6.009: Fundamentals of Programming

Lecture -1: Programming Beyond 6.009

- Review of 6.009 Big Ideas
- What's Next?

Adam Hartz

`hz@mit.edu`

17 May 2021

6.009: Goals

Our goals involve helping you develop your programming skills, in multiple aspects:

- **Programming:** analyzing problems, developing plans
- **Coding:** translating plans into Python
- **Debugging:** developing test cases, verifying correctness, finding and fixing errors

So we will spend time discussing (and practicing!):

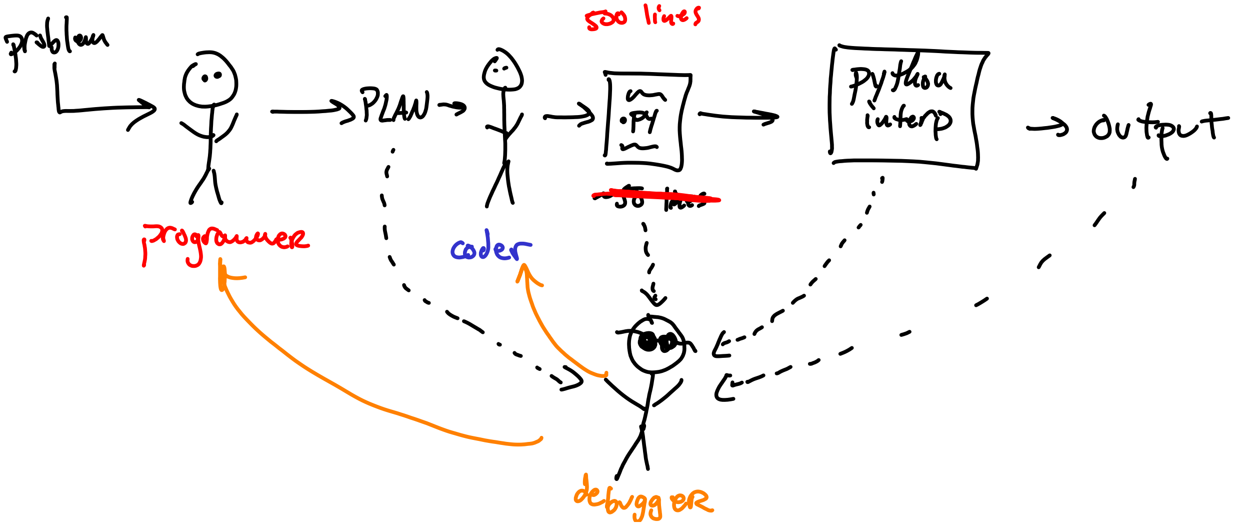
- high-level design strategies
- ways to manage complexity $\rightarrow$ style (PEP 8), abstraction
- details and "goodies" of Python
- a mental model of Python's operation
- testing and debugging strategies

Lots of Cool, Challenging Problems!

- Audio Processing
- Image Processing
  - Convolutional Filters
  - Color Images
  - Seam Carving
- Bacon Numbers / Path Finding
- Path Planning in the USA
- N-dimensional Minesweeper
- SAT Solver
- Autocomplete
- Symbolic Algebra
- LISP Interpreter

6.009 Overview

- improving “behind the scenes” understanding
- managing complexity as programs grow
- filling your “toolbox” with common techniques/strategies
- practice with programming, coding, and debugging



Growth, not Perfection

ONE YEAR
Baby step
Baby step
Baby step
Baby step
© Sarah Andersen

What's Next?

Two perspectives:

- ~~What else exists within Python?~~ what next programming?
- What comes next in terms of subjects?

course 6

Python Standard Library Highlights

Another reason to like Python (which we've not really utilized so far) is that it has a huge *standard library* of useful modules/functions/classes. We certainly can't talk about it all here (see `https://docs.python.org/3/library/index.html`, the list is **huge**), but we can talk briefly about a couple of highlights:

- various collections (beyond lists, sets, etc): `collections`
- tools for working with iterators and functions: `itertools`, `functools`
- mathy things: `math`, `cmath`, `random`, `statistics`
- rational numbers: `fractions`
- tools for working with functions: `functools`
- implementations of built-in operations as functions: `operator`
- tools for interacting with operating system: `os`, `sys`
- tools for dealing with errors/reporting: `traceback`, `logging`
- tools for creating/interacting with Internet protocols/etc
  - `email`, `smtplib`, etc
  - `http.server`, `urllib.request`, etc

lambda x,y: x+y
lambda x,y: x-y

These modules can be super useful, but aren't really worth talking about here.

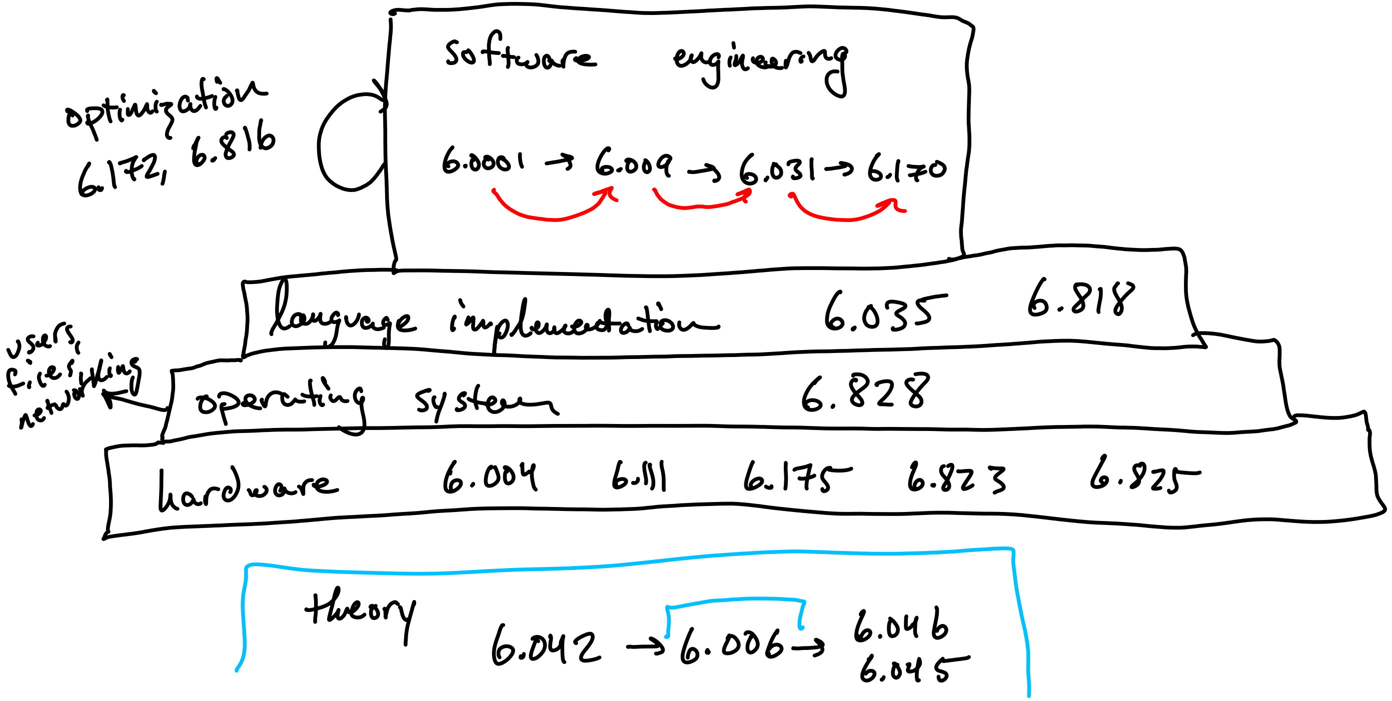
External Packages

Outside of the standard library, there are a wealth of other useful packages!

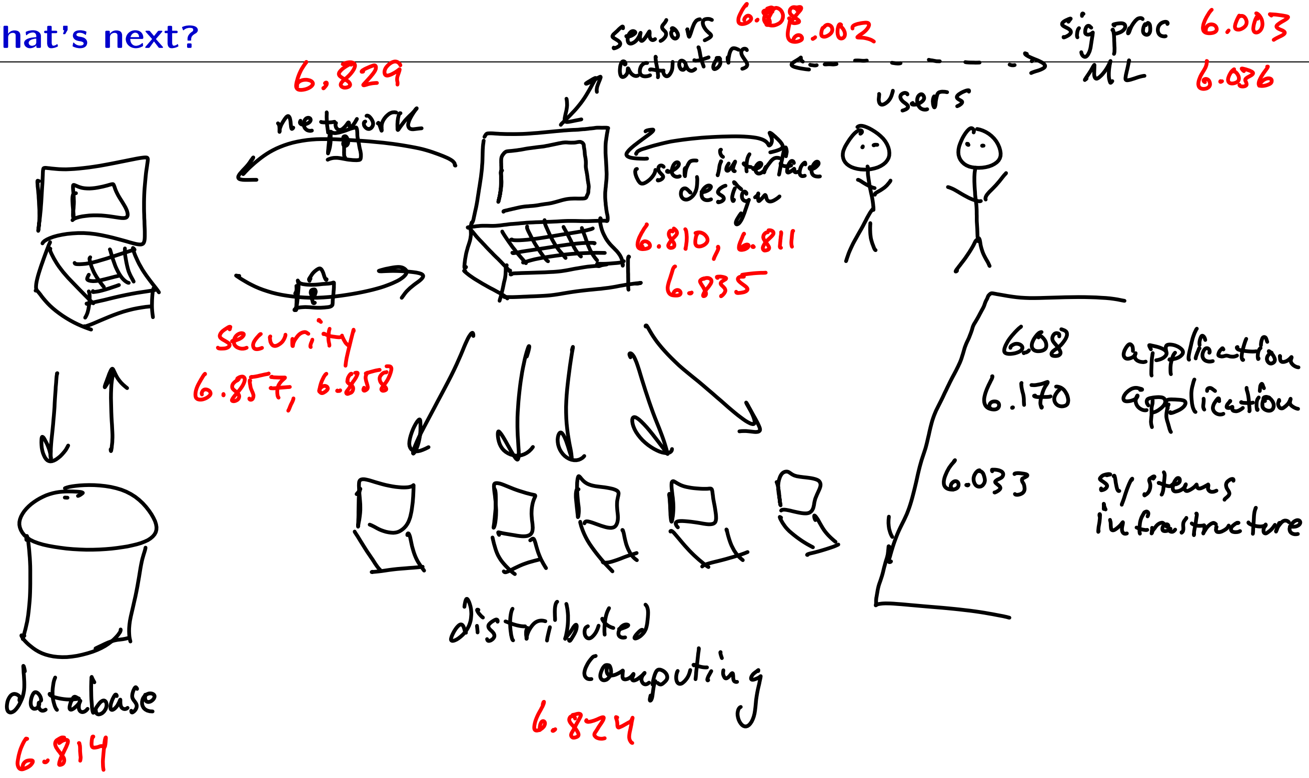
Examples:

- `sympy` for symbolic algebra
- `numpy` for numeric computation (fast operations on large multi-dim arrays+matrices)
- `matplotlib` for generating plots
- `nltk` for natural language processing
- `mypy` for static analysis of code
- etc, etc, etc

What's next? in terms of course 6 subjects



What's next?


sensors actuators
6.08 6.002
sig proc 6.003
ML 6.036
6.829
network
security
6.857, 6.858
user interface design
6.810, 6.811
6.835
users
6.08 application
6.170 application
6.033 systems infrastructure
database
6.814
distributed computing
6.824