CS3157: Advanced Programming

Lecture #1

Sept 6

Shlomo Hershkop
shlomo@cs.columbia.edu

1

Welcome!

- Welcome to Advanced Programming
- Fall 2006
  - new course
  - redesigned
  - feedback
  - focused on internal implementation

2

Overview

- Today:
  - Basic overview of the course and objectives
  - Setup basics
  - Basic setup
- Goal:
  - Thing are much easier if everyone knows why they are here, and what we are trying to accomplish.
  - I will not stand here an 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.

3

What?

- CS3157: Third course for CS majors.
- 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.
    - Need to learn tool of the trade

4

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
  - Plus more!

5

Point

- Hopefully you are familiar with at least one programming language
  - Java ?
- Programming is not Java !
- Computer science != programming

- Give you a feel of the real world:
  1. Problem description blurry
  2. Many choices for programmer
  3. Will learn best tools to stay lazy ☺
  4. Please ask for help if you need it

6

Background

- Instructor: Professor Shlomo Hershkop
  - (shlomo@cs.columbia.edu)
  - aim: Prof Hershkop
- My Background
  - Research Areas
  - Current work
- Meeting times
  - Mondays lectures
  - Wednesday: short Lecture plus lab slot

7

Resources

- Class website:
  - cs.columbia.edu/~sh553/teaching/3157-f06/
  - Check it regularly (at least twice a week).
    - See announcement sections for update info.
- CS account
  - need an account for the class
  - mice.cs.columbia.edu
    - apply for swipe access to clic lab (486 mudd)
- TA's:
  - William Mee
    - wjm2107@columbia.edu

8

Requirements

- Interest to learn about Computer Science
- Learn to use cool tools
- Learn to make your own tools
- 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
- Make sure you understand the material..

9

Textbook

- Textbook can be viewed online (Safari), acquired online or purchased at the Columbia Bookstore.
  - Else: borrow, threaten, or 'acquire' a book
  - Motto: No questions asked!
- Perl
  - Programming Perl
    O'Reilly
- C/C++
  - Deitel and Deitel

10

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

11

Course Structure

- 9 Labs – 120 points
  - Out Wednesday, Due by Sunday
- 3 Homeworks – 60 points
  - Will have about 2 weeks per homework
- Midterm (20 points)
- Final (60 points)
- 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!

12

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.

13

Homework & Projects

- Theory and programming :
  - Online submission
  - Must be able to run on cs system (this is important).
- 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).
  - Only will review answers once everyone submits

14

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

15

Cheating

- Don’t

16

Cheating Policy

- Plagiarism and cheating:
  - I'm all against it. It is unacceptable.
- You're expected to do homeworks 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
  - http://www.cs.berkeley.edu/~aiken/moss.html
  - 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…

17

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
- Summer is short, so provide feedback !
- Please feel free to show up to office hours or make an appointment at any time

18

Shopping List

- You need either a CS account ASAP
  - CS: 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
  - either make sure you have online access to safari
  - purchase text

19

- Any Questions ?

20

Short Survey 1

- Education Background
- Programming background ?
- Do you have access to a desktop/laptop?
- Any cool technologies you would like to see covered ?

21

Example:

- Task:
  - Create a program to run a web based game, which will be marketed to both desktop and phone users.
  - Any ideas on how to design the programming backend?
  - Ideas on how to measure requirements.
  - What else is important?

22

Last plug

- 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.

23

Programming Language

- 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 ☺

24

Lets 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 clic machine or compute.cs.columbia.edu or cluster.cs.columbia.edu

25

quick question

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

26

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 ??

27

Perl

- *Perl* was originally designed as a logging tool, released by Larry Wall in 1987.
- Open source and cross platform. Current version 5.8.7.
- 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 ☺

28

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

29

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

30

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…

31

Programming perl

- Ok so lets get started already
- Lets go over some conventions and rules before coding

32

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

33

Pre-programming II

- Naming a perl script ?
  - Something.pl
- Many times its 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

34

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
```

35

- 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

```
chmod +x test.pl
  ./test.pl
```

- The other way is to call perl directly

```
perl test.pl
```

36

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”;`

37

test.pl

```
#!/usr/bin/perl

#your first perl program

print “hello everyone\n”;
```

38

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

39

Environment

- I want you to learn to use eclipse
  - 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

40

Something NEW!

- Most languages you know already, variables need to be declared ahead of time, what type they will deal with
- By default perl, will try to figure out what you mean
- Which means as soon as you use a variable for the first time, perl will assign it a type
- So initialization and declaration/assignment happen at once
- Why might this be bad ??

41

Variables

- Variables
  - Data dependant
  - No space in name
  - names consist of letters, digits, underscores; up to 255 chars
  - CASE SENSITIVE
  - Should start with letter or underscore
  - Initialized variables have the value of **undef**
    - Can use it later to test if a variable has been used/assigned

42

Data types

- The basic data types are as follows, we will go through each in turn

- scalars ($)
- arrays (@)
- hashes (%)
- subroutine(&)
- typeglob(*)

43

Scalars

- This type of variable starts with a $
  - $first
  - $course
- Can hold: int, real, string
  - 234
  - -89
  - 36.34
  - “hello world”
- Context dependant
  - $name = “shlomo”; #perl sees this is a string
  - $n = 123; #perl sees this as a number

44

Arrays

- Starts with @
- Order list of scalars
- `@class3157 = (“shlomo”,”weijen”,”edward”);`
- The scaler type in the array can be anything a scalar can hold
  - So can mix numbers and strings etc
- To reference elements, use the variable name with a dollar in front and subscript

```
$class3157[0]; #is shlomo
```

- Since perl tries to be useful what do you think this should be:

```
$class3157[-1];
$class3157[14];
```

45

Next time

- setup cs account
- ask for swipe access
- think about book
  - do reading

46