Comp 151

Introduction to Computer Science
Introductory Lecture
First

- Lets go over the syllabus
- Then find out who you all are.
- Book for class.
What is computer Science

- Architectural discipline
  - Much like engineering
  - Includes both design and implementation
    - Need to know how to build from iron before can design a good bridge.
    - Need to know how to program before you can build great software.
- Need to think logically
  - Understanding how computers 'think' is hard for some people.
- Over the course of the semester
  - We will learn a lot of programming and a bit of the design
What else does computer science include?

- Later in your computer science career you will also like need to work with
  - Database design
  - Logic
  - Technical writing
  - User interface design
  - Program and system security
  - Intra and inter process (program) communication.
  - System administration
Computer Science is hard

- National statistics
  - Drop rates
  - New approach to course changing that
Why is computer science hard part I

• There is a right answer
  • If the program does what its supposed to without errors it is right
  • Otherwise it is wrong
  • No scope for “well argued but wrong” or “worked hard”
    ✓ Partial credit for partial completion is available.

• Mathematical rigor
  • Some people hate math – some computer science requires math
Why computer science is hard II

- **Strict grammar**
  - No such thing as close enough
  - Computer understands a few ways to be told what to do
    - Telling it any other way won't work.
- **Out of class programming projects**
  - Computer science closest thing to engineering we have at Bridgewater State
  - We build things
- **Actions have consequences**
  - Programming errors → money loss, car problems etc
Experiences I've Seen

- Three types of people as far as computer science goes
  - Easy and pie
    - It just makes sense
    - Fairly small group
  - Regular people
    - Who have to work at it
    - Eventually understand it
    - Your instructor was in this group
  - I hate this it doesn't make any sense
    - A few people will just not be able to handle the different way of thinking (not many in this group)
Why should I study Computer Science then

- So why study CS if you are not in the lucky few to which it comes easily?
  - like many disciplines, once you adjust to the CS way of thinking, very satisfying
    - similar to engineering disciplines
    - hard but satisfaction of building things.
  - abstract mathematical beauty
  - good jobs when you graduate
  - Python makes that first step into programming much easier.
Hollywood vs. Reality

- Hollywood give a distorted impression of reality
  - Presentation of computer science is no exception
  - Computer science is about building programs and software systems
    - Not about rebuilding hardware
      - That's a technical degree
    - Or designing hardware
      - That's electrical engineering
  - Most software you've used is graphical/window based
    - But much (most?) software is not...
What is a computer?

- Today what sorts of things do computer scientists program?
What is a computer?

Today what sorts of things do computer scientists program?

- Servers/desktops/laptops
- Tablets/phones
- Cars
- Coffee makers
- Cash registers
- Set top boxes (tivo and friends)
- Thermostats
- Airplanes
- Pretty much everything electronic

- Lots more non-windowed programming
Programming Languages

• A *program* is a set of instructions a computer follows in order to perform a task.
• A programming language is a special language used to write computer programs.
• A computer program is a set of instructions that enable the computer to solve a problem or perform a task.
• Collectively, these instructions form an *algorithm*
Programming Languages

• An *algorithm* is a set of well defined steps to completing a task.
• The steps in an algorithm are performed sequentially.
• A computer needs the algorithm to be written in *machine language*.
• Machine language is written using *binary numbers*.
• The binary numbering system (base 2) only has two digits (0 and 1).
Programming Languages

• The binary numbers are encoded as a machine language.
• Each CPU has its own machine language.
  - Motorola 68000 series processors
  - Intel x86 series processors
  - DEC Alpha processors, etc.
• Example of a machine language instruction:
  1011010000000101
Programming Languages

- In the distant past, programmers wrote programs in machine language.
- Programmers developed higher level programming languages to make things easier.
- The first of these was assembler.
- Assembler made things easier but was also processor dependent.
Programming Languages

• High level programming languages followed that were not processor dependent.
• Some common programming languages:
  • Python
  • C#
  • php
  • C
  • C++
  • Java
Getting there from here

- So how do we get to those machine instructions from the high level language?

  - **Compiler:**
    - A program that translates the entire high level program into machine code and saves the file full of machine code
    - Creates an executable program
    - Compiled languages:
      - C/C++ and sort of java and C#

  - **Interpreter:**
    - A program that reads the high level program line by line and translates it into machine code as the program is run
A note on computers and programming

- The computer does not hate you
  - Despite what you might read about or see in movies
    - AI is still either primitive or very expensive
    - There is no AI running on these computers
  - The computer does exactly what you tell it to do
    - Unfortunately not what you meant to tell it to do
- Programming environment has grammar
  - Python more flexible than some languages.
  - Still non-grammatical statements are rejected
  - Syntax error – dreaded message
- But making errors, learning what they are, then fixing them is fine way to learn
Now let's look at a basic Python program

- Walk through Python startup
- Open editor
  - Type in
  - `print("hello there")`
  - Save and then run the program
  - Very simple
  - `print` in Python is a special function for printing to command line (Python 2.7/3.x distinction)
    - *Command line vs windowed programming discussion*
    - *Characters within “” are text or strings (more on that later.) print will print them as written*
    - *() are like function parentheses in math*
      - *Print will print whatever is in the parentheses*
Now let's make a mistake

- Now in your file write
  - print("hello there")
  -
- Without the final parentheses.
- Run this and what happens?
Now let's make a mistake

- Now in your file write
- `print("hello there")`

- Without the final parentheses.
- Run this and what happens?
  - File "pythontest.py", line 2
  - `SyntaxError: invalid syntax`

- Your program can't find the end of the function call and gets confused.
- Put the paren back and run the program again
  - Works again
Arithmetic in python

- Type basic arithmetic in python as normal
- Put this into your program
- \[(8+4)/3*5\]
- Run your program
  - No errors, but no indication of the new line
    - Value was computed, but provided to computer
    - You didn't tell the computer to do anything with that value so it didn't.
  - Lets keep track of the value
  - Rewrite your new line to look like this
  - answer = \( (8+4)/3*5 \)
  - Answer now holds the value of the arithmetic
answer = (8+4)/3*5

- Lets have closer look as what's going on here.
  - Whenever we have an '=' symbol
  - Python treats it as assignment
  - Some value on the right hand side is (possibly calculated and then) placed into the 'variable' on the left hand side
    - Not a statement about equality
  - Variables are named storage places for values
  - The first time the variable is used it has to be on the left hand side of the equals sign.
    - Whatever value is on the right hand side will be placed into the variable
  - If you put the variable on the left hand side of an equals sign the second time, the new value from the right hand side will be placed in the variable
Variables and program order part II

- Simple programs like the one we've worked on so far
- Start from the top line
  - Then execute each line after that in order till the last line in the file is executed
  - We'll see exceptions to this later.
- Once a variable has been given a value in a line
  - It can be used in succeeding lines
  - Lets add one more line our short program
  - Put this line last
  - print(answer)
- Now run your program again
- You see the value of answer printed out too
- You can use answer anywhere in your program now to use the value that answer holds
Assignment

- Lets stop a bit early on the first day.
- Read to chapter 1 in your book
- Try a couple of more programs yourself.
  - See you next time.