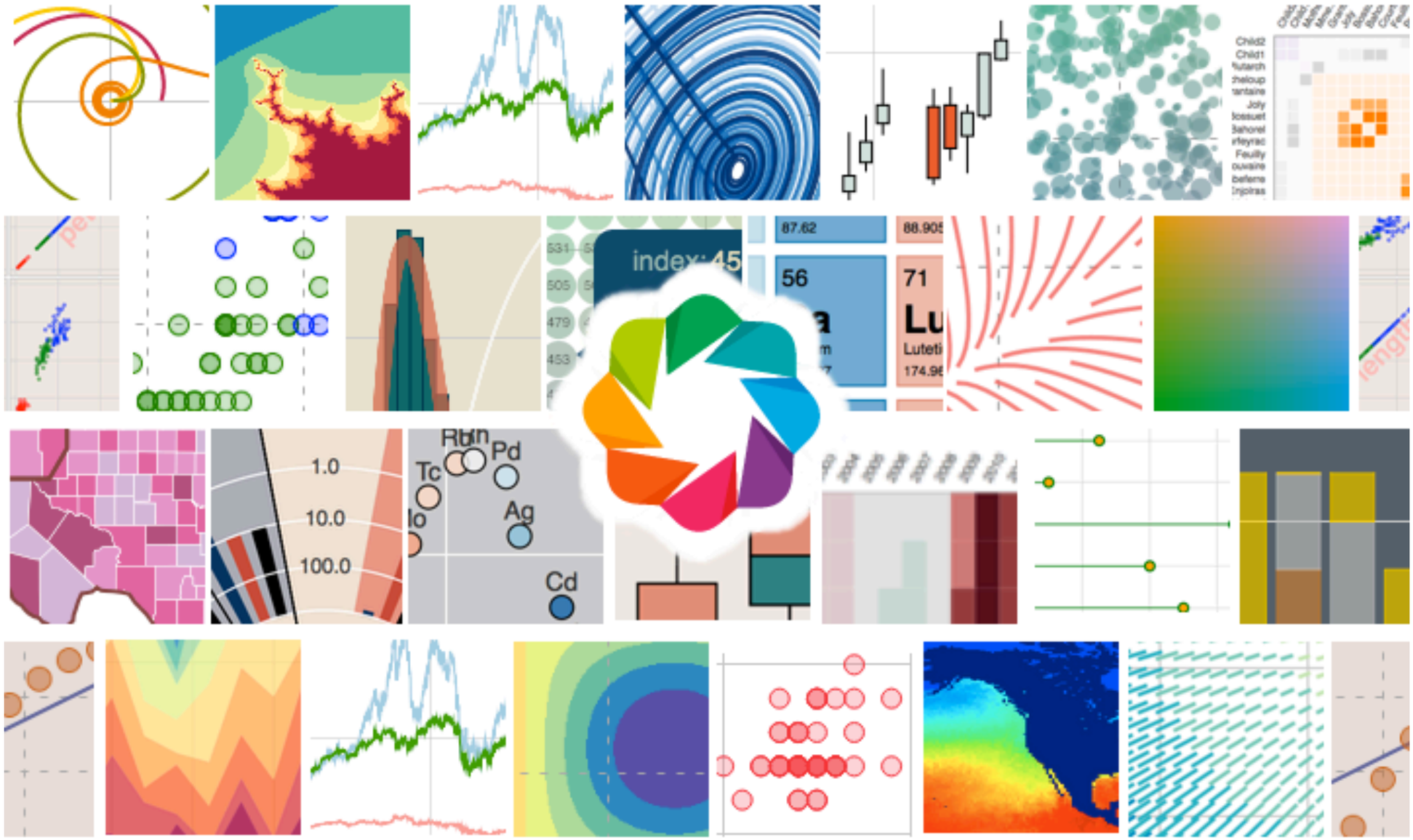


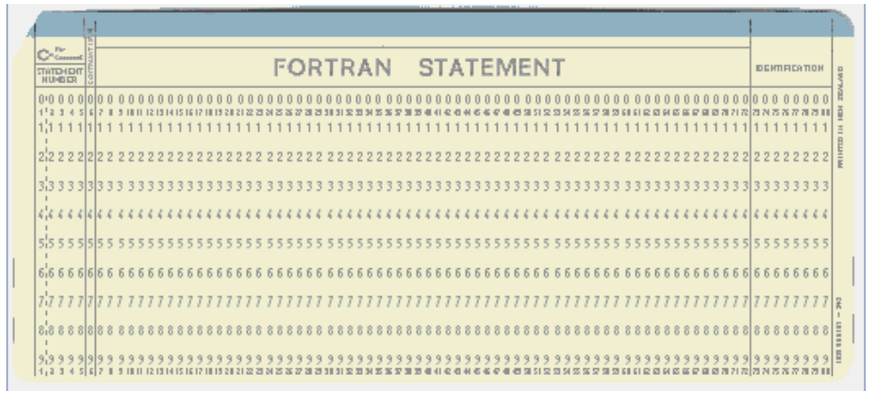
Python Visualization

Or How I Learned to Stop Worrying and Love ~~Pandas~~

astropy.Tables

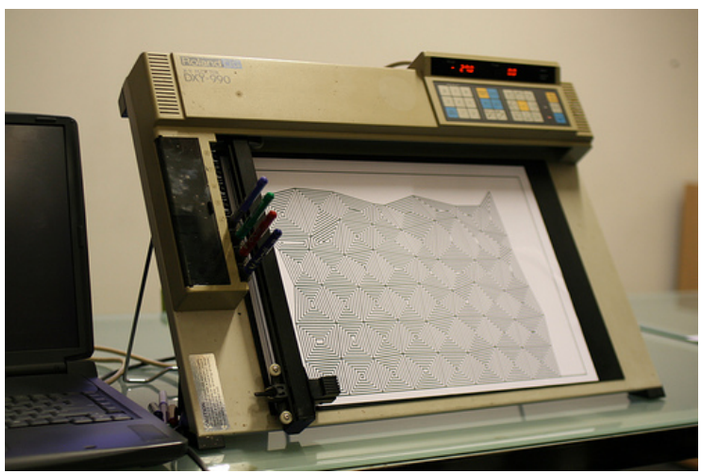
“Computers” circa 1890

Computers become machines

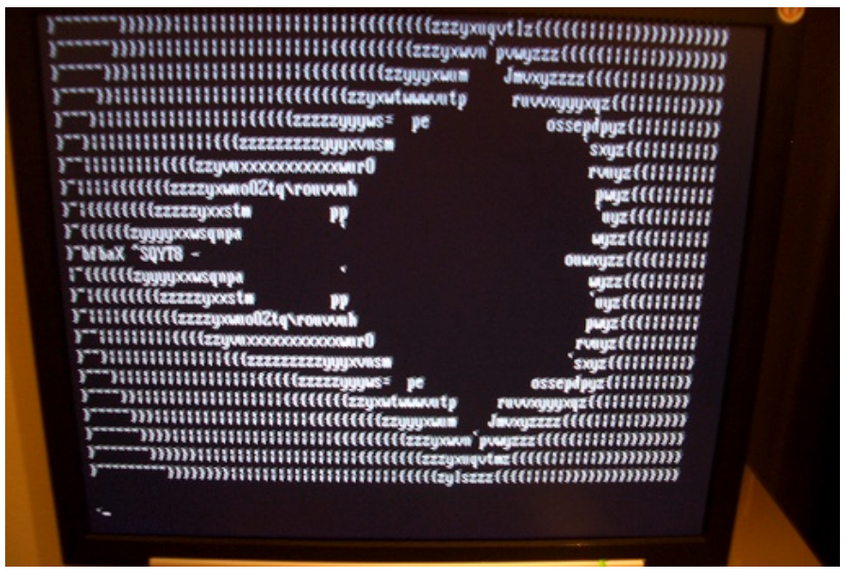


Punch cards and readers

VT100 “dumb" terminal attached to a mainframe



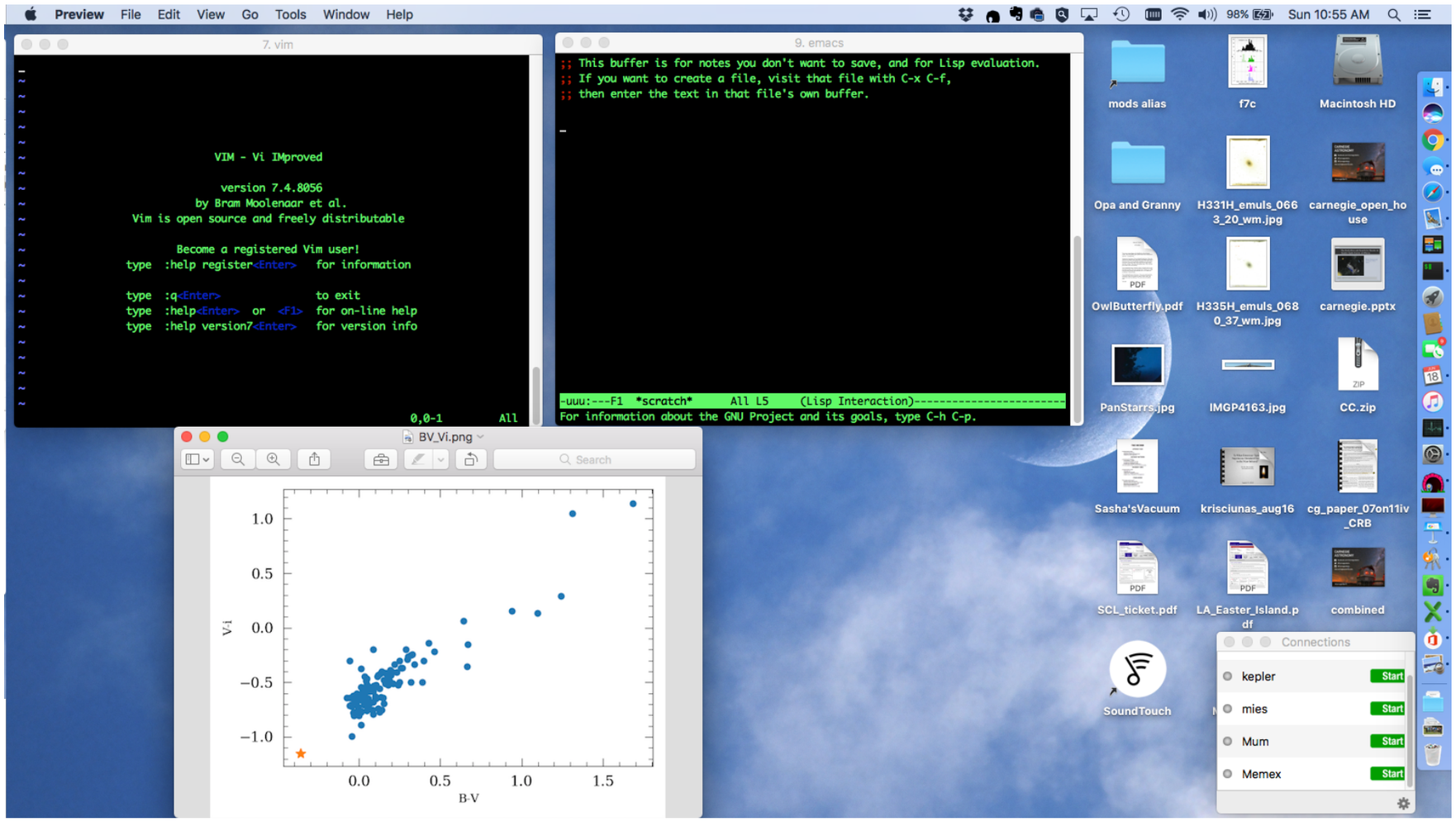
Pen plotter



Mandelbrot on PDP-11

Commodore vic-20

We've Come a Long Way… but

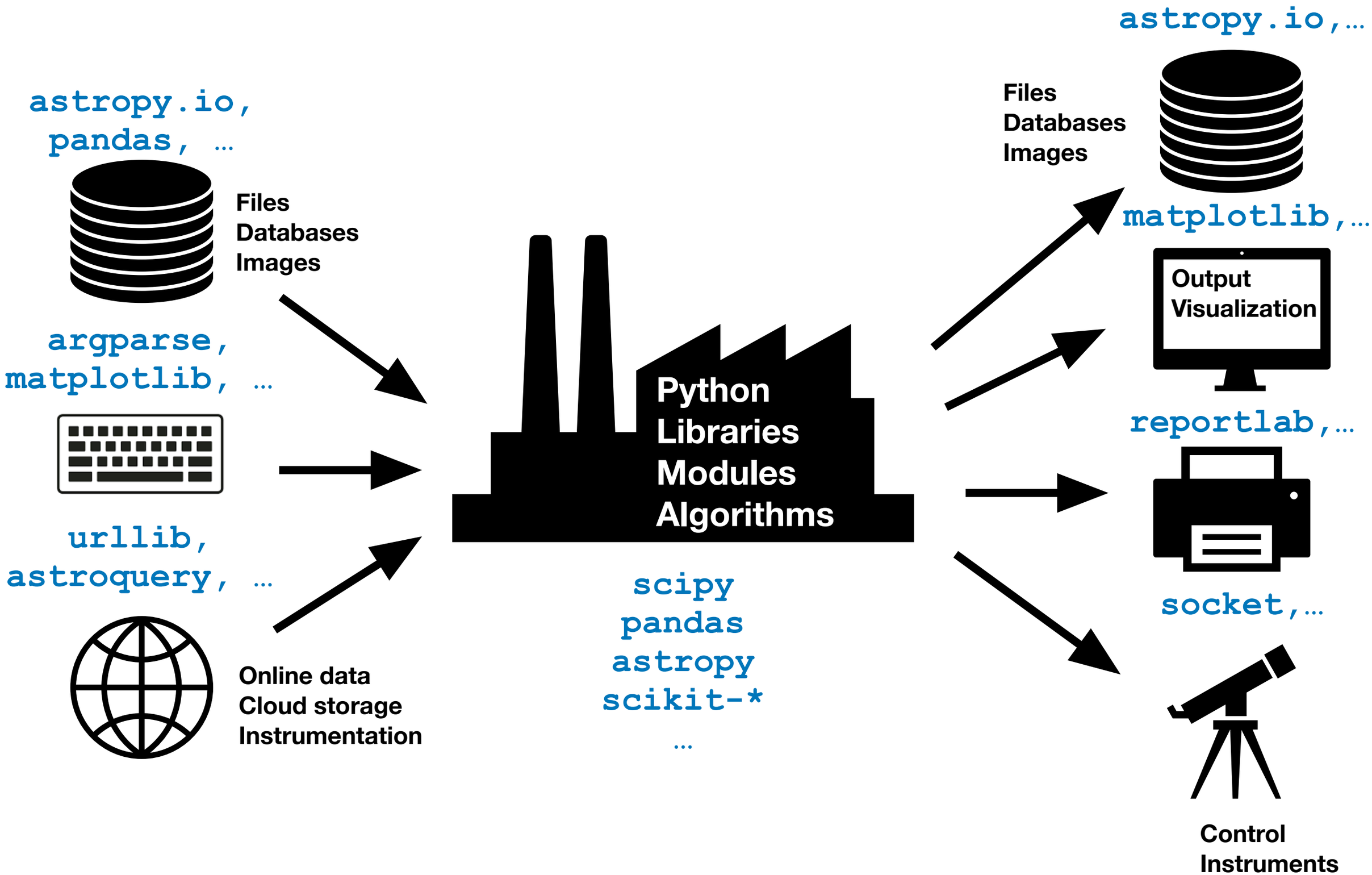


MacOS still has a built-in VT100 terminal emulator and runs on top of a UNIX operating system (Darwin).

It comes with (*and I still use*) the vi text editor, written in 1976.

You can still ruin your life with `rm -rf *`

Nevertheless, Programming is still the same basic concept



GUI vs CLI

(How to start a Flame War with 2 acronyms)

GUI

Pros:

- Everything is “exposed”
- Visualization is easier for the user
- User errors are minimized

Cons:

- Programming is MUCH more time consuming:
  - GUI libraries are more complex
  - All possible conditions must be handled.
  - Asynchronous programming
- Remote connections require high bandwidth
- Can’t run on a cluster job scheduler

CLI

Pros:

- Straightforward to code
- OS-agnostic
- works well over remote connections, even low bandwidth
- Can submit to cluster scheduler

Cons:

- More prone to user errors
- Nothing is exposed (need to memorize commands, arguments, etc).
- No visualization in general
- Not readily available on all platforms (Windows, iOS, etc)

Why Python?

- Fast to code
- Both GUI and CLI interfaces are built in and don’t depend on OS (in principle)
- Forces you to write readable code.
- (Relatively) easy to debug: excellent trace-backs
- Vast (ever increasing) library of modules (309,996 on pypi.org).
- (Relatively) easy to extend using C/C++/Fortran libraries
- Named after Monty Python’s Flying Circus. Not the snake.
- It’s free and open source.

Why *Not* Python?

Python is a fantastic and versatile language, but has some drawbacks.

- It's not fast. Not even close to the numerical performance you get from compiled languages like C/C++/Fortran
- To share your code with others, they have to have a compatible version of python, the same packages, etc.
  - The dreaded "dependency cascade".
- Vast (ever increasing) library of modules (309,996 on pypi.org). What is "standard"?

Top Data Visualization Modules

- **Matplotlib:** 2D and limited 3D plots. Fast becoming primary plotting library. http://matplotlib.org

- **Bokeh:** 2D plots via HTML. Emphasis on interactivity. Fairly new, but so far looks very promising. http://bokeh.pydata.org

- **Mayavi:** 3D visualization. GUI interface as well as scripting from python. http://mayavi.sourceforge.net

- **VTK:** High-performance imaging (used for medical imaging). Interface is very low-level (practically C++) http://vtk.org

- **Vpython:** "3D Programming for Ordinary Mortals". Scenes, ray-tracing, animation. http://vpython.org

- **YT:** Visualizing volumetric data. http://yt-project.org

- 1000's of usage-specific modules that use the above. http://pypi.python.org

The Hardest Thing about Programming in Python: Version consistency!

- Not only the version of python itself, but all those open-source packages
- Anaconda (conda) is one of the attempts to keep consistency: sets of packages that work together
- Try to minimize the number of external packages you need. Stick to the “main” ones if at all possible (numpy, scipy, astropy, matplotlib, etc)
- If you need a very specific set of packages and versions, use conda environments. They’re also handy for replication.