

Programming for Chemical and Life Science Informatics

I573 - Week 1

Rajarshi Guha

13th January, 2009

Why Program?

- Much of science involves computers for data analysis
- In most cases, prepackaged software does the job
 - ▶ Especially true of non-computational disciplines
- Lots of programs available to do cheminformatics
 - ▶ Schrödinger
 - ▶ Accelrys
 - ▶ ICM
 - ▶ ChemAxon
 - ▶ OpenEye

Why Program?

- You can get by with using prepackaged software
- But in many cases, you have to develop a *workflow*
 - ▶ Certain repetitive tasks need to be performed
 - ▶ Certain tasks will require user input
- No sense in doing these types of things by hand

*The three chief virtues of a programmer are:
Laziness, Impatience and Hubris*

Larry Wall

- But prepackaged software may not do the things the way you want
- Prepackaged software may not do the things you want in the first place

Levels of Programming

- Programming may be as simple as a small shell script that runs one program after another - glue
- Using the scripting language of a software platform to perform a sequence of tasks (MOE, Schrödinger etc)
- Use a toolkit to build a new application that does not exist
- Write a programming library to perform domain specific tasks

Not Just Data Analysis

- Programming is just way to express *computational thinking*
- Computational thinking is nicely discussed by Jeannette Wing in *Comm. ACM*, **2006**, 49(3), 33–35
- It doesn't imply that you think like a computer - instead it's a way of thinking that lets you analyse and solve problems using computational tools
 - ▶ Being able to find specific cases and generalize and then go back to specific cases
 - ▶ Using suitable levels of abstraction
 - ▶ Choosing an appropriate representation
 - ▶ Identification and use of invariants
 - ▶ Recognizing the need for modularity or caching
 - ▶ Prevention and protection against worst case scenarios

Computational Thinking

- Applicable to every day life
 - ▶ *Backtracking* - trying to find your car keys
 - ▶ *Caching* - keeping a coffee mug in the office
 - ▶ *Performance modeling* - choosing which line in a supermarket
- A lot of these ideas were also described in *How to Solve It*, by George Polya
- A classic, oriented towards solving mathematical problems
- Applicable to any scientific area

Computational Thinking & Programming

- Everybody can code
- But not everybody will code well
- Coding well requires a mental “vision” of the problem in a computational framework
 - ▶ What does your code need?
 - ▶ How will you represent it?
 - ▶ What architecture is suitable?
 - ▶ What algorithm is suitable?
 - ▶ Should it be reusable?
 - ▶ What should it return?
- You don't think about tying your shoelaces - so why think about reading lines from a file? Think about the bigger things!
 - ▶ Know your toolkits and libraries - **don't** write an XML parser by hand!

Computational Thinking & Programming

- Everybody can code
- But not everybody will code well
- Coding well requires a mental “vision” of the problem in a computational framework
 - ▶ What does your code need?
 - ▶ How will you represent it?
 - ▶ What architecture is suitable?
 - ▶ What algorithm is suitable?
 - ▶ Should it be reusable?
 - ▶ What should it return?
- You don't think about tying your shoelaces - so why think about reading lines from a file? Think about the bigger things!
 - ▶ Know your toolkits and libraries - **don't** write an XML parser by hand!

Elegance

- Elegance might seem superfluous
- Just write the code to get the job done - right?
- In many cases an elegant solution doesn't just look good, it will
 - ▶ perform better
 - ▶ be maintainable
 - ▶ be extensible
- You can push it too far - shouldn't sacrifice performance and correctness for elegance!
- But brute force does have it's uses
 - ▶ Linear scans over a database versus indexed queries

Program For Who?

- There are many reasons why you'll want to write code
- But equally important is who you are writing for
 - ▶ Yourself
 - ▶ Computational scientist
 - ▶ Non-computational scientist
- The audience will guide many decisions
 - ▶ command line or GUI?
 - ▶ version control? documentation?
 - ▶ standalone application or library?
 - ▶ nature of I/O?

Computational Research

- Your research is computational, don't end up as a button pusher
- Knowing how to glue tools together is fine - but don't end up as technician
- Know your tools, algorithms, architectures
 - ▶ Rough analogy to the Sapir Whorf Hypothesis
 - ▶ If you know how to extract small molecules from a PDB file, you can start asking questions about the protein **and** ligands
 - ▶ If you know about XML libraries you can generate XML output and then use XSLT to generate output in HTML, plain text, PDF etc
- Domain knowledge is vital - differentiates you from any other programmer
- Aim for elegance

Goals of the Class

- The class will have a practical focus - how do we do certain tasks
- We will go over common toolkits for cheminformatics and bioinformatics
 - ▶ Requirements and capabilities
 - ▶ Discuss the underlying models encoded by the toolkit
 - ▶ What does it mean when programming?
 - ▶ Examples
- We'll also look at some cases of general purpose software that are especially useful for cheminformatics and bioinformatics
- We'll also cover theoretical topics that are relevant when programming applications

Goals of the Class

- We'll also cover some more general topics relevant to programming
 - ▶ Debugging, version control, scientific software design
- Part of the class will also look at web-based programming and web technologies that are relevant for cheminformatics and bioinformatics
 - ▶ Web services, XML technologies
 - ▶ Blogs, wiki's - not programming *per se*, but does offer interesting programming opportunities
- We'll also consider some of the technologies that allow us to easily integrate cheminformatics into generic applications such as Google Spreadsheets

What the Class Won't Cover

- Learning any specific language
- We will not cover scripting languages specific to a prepackaged solution
 - ▶ So, we won't look at MOE's SVL etc
- We won't go into (too much) mathematical detail of the algorithms
- No in depth tutorials

What Should You Get Out of It?

- Familiarity with one or more toolkits
- An understanding of what goes on under the hood for some common problems in cheminformatics and bioinformatics
- The ability to put together an application that does something novel
 - ▶ Gluing a series of program to perform a computational study
 - ▶ Write an application to solve a cheminformatics or bioinformatics problem
 - ▶ Implement some form visualization for a chemical or biological dataset
 - ▶ Extend an open source toolkit with new functionality
- *Become fluent in computational thinking*

Requirements

- You should be comfortable in one language
- I'm going to be running the class in Java and Python
 - ▶ The toolkits we'll use are accessible from C, C++, Java and Python
 - ▶ The examples should be understandable if you know any language
- Should be familiar with concepts in cheminformatics and/or bioinformatics
- It'll be useful if you have some familiarity with basic statistics

Time & Place

- Informatics 105
- Tuesdays and Thursdays
- 9:30 am – 10:45 am
- Office hours by appointment, rguha@indiana.edu

- There's no specific book for this class
- I have listed a few books that are generally useful on the course web page
 - ▶ *Introduction to Scientific Computation and Programming* by Daniel Kaplan
 - ▶ *Writing Scientific Software: A Guide to Good Style* by Suley Oliviera and David E. Stewart
- When we cover specific programming topics, I'll provide references
- Blogs are a useful resource - see what people are doing, tips and tricks, pitfalls, references

- Logins will be provided on *cheminfo.informatics.indiana.edu*
- The server has the following toolkits and software installed
 - ▶ OEChem
 - ▶ BCI
 - ▶ BioPython
 - ▶ R
- You're free to work on your own machine, but you won't be able to access the commercial tools (OEChem and BCI)

Evaluation

In-class participation	10%	
Homework	30%	Small exercises based on class material
Project	50%	A moderately large application, that will address some problem in cheminformatics or bioinformatics
Presentation	10%	A short (10 minute) presentation on the project

Assignments

- Usually general programming problems
- Some of them will make use of toolkits
- Some will require you to develop code from scratch
- Work in any language you want - but I can only help with C, C++, Java, Python, R

Projects

Goals

- Gain experience in writing a moderately large application
- Learn a cheminformatics or bioinformatics toolkit
- Learn (or apply) chemistry and/or biology knowledge
- Apply good software design principles

Requirements

- Projects are individual
- I'll provide a list of projects
- You are free to suggest a project - we will discuss it

Caveats

- In general a project should not be simply be a wrapper around pre-existing software
 - ▶ Web frontend to something like BLAST
- Glueing programs is OK, if the goal is a new interface to the combination or if the goal is to perform some computational study