Software Carpentry Bootcamp -- Day 1

http://swcarpentry.github.io/2013-08-23-harvard/

Twitter: @swcarpentry #dst4l

Listservs: harvard@lists.software-carpentry.org
dst4l@calists.harvard.edu

DST4L intro video: http://www.youtube.com/watch?v=eO_6yQLU7tA

Class-related URLs: https://pinboard.in/u:dst4l (Christine went rogue and started a pinboard account for class-related links; ask if you want
the login, to add links yourself)

Stack Overflow is your friend.

Instructors:
Jessica McKellar (Cambridge-based startup)
David Murry (Pyth
Phillip (working at MIT)
Erik Bray (space telescope science institute in Baltimore)
Michael Selik (Infochimp)

Why Python?
You're able to use it for many, many applications:
database interaction, visualization, web.
It's a good investment because it's a very general tool.
Python is fun.
Python is a great lanuage to transform data between forms.
It's being improved for statistics.
It's easier to learn than R.
Scientists and educational institutions are consolidating around Python.

Shell I
http://swcarpentry.github.io/2013-08-23-harvard/lessons/shell/tutorial.html

Why use the shell?
It is hard to write software that is graphical. Many programs are designed to run in a command-line interface. ("Programmers are lazy.")

If you can get good at it, it's more efficient. Less clicking, shortcuts with wildcards, etc.

If you look for help, many answers will ask that you open the command line.

What is a shell? A user interface to allow you to run other programs. GUIs are shells. So is the command line!
The command line takes in input, processes it and gives the user output.

bash: born-again shell

- $ ls: list files in current directory; you will use this a lot

in a graphical interface, you can have many folders open. in a command line shell, commands are executed
within the context of a specific directory. The shell assumes you are working in the current directory when calling files or commands.
To find the current directory,

- $ pwd: show current directory ("present working directory", or maybe "path to working directory", or "it's just pwd; that's what it is")

Home directory: where your stuff is; the notion of home directories has its roots in multi-user computers of bygone days

"Directory" and "folder" are interchangeable, but programmers tend to say "directory".

- $ ls -F the '-F' is a flag or parameter (case sensitive) shows some extra symbols to help differentiate between folders and files
  - Double dashes can take more and longer parameters, like full words.
  - Commands take a --help (windows) -help (mac) command to display all of the parameters that are allowed for a particular command

- (Mac) man: manual; type q to exit
- command: man ls to get the help screen

within manual text display to search for a word use '/[term]' (e.g. /color)
Windows default command line (cmd) is completely different from bash, but it's widely considered better than the Windows one.
POSIX standards

To paste into windows git bash, hit the insert key. Also, you can right click on the window title bar, choose Edit, and select paste.
http://stackoverflow.com/questions/2304372/how-do-you-copy-and-paste-into-git-bash

- $ unzip shell-data.zip

- $ cat shell-data/molecules/methane.pdb
    - cat: concatenate - shows any plaintext file. There are python libraries that can work with spreadsheets, but why would you want to

work with anything other than plaintext? Just extract it from excel as a .csv and go forth in plaintext.

- can string multiple file names into one command line
    - $ cat shell-data/molecules/methane.pdb shell-data/molecules/ethane.pdb

tab to autocomplete. tab twice to display all files that start with the characters you have started typing. Use
the up arrow to access a command you've just run. You can go back multiple commands. The down arrow walks forward.

- $ ls shell-data
    - shows you the contents of another folder, if you aren't in it (in this case, shell-data). Let's move to the folder with the cd command

- cd: change directory
    - "cd .." goes to the directory above your current directory. ".." is usable in file paths as well.
    - "." references current directory

tilde ~ is shorthand for home directory

relative path (relative to the current working directory)
absolute path (contains full path; starts with /) (for windows users, don't use the drive letter as you might in cmd)

- ls /

Commands are just programs too. they mostly live in /bin
You can run ls or /bin/ls -- they are identical.

"A lot of the installation issues you have had this morning, and will inevitably have in the future, are related to this."

Environment variables: case sensitive! ($PATH for instance) these are the variables for the settings for the shell.

- $ echo $PATH

    - shows the directories that the shell will look for the program in. for instance, ls should live in /bin. This can be a problem for programs that have the same name, as the shell might find the wrong program first and run it. You can use a full path name to help force the correct program.
- $ which python
    - shows the path name to the directory for the current running python(may not work for Windows).

Shell II

Hidden files: any file whose name starts with a period; use $ ls -a to see in directory

one command to cat them all!

- $ cat *.pdb
    - wildcards * to transform/manipulate large sets of files
        - $ ls p*.pdb

- $ less
    - type 'h' to get help
    - allows you to arrow through the file. (Macs - 'space bar' to go down and 'b' to go back)
    - "$ less is more than $ more"

How do we use less to view the output of /molecules/ cat?

Output redirection:

- $ cat *.pdb > all_molecules.txt
    - will create file if it does not exist
    - will overwrite file if it does exist
    - This puts the output of the cat command into a file instead of displaying it on the screen
- $ cat *.pdb >> all_molecules.txt
    - using double angle brackets appends rather than overwrites
    - not an intelligent append, it will repeat data if already present
- $ less all_molecules.txt
    - Now we can see all the .pdb files all in one text file and use the less function (which has more).

Pipe redirection:
Compose multiple programs together

- $ cat *.pdb | less
  - takes the output of "cat *.pdb" and passes it along to "less," the character is called a pipe.
  - Means you don't need to have a temporary file to capture data between programs

- grep: search for text in files; "general regular expression parser"
  - line-oriented
  - case-sensitive by default
  - finds the lines that contain the string you pass it. you can get more detailed with regex (tomorrow)
  - example: $ grep "ATOM" cubane.pdb (all lines containing "ATOM" in file)
  - shell wildcard symbols ARE NOT THE SAME as regex. * will behave differently in grep and the shell
  - can take output from a pipe: $ cat *.pdb | grep "ATOM" | less
    - cat *.pdb | grep "AUTHOR" | sort | uniq
      - just the author line, sorted and unique

Text editing: $ nano
(or use a gui text editor)

#! "hash bang" "shabang"

type into editor:

#!/bin/bash
cat *.pdb | grep "ATOM"

write file (save) as: atoms.sh
If you are using a GUI text editor, you will want to save to your home directory

- $ echo ~
  - this will display your home directory

run:

- "not all files are executable files by default ... this is a sort of safety valve"
- make file into an executable: $ chmod +x ./atoms.sh
- and then run: ./atoms.sh

~~**LUNCH**~~
TELESCOPE

Python I

In Canopy: select Editor
In Anaconda: select ipython-qtconsole

Some things you can do with Python:

- data visualization (matlibplot: "Excel on crack")
- statistical analysis
- machine learning
- web scraping, turning unstructured data into structured data, using structured data more productively (Scrapy library - http://scrapy.org/)
- natural language processing (NLTK)
- building interactive & database-driven web sites
- building games (pygame)

Jessica will send out notes later

1 / 2 = 0
1.0 / 2 = 0.5
Integer vs. float (variable types)

Python, where the whitespace is ignored and the points don't matter. (Unless you are using conditional statements and need some indentation.)

Strings:

- in quotes
- single or double doesn't matter
- concatenate with +
- can be multiplied with *

Variables don't need to be declared before we use them

- "Hello" + 1
  - cannot concatenate integers and strings
- "Hello" + "1"
  - now we have two strings!

type(True) is bool
boolean is a variable type

Comparison operators in Python

- 1 == 1 (is 1 equal to 1?) (this has to be a *double* equal sign. Single = is for variable assignment *only*)
- 2 != 1 (is 2 not equal to 1?)
- Also >, >=, <, <= for greater than, greater than or equal, etc.
- "in" & "not in" both return boolean responses
    - These can be used to test strings contents like "a" in "happy" (True)
    - They can also test list or set membership
- Use the keywords "and" and "or" to create compound expressions like "if a > 1 and b == 0"
    - chained conditions are evaluated left to right

Conditional statements:

```
"if ...some boolean condition...:
    ...do this stuff...
"
```

Note: Everything under the if statement should be preceded by 4 spaces. That's how we know that line should be executed *only* if the if statement is true.

if and else:

```
sister_age = 15

brother_age = 12

if sister_age > brother_age:
    print("sister is older")
else:
    print("brother is older")
```

sister is older

else if:

```
 if sister_age > brother_age:
    print("sister is older")
elif sister_age == brother_age:
    print("same age")
else:
    print("brother is older")
```

"tracebacks": error messages

- SyntaxError: This generally means you mistyped something; Python is considered fairly syntactically flexible but there are still some hard limits most of which you'll learn with practice
- IndentationError: This means you indented one line with a different number of spaces than a line. For example under an "if" statement if you indent one line by 4 lines but the next line by 5 you might get this error. Python is very picky about indentation--make sure to use it consistently

Functions:

format: function(argument)

use type() to find out what you are dealing with

You can use single quotes and double quotes in the same way, as long as you don't interchange them.

print() is used to produce output from a file when not typing commands into interactive interpreter

exercises:
http://www.codecademy.com/courses/python-beginner-en-kSQwt/0/1#!/exercises/1

https://openhatch.org/wiki/Python_files

Python Questions:

- Q?

Discussion

For what is this useful in Access Services?

Computer literacy
Understanding new research tools
Manipulating/analyzing data produced by access services
Data too big for Excel

Jeremy's project at Wolbach:
Mapping which institutions are collaborating, where, how often

API: Application protocol interface
Just like piping in shell!