Welcome to CS 111!

Computer science is not so much the science of computers as it is the science of solving problems using computers.

Eric Roberts

• This course covers:
  • the process of developing algorithms to solve problems
  • the process of developing computer programs to express those algorithms
  • other topics from computer science and its applications
Computer Science and Programming

- There are many different fields within CS, including:
  - software systems
  - computer architecture
  - networking
  - programming languages, compilers, etc.
  - theory
  - AI

- Experts in many of these fields don’t do much programming!

- However, learning to program will help you to develop ways of thinking and solving problems used in all fields of CS.

A Breadth-Based Introduction

- Five major units:
  - weeks 0-4: computational problem solving and “functional” programming
  - weeks 4-6: a look "under the hood" (digital logic, circuits, etc.)
  - weeks 7-8: imperative programming
  - weeks 9-11: object-oriented programming
  - weeks 11-13: topics from CS theory

- In addition, short articles on other CS-related topics.

- Main goals:
  - to develop your computational problem-solving skills
    - including, but not limited to, coding skills
  - to give you a sense of the richness of computer science
**A Rigorous Introduction**

- Intended for:
  - CS, math, and physical science concentrators
  - others who want a rigorous introduction
  - no programming background required, but can benefit people with prior background

- Allow for 10-15 hours of work per week
  - start work early!

- Less rigorous alternatives include:
  - CS 101: overview of CS
  - CS 103: the Internet
  - CS 105: databases and data mining
  - CS 108: programming with a focus on web apps
  - for more info: [http://www.bu.edu/cs/courses/divisional-study-courses](http://www.bu.edu/cs/courses/divisional-study-courses)

**Course Materials**

- Free online textbook: *CS for All*
  - by Christine Alvarado, Zachary Dodds, Geoff Kuenning, and Ran Libeskind-Hadas
  - [www.cs.hmc.edu/csforall/index.html](http://www.cs.hmc.edu/csforall/index.html)

  - bring to every lecture
  - register it on Blackboard
  - the standard BU clicker

- Another good online textbook: *How to Think Like a Computer Scientist*
  - by Bradley Miller and David Ranum
  - [interactivepython.org/courselib/static/thinkcspy/toc.html](http://interactivepython.org/courselib/static/thinkcspy/toc.html)
Traditional Lecture Classes

• The instructor summarizes what you need to know.
• Readings are assigned, but may not actually be done!
• Dates back to before the printing press.
• Many technological developments since then!

Limitations of the Traditional Approach

• You get little or no immediate feedback.
• Research shows that little is learned from passive listening.
  • need to actively engage with the material
• Homework provides active engagement, but...
• After college, you'll need to continue learning on your own.
  • should get good now at learning from a textbook
Lectures in this Class

• Based on an approach called *peer instruction*.  
  • developed by Eric Mazur at Harvard

• Basic process:
  1. Question posed (possibly after a short intro)
  2. Solo vote (no discussion yet)
  3. Small-group discussions (in teams of 3)  
     • explain your thinking to each other 
     • come to a consensus
  4. Group vote  
     • each person in the group should enter the same answer
  5. Class-wide discussion  
     • why is the correct answer correct?  
     • why are the wrong answers wrong?  
     • possibly some clarification/explanation by me

Benefits of Peer Instruction

• It promotes active engagement.

• You get immediate feedback about your understanding.

• I get immediate feedback about your understanding!

• It promotes increased learning.  
  • explaining concepts to others benefits you!

Crouch, C., Mazur, E.  
*Peer Instruction: Ten years of experience and results.*
Drawback of Peer Instruction

• Less time to catch up on sleep.

Some people talk in their sleep. Lecturers talk while other people sleep.
– Albert Camus

Preparing for Lecture

• Short reading from the textbook (or other resource)

• Short online reading quiz

  • complete by 9 a.m. of the day of lecture (unless noted otherwise)
  • don't need perfect correctness
  • your answers should show that you've done the reading
  • no late submissions accepted

• Reading the textbook is essential.

  • prepares you for the lecture questions and discussions
  • we won't cover everything in lecture
Course Website
www.cs.bu.edu/courses/cs111

• not the same as the Blackboard site for the course

• use the Lectures link to access Blackboard, which will be used for:
  • the reading assignments
  • the reading quizzes posted at least 36 hours before lecture
  • the lecture notes – posted after lecture

Labs

• Attendance is required

• Will help you prepare for and get started on the assignments

• Will also reinforce essential skills

• ASAP: Complete Lab 0 (on the course website)
  • register your clicker
  • setup a CS account before your first lab session
  • some other tasks to prepare you for the semester
Assignments

• Weekly problem sets
  • most have two parts:
    • part I due by 10 p.m. on Thursday
    • part II due by 10 p.m. on Sunday
  • Final project (worth 1.5 times an ordinary assignment)
  • Can work on some (but not all!) problems with a partner
    • see the syllabus for details
  • Can submit up to 24 hours late with a 10% penalty.
  • No submissions accepted after 24 hours.

Collaboration

• Two types of homework problems:
  • individual-only: must complete on your own
  • pair-optional: can complete alone or with one other student

• For both types of problems:
  • may discuss the main ideas with others
  • may not view another student/pair’s work
  • may not show your work to another student/pair
  • don’t consult solutions from past semesters
  • don’t consult solutions in books or online
Collaboration (cont.)

- For pair-optional problems:
  - work with at most one partner per assignment
  - work **together at the same computer**
    - screen should be visible to both of you
    - one person types, while the other plans/critiques
    - switch roles periodically
  - may **not** split up the work and complete it separately
  - both submit the same solution and clearly indicate that you worked as a pair.

- After finishing the problems:
  - each person should have contributed equally
  - both could complete the problems on their own

Grading

1. Weekly problem sets + final project (40%)

2. Exams
   - two midterms (25%) – **on Wed night from 6-7**; no makeups!
   - final exam (25%)
     - can replace lowest assignment and lowest midterm

3. Preparation and participation
   - pre-lecture reading quizzes (5%)
   - attendance/participation (5%) – full credit if you:
     - make 85% of the votes over the entire semester
     - attend 85% of the labs
   - using a clicker for someone else is **not** allowed

- To pass the course, you must earn a passing grade on each of these three components.
Course Staff

- Instructors: Dave Sullivan (A1 and C1 lectures)
  Aaron Stevens (B1 lecture)

- Teaching Fellows/Assistants (TF/TAs):
  Will Blair  Cody Doucette
  Alex Breen  Sahil Tikale

- Undergrad Course Assistants (CAs):
  Linshan Jiang '17  Tania Papandrea '16
  Alison Kendler '16  Molly Shopper '17
  Sandra Lefdal '17  Anbita Siregar '16
  Sophie Lehar '16  Chris Sullivan '15
  Zach Lister '16  Nick Sweetser '17
  Steve Man '15  Enze Yan '16
  Vidhu Nath '15

- Office hours: http://www.cs.bu.edu/courses/cs111
- For questions: cs111-staff@cs.bu.edu or post on Piazza

Algorithms

- In order to solve a problem using a computer, you need to come up with one or more algorithms.

- An algorithm is a step-by-step description of how to accomplish a task.

- An algorithm must be:
  - precise: specified in a clear and unambiguous way
  - effective: capable of being carried out
Example of Defining an Algorithm

- diapering a baby

Algorithm for Finding My Office

1. Go to the entrance to the MCS (math/CS) building at 111 Cummington Mall – behind Warren Towers.

   \[\text{Do not enter this building!}\]

2. Turn around and cross the street to the doors across from MCS.

3. Enter those doors and take an immediate right.

   \((\text{continued on next slide})\)
Algorithm for Finding My Office (cont.)

4. As you turn right, you should see the door below. Open it and go up the stairs to the second floor.

5. As you leave the stairs, turn right and then go left into a small hallway. My office is the first door on the left (PSY 228D).

Programming

• Programming involves expressing an algorithm in a form that a computer can interpret.

• We will primarily use the Python programming language.
  • one of many possible languages
  • widely used
  • relatively simple to learn

• The key concepts of the course transcend this language.

• You can use any version of Python 3
  • not Python 2
  • see Lab 0 for details
Picobot

• Python is a relatively simple language, but...

• To allow for interesting problems right away, we're going to start with something even simpler!

• Picobot!
  • a special-purpose language
  • we'll learn it all in the next week
  • controls a robot based on the Roomba vacuum cleaner robot

Picobot (cont.)

• Goal: to have the robot "vacuum" a small room.
  • there may be obstacles!
  • it can't remember where it's been
  • it can only sense its immediate surroundings

• We'll write rules that tell the robot what to do.
  • see the first reading assignment for details!
Your Current Tasks

1. Complete Lab 0 ASAP.
   • on the course website (www.cs.bu.edu/courses/cs111)

2. Complete first reading assignment and quiz by 9 a.m. on Friday.
   • on Blackboard (can use the Lectures link to access)