First lecture

What we'll cover

- General course structure
- What is programming?
- Why use programming?
- The Unix environment.

General course

- First six weeks are the fundamentals of programming
- Second six are special topics
- Feel free to bring in your programming challenges

Literature

- "Practical Computing for Biologists" (Haddock and Dunn)
  - Great for beginners
  - Mostly geared towards text editing

- O'Reilly Books
  - "Bioinformatics Programming in Python"
    - In Python 3, but good methodology

- http://greenteapress.com/thinkpython/thinkpython.pdf
  - Free!

Literature - The internet

- Stack Overflow
  - The answer is there, but it might be snarky

- Software Carpentry
  - Lot's of great free lessons
  - Host lectures - keep an eye out

What is a program?

How to get *inside* your computer

What is a program?

- A series of commands for your computer

What is a program?

- A series of commands for your computer
  - Computers are dumb

What is a program?

- A series of commands for your computer
    - Computers are dumb
    - Computers read binary (0s and 1s)

What is a program?

- A series of commands for your computer
    - Computers are dumb
    - Computers read binary (0s and 1s)

- So we write programs in a language that is more readable to humans
    - These are translated to binary by an assembler language that is in-between your script and the computer itself.

What is programming?

- One or more scripts saved in text files
  - Must be accessible to the operating system

- Creating software and scripts is the goal.
  - Your operating system itself is just a collection of scripts that interoperate

- Why programming?

Repeatability

- A script can be a record of what happened
  - Especially important when things go wrong
  - Publishing scripts is cool - you want to be cool

- Software builds on itself
  - Take advantage and be part of evolution

- E.g.:
  - Make a software pipeline to collect, catalogue, and align sequences in a repeatable and well-documented fashion. Now give it to somebody else so they can do it too.

Faster

- The first and most central goal of computer science
  - People have been working on this for over 50yrs. Take advantage of them.

- E.g.:
  - "I wrote down all this data, and now I need to divide every number by 4.28!"
  - "My NGS text file is too big to be opened by any text editor known to man!"

Automation

- Do the same thing lots of times
  - Let's face it, some tasks are simply below you.
  - Nothing is below a computer, and it's way better at this than you are anyway.
- E.g.:
  - "I collected two months of data on color, sex, body size, and gut content of five different species at 7 different field sites, but my advisor says only take sex and color from 2 species at 5 field sites. How do I put all this in one text file in under 2 seconds?"

Elements of Style

- Which language to use?
- Is your code readable by others?
- Is your code readable by you?
- How can you appropriately break up tasks?

Languages!

- There are many, many computer programming languages.

Languages!

- There are many, many computer programming languages.
- Things to consider:

Languages!

- There are many, many computer programming languages.
- Things to consider:
    - Speed versus readability

Languages!

- There are many, many computer programming languages.
- Things to consider:
  - Speed versus readability
  - Documentation

Languages!

- There are many, many computer programming languages.
- Things to consider:
  - Speed versus readability
  - Documentation
- What are people in your field are using?
  - Stats - R
  - Dense computation - C & C++
  - Next-Gen - Perl & Python & Unix
  - Unix is often used as "glue" in workflows

Why Python?

- General concepts almost universal

Why Python?

- General concepts almost universal
- Readable

Why Python?

- General concepts almost universal
- Readable
- Popular

Why Python?

- General concepts almost universal
- Readable
- Popular
- Well-documented

Why Unix?

- General concepts almost universal

Why Unix?

- General concepts almost universal
- Operating system written in C

Why Unix?

- General concepts almost universal
- Operating system written in C
- Very fast

Why Unix?

- General concepts almost universal
- Operating system written in C
- Very fast
- Almost universally used in computers, supercomputers and file systems
    - This is how most programmers manage and organize files

A taste of Eunuchs

- Commands are small programs
  - Type name of command and hit "enter"
  - Unix searches for the program's text file, and executes it.
- Programs have preset arguments which change their behavior
  - Find these in the manual pages
- They interact with files that are in the folder (directory) that you're in

A taste of Unix

- Interact with Unix via a "shell"
  - The shell channels information between the user and the Unix programs through "standard streams"

- Information on screen is called standard output or "stdout"

- Input to programs is "stdin"

- Also, "sterr" - will be useful later

File systems

- Your computer contains a nested hierarchy of directories.

File systems

- Your computer contains a nested hierarchy of directories.
    - Keeping track of where you are in the file structure of your computer is an important component of programming.

File systems

- Your computer contains a nested hierarchy of directories.
  - Keeping track of where you are in the file structure of your computer is an important component of programming.
  - The highest level is the root (denoted: /)

File systems

- Your computer contains a nested hierarchy of directories.
  - Keeping track of where you are in the file structure of your computer is an important component of programming.
  - The highest level is the root
- There are several high-level directories that users don't usually go into where programs files are stored
  - /usr/bin
  - /usr/lib

A note on backups

- Everyone should back up their computer regularly
- We will discuss some commands today that can remove files
    - They can be strung together to remove your whole file system

File path

- Every file has an address on your computer
    - This is the filepath

File path

- Every file has an address on your computer
  - This is the filepath
- If you are going to do an operation on a file, you'll need it's address

File path

- Every file has an address on your computer
  - This is the filepath
- If you are going to do an operation on a file, you'll need it's address
- Bash has a few filepaths where it automatically looks for program files
  - This is useful for calling programs
  - You can check which filepaths these are by typing "echo $PATH"

A few important paths

Here	.
One level up	..
Home	~ or $HOME
Root	/

.. and . -----> "relative paths"

~ or /usr/bin -----> "absolute paths"

Commands for Getting Around

1.) Common commands

2.) Working on files

3.) Stringing them together

nano

- nano is Unix's default text editor
- Type 'nano' to access it
- This will open a text editor within your terminal
- Saving, exiting and other file functions are controlled with ctrl + letter keys
- If you create a document and write to it, saving it will add the document to the current directory

Commands for Getting Around

cd	Change Directory
mkdir	make directory
ls	List
rm	Remove
pwd	Print working directory
man	Manual

Commands for Getting Around

cd	cd : takes you home cd .. : takes you up one level (to the containing directory)
mkdir	mkdir filename
ls	ls -a : shows hidden files ls -l : shows files along with sizes and timestamps
rm	rm -r : remove recursively rmdir: remove directory **CAUTION** with power comes danger!

Getting Comfortable

tab	Auto complete
*	Wildcard
Up arrow	Last command
Ctrl + C	Escape process
Ctrl + L	Clear screen

Getting Comfortable

tab	Enter enough unique characters and press tab. This will complete the filepath or command.
*	Matches every character in a filename.

File operations

grep	print line with matching plain text string
cat	Concatenate, stream to "standard out"
head/tail	Print the first or last lines in file
|	Send output of one command or program to another as input
wc	Word count
cp and mv	Copy and move

File operations

grep	grep word filename
cat	cat file1
head/tail	head -n1 file1 tail -n4 file1
|	ls -l | wc -l
wc	wc -l counts number of lines wc filename counts the words in the file
cp and mv	cp file folder makes a copy of a file into a folder mv file folder moves that file, leaving no copy

File operations

**Looking at the manual for all the commands we are showing you is worth your while. Typing 'man command name' will show the manual file

Or just Google it!

Redirection

- > versus >>
  - > overwrites file content with whatever is on the left side of the redirect symbol
  - >> appends whatever is on the left side to the file on the right side
- Between the pipe and the redirect, you can write a one-line custom program for text editing
  - "Get all sequence names from a sequence file"
  - `grep ">" file1.fas | cut -d ">" -f 2 >> seqs.txt`

Tasks

- Create a file and a directory. Put some words in the file. Copy the file into it. Now, go into the directory and delete the file. Change back into the original directory and move the file into the directory. How is this different than copying?
- Create a second file and move it into your directory. Count how many files are in the directory using a simple script.
- Copy the first line of each of your files to a new file

Bonus task

- Copy all the tree files to home
- Remove all the tree file in home
- Concatenate all the tree files in a file called trees.txt in home
- How many trees are in this file?
- The second tree is unrooted and has node labels. Make a new file with just the second tree from each of the tree files called trees2.txt