Objectives

- Concluding CSCI111
  - Other programming languages
  - What is computer science?

Applying What You Know To Other Languages

- At the beginning of the semester, some of you wondered
  - "Why Python?"
  - "Will I be able to read/write programs in other programming languages?"
- We'll answer the first question by showing that you can do the second

Applying What You Know To Other Languages

- **Syntax**: symbols used
- **Semantics**: what the symbols *mean*

What is the Python Program Doing?

- Page 4 of handouts

What is the Python Program Doing?

- Getting a line of input from "standard in" (from the user)
- Splitting the input into integers
- Calculating the result to a formula
- Deciding if a student is admitted, based on the result of the formula

Admissions Problem

- Binary University decides to admit students based on a formula that weighs various factors
  - Scores of 70 or better are admitted
- Input: single line, 4 integers, in order below

Category	Range	Weight Factor (Multiplier)
High School GPA	0 - 10	0.25
SAT score	600-2400	.01
AP Courses	0-10	10
Intangibles	1-10	8

Example Input/Expected Output

Input	Expected Output
0 1 0 300	DENY
6 10 99 2390	ADMIT
0 7 82 1500	ADMIT
2 5 0 990	DENY
2 5 0 1000	ADMIT
2 5 0 1010	ADMIT

What is the Python Program Doing?

- Getting a line of input from "standard in" (from the user)
- Splitting the input into integers
- Calculating the result to a formula
- Deciding if a student is admitted, based on the result of the formula

Identify these pieces in the other programs

Example Bash Program

- `printLab.sh`

Comparing Programming Languages

- How is the syntax/semantics of these languages different from Python?
- What is easier or harder to do in these other programming languages than in Python?

Comparing Programming Languages

- Benefits of Python:
  - Simpler syntax (e.g., fewer {} and ())
  - Can cover some content with less overhead
- Drawbacks
  - Data types aren't explicit (static)
    - Can be harder for you to remember and keep straight
  - Not compiled explicitly beforehand
    - Keep executing to find all the syntax bugs
    - Doesn't check: "you're passing a file instead of a string"
  - Allows you to do some crazy stuff that won't work in other programming languages

Who Uses Python?

- Google
  - Backends of Gmail and Google Maps and search-engine internals
- NASA
  - Collaborative engineering
- Yahoo
  - Groups: Maintain discussion groups; Maps
- RedHat Linux
  - System infrastructure
- Original BitTorrent client; Youtube; Civilization IV

Source: `http://wiki.python.org/moin/OrganizationsUsingPython`

Tiobe Index

Based on number of hits on web

Position Apr 2011	Position Apr 2010	Delta in Position	Programming Language	Ratings Apr 2011	Delta Apr 2010	Status
1	2	⬆	Java	19.043%	+0.99%	A
2	1	⬇	C	16.162%	-1.90%	A
3	3	=	C++	9.225%	-0.48%	A
4	6	⬆⬆	C#	7.185%	+2.75%	A
5	4	⬇	PHP	6.584%	-3.08%	A
6	7	⬆	Python	4.931%	+0.73%	A
7	5	⬇⬇	(Visual) Basic	4.682%	-1.71%	A
8	11	⬆⬆⬆	Objective-C	4.386%	+2.10%	A
9	8	⬇	Perl	1.991%	-1.56%	A
10	10	=	JavaScript	1.513%	-0.96%	A
11	12	⬆	Ruby	1.482%	-0.74%	A
12	20	⬆⬆⬆⬆⬆⬆⬆⬆	Lua	1.035%	+0.51%	A
13	9	⬇⬇⬇⬇	Delphi	1.034%	-1.68%	A
14	-	=	Assembly	0.967%	0.00%	A
15	23	⬆⬆⬆⬆⬆⬆⬆⬆	Lisp	0.934%	+0.45%	A

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Computer Science != Programming

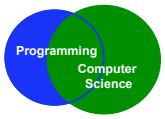
programming : CS ::

machining : engineering

grammar : literature

equations : mathematics

walking : W&L



a vehicle, not a destination

Computer Science Fields

Systems
- Architecture
- Operating systems
- Networks
- Distributed and parallel systems
- Databases
- Security
- ...

Software
- Compilers
- Graphics
- Software engineering
- Software testing and verification
- ...

Theory
- Algorithms
- Theory of computation
- ...

Other
- Artificial intelligence
- Robotics
- Natural language processing
- Bioinformatics
- Visualization
- Numerical analysis
- ...

- Often research involves combinations of these fields
- Not just programming!
  - But programming is a tool to do much, much more!

Computer Science Fields

Systems
- Architecture *
- Operating systems *
- Networks *
- Distributed * and parallel systems
- Databases *
- Security*
- ...

Software
- Compilers *
- Graphics *
- Software engineering *
- Software testing * and verification*
- ...

Theory
- Algorithms *
- Theory of computation
- ...

Other
- Artificial intelligence *
- Robotics *
- Natural language processing*
- Bioinformatics
- Visualization*
- Numerical analysis
- ...

* = field we discussed or did a problem in
  - Some are a stretch :)

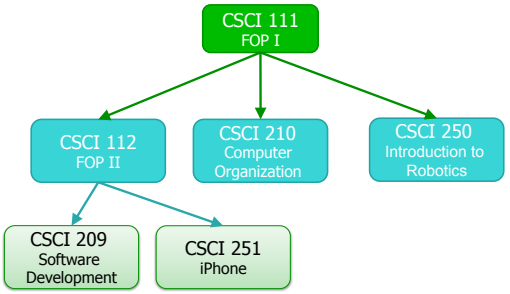
CS == Complexity Science

- Study of Complexity
  - How can it be done?
    - Based on **information**
    - Managing, manipulating data
    - Possible algorithms
  - How well can it be done?
    - Most **efficient** algorithm in terms of time and/or space
  - Can it be done at all?
    - Often, proof is a program--an implementation of the above

Conclusions

- See impact of computer science on your life
- Understand some of the computing issues better
  - Taking out some of the mystery
  - Security, testing, debugging, efficiency
- Algorithms are everywhere
  - Process for solving problems, **efficiently**
  - Mapping human intuition to systematic/automatic process

Where Can You Go from Here?



Reminders

- Finals are taken in the lab classroom
  - No computers
  - If need to change your time, sheet upstairs outside the CS department office
- Evaluations due Sunday at midnight on Sakai (tests and quizzes)
  - Last checked: 6 submissions
- Take-home essay due Friday at 5 p.m.
  - End of exam period
- All lab work must be submitted by **Wed**

Broader Issues

- Articles:
  - Tech education, Puzzles of Cyberspace, DARPA Urban challenge, Excel Bug, Google Maps Bugs, Digital Humanities (culture genome/art fraud), Sensor Networks, Social Networks, OLPC
- Questions
  - Most liked article? Least liked article?
  - Who found the articles overall least interesting?
  - Most interesting?

Nick, Will, Minh, Callie

Yates, Lida, Anh, Colin

Meng, Jean Paul, Ola

Broader Issues

- One Laptop Per Child
  - An experiment on bringing cheap but educational technology to poor children
- What challenges did OLPC face and how did that affect their design decisions?
- What are some unusual features of the laptop?
- What does this technology mean for better-off countries?
- Is this project worthwhile?
- What has changed (relevant to this project) since this article in 2007?

Discussion

Challenge	Design Decision
Lack of power	New, cheap battery; Consumes less power; Alternative power sources: solar power, pull cord
Software bloat	Rewrite code more compactly, efficiently
Environment	Dust proof, drop proof, light
Users: children	Simple user interfaces; tiny keyboard; lightweight; applications keep students interested
Cost	Linux, Python, open-sources tools; cheaper battery; no hard drive; no CD/DVD drive