-]\Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

All content is publicly available under the Creative Commons Attribution License: https://creativecommons.org/licenses/by/4.0/

Download data-shell.zip from this link:td

- '/' has 2 meanings:
- - root of the file system
- - file separator
- --help gives usage documentation for a command (works on windows and linux)
- man [command] gives the manual page for a command (not necessarily available across all platforms or programs written by individuals) (works on mac and linux)
- **Navigating Files and Directories**
- http://swcarpentry.github.io/shell-novice/02-filedir/index.html
- cd command changes working directory from current directory (can only go deeper - if you need to go sideways, need absolute path - path from root all the way to folder)
- cd .. command goes up one (can be iterated ../../../.. one level for each '..')
- Exercise: Relative Path Resolution

1
2
3
4XXX XXXXXXXXXXXX

- **Working with Files and Directories**
- http://swcarpentry.github.io/shell-novice/03-create/index.html
- mkdir command creates a new directory in current directory
- nano draft.txt drops into nano text editor in a file called draft.txt
- '^' means control key
- ^o write file draft in current directory
- ^x quits the program
- rm [filename] command removes file from current directory
    - This is forever - you will never get the file back
    - Either use a version control system (VSC) or know you never need it again
    - -r modifier allows it to remove directories (this is recursive and delets everything)
    - -i modifier prompts before the removal of each file
- mv [filename] can move a file from one location to another
    - It can also be used to rename a file
- 'Tab' will complete with either a unique file or will show all files/directories in current directory that begin with those letters
- '.' points to current location
- Never put whitespace in filenames as it will mess with bash (see Naming Files below)
- **Pipes and Filters**

- http://swcarpentry.github.io/shell-novice/04-pipefilter/index.html
- Underlying philosophy of UNIX Shell: Can build complex programs with simple commands
- wc command gives wordcount for file (lines words characters)
  - -l modifier gives just lines
- * command is a wildcard - UNIX shell looks for all the files that match the pattern (e.g. *.pdb finds all files in directory ending in .pdb)
- ? is a single character wildcard
- '>' pipe command - takes the output of one command and puts it into another location
- 'cat' concatenate [file] to standard output
- 'sort' command sorts the file based off modifier
  - -n sorts by number
- '>>' command adds lines to file (appends data)
- '|' pipe operater in unix that allows commands to run one after another taking output from the first to the second and so on (wc -l *.pdb | sort -n will return a sorted wordcount)
  - Can think about the pipe | as the word "then" i.e. do this command first then do a second command.
- 'head' command returns the first 10 lines of each file
  - -n [number] flag returns a specific number of lines. A negative number returns all but the last -n lines of the file
- 'tail' command returns the last 10 lines of each file
  - -n [number] flag returns a specific number of lines
- 'cp' command copies file from one document to a new clone

Loops

http://swcarpentry.github.io/shell-novice/05-loop/index.html

- For Loops:
-
  - for [keyword] in [files] (* will do all filles in the working path in alphabetical order)
    - do
    - [commands] $[keywoard] ($ calls for the variable of the keyword rather than the string)
    - done
  - for [keyword] in [files]; do [command]; [command]; ... [command]; done --> The semi colon marks the end of a line for doing these loops in a single command line.
- 'echo' command returns the string without executing the command (useful for debugging)
- 'clear' to clear the terminal

for filename in c*; do ls $filename; done

for filename in c*

- do
- ls $filename
- done

1. No files are listed
2. All files are listed
3. Only cubane octane pentane are listed
4. Only cubane is listed

1
2
3
4XX XX XXXXXXXXXxXXXxXX

- Add echo before the command that you'd like to run in a for loop - this will show you the commands that will be executed without actually executing them
- e.g.
- for filename in *.dat
- do
- echo cp $filename original-$filename
- done
- > cp basilisk.dat original-basilisk.dat
- > cp unicorn.dat original-unicorn.dat
- **Shell scripts:**
- http://swcarpentry.github.io/shell-novice/06-script/index.html
    - A way to save commands that you want to use over and over again
    - have .sh filetype
    - run shell scripts by using bash [shellScript]
    - hashtag (#) comments in shell scripts
- ***Miscellaneous links:***
- *How To Customize The Git For Windows Bash Shell Prompt*
- https://alanbarber.com/2015/12/30/how-to-customize-the-git-for-windows-bash-shell-prompt/
- *Changing your Terminal Prompt In Mac OSX*
- https://mattmazur.com/2012/01/27/how-to-change-your-default-terminal-prompt-in-mac-os-x-lion/
- Naming Files
- https://speakerdeck.com/jennybc/how-to-name-files?slide=5
- *Bash Special Variables*
- https://www.mylinuxplace.com/bash-special-variables/
- The course page:
- https://uwescience.github.io/2019-01-15-uw/
- Version Control with Git
- https://swcarpentry.github.io/git-novice/
- Download Visual Studio Code (nano replacement; modern industry standard):
- https://code.visualstudio.com/download
- Download iTerm for Mac (terminal replacement; modern industry standard):
- https://www.iterm2.com/downloads.html
- **Day 2: Git Version Control**
- Clearing your git credentials: https://stackoverflow.com/questions/15381198/remove-credentials-from-git
- Git ignore files on github:
- https://github.com/github/gitignore
- Git branching model:
- https://nvie.com/posts/a-successful-git-branching-model/
- Git kraken, one of the suggested git clients:
- https://www.gitkraken.com

Day 3: Python

SWC has 2 different python lessons:
 http://swcarpentry.github.io/python-novice-inflammation/ (older, focus on numpy)
 http://swcarpentry.github.io/python-novice-gapminder/ (newer, focus on pandas)

Sample data today (WEST ROOM):
 http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip
 make a folder on your desktop (mine is called swc_py) and unzip the sample data there.

If you didn't install anaconda:

http://swcarpentry.github.io/python-novice-gapminder/setup/

Jupyter notebooks is convenient for running python, but there are other interfaces (e.g. spyder, which is like RStudio for R)

- (if running on linux distribution, jupyter will open in firefox but doesn't like chromium)
- Can run python in terminal by typing python which opens a python shell where you can execute commands
    - quit() to exit shell
- In terminal --> python [name_of_script.py] will run all of the commands in a script
- IDE - Integrated developer environment (spyder comes with most anaconda distributions)
- Cell is the box in the jupyter file
    - edit mode (green box, accessed by hitting enter)
    - command mode (blue box, accessed by hitting escape)
        - 'h' in command mode brings up all the shortcuts
        - 'a' adds cell above current cell
        - 'b' adds cell below current cell
        - 'dd' delets current cell
        - 'z' undo
        - arrow keys move up and down from cell to cell
        - 'm' makes a cell become commentary (headings or commentary) - a markdown cell
            - Use a '#' for heading sizes
            - '- ' creates a list
            - enclosed in '$ $' gives lotec
            - 'shift + enter' renders markdown
        - 'y' makes a cell become a code cell
        - '#' in a code cell creates an in line comment
        - "shift + enter" runs a code cell
- Assigning variables
    - Give variable name, set equal to value (e.g. age = 42, first_name = 'Ahmed')
    - 'print' automatically inserts a space between each of the arguments it is given
- Jupyter:
    - kernel --> restart & run all will clear the memory and run in order (useful because jupyter keeps track of the order you run in and can create problems in your code)

For python in git bash in windows: https://stackoverflow.com/questions/32597209/python-not-working-in-the-command-line-of-git-bash

Edit this file:

- `nano ~/.bashrc`

Add this to a new line in .bashrc:

- `alias python='winpty python.exe'`

Save this change (Ctrl+O, Ctrl+X), then run:

- `source ~/.bashrc`

now you should be able to run in git bash:

- `python`

useful ascii art showing how slicing works:
https://stackoverflow.com/questions/509211/understanding-slice-notation

POST-COFFEE BREAK POLL (vote with an "x"):

too fast   :
just right :XXXXXXXX
too slow   :XXXXX

Here are the two Jupyter Notebooks and the standalone script from the 2 days of python lessons in the "Meeting room"
http://dan.mccloy.info/data/swc/