6.009: Fundamentals of Programming

Lecture 0: Welcome to 6.009!

Adam Hartz

`hz@mit.edu`

15 February 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:

- high-level design strategies
- ways to manage complexity
- details and "goodies" of Python
- a mental model of Python's operation
- testing and debugging strategies

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:

- high-level design strategies
- ways to manage complexity
- details and "goodies" of Python
- a mental model of Python's operation
- testing and debugging strategies

...but discussion only goes so far!

6.009: Pedagogy

Learning to program is a lot like learning a musical instrument or a sport.

How does one learn those things?

6.009: Pedagogy

Learning to program is a lot like learning a musical instrument or a sport.
How does one learn those things?

Just like with music/sports, **deliberate practice** is key!

To improve as a programmer, it helps to:

- watch how experienced programmers approach problems
- program!
- receive feedback from more experienced programmers

6.009 aims to provide you with *lots* of opportunities for all of these

6.009: Pedagogy

Learning to program is a lot like learning a musical instrument or a sport.
How does one learn those things?

Just like with music/sports, **deliberate practice** is key!

To improve as a programmer, it helps to:

- watch how experienced programmers approach problems
- program!
- receive feedback from more experienced programmers

6.009 aims to provide you with *lots* of opportunities for all of these

- Labs give opportunities to practice new techniques/skills to solve interesting problems.
- Lectures/recitations equip you with tools useful for attacking those problems.
- Checkoffs and office hours give opportunities to receive expert feedback.

6.009: A Typical Week

A typical week centers around a lab assignment, supplemented by instructor presentations and with lots of help available.

- **Lecture:** Release ~Friday afternoon, Q&A session Monday 11am ET
- **Recitation:** Wed, 1-hour blocks from 8am-7pm Eastern

- **Office Hours**
  - Monday Evenings, 7pm-10pm Eastern
  - Tuesday Mornings, 7am-10am Eastern
  - Tuesday Evenings, 7pm-10pm Eastern
  - Wednesday Evenings, 7pm-10pm Eastern
  - Thursday Mornings, 7am-10am Eastern
  - Thursday Evenings, 7pm-10pm Eastern
  - Fridays, 7am-10am and 11am-5pm Eastern
  - Sundays, 7am-10pm Eastern

Labs: the Heart of 6.009

Logistics:

- Typically issued Fridays at ~6am Eastern
- Mix of conceptual questions and writing code (Python 3.6+, 3.9 recommended)
- Sometimes, some questions are due Mondays at 11am (before the nominal lecture time)
- Bulk of the lab is due the following Friday at 5pm Eastern
- Checkoff meetings are due on Wednesday at 10pm Eastern
- Lateness policy described on web site

Cool Problems!

- Audio/Image Processing, Minesweeper, SAT Solver, LISP Interpreter, ...

6.009: Web Site

Just about everything in 6.009 happens via the web site:

`http://mit.edu/6.009`

Labs: Logistics / Infrastructure

(demo)

Getting the Most Out of 6.009

Getting the Most Out of 6.009

Lectures/Recitations:

- Step 1: Come to lecture/recitation, and participate!
- Take notes *in your own words* and review them later
- **Ask questions!** We want to have a conversation.

Getting the Most Out of 6.009

Lectures/Recitations:

- Step 1: Come to lecture/recitation, and participate!
- Take notes *in your own words* and review them later
- **Ask questions!** We want to have a conversation.

Labs:

- Start early (labs are week-long assignments)
- Formulate a plan before writing code
  - Try to understand the problem thoroughly before writing code
  - When things go wrong, step away from the code and revisit the plan
- Work through problems on your own
- Ask for help when you need it!
  - Labs are intentionally challenging
  - Bugs are a natural part of life
  - Lots of opportunities for help (office hours / forum)

Growth, not Perfection

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

Check Yourself!

What happens when the following program is run?

```
functions = []
for i in range(5):
    def func(x):
        return x + i
    functions.append(func)

for f in functions:
    print(f(12))
```

1. It prints 12, then 13, then ..., then 16
2. It prints 13, then 14, then ..., then 17
3. It prints 16, then 15, then ..., then 12
4. It prints 17, then 16, then ..., then 13
5. A Python error occurs
6. Something else

This Week: Python Mental Model