Welcome to The Carpentries Etherpad

This pad is for the Software Carpentry workshop: https://softwaresaved.github.io/2020-11-10-ssi-online/, organised by the UK's Software Sustainability Institute (https://software.ac.uk/).

The pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents. We will use the pad during the workshop to share notes and bits of code. The pad will remain available after the workshop too.

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

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

The pad is public so make sure you do not share any private information here.

Useful Links

- Workshop website: https://softwaresaved.github.io/2020-11-10-ssi-online/

Attendance

Please sign in your name below.

1. Susana Direito +1
2. Matthias Mimault +1
3. Nicole Rosik +1
4. Martin Spurr +1
5. Emel Yusuf
6. Callum Highmore +1
7. Manuel Romero +1
8. Aneesa Nabi +1
9. Simone Lucanto +1
10. Beatrice Landoni +1

 # DAY 1

1. Bash/UNIX Shell

Download the data for this session - direct URL: You can download from here:
https://swcarpentry.github.io/shell-novice/data/data-shell.zip
Unzip the file to your desktop

Bash shell commands in the command line interface

pwd – print working directory

ls – list(s currend directory's content)

ls -F (added '/' to the folders)

ls -F / (command: list, option: list folders, where: in the root directory at the top of file system hierarchy)

man ls (opens the manual of 'ls' command, what optiones/ arguments we could use, navigate through by hitting space, press 'q' for quit)

Below is a mac-specific response:
ls --help
ls: illegal option -- -
usage: ls [-@ABCFGHLOPRSTUWabcdefghiklmnopqrstuwx1%] [file ...]

cd Desktop/ ('change directory' to Desktop)

cd / (go to the root)
cd .. (up one level)

cd ../molecules/

(base) uname:Desktop robertn$

but sometimes you don't have visual feedback, thus you could reassure you are in the right directory:
can use 'pwd' command to check your current location

absolute path (start from root):
cd /Users/[your_user_name]/Desktop
your user names indicate the 'Home directory'
cd /Users/

cd ~/Desktop (Desktop folder access path from your home directory)

cd ~ (shortcut to bring back to home dir.)

Excercises

1. explore ls -l and ls -h options of list (ls) command

Can combine multiple options separately and in a concateneted way

e.g. :
ls -t -r
ls -tr

2. Understand and practice navigating through directory tree
https://swcarpentry.github.io/shell-novice/02-filedir/index.html

'Clear' console:
Ctrl (Cmd) + L

Creating new folder:

mkd thesis (makie directory named 'thesis')
if 'mkd' command not found can use 'mkdir' alternatively

creating new file:

nano thesis.txt (usineg the built-in nano editor to create a new empty text file called 'thesis.txt')

- can add some content

then Save: ^O (write out = save the file) --> type Y
then Exit: ^X

'^' denotes Ctrl (on a Mac ^ symbol is on control button)

Moving/ renaming the text file, e.g.:
mv thesis/thesis.txt thesis/quotes.txt (rename thesis.txt in the thesis dir under the 'quotes.txt' name)

check:
ls -F thesis/

move the file upwards in the folder hierarchy:

navigate data-shell directory (one level up):

cd ..

then:

mv thesis/quotes.txt . (dot at the ed is important)

copying file to a new location

cp quotes.txt thesis/quotation.txt (not only copy file content to new location but can rename it at the same time)

copying a folder, e.g.:

cp -r thesis thesis_backup ('-r' flag or option refers to 'recursive' operation: copying everything inside)

delete file (careful! delete stuff forever, there is no recycle bin)

rm quotes.txt

delete folder (again need to use a recursive operation)

rm -r thesis

Navigate to molecules directory

cd /Users/[your_user_name]/Desktop/data-shell

cd molecules/

wildcardas (e.g. '?', '*')

e.g. listing only files with .pbd extension:

ls -F *.pdb (pattern matching, * indicates, any caharcter string and we only specified the file extension we are interested in)

Being more specific e.g. interested in retrieving specific files in our folder:

ls -F *t??ne.pdb (* = replacing whatever character string is there, then need a chaacter 't' then '?' replacing individual characters, then .pdb)

Alternatively, can use without the '-F' flag or operator:

ls *t??ne.pdb

How many lines in a sepcific file:

using the wc (word count) command

But we are interested in the lines only: can use '-l' flag

wc -l *.pdb

redirect standard output (bash/ terminal console) to a user-specified file
utilising '>' operator

wc -l *pdb > lengths.txt (for further automated exploration and processing)

cat lengths.txt (look into the file we've just created)

we can sort (asc/ desc) file content - by default is alphabetical order, but here we are interested in the numeric order -> use the '-n' option

sort -n lengths.txt

can sorted results (by default outputted to the console) to a file

sort -n lengths.txt > sorted_lengths.txt

cat sorted_lengths.txt | head -n 1

head -n 1 sorted_lengths.txt

Without creating intermediate file:

wc -l *.pdb | sort -n | head -n 1

Excercise(s)

Pipe reding comprehension: https://swcarpentry.github.io/shell-novice/04-pipefilter/index.html

Using a **for loop** to execute commands on multiple files within a folder

```
$  for filename in basilisk.dat minotaur.dat unicorn.dat
>  do
>      head -n 2 $filename | tail -n 1
>  done
```

A more convenient way for task automation:

```
$  for filename in *.dat
>  do
>      head -n 2 $filename | tail -n 1
>  done
```

create a new bash script file in nano

nano middle.sh ('.sh' extension is a convention to indicate shell script files)

e.g. you could record your function or for loop in a file, record the following in the .sh file:

head -n 15 octane.pdb | tail -n 5

after saved you just need to execute the .sh script file, use the 'bash' command to do so:

bash middle.sh

can generalise your script, by using "$1" in the .sh file replacing octane.pdb --> useing the first argument

now when executing the cript need to define what file you would execute the script on, e.g.:

bash middle.sh octane.pdb

Giving more flexibility to the script by enabling the user to define all the 3 key parameters - thus let's modify our script content as follows:

head -n "$2" "$1" | tail -n "$3"

Next: create a script file for stataistical data analysis

cd ../north-pacific-gyre/

Indentation is by convention four ‘ ‘ (white space)-long

nano do-stats.sh

```
for datafile in *A.txt
do
   echo $datafile
   bash goostats $datafile > stats_$datafile
done
```

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

if we aiming for writing out multiple file content into a specific file by using by using a for loop
'>'
in each iteration the given output file's contant will be overwritten

However, if we are using '>>' this would allow to append files' content

Final excercise covered: https://swcarpentry.github.io/shell-novice/06-script/index.html

---

Feedback on Day 1 morning session (shell)

What went well, what did you learn that you are excited about using in your work?

- Very useful intro from basics to scripts
- Very great introduction to Shell
- Great introduction to shell including very useful commands
- Very useful
- I did not know I could add variables to my script to make it more universal.
- Dynamic and interactive. Good help system without grieving the whole session. Excited about the

pipe system

What could be improved, what was not clear, what needs more explanations?

- Compatability with tutorial-suggested setup (git-bash) esp on Windows
- Fast paced, it was sometimes easy to not follow one step and then miss the next steps
- Too fast sometimes
- I knew what was in the first half already, and some people found the session too fast. Maybe having something to practice on before the course starts would be nice to give everyone some basic knowledge to start with.
- A bit fastpaced at the end.
- Location of downloaded files could be more explicit

[Afternoon session] ----------

2. Building programs with Python (Anaconda, Jupyter Notebook or JupyterLab, Python 3.x)

file(s) to download (if any)/ sources of help/

gapminder dataset:
http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

https://stackoverflow.com

https://python-forum.io/Forum-General-Coding-Help

or simply google (or duckduck) on your question (or the error message you happened to receive)

Cheatsheets:

https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf

Commands/

can write your python script in the terminal (command line interface)
nano myscript.py

in the script:

e.g.
print('Hello world!') save then run from Terminal

Below is executed from JupyterLab or JupyterNotebook

Open up a new JupyterNotebook file

shift+enter to execute the code in a cell

Assignment (operator '=' to assign a value stored at a memory address in a named object)

age = 42
atom_name = 'helium'

Can retrieve specific characters from a character string:
print(atom_name[1]) # **using indexing** and in Python indexing starts from 0, while in other languages indices may start from 1, e.g. in R --> so here using index [1] Python will return with the 2nd character

Slicing a string

print(atom_name[0:3])

using parentheses around to wrap an expression e.g. (atom_name[0:3]) aslo printing its return value on the console

full_name = "Adam" + " " + "Walsh"

separator = '=' * 10 (repeat the symbol/ character 10 times)

can print on the console in a single step:
(separator = '=' * 10)

Data types

print(1 + int('2')) (implements: arithmetics - addition)

print(str(1) + '2') (implements: concatenation)

print('three squared is: ', 3**2)

linear value assignment

first = 1
second = 5 * first
first = 2
print('first is ', first, 'second is ', second)

Built-in functions

Works on both chacarter strings and numeric values/ vectors/ lists

print()
max()
min()

NB: order of string characters --
http://support.ecisolutions.com/doc-ddms/help/reportsmenu/ascii_sort_order_chart.htm

Built-in functions for numeric values

round()

```
result = print('this')
print('print result', result)
```

Asking for help:

```
print?
help(print)
```

Lists (only concept-level)

```
my_list = []
my_list.append()

my_list2 = []

my_list.extend(my_list2) # added list elements were simply added to the original list

my_list.append(my_list2)  # keep appended list as a 'sub-list'
```

Indexing

can use negative numbers starting from the end of the string or list/ vector, starting from -1 that returns with the last element

When subsetting a fragment of a list or a character string as [start_included : stop_excluded] --> the 'stop' index is excluded from the returning fraction

And since in Python (left->right) indexing starts from index=0, therefore 'start' index would return

(index+1)th element

For loops - iterating through objects/ elements

http://swcarpentry.github.io/python-novice-gapminder/12-for-loops/index.html

(you may recall what you've learned durig the bash shell session - syntax is slightly differ but construct might be familiar already)

```
for number in primes:
    print(number)
```

Interpreting error messages (...)

Excercises/

http://swcarpentry.github.io/python-novice-gapminder/04-built-in/index.html

http://swcarpentry.github.io/python-novice-gapminder/11-lists/index.html

Questions/ Comments

Feedback on Day 1 afternoon session (Python)

What went well, what did you learn that you are excited about using in your work?

- The pace that the workshop was carried out was brilliant for an absolute beginner, it has built my confidence to do further reading and exploration on using Python.
- good insights regarding the use of the environment around the code: markdowns, auto-completion, report organisation
- I liked the pace of the workshop, the concepts were explained in a clear way. I really enjoyed it and I'm looking forwrd to the next session :)
-

What could be improved, what was not clear, what needs more explanations?

- Could be useful to go through the questions together as a group, some people couldn't find the exercies (just like in the shell part of the workshop)
-
-

 # DAY 2

1. Python

Slides not for this particular workshop but overall some might be helpful:

https://github.com/robertn01/Data-Carpentry-Social_Sciences-Python/blob/master/presentation/1_2_July_SFC_DataCarp_DigiHum_Python_JupyterLab_episodes_1_4_v07072020RN.pdf

Commands (referred to website)

Libraries

http://swcarpentry.github.io/python-novice-gapminder/06-libraries/index.html

Reading data, Pandas library

http://swcarpentry.github.io/python-novice-gapminder/07-reading-tabular/index.html

gapminder dataset:
http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

import pandas as pd

```
data = pd.read_csv('data/gapminder_gdp_oceania.csv')
print(data)
```

Indexing:

iloc (index-based location)

loc (named location)

Plotting

http://swcarpentry.github.io/python-novice-gapminder/09-plotting/index.html

Conditional statements

http://swcarpentry.github.io/python-novice-gapminder/13-conditionals/index.html

Useful URLs/ further ref's

https://docs.python.org/3/library/os.html

Feedback on Day 2 morning session (Python)

What went well, what did you learn that you are excited about using in your work?

- I really appreciate that Toni said what she was coding as she did it,which helped me as I have to switch between tabs.
- Very informative
- I liked the pace at which the material was covered, extremely informative , I will definitely start using Python when doing data analysis, hopefully will start writing my own programmes.
- Very informative and well explained.
- Happy to have learn named arguments and programming style
-

What could be improved, what was not clear, what needs more explanations?

- There was a lot of content introduced and not that much time to do exercises/practice what we've learned
-
-
-
-
-

2. Git/Hub - shell terminal/CLI

Readings/ URLs:
Lesson overview: https://swcarpentry.github.io/git-novice/

https://gmd.copernicus.org/preprints/gmd-2020-270/
https://github.com

bash/ unix commands

git config --help
git --help

Configuration of environment

git config --global user.name "[your name]"
git config --global user.email "[your email - present on github]"
git config --global core.editor "nano -w"

on Windows machines also:
git config --global core.autocrlf true

git config --list

git config --global core.editor "'c:/program files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin”

Create a repository

mkdir planets
cd planets/
git init

ls -a (show hidden files, starting with a dot)

ls .git/

git status

mkdir moons

cd moons/

git init

cd ..

ls -a moons/

ls -a

rm -rf moons/.git

ls -a moons/

nano mars.txt

cat mars.txt

git status

git add mars.txt

git status

git commit -m "Commit message"

git log

nano mars.txt

(add content to file, save Ctrl + O & Ctrl + X)

git status

git diff mars.txt

git add mars.txt

git status

git commit -m "Updated commit message"

git log

less mars.txt

Managing multiple files

echo "Maybe I should start with a base on Venus" >> mars.txt

echo "Venus is a nice planet and I definitely should consider it as a base" >> venus.txt

git status

git add mars.txt venus.txt

git status

git commit -m "Updated commit message"

git log

Explore / review change log

add change to e.g. mars.txt

echo "Demonstrative changes" >> mars.txt

git diff mars.txt

git diff HEAD mars.txt

git diff HEAD~1 mars.txt (HEAD~n: nr of steps back in time)

git stash pop ('static' overview of changes)

Absolute (explicit/ specific) comparison using the generated hash can be seen in the 'git log' return

git diff 76894521f007f1a3e01ae019ed33ab5e3c6d9a99 mars.txt (hash: alphanumeric astring is unique to your own system/ session/ files)

Can use the first 7 characters from the hash --> returns with same result

If don't want to keep particular changes use:

git checkout mars.txt

git status

git checkout HEAD mars.txt

git log

git checkout 8d715f0 mars.txt (again, hash string is unique to your session)

git diff mars.txt

(git diff --help)

git restore --staged mars.txt

git status

git diff mars.txt

git checkout mars.txt

git status

(cat mars.txt)

Ignore changes (no tracking)

touch ganymede.dat jupyter.dat

git status

nano .gitignore

(add files to be ignored)

cat .gitignore

git status

git status --ignored

git add .gitignore

nano .gitignore
(*.dat
!ganymede.dat
!g*.dat)

git status

git checkout .gitignore

git status

(etc.)

Upload local files to your github (cloud)

git remote -v
git remote add origin [replade your URL - https://...]
git remote -v

git branch -v

git push origin master

cd ~/Desktop/

git clone https://github.com/[your username]/planets.git planets-collab

cd planets-collab/ (replace your dir name)

ls ../planets
ls -a

git log --oneline

nano pluto.txt
(add content)
cat pluto.txt

(then commit, etc)

Create a text file for Saturn

(...)

git add saturn.txt

git status

git commit -m "Added Saturn comment"

git log --oneline

git push origin master

cd ../planets/

ls

git branch -v

git log --oneline

git pull origin master

(cd -)

echo "Some added change to pluto" >> pluto.txt

git add pluto.txt
git status
git commit -m "Commit msg"
git push origin master
git log --oneline

(add further information to pluto.txt)

--- As a basic routine ---
1. pull changes from cloud repo
2. then push whatever changes we have locally to the cloud (cloud is THE collective/collab backup)
---
git pull origin master (~fetching cloud master baranch content/ status)

git status

git add pluto.txt
git status
git commit -m "Comment"
git log --oneline

git push origin master

(...)

git pull origin remote

git pull origin master

git log --oneline

Conflicts (course notes)

https://swcarpentry.github.io/git-novice/09-conflict/index.html

Questions/ Comments/ Issues to resolve

$ git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
        [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
        [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
        [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
        <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone   Clone a repository into a new directory
   init   Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add   Add file contents to the index
   mv   Move or rename a file, a directory, or a symlink
   reset   Reset current HEAD to the specified state
   rm   Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect   Use binary search to find the commit that introduced a bug
   grep   Print lines matching a pattern
   log   Show commit logs
   show   Show various types of objects
   status   Show the working tree status

grow, mark and tweak your common history
   branch   List, create, or delete branches
   checkout   Switch branches or restore working tree files
   commit   Record changes to the repository
   diff   Show changes between commits, commit and working tree, etc
   merge   Join two or more development histories together
   rebase   Reapply commits on top of another base tip
   tag   Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch   Download objects and refs from another repository
   pull   Fetch from and integrate with another repository or a local branch
   push   Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.

* Zenodo (examples)
(scripts)
https://zenodo.org/record/3997301#.X6vg7dunzUL
(data / inputs)
https://zenodo.org/record/3997165#.X6vhCNunzUJ [NAEI]
https://zenodo.org/record/3997271#.X6vhR9unzUJ [General]
(paper)
https://gmd.copernicus.org/preprints/gmd-2020-270/

https://choosealicense.com

Feedback on Day 2 afternoon session (Git/Hub)

What went well, what did you learn that you are excited about using in your work?

- Github seemed quite abstract and scary, but now I feel I can give it a try
- Great introduction from starting a new project to usage in various scenarios for sync/update
- I didn't use github before so at the beginning it seemed a bit scary but it turns out, it's extremely useful. I'll try to learn it a bit more
- Great coverage of Git and extension to question of licensing
-

What could be improved, what was not clear, what needs more explanations?

- I got a bit lost part way through and it took some time to catch up, but because of the online resources and recorded session I'm not worried about losing out, so thanks for those failsafes.
- A lot of material was covered in a very short time, it was easy to get lost
-
-
-
-