CS3157: Advanced Programming

Lecture #1
Jan 17
Shlomo Hershkop
shlomo@cs.columbia.edu
Overview

- Welcome to Advanced Programming
- Today:
  - Basic overview of the course and objectives
  - Setup basics
  - Basic setup
- Goal:
  - Things are much easier if you know why you are here, and what we are trying to accomplish.
  - I will not stand here and lecture (although there will be some of that). This is going to be a very interactive course.
  - We will learn about programming ideas while trying to have fun.
  - Will end up with a lot of useful “marketable” skills for use in anything to do with computers
Background

- Instructor: Professor Shlomo Hershkop
  - (shlomo@cs.columbia.edu)
  - aim: Prof Hershkop

- My Background
  - Research Areas
  - Current work

- Meeting times
  - Mondays: Lectures
  - Wednesday: Lab

- Will take a poll to see which slots works best
What?

- **CS3157:**
  - Third course in the CS intro sequence

- **Prerequisites:**
  - Intermediate knowledge in Programming
  - **Object Oriented Programming:**
    - What, why, how, and when.
  - **Program Designs.**
    - Not enough to know how to write the program, need to know how to do it correctly.
    - Become familiar with tools of the trade
So what are we going to be doing?

- Cover some practical languages:
  - Perl
  - C
  - C++

- Cover practical skills:
  - Debugging
  - Environmental setup
  - CGI/Web based programming
  - Regular expressions
  - Web scripting
  - Ruby
  - Plus more!
Hopefully you are familiar with at least one programming language

- Java?

- Programming is not Java!
- Computer science != programming
Design of lab part of the course:

- Give you a feel of the real world
  - Problem description blurry
    - Not going to get a simple recipe
    - Goal oriented tasks
  - Many choices for you as a programmer
  - Being lazy is a plus:
    - Will learn best tools to stay lazy

Please ask for help if you need it

- We are here to help
Most popular question:

- How much work is involved/expected in this course?
Resources

- Class website:
  - cs.columbia.edu/~sh553/teaching/3157-s07/
  - Courseworks.columbia.edu
  - Check it regularly (at least twice a week)

- CS account
  - need an account for the class
  - mice.cs.columbia.edu
    - apply for swipe access to clic lab (486 mudd)

- TA’s:
  - Robert Lin
  - Aniruddha Dutta
  - Brody Berg
Requirements

- Interest to learn about Computer Science
- Learn to
  - use cool tools
  - make your own tools
  - Why things work in a certain way
- Feel free to explore
  - no reason to limit yourself to the basic requirements….if you want try to add something cool to a lab/homework or make use of outside work please talk to me
- Make sure you understand the material..
Textbook

- Textbook acquired online or purchased at the Columbia Bookstore.
  - Else: borrow, threaten, or ‘acquire’ a book
  - Motto: No questions asked!
  - Don’t ask don’t tell

- Perl
  - Programming Perl
    - O'Reilly

- C/C++
  - Deitel and Deitel

- Others
  - Google or Safari online (Columbia has a library subscription)
Reading

- I will be posting reading on the website and in class notes

- Please try to keep up with the reading
  - I will try to make up examples for class, but there are random stuff which the book covers which is good to see in print
    - Feel free to ask questions from anything you see in the book
    - Or on the internet (although might need large grains of salt sometimes)
Course Structure

- Reminder: this is a 4 credit course
- ~10 Labs – 120 points
  - Out Wednesday, Due by Sunday
  - Will get to use class time to do work
  - Average about 2-4 hours
- 3 Homeworks – 60 points
  - Will have about 2 weeks per homework
  - Overlap with labs, can reuse your own work
- Midterm (20 points) and Final (60 points)
Word of caution:

- Homework/Lab is very important:
  - Firm believer in hands on learning
  - Start early
  - Come to office hours, and ask questions
    - We are here for YOU!

- For your own benefit
  - If you feel differently either:
    - Mumble under your breath
    - Talk to me
Class participation and Attendance

- Attendance and participation is expected
  - Very interactive lectures & Labs
  - Part of your learning if by doing in class examples.
  - Class anonymous feedback system

- If you have to miss class, I expect you to catch up.
  - There will be class notes posted to the website usually within 24 hours.
  - There will be many examples in class only, so make sure to get someone’s notes.
Homework & Projects

- Homework/Labs:
  - Online submission through your cs account
  - Must be able to run on cs system (this is important)
  - Else grade hiccups and headaches

- Late policy:
  - Homework’s are due as stated
  - If you need more time, I will grant extensions on a case by case basis (for good reason) but you need to contact me before the deadline
  - Don’t believe in posting solutions, you are responsible for following up on correct answers/procedures
  - Will try to get back grades as soon as we can
Labs

- Generally will create a few programs with common theme...
- Will be working in the clic lab...
- Online submissions
- Will be around to answer questions hints

- Can NOT ask for code from other students
  - Can ask input/output
  - General ideas
  - Use your best judgment for anything else
  - Does not include team members
Cheating

Don’t
Cheating Policy

- Plagiarism and cheating:
  - I’m all against it. It is unacceptable.
  - Had to fail 3 students last semester
- You’re expected to do work by yourself
  - This is a learning experience.
  - You will only cheat yourself.
  - My job is to help you learn, not catch you cheating, but....
- Automated tools to catch plagiarizers
  - Moving stuff around, renaming, etc. doesn’t help
- Results: instant zero on assignment, referral to academic committee
  - Columbia takes dishonesty very seriously
  - I’d much rather you come to me or the TAs for help and not resort to cheating...
Feedback System

- Last minute of class will be set aside for feedback:
  - Please bring some sort of scrap paper to class to provide feedback.
  - Feel free to leave it anonymous.
  - Content: Questions, comments, ideas, random thoughts.

- I will address any relevant comments at the beginning of each class

- Please feel free to show up to office hours or make an appointment at any time
Shopping List

- You need a CS account ASAP
  - https://www.cs.columbia.edu/~crf/accounts
  - Try to log into the account once you get approval
- Check out the class page
- Make textbook plans for both perl and c
- Apply for lab access through mice.cs.columbia.edu
  - Separate from your cs account (although might need one to apply)
Any Questions ?
Short Survey 1

- Let me pass out a brief survey and get some feedback from all of you now
One of the points of computer science is to teach you how to think, learn, and analyze computational related information.

Each course is a tool which you will collect for later use.

Lots of tools in this course, since we will be covering many different topics and subjects.
A *programming language* specifies the words and symbols that we can use to write a program.

A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid *program statements*.

We will cover different programming languages:

- The only part to learn a new language involves recognizing the RULES which form the instructions.
- In addition the theory behind design choices.
- But anything can be turned into a programming languages – even colors or water pipes 😊
Let's get started

- Let start to learn about perl
  - Historical perspective
  - Practical example
  - Misc resources and advice

- You have a choice if you want to work out of your clic account or local machine

- work must be submitted on cs
  - log into a clic machine name.clic.cs.columbia.edu
  - compute.cs.columbia.edu
  - cluster.cs.columbia.edu
quick question

Anyone know how to check for what operating system is running on a non-windows machine?

what about windows machine?
- ver
- uname -a
Perl

- Perl
  - History
  - Version 5.8+
  - Rumblings of version 6
- What is it?
  - Scripting language
  - Aims to be a USEFUL language
  - Base + tons of libraries
  - Both a compiler and byte code executable
- Where to get it?
  - cpan.org
  - www.activestate.com/Products/ActivePerl/
- Why am I starting with perl ??
Perl

- *Perl* was originally designed as a logging tool, released by Larry Wall in 1987.
- Open source and cross platform. Current version 5.8.8
- Derived mostly from c, awk, unix shell, and anything lying around which was useful
- Referred to as “duct-tape” of the internet
  - Will quickly learn why 😊
Difference: Java and PERL?

- **Java**
  - High Level Language
  - Source code is compiled to byte code
  - Byte code = java execution instructions
  - Byte code executed by java
  - Most functionality built into libraries, very strong graphic capabilities

- **Perl**
  - Scripting language
    - Very very non rigid structure (i.e. what ever you want)
  - code can be interpreted line by line in real time
    - i.e. compiles and executes each time invoked
  - A lot of functionality built into base language
    - String handling second to none
Remember

- Perl evolved over time
- Important to check local system for version
  - Either manually
  - Or within the program
  - Might require a minimum version to work

- Will get a chance to mess with this in labs
Environmental Hazards

- Depending on the local system will behave differently:
  - each operating system has different end line denotations...be aware of this
- Cunix
  - Anyone know what operating system they run ?
- CS has both of these main os’s :
  - Linux
  - SunOS
- Windows
  - Active perl
  - Cygwin
    - Perl
- VNC
  - Allow you to remote connect to CS if you have an account
  - everything already setup for you (basics)
  - session live across log ins...
Programming perl

- Ok so let's get started already

- Let's go over some conventions and rules before coding
Pre programming

- Your perl program will essentially be a text file
- A perl interpreter will run through your text file and execute code on your behalf (convenient lie)
- Can either invoke the perl interpreter explicitly or implicitly
Pre-programming II

- Naming a perl script?
  - Something.pl

- Many times it's necessary to check which version of perl is being used:
  - perl –v

- On unix/linux/sun can see which perl compiler is called by default:
  - which perl

- Need if you want to tell the system which perl you really want to use
Perl modules

- Another type of perl program is called a module
- Reuseable code
- Name something.pm

- Import it by saying
  - use *amodule*
Compiler/interpreter

- Perl is interpreted
- The script needs to tell the system where the interpreter is sitting
- Accomplished by special command on the first line of your program:

```
#!/usr/bin/perl
or
#!/c:\perl\bin
```
hint

- which perl

- locate perl
Comments start the line with a hash, will continue to end of line mark
- will talk later about multiple line comments
- system is called POD

In addition, on unix/linux need to tell system to execute your perl script

Can call perl directly
  perl test.pl

Or can turn it into executable
  chmod +x test.pl

./test.pl
Built in functions

- Can call tons of built in functions to do stuff in perl
- Can define your own (later today)
- One is the print command

print "something\n";
test.pl

#!/usr/bin/perl

#your first perl program

print "hello everyone\n";
Technical details

- By default the start of your code is the equivalent of “main”
- Will run each line in turn, execute and then next line
- Will end when reach end of code
Environment

- I want you to learn to use eclipse
  - [www.eclipse.org](http://www.eclipse.org)
- Need EPIC plug in for perl
  - how to get it

- try to get the previous example to run over the weekend
- do the reading