Programming for Biologists and Biochemists

Lecture 1: Introduction to Computing and Python Programming

Brian Kidd
March 29, 2011
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
We are drowning in information and starving for knowledge.

–Rutherford D. Roger
1. Assembling the human genome
2. Solving the 3D structure of proteins
3. Searching for information
4. Reconstructing evolutionary histories
5. Mapping gene/protein networks
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Objectives:

Become familiar with programming tools for solving biological problems*

Learn programming skills that are useful for careers in biology and the life sciences

Learn problem solving skills that apply to computational and experimental disciplines

*Expected to write your own Python program from scratch by the end.
Lectures and Labs

Lectures:
Date/Time: Tues 6-9:30p.m.
Location: Social Science II, 71

Mandatory Lab Sections:
Date/Time: Mon/Wed 5-7:00p.m.
    Thu 4-6:00p.m.
Location: Social Sciences I Mac Lab, 135
Course Resources

Class Website:
http://www.soe.ucsc.edu/classes/bme060/Spring11/

Textbook:

Bioinformatics Programming Using Python, Mitchell Model (req*)
Python Pocket Reference 4th ed, Mark Lutz (opt*)

Other resources that might be helpful are on listed on the class website

*Available for free online through Safari
Class Policies

Grading:

Programming assignments, quizzes*, final/project**, and participation

Late Assignments:

64 late hours without penalty
Deduct 1% from grade every hour afterward

Collaboration:

Encouraged, groups of ≤ 3, list teammates
Cite code snippets

*Lowest quiz grade dropped

**Project for upper division students 160 & 180
Class Format

Lectures:

Quiz
Topic 1
Break
Topic 2
Break
Challenge Questions and inClass Exercises

Labs:

Demo/intro
Assignment given out (time starts)
Course Overview

1. Computing statistics on biological sequences
2. Writing pattern discovery methods
3. Manipulating a wide range of data formats
4. Communicating with remote biological databases
5. Calling and parsing output retrieved from local and remote servers
Course Topics

Introduction to computing and python programming
Manipulating Data
Flow Control
Files and Functions
Searching Text and Pattern Matching
Large Datasets and Statistics
Biopython
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Brief History of Computing

Computing is about getting devices of various sorts, “computers” to carry out mathematical operations for us.
Renaissance of Computing

Analytic Engine:
The first mechanical computer

Proposed 1821
Built 1991
Analytical Engine Video

http://www.youtube.com/watch?v=0anIyVGeWOI&feature=fvw
Electrical Computing

ENIAC:
Early electrical computer
ENIAC Video

http://www.youtube.com/watch?v=VAnhFNJqNYY
Portable Computing

Apple’s iPad
Modern portable computer

2011
Super Computing

Blue Gene Super Computer: Super computing cluster @ UCSD 2008
Distributed Computing

Folding@home: Connecting 1000s of computers

2000
Evaluating Computing

FLOPS - floating point operation per second, e.g. $1.0 \times 2.0$
Entering an Age of Exploration

21\textsuperscript{st} century science has shifted from being data-limited to hypothesis, and analysis limited.

“Tools are needed to support the research cycle—from data capture and data curation to data analysis and data visualization.”*

Programming Aids Discovery

Sequence Analysis

Structure Analysis, Prediction, and Design

Biological Function

Personalized Medicine
Hot Topics for Life Sciences

1. Sequence analysis

2. Pattern matching and discovery
   Toussaint, et. al., NAR 2009.

3. Structure prediction and design

4. Network mapping and modeling

5. Personalized medicine
Take Home

1. Computing power keeps increasing

2. Programming necessary for both analyzing/interpreting scientific data and discovering new insights

3. Shift from data collection to data analysis and interpretation
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Computer Architecture
What’s a Computer?

Software versus Hardware

Information Flow in a Computer

Input: (keyboard, mouse, other device)

Interpret/Process: (OS, CPU)

Output: (screen, printer, storage)
UNIX

Operating System Developed in 1969

Powerful and Flexible, i.e. Open Access

Enter Commands Through a Terminal*

In UNIX, almost everything is a file

*Use Darwin terminal or X11 on Mac OS X.
Files

A file can be a program: computer instructions (recipe)

A file can be data a program uses (ingredients)

Can read them in and write them out

Can organize them into directories
File System is a Tree of Folders

In UNIX, Brian's program is at:

/home/bkidd/lab01/hello.py
or
~bkidd/lab01/hello.py
Communication with the UNIX operating system is via the command line

Enter commands in a terminal at a prompt ($, >)

*Neal Stephenson, “In the Beginning... was the command line”, 1999.
Interacting with UNIX

Type in commands to a prompt ($, >)

$ mkdir lab1

prompt command arguments
Type in commands to a prompt ($,>)

$ bl2seq -p blastp -i seq1 -j seq2 > results
1. Computers have a specific information flow: in $\rightarrow$ process $\rightarrow$ out

2. UNIX is an operating system for controlling this information flow via the command line
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Concepts in Computer Science

**Program** - a detailed, step-by-step set of instructions telling a computer EXACTLY what to do

**What can be computed?** - the field of computer science is obsessed with this question.
  design / analysis / experimentation

**Algorithm** - step-by-step process for achieving a desired result

**Natural language** - great for humans, but ambiguous
  “I saw the woman in the park with the telescope who had the telescope, who was in the park?”

**Run / Execute** - carry out the steps of a program
What is a Computer Program?

Sequence of commands written in a particular language that can be translated into operations a computer can carry out.

Analogous to a recipe:

A recipe contains instructions that, when followed, allow a person to produce an entrée, appetizer, dessert, etc.
Value of Programming

PROGRAMMING IS:

**Challenging** – must be able to see the forest (big picture) and pay attention to the trees (small details)

**Empowering** – you truly become a master of your computer!

**Creative** – combines art and science
Programming is a Tool

Use it wisely to answer questions

Known:

A genetic mutation leads to a change in the protein structure of hemoglobin, causing the molecules to form fibers that distort the shape of a red blood cell to form a sickle.
Question:

What is(are) the mutations in the protein sequence that are associated with sickle cell anemia?
How Programs Work

Source Code:

“Human-readable” instructions that contain all the logic for the program to follow

Object Code:

“Computer-readable” instructions that translate the source code procedures into steps the computer will carry out

Compiler:

source code ➔ object code
Syntax versus Semantics

Learning how to program is analogous to learning a new language

Syntax – the precise structure or form (vocabulary) of the language

Semantics – the precise meaning and content of the language
Program Statements

A program is a series of **statements**

- Analogous to steps in a recipe
- In Python each statement ends at the end of the line
- Statements can be grouped into blocks based on indentation
Program Comments

Comments are essential for good code

Allow others to follow what you’ve done

Remind yourself when you revisit code at later point

Points will be deducted from your labs if it doesn’t contain comments (also helps with partial credit)

#In Python each comment starts with a pound sign
The Programming Process

1. Plan/Outline
2. Edit
3. Compile/Interpret* 
4. Link/Load** 
5. Debug***

*Done automatically with Python. 
**Unnecessary in Python. 
***Process of elimination.
Problem Solving 101

Understand the problem statement

What are the inputs?

What are the outputs?
Creating a Master Plan

Most important step

Outline what the program will do
“pseudocode”

Capture essential steps, functions, and order of statements
• take in sequences
• compare letters
• print out differences
Example Pseudocode

Ask person to type in their name
  Input

Get whatever was typed
  Process input

Say bye in a personal way
  Output, print to screen
Exercise

Create pseudocode for the sequence comparison example

```
bl2seq -p blastp -i lgzxB.fasta -j 2hbsB.fasta
```

1. Specify inputs and outputs
2. Include error checking
3. At least 5 main steps to outline
Exercise

Create pseudocode for the sequence comparison example

Example:
- read in sequence files
- determine if sequence is protein or nucleic acid
- check if files are in the correct format
- compare two sequences
- print out results to screen
Editors

Program used to read/write/save the source code

We will teach you IDLE* in this course

Good editors:

Allow you to jump directly to a specific line of code
Color Python-specific commands (syntax highlighting)

*Many good editors exist for Python.
Debugging

Easy: Remove syntax errors from the code

Hard: Ensure program executes correctly
Development Environments

Designed to help programmers develop code, i.e. “one stop shop” for:

- Editing
- Executing
- Debugging

IDLE is a free development environment distributed with python
Programming Modularity

Turn complex problems into simpler ones
Take Home

1. Understand what the problem is and what you are asked to solve

2. Take the time to write plan out your program using pseudocode

3. Divide complex task into simpler ones that are easier to solve
Overview

1. Why programming matters
2. Course logistics and overview
3. Brief history of computing
4. Fundamentals of computers
5. Programming basics
6. First program in Python
Brief Background on Python

Developed by Guido van Rossum

- started in 1989
- first released: 1991
- version 1.0: 1994
- version 2.0: 2000
- version 3.0: 2008
What is Python?

Python is a computer language that is:

- Interpreted - fast development
- Dynamic – data handling
- Smart – memory management
- Readable – string / text oriented
- Object Oriented
- Community Supported
What Does Interpreted Mean?

Source Code – what you write

Byte Code – what Python “compiles”

Interpreter – reads and executes code

*Figure 2-2 from Learning Python by Mark Lutz.*

Interpreter simulates a computer that understands a high-level language, rather than translating the code, the interpreter executes instructions line-by-line.
Strengths of Python

Python is:

Free – http://www.python.org

Portable – (Unix/Linux, Mac, Windows, desktops, laptops, and mobile devices)

Powerful

Mixable – wrappers and glue code

Easy to use/learn! – http://docs.python.org/tutorial/index.html
Different Types of Data

Data comes in different types

Simple types like numbers and strings
Python infers data type

Boolean, Integer, Float, String

Collections of simples types
Python calls these lists or tuples
Encapsulated in [] or ()

Associations between data
Python calls these dictionaries
Encapsulated in {}}
Primitives

Simple values describe numbers and strings

7 # integer value
1.5 # float value
2e4 # float (scientific notation)
True # boolean
“Hello” # a string value can be
# single ‘Hello’ or
# double quotes “Hello”
Primitives Continued

Don’t have to explicitly define these!

However, pay attention to them when doing arithmetic

\[
\begin{align*}
1/3 & = 0 \\
1.0/3.0 & = 0.3333333333333333
\end{align*}
\]

Can explicitly define type (called casting)

\[
\begin{align*}
\text{float}(1)/3 & = 0.3333333333333333 \\
\text{int}(1.0)/3 & = 0
\end{align*}
\]

*If unsure, use type(x).*
Expressions I

Numeric Operators (think math):

2 + 2    # addition
4 * 3    # multiplication
4 ** 2   # power
4 / 2    # division
11 // 4  # floor division
11 % 4   # remainder
Expressions II

Logical Operators (think logic):

- `not True` # negation
- `True and False` # and operator
- `True or False` # and operator
Expressions III

String Operators (think words):

‘TATA’ in ‘TATATATATATA’  # substring?
‘AC’ + ‘TG’                # concatenate
6 * ‘GC’                   # repeat

‘VLSPADKTNVKAA’[0]         # subscription
‘VLSPADKTNVKAA’[-1]        # subscription
‘VLSPADKTNVKAA’[1]         # subscription

‘VLSPADKTNVKAA’[1:4]       # slicing
‘VLSPADKTNVKAA’[4:-1]      # slicing
‘VLSPADKTNVKAA’[4:]        # slicing
Calls

Function call (more to come):

len('TATA')    # number of characters
print('ACTG')   # print out string
'TATAT'.count('A') # count character
The following 2 rules will save us a lot of pain:

The first line in any of your programs should always be:

```bash
#!/usr/bin/env python
```

The second and third lines in your program should be:

```bash
# Name: Brian Kidd (bkidd)
# Group Members: None
```
Documentation

The first line in any of your programs should always be:

```bash
#!/usr/bin/env python
```

The second and third lines in your program should be:

```bash
# Name: Brian Kidd (bkidd)
# Group Members: None
```

Next should be a program overview

```quote
"""
This program does ...
inputs:
outputs:
"""
```
Homework

Reading:

Model chapter 1, appendix A (449-451)
Lutz (Learning Python) chapter 2
Python - All a Scientist Needs

Quiz:

Quiz 1 next Tues (4/5)
Covers readings, lectures, and lab

Pre Lab 1:

Email TA: eyliaw@soe.ucsc.edu to get on class list
Lab 1 Preview

Get familiar with environment

First Python program

Collect student information

Practice debugging a program