CIS 210: Introduction to Computer Science

Instructor: Michal Young
Graduate Assistants: Daniel Ellsworth, Kanika Sood
Undergraduate assistants: Zak Remer, Noah Cooper, Jason Jensen, Kevin Beick
Fall 2012

Obtaining Course Info

Read the class web page:
• [http://www.cs.uoregon.edu/classes/12F/cis210](http://www.cs.uoregon.edu/classes/12F/cis210)
  All basic class information is there

Follow the class blog:
  Announcements will appear there first

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

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: People who skip lecture do poorly on assignments and exams

Textbook

*Introduction to Computing Using Python: An Application Development Focus*
by Ljubomir Perkovic

Read assigned chapters *before* lecture
come to class with questions
Experiment!
try examples from the book, and try variations
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. 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 Python language
   • but the programming concepts apply to other languages

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
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.

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.

First Assignment

On the course web site now
2 parts
  1 – “paper and pencil” (actually text file), individual
  2 – programming, pair exercise
Due Friday 5pm. Submit files on Blackboard.
Draw with turtle graphics:

```python
import turtle
s = turtle.Screen()
t = turtle.Turtle()
t.forward(d),
left(a), right(a), setheading(a)
goto(x, y), penup(), pendown()
```

Design questions for the rough drawing at right:
- How will you break the problem down into parts?
- How will each part work?
- How will you organize and simplify your program?

First step:
Test turtle graphics with a trivial program based on examples and documentation.

```python
import turtle
s = turtle.Screen()
t = turtle.Turtle()
t.forward(60)
t.left(90)
t.forward(40)
s.mainloop()
```

Building blocks we can use:
```
t.left(degrees), t.right(degrees)
```

```
for step in range(n):
    do something
```

```
def some_action()
    steps to do the action
```

Programming and CS

*Why the CS major starts with programming*

Learning to program is just part of CS
But programmability (universality) is the essence

You must understand programming to understand CS

Python is (just) a reasonable example to start with