CIS 210: Introduction to Computer Science

Instructor: Michal Young
Graduate Assistants:
  Daniel Ellsworth
  Mahshid Yar Mohammed
Fall 2011

Why come to class?

Slides will (mostly) be available after class
But ... Lecture is more than reading the slides, and I don’t do all the talking.

Observation from fall 2010: People who skip lecture do poorly on assignments and exams

Obtaining Course Info

Read the class web page:
  • http://www.cs.uoregon.edu/classes/11F/cis210
    All basic class information is there

Follow the class blog:
  • http://uo-cis210.blogspot.com/
    Announcements will appear there first

Keep current! It is your responsibility. Suggestion: Subscribe to email notifications for the blog

Textbook(s)

Python for Software Design: How to think like a computer scientist
  by Allen B. Downey
  http://greenteapress.com/thinkpython/

Building Java Programs
  by Stuart Reges and Marty Stepp

Read assigned chapters before lecture come to class with questions Experiment! try examples from the book, and try variations
Two books? Two languages?

Yes.

Python to Java

10 weeks to learn Python, and learn Java, and
learn more about programming languages

• Versus 10 weeks for Java alone in Fall 2010

Crazy? Maybe not.

Sometimes two perspectives are clearer than one.

Introduction to Computer Science

Programming is an important part of computer science

Important

It makes everything else possible.

But just a part

There is much more to computer science.

“CS may be more than programming, but it is not less than programming.”


Q: What is Programming?

A: Solving problems

The computer is a tool.

• A carpenter must know how to use a hammer, but knowing how to use a hammer doesn’t make you a carpenter.

A programming language is also a tool.

• You will learn Python and Java. You will also learn to program. Not the same thing!

Programming is mostly about logical analysis and problem solving.
Goals for CIS 210

Learn computer science concepts
Problem solving with computation

General programming skills
  • includes designing programs to be understood and modified by humans
  • includes testing, debugging

Expressing programs in the Java language
  • but the programming concepts apply to other languages

Getting Help

Labs are excellent opportunities to get help
Instructor and GTFs also hold office hours. We want to see you there!
  • But if you skip the lecture, don’t ask me to repeat it in office hours.
    I won’t do that.

Email is also useful.
cis210-help@cs.uoregon.edu
We try to answer quickly, usually within 24 hours.

Don’t wait to the last minute
If the assignment is due in two days, and you are completely lost, I probably can’t help you much.

Labs

Lab attendance is mandatory
It counts toward your grade!

Labs cover material not in lecture
It’s your best chance to understand how to solve the homework problems

Pair Programming

Pair programming is allowed on some assignments
  • Pair programming is done with two people working together at one computer: One driver and one observer. Trade roles often.
    – Pair programming does not mean letting someone else do your assignment. You must understand every bit of it.
  • Keep a log of meetings.
  • Each partner turns in program listing both authors

Always document contributions of all authors
Other Collaboration

**DO** discuss the problems
  Discuss general approaches to solving them. Learn from each other.
  If you rely on ideas from someone else, or somewhere else (e.g., a web site), document it in your solution.

**DON’T** copy or plagiarize
  Write every line of program code yourself.
  We *can* tell. We *do* enforce UO academic honesty policy.

[Reges & Stepp, 1.1 > basic computing concepts]
[Downey, 1.2 > way of the program; variables, expressions, and statements]

**COMPUTATION AND COMPUTERS**

Computers used to be bigger and more expensive ... this is *part* of the Bell Relay Computer

These cables are the program

Programming was tedious
Two big ideas ...

1. Instead of connecting cables, let’s store the program in memory!
   (stored program machine)

2. Let’s write a program that simulates a more convenient computer!
   (Universal Turing Machine)
Universal Computing Devices

A computer executes instructions

A *program* is a collection of instructions

A program can simulate another (more convenient) computer.

Almost *any* computer can simulate any other (with the right programs).

Virtual (imaginary) Computers

Now it’s simulating a (virtual) Java computer, which executes Java programs, or a Python computer, or …

The computer executes programs expressed in binary machine language.
A Python Program

```
x = 5
y = 2
x = x + y
print x
```

A Python Program

```
x = 5
y = 2
x = x + y
print x
```

It's really binary, but usually we don't care.
A Python Program

```
x = 5
y = 2
x = x + y
print x
```

(Let’s do it ... )

Dumb but fast

Basic computer operations are really this simple
(but really, really fast)

To do anything interesting, we need repetition

Repetition in Python

```
x = 1
while x < 7 :
  x = 2 * x
print x
```
\begin{align*}
  &x = 1 \\
  \text{while } x < 7 : \\
  &\quad x = 2 \times x \\
  &\text{print } x
\end{align*}
\[ x = 1 \]
while \( x < 7 \):
\[ x = 2 \times x \]

print \( x \)

\[ x = 1 \]
while \( x < 7 \):
\[ x = 2 \times x \]

print \( x \)

\[ x = 1 \]
while \( x < 7 \):
\[ x = 2 \times x \]

print \( x \)

\[ x = 1 \]
while \( x < 7 \):
\[ x = 2 \times x \]

print \( x \)

Now what?
### One by one ...
Repetition on collections of data

```python
x = 1
while x < 7:
    x = 2 * x
    print(x)
```

### Combined (nesting)

```python
for x in ["yabba", "dabba"]:
    for y in ["skiddly", "doo"]:  
        print(x, y)
```

### Nested loop execution

```python
for x in ["yabba", "dabba"]:  
    for y in ["skiddly", "doo"]:  
        print(x, y)
```

**Most programming languages have similar constructs for repetition**

**What do you expect it to print?**
**Let's try.**

**OK, it's not great poetry**
Week 1 project

Goal:
- Learn to edit and run Python programs
- Learn to use system for turning in work
- Begin learning Python programming language

A “warm-up” project
- They will get harder, and more interesting

Go to lab to get started

What now?

Read Downey, *Think Python*
- *Chapter 1*, The way of the program
- *Chapter 2*, Variables, expressions, and types

Go to lab

Start programming
- Install Python (version 2.7 recommended)
- Play
- Play some more. Discuss.
- Try the assignment (due Friday)