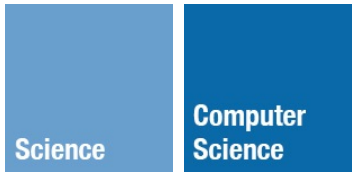
Preliminaries



CS 331: Data Structures and Algorithms

Michael Saelee <lee@iit.edu>

Michael (Sae) Lee

- lee@iit.edu
- http://moss.cs.iit.edu
- Office: SB 226A

Agenda

- Course overview & Administrivia
  - Prerequisites
  - Topics & Resources
  - Grading
  - Dev environment & Class procedures

Data Structures

- How do we store, organize, and retrieve data on a computer?

& Algorithms

- How can we efficiently (in space/time) carry out some typical data processing operations?
- How do we analyze and describe their performance?

Prerequisites

- I assume you are …
  - fluent in some programming language
    - familiar with procedural & OO paradigms
  - comfortable with development processes:
    - compilation, debugging, testing

Python

- We'll use the Python programming language to explore data structures & algorithms
- Easy-to-learn, clean ("one obvious way to do" things), and popular language
  - Ton of useful, powerful libraries

Topics

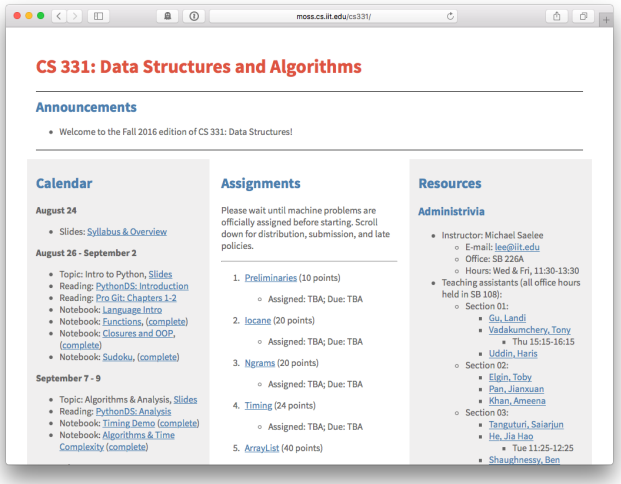
- Python crash course
- Algorithmic analysis
- Linear data structures (Lists, Stacks, Queues)
- Hashing and Hashtables (aka Maps)
- Recursion and Trees

Online resources

1. Course website: moss.cs.iit.edu/cs331

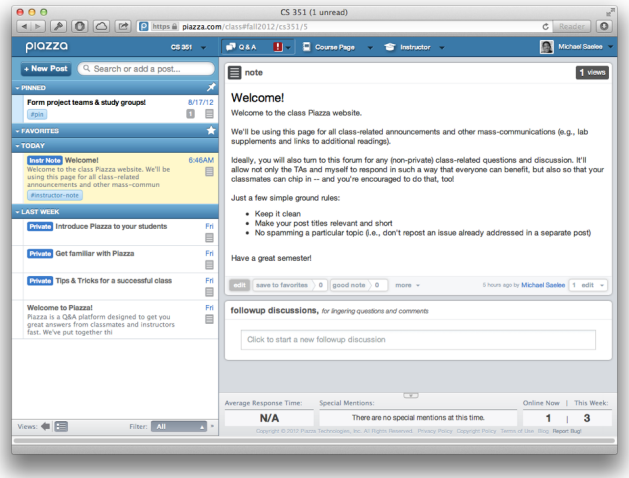
   - static information

     - lecture calendar, lab writeups, slides, screencasts, links, etc.



Online resources

2. Piazza: discussion forum

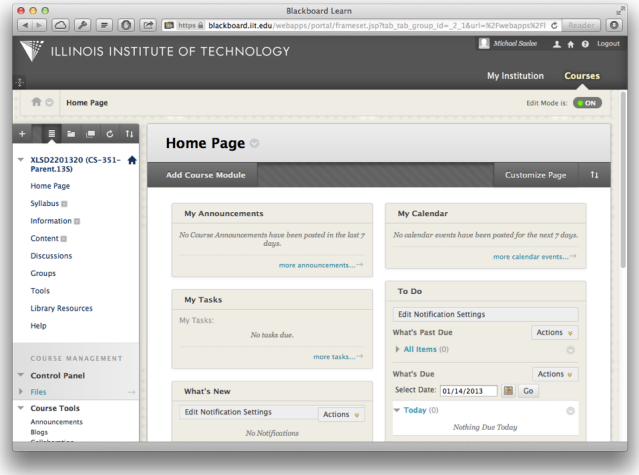


- all class-related questions
- monitored by TAs
- scales *way* better than e-mail
- announcements, links to additional readings & resources

Online resources

3. Blackboard

   - *only for grade reporting*!

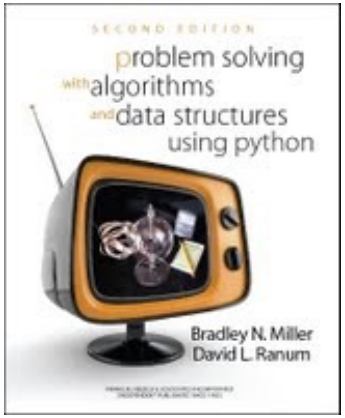


Online resources

4. Live online discussion forum for remote students
   - mechanism TBA (Slack?) by TA
   - virtual lab session + office hours

Supplements

- The Python Tutorial (docs.python.org/3/)
- Problem Solving with Algorithms and Data Structures Using Python



Grading

- 40% Machine Problems
- 20% Quizzes / Self-evaluation
- 40% Exams (3 total: 2 midterms + final)

On Exams and Scores

- Exams are all cumulative
- Higher scores on later exams will replace lower-scoring, earlier exams

```
>> scores = [60, 80, 75]
>> [max(scores[i:]) for i in range(3)]
[80, 80, 75]

>> scores = [75, 80, 100]
>> [max(scores[i:]) for i in range(3)]
[100, 100, 100]
```

Machine Problems

- Programming assignment(s) every 1-2 weeks
- All assignments are retrieved and submitted via the class Jupiter Notebook server: braeburn.cs.iit.edu
  - Log in using @hawk.iit.edu Google ID

Jupyter Notebooks

- In-browser Python development platform
  - "Cells" can contain plain text, code, output (and more)
  - All lecture notes, demos, and assignments will be distributed as notebook files

Jupyter Notebooks

- You should install a notebook server locally for convenience and in-class work
  - Install via Anaconda (with Python3) — see http://jupyter.org/install.html
  - But *all work must be tested and submitted* on the class server! (Lab 1 will go over this)

Class procedure

- Review reading before arriving to class
- Download starter notebooks before class
- Class will consist of lots of interactive demos (code along with me!)
- Completed notebooks are always posted

For Friday

- Start reading chapter 1 of PythonDS
- Install Jupyter notebook server locally (demo in a moment)
  - Open up “Language Intro” notebook to verify your setup works
- Log in to class server to confirm account