**iTerm2: command lines**

Important commands:

* pwd (present working directory: tells you where on your computer you are)
  + asking the computer a question; where in the entire directory structure of our computer are we located, i.e.e, where are we?
    - could give you something like /Users/carolinebauman
    - Think of computers as a tree, the root is the very base level of your computer.
* quit() ⇒ quits program
* ls (list: lists files and folders in current directory that you’re in)
  + the menu of things that are available to manipulate
* cd (change directory: like opening up a folder on your desktop. Also going to take an ‘argument’ - change to new directory)
  + allows you to move around; but need to supply with an “argument” to tell where you want to go (i.e. cd desktop)
  + cd D (hit tab) - knows it’s looking for things that start with a D
* At dot (or period) means the current directory you’re in
* Two dots means the directory before the one you’re in (down one level of the tree)
* Mkdr ⇒ Creates a directory
* How to get github repository into Terminal (change to SSH)
  + create repository
  + Move into folder that you’re keeping code in (using cd command)
  + git clone (then past [git@github.com](mailto:git@github.com) link from your repository page)
  + i.e. git clone [git@github.com](mailto:git@github.com):CarolineBmn/DataTest.git
  + type ls -l
  + move into repository using cd reposity name (i.e. cd datatest)
  + ls -l

**Week 3 Monday (Jackie Kazil)**

Git and Github

* Git is the tool, github is the hosted service for that tool
* Git = a version control system that allows you to record the differences in your work as you change code
* <https://help.github.com/articles/generating-an-ssh-key/>
* SSH key
  + Creating two keys: one that is public that you’ll share with Github and a private one that you’ll never share with anyone
* <http://nvie.com/posts/a-successful-git-branching-model/>
* PUSH
  + taking a file and entering it into version control (Git)
  + Take repository URL and clone into terminal
  + Save file in folder that’s the clone of your repository
  + Can check to see what’s not pushed by using git status
  + To push to repository:
    - git add .
    - git commit -m 'Add filename file'
    - git push
* PULL
  + Taking a file from version control (Git) and putting the file onto your machine
  + On Git, find url link on main repository page next to HTTPS
    - Copy link to clipboard
    - In iTerm, type “git clone link”
    - “Git pull”
      * lets you pull specific files. Navigate (cd) into folder (i.e. advanced-data-journalism/) then use git pull

**Week 4 Monday (Python in Terminal)**

* Typing python gets you into python
* Typing exit() gets you out of python
* Concepts:
  + Class: like a telephone. In Python, strings, lists. Big umbrella concept that describes object we can work with in python.
* Import ⇒ brings tools into python. Like “import urllib2” for importing from web
* STRINGS:
  + variable = “String name”
  + ex. example = “Hello”
  + Can apply something to the string, like make all upper case. Called methods
  + example.upper()
  + 'HELLO'
  + Formula for methods = objects.methodname()

**Sublime Text 2**

* Save as .py to run as a python program
* “if” statement: if a certain criteria is filled, then it will print a certain thing
  + ex: if contribution\_amount > 1000.00:
  + print 'That is a lot of money!'
* Second half of if statement is “else” clause; so if the condition ISN’T satisfied, what to do then.
  + ex: if contribution\_amount > 1000.00:
  + print 'That is a lot of money!'
  + esle:
    - print “That is not much money ….”
* “elif” clause ???
* LOOPS
  + Lists: abc = ['a', 'b', 'c']
  + To create a loot: for letter in abcs:
  + print letter
  + A loop is saying that is will apply what you want to every part of the list; doesn’t have a condition that needs to be satisfied unlike the “if” clause
  + If combining both “if” statement and lists:
  + abc = ['a', 'b', 'c']
  + for letter in abc:
  + if letter == 'a':
  + print 'AAAAAAA!!!!'
  + Dont forget to indent print function four spaces

.replace (old, new)

.zfill(width)

**Week 6 Monday**

* git clone link
  + Clones the entire repository (downloads the entire file)
* git pull
  + Updates the file with new changes
  + Navigate into the file and: git pull
* ls -a shows hidden files
* CSV is comma separated values file
* Paths and Relative paths
  + pwd will show you where you are
  + Starting with the directory where the program is, use relative path structure ./

import csv

csvfile = open('./data/sample.csv', 'r')

outfile = open('./data/sample-clean.csv', 'w')

# Now a DictReader and DictWriter

# DictReader and DictWriter are imported libraries

reader = csv.DictReader(csvfile)

writer = csv.DictWriter(outfile, reader.fieldnames)

# DictWriter writes to outfile

#reader.fieldname refers to the headers

# Write headers

writer.writeheader()

# Clean and write the data to output

for row in reader:

row['first\_name'] = row['first\_name'].upper()

writer.writerow(row)

**Week 6 Wednesday**

import csv

# Open our input and output files

csvfile = open('./data/sample.csv', 'r')

outfile = open('./data/sample-clean.csv', 'w')

# Now a DictReader and DictWriter

# DictReader and DictWriter are imported libraries

reader = csv.DictReader(csvfile)

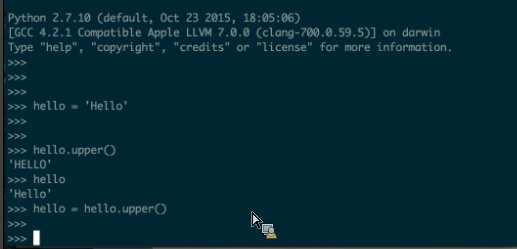
for row in reader:

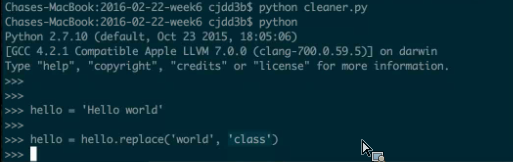
print row[‘last\_name’]

THEN TO RUN PROGRAM: go to terminal python cleaner.py

* Methods are lowercase
* Objects are uppercase

**Week 6 Friday**



**Google “python string methods”**

Zip codes are strings, not integers

>>> chase\_zip = ‘02139’

>>> int (chase\_zip)

2139

>>> chase\_zip\_int = int(chase\_zip\_

>>> chase\_zip\_int

2139

>>> chase\_zip\_dirty = ‘2139’

>>> chase\_zip\_clean = chase\_zip\_dirty.zfill(5)

>>> chase\_zip\_clean

‘02139’

**Week 7 Friday**

* Have a lot of control over the client; only some control over the server
* Request leads to a response. Response pulls the HTML
* Two types of headers
  + Request headers
  + Response headers
* Cookies are stored pieces of information
  + Used for tracking and user experience reasons
* Network tab of Chrome Developer tools

Scraping websites

* Must be familiar with Chrome Developer Tools
  + View ⇒ Developer ⇒ Developer Tools
  + If you go to a site and have the Network tab open, it records all request and response transactions

**Week 8 Monday**

Concept of greater than/less than is different for strings and numbers

Data would need to be cast as an integer or a float; it won’t work as a string

* Get and post http
  + Gets information from forms on websites
  + Information passed along to server
* The stuff after Google.com is the “get parameters”
  + <https://www.google.com/?gws_rd=ssl#q=burrito>
  + They are visible
  + q=burrito
  + Uses burrito parameter to find results related to burritos
  + q=America
* Post parameters are passed in the background
* Boone County Jail
  + Get all the inmates on one page
    - At the end of Web address: ?max\_rows=500
      * Shows 500 rows
* UNDERSTAND: Request, response, get, post, and the developer console

**Week 8 Wednesday — NO CLASS (NICAR Conference)**

**Week 8 Friday — NO CLASS (NICAR Conference)**

**Week 9 Monday**

* Scraping is about HTML and understanding how the Internet works
  + Worth investing the time and learning
  + Scraping is pulling what’s on the Web page and getting it onto an Excel or CSV format
    - For example, getting Boone County jail residents matched up with the Mizzou football team roster
    - Involves writing loops
      * Each list has multiple rows
      * Go through each row one at a time, then go to each cell one at a time
      * Configure tools in Python to make the program find the rows in the table, the cells in the rows, and then the information in the cells
        + This is where the Web inspector comes in
    - Use right-click menu ⇒ Inspect
      * Highlights what you clicked on in the inspector
* Need to know CSS
  + View ⇒ Developed ⇒ view source
  + Website uses HTML, CSS, JavaScript
  + HTML is the bones of the website
    - Structured markup language
      * Table tags
      * Divs are structural pieces of the site
      * Paragraph tags
      * Image tags
* (On-call phone rings)
  + CSS is style elements
  + Web designers have to use different names to target different elements on the page
    - To apply a certain style to a certain div, it must be identifiable and targetable
    - class =”BCSDTitle”
      * Class is applied to multiple rows on a table
    - There are also div IDs
      * ID and class can be functionally interchangeable elements on a page
      * Div is an HTML tag that you can target. It’s a block-level argument, a chunk of the page
* Need to know which part of the page contains the information you want
  + Isolate and extract one piece of the page that contains the information you care about
    - On Boone County Jail site, we want the stuff that lives inside the table
    - How to define in HTML terms this table uniquely?
      * This is where the “science” comes in.
      * Seems to be the only table on the page
        + Look for table tag ⇒ **<table**

But there are lots of tables on the page. This is a sign of bad Web design

People used to lay things out on the grid

Need to find new way to search for table

* + - * + Use Inspector tool to see if there’s something unique about this table

Right-click Last name and Inspect

Find table tag that corresponds to

BCSDTable

Collapse and shadow seem generic

Find out if BCSD exists anywhere else on the page

BCSD and center alignment

Isolating what we want solves 70% of the problem

* Use “Inspect” to see attributes of a particular element on the page
* Use “View Source” to search to see if that element is unique
* On Wednesday we’ll figure out how to get stuff out of the rows