

6.009

Fundamentals of Programming

Lecture 1:

Python Notional Machine

Adam Hartz
hz@mit.edu

6.009: Staff

Instructors:



Duane Boning



Erik Demaine



Adam Hartz



Armando Solar-Lezama

6.009: Web Site

Just about everything in 6.009 happens via the web site:

`http://mit.edu/6.009`

6.009: Goals

Our goals involve helping you develop as a programmer, in multiple aspects:

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

6.009: Goals

Our goals involve helping you develop as a programmer, 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 as a programmer, 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, practice is key!
To improve as a programmer, you have to program.
And 6.009 asks you to program...a lot!

- Labs give opportunities to practice new techniques/skills to solve interesting problems.
- Lectures/tutorials equip you with tools useful for attacking those problems.

6.009: A typical week

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

- **Lecture:** Tue 9:30-11am, 34-101
- **Tutorials:** Wed 9-11, 10-12, or 1-3
(first 30-60 minutes of session)
- **Office Hours:**
 - ▶ Wed 9-11, 10-12, or 1-3
(last 60-90 minutes of session)
 - ▶ Fri 9-11, 10-12, or 1-3

Labs: the Heart of 6.009

Logistics:

- Issued Tuesdays at 11am
- Typically a mix of conceptual questions and writing code (Python 3.5+)
- Submissions to questions/code due the following Monday at 10pm
- Checkoff meetings during office hours; due 4 days after submission (nominally, Fridays at 3pm)
- Lateness policy described on web site

Cool Problems!

- Image Processing, Minesweeper, SAT Solver, LISP Interpreter, Platforming Game, ...

6.009 Grading

- 9 labs, two of which are 2-week “double labs”. (44 total points)
- 3 quizzes, 20 points each. Resubmission opportunity for partial credit.
- See web site for more details and points-to-grade mapping.

Getting the Most out of 6.009

Getting the Most out of 6.009

Lectures/Tutorials:

- Step 1: Come to lecture/tutorial!
- 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/Tutorials:

- Step 1: Come to lecture/tutorial!
- 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, 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 / Piazza)

Collaboration Policy

Our goal is that *every student* develops these skills throughout the course. Collaborating too closely with others (or outsourcing pieces to other students or StackOverflow) can rob you of an opportunity to develop those skills in yourself.

Please read our collaboration policy carefully.

<https://eecs6009.mit.edu/spring18/collaboration>

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))
```

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

The Rest of Today

- Python Notional Machine

