==========
==========

Welcome to Software Carpentry Etherpad for the January 13th-14h. workshop at the University of Connecticut!

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 Software Carpentry and Data Carpentry community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org/).
Users are expected to follow our code of conduct: http://software-carpentry.org/conduct.html

We will use this Etherpad during the workshop for chatting, asking questions, taking notes collaboratively, and sharing URLs or bits of code.

Website: https://carpentries-uconn.github.io/2020-01-13-uconn/

Socrative Login: https://b.socrative.com/login/student/
Room: UCONNSWC

gapminder data for python workshop:
https://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip
https://jeremy9959.net/data/gapminder_data.csv

data-shell for the UNIX lesson
https://swcarpentry.github.io/shell-novice/data/data-shell.zip

Pre-Workshop Survey
https://carpentries.typeform.com/to/wi32rs?slug=2020-01-13-uconn
lspwd
==========
==========

Instructors:

* Tim Moore (EEB) - timothy.e.moore@uconn.edu
* James Mickley (EEB) - james.mickley@uconn.edu
* Pariksheet Nanda (MCB) - pariksheet.nanda@uconn.edu / hpc@uconn.edu
* Dyana Louyakis (MCB) - artemis.louyakis@uconn.edu
*Jeremy Teitelbaum (MATH) - jeremy.teitelbaum@uconn.edu

Helpers:

* Cera Fisher (EEB) - cera.fisher@uconn.edu
* Kendra Mass (MCB) - kendra.mass@uconn.edu

Attendees: (Put your name & dept here):

Huidi Tian Chemistry
Lei Wang Chemistry
Mariya Topchy Marketing

Dan Phillips (MCB)
Van Augur, DPP (Public Policy)
Rebecca Thomas (Psychological Sciences)
Shaun James (Physiology and Neurobiology)
Cera Fisher (Ecology and Evolutionary Biology)
Yanning Wei (Geography)
Kevin Manning (Psychiatry, Farmington)
Amy LaFountain (Ecology and Evolutionary Biology)
Gillian McNeil (MCB)
Tim Specht (Finance)
Marcy Balunas (Pharmaceutical Sciences)
Burton Guion (Public Policy)
Meghan Maciejewski (EEB)
Xiu Zhai (ECE)
Megan Chiovaro (Psych Sci)
Essam Boraey (public Policy)
==========

Nearby options for lunch:
==========
- Cafe Coop, UConn Coop
- The Student Union
- Bookworms Cafe, Homer Babbidge Library
- The Benton Cafe

Day 1 - Shell - morning

history of sort lengths

```
  148  sort -n lengths.txt > sorted-lengths.txt
  149  cat sorted-lengths.txt
  150  cat lengths.txt
  151  ls
  152  rm lengths.txt
  153  rm sorted-lengths.txt
  154  rm pi.pdb
  155  wc -l *.pdb > lengths.txt
  156  cat lengths.txt
  157  history 10
```

Using loops in the unix shell

Start in the 'data-shell' directory
use cd to move to the creatures directory

- cd creatures/

looking in creatures directory we see three files

- ls
- basilisk.dat minotaur.dat unicorn.dat

we want a particular line out of these files, the classification line. We can use the head command with multiple files to take a peek at what they contain:

head -n 3 *

the output shows us that there's a line that starts with
CLASSIFICATION
that is the line that we want to work with. We're going to use a loop to work with these files.

- #### shell output for the exercise Pariksheet walked us through ####

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

- CLASSIFICATION: basiliscus vulgaris
- CLASSIFICATION: bos hominus
- CLASSIFICATION: equus monoceros

Socrative question explanation:
The two loop commands do not output the same information because are giving slightly different commands, because of the use of the variable name $datfile

Example: Printing the 10th thru 15th line of a file

From the molecules directory type
head -n 15 octane.pdb | tail -n 5

This gives us the last five lines of the first 15 lines -- which would be the 10th thru 15th lines.

We can put this command into a script and use variables to pass arguments to that script. Use nano to open a file called middle.sh
$ nano middle.sh

In that file, type
head -n 15 "$1" | tail -n 5

save the file (Ctrl-O, Ctrl-X) and then run your new program from the command line using the command bash

bash middle.sh octane.db

In this example, "octane.db" is the first argument we passed to our script. The variable we used, $1, will

pick up the first argument (argument 1) and use it in the command in our script.

https://hpc.uconn.edu
https://wiki.hpc.uconn.edu
https://github.uconn.edu/HPC/parallel-slurm <- We ran example 3 today

If you're interested in using HPC to speed up your research:

1. Create an account on the Storrs cluster here by clicking on "Get an account": https://hpc.uconn.edu
2. E-mail our shared ticketing mailbox hpc@uconn.edu for an appointment to review your research question, installing your required software on the cluster, and how we can optimize your workflow.

Day 1 - Python - afternoon

To setup:
Create a directory called "PythonWorkshop" (on your desktop or in your homedirectory)
[e.g.
cd
cd Desktop
mkdir PythonWorkshop

then put the file
https://jeremy9959.net/data/gapminder_data.csv
in that directory.

My SOCRATIVE Home is TEITELBAUM4084

For later reference, the cheat sheet
https://jeremy9959.net/data/Pandas_Cheat_Sheet.pdf
might be useful. But it's for later.

Open a shell window
go to the PythonWorkshop directory
jupyter lab

For the notes from the first part, go to:

http://jeremy9959.net/Python-1.html

Day 2 - Python - morning

GOOD MORNING EVERYONE! REMEMBER TO GET YOUR RED/GREEN STICKIES handy.

Make sure the etherpad is handy

If you need something to work on as a warmup:

- take the original pandas dataframe "data" that had one row for each country and year
- use the pivot table command to make a data frame that has years in the index, one column for each country, and has the gdpPer capita in each cell.
- plot the gdp percapita over time as a bar chart for Germany, Italy, France, and Sweden
- Put a title on your plot

Becca:
gdp_by_year_in_countries = pd.pivot_table(data, index='year', columns='country', values='gdpPercap')
gdp_by_year_in_countries[['Germany', 'Italy', 'France', 'Sweden']].plot(kind='bar', figsize=(10,10), title="GDP By Year in Germany, Italy, France, and Sweden")

Megan:
- data_by_country = data.set_index("country")
- gdp_by_country_over_time=pd.pivot_table(data_by_country, index='country', columns='year', values='gdpPercap')
- transpose = gdp_by_country_over_time.T;
- transpose[['Germany', 'Italy', 'France', 'Sweden']].plot(kind='bar', figsize=(10,10), title='gdpPercap per Year Across Country')

Marcy:
data_day2 = pd.pivot_table(data,index='year',columns='country',values='gdpPercap')
data_day2[['Germany','Italy','France','Sweden']].plot(kind='bar',title='GDP Over Time in Europe')

For thos who want to have the (rough) notes for today's session handy, they are at
http://jeremy9959.net/Python-2.html

Pivot tables, grouping and plotting

```
import matplotlib.pyplot as plt
plt.style.use('ggplot')
import pandas as pd
data = pd.read_csv('gapminder_data.csv')
data_by_country = data.set_index('country')
gdp = data.pivot_table('gdpPercap', ['continent', 'country'], 'year')
gdp
gdp.groupby('continent').mean()
gdp.groupby('continent').min().T.plot()
gdp.groupby('continent').max().T.plot()
## Begin tangent of figuring out why gdp per capita crashed in Asia after the 70s.
gdp.loc['Asia', 'Kuwait'].T.plot()
## End tangent
```

Faceting plots

It would be nice to use a command to plot each continent separately.
In markdown, hashes make headings; here's a cheatsheet:
##https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
Iteration
for prime in [2,3,5,7,11,13,17,19]:

- print(prime)

For number in range(10):

- print(number)

range(x,y,z) is x=start y=end z=step size
for number in range(1,10,2):

- print(number)

indent can be four spaces or tab (for jupyter)
L=[]
for x in range(10):

- L.append(x) #take the element x and put it in the list L

print(L)

import numpy as np
for x in np.arange(0,1,.1):

- print(x)

sum=0
for x in range(20):

- sum = sum + x

print(sum)

sentence = "You have nothing to lose but your chains"
words = sentence.split() ## split means split a string at white space by default or another character if you give it one
words
sum=0
for x in words:

- print(x, len(x)
- sum = sum+len(x)

print('Total length is ', sum)

Problem to think about:
Given L = [1,2,3,4,5...] (any list)
make a new list:
[1, 1+2, 1+2+3, ...]
How could you make a for loop to do this?

Van:

```
sum=0
M=[]
L=range(1,11,1)
for x in L:
    sum=(x+sum)
    M.append(sum)
print(M)

L=[1,2,3,4,5,6,7,8,9,10]
sum=0
L2=[]
for x in L:
    sum=sum+x
    L2.append(sum)
print(L2)
```

Please download and unzip these files to your workshop directory:
http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

Iteration over files

```
from glob import glob
for file in glob('data/gapminder*.csv'):
```

- print(file)

```
americas = pd.read_csv('data/gapminder_gdp_americas.csv', index_col='country')
americas.head()
americas['gdpPercap_1952'].min() # finds smallest value in column
americas['gdpPercap_1952'].idxmin() # tells you which row (by your index value) has smallest value

for file in glob('data/*gdp*.csv'):
```

- data = pd.read_csv(file, index_col='country')
- print(file,data['gdpPercap_1957'].idxmin())

```
for file in glob('data/*gdp*.csv'):
```

- data = pd.read_csv(file, index_col='country')
- print(file,data['gdpPercap_1957'].idxmin(),data['gdpPercap_1957'].idxmax())

Nested loops

```
for fruit in ['apple','pear','grape']:
```

- for color in ['green','orange','yellow']:
    - print("I wish I had a ",color,fruit)

```
for file in glob('data/*gdp*.csv'):
```

- for year in ['1957','2002']:
    - data = pd.read_csv(file,index_col='country')
    - continent = file.split('_')[2].split('.')[0].upper()

- key = 'gdpPercap_'+year
- print(continent,'\nPoorest in '+year+':',data[key].idxmin(),
    - '\nRichest in '+year+':',data[key].idxmax(),
    - '\n\n')

#data.shape returns (rows, columns)
data.shape(0)

write a loop that goes through the gdp files and finds the one with the fewest rows
from socrative:
import glob
import pandas as pd
fewest = ____
for filename in glob('data/*.csv'):
    dataframe = pd.____(filename)
    fewest = min(____, dataframe.shape[0])
print('smallest file has', fewest, 'records')

Solution to Socrative exercise:
from glob import glob
import pandas as pd
fewest = pd.np.inf
for filename in glob('data/*.csv'):
   dataframe = pd.read_csv(filename)
   fewest = min(fewest, dataframe.shape[0])
print('Smallest file has', fewest, 'records')

Functions

def last_letter(x):

- print(x[-1])

last_letter ## is a function
last_letter('jeremy') ## will return last letter, y
last_letter(['ab','bc','ft']) ## will return last element, ft

def square(x):

- y = x**2
- return y

square(5)
z = square(5)
print(z)

ll = last_letter('jeremy')
def last_letter(x):

- print(x[-1])
- return x[-1]

last_letter('jeremy')
ll = last_letter('jeremy')
print(ll)

# if we want to take a numerical month, numerical day, numerical year and convert to a formatted string, make a function :
# month,day,year are all integers - these are the ARGUMENTS of the function
def format_date(month,day,year):

- joined = str(month)+'/'+str(day)+'/'+str(year)
- return joined
- ## will give no response

format_date(10,12,2020)
# will return 10/12/2020
"My birthday is "+format_date(9,9,2020)
joined = "this is a variable called joined"
format_date(11,3,2020)
joined
will print our GLOBAL variable joined, not the LOCAL variable joined within the function
def format_date(month,day,year):

- joined = str(month)+'/'+str(day)+'/'+str(year)
- print(joined)
- return joined
- ## now the joined within the function will be returned when we run the function

def a_demo(x):

- x = x+5
- return x

a_demo(8)
# returns 13
x=0
a_demo(3)
#returns 8 because function uses local variable x in the function
x
#returns 0 because this is the global variable outside the function

caveat is lists - when you pass a list or an array through a function, it will change the elements within that list globally:
def interesting_case(L):

- L.append('aha!')
- return 'hmmm'

L=[1,2,3]
interesting_case(L)
# returns 'hmmm'
L
returns [1, 2, 3, 'aha!']

def different_case(L):

- S = L + ['aha!']
- return S

different_case(L)
# returns [1, 2, 3, 'aha!', 'aha!']
L
but L still returns [1, 2, 3, 'aha!'] because the different_case function isn't appending to the list

back to gapminder data
data=pd.read_csv('gapminder_data.csv',index_col='country')
gdp_per_capita_by_year = pd.pivot_table(data, index='country', columns='year', values='gdpPercap')
gdp_per_capita_by_year.plot()
gdp_by_year = gdp_per_capita_by_year.T
gdp_by_year.plot(legend = None)

def gdp_plot(country):

- gdp_by_year[country].plot()

gdp_plot('United States')

def gdp_plot(country):

- gdp_by_year[country].plot(title='GDP over time for '+country)

gdp_plot('United States')

gdp_by_year.loc['1957':'1972']
def gdp_plot(country, start_year, end_year):

- gdp_by_year.loc[start_year:end_year][country].plot(title='GDP over time for '+country)

gdp_plot('United States','1952','1967')

More Plotting
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [1, 4, 9])

import numpy.random as rnd
rnd.choice([-1, 1]) # Picks an element from a list at random
# Print a random number from 0-9 20 times.
for i in range(20):
    print(rnd.choice(range(10)))

# This function covers a lot of things we've already done.
# Flips a coin.
def random_walk(N):
    spot = 0
    L = []
    for i in range(N):
        spot = spot + rnd.choice([-1, 1])
        L.append(spot)

```
    return L
random_walk(10)
plt.plot(range(100), random_walk(100))
# Plots stack on eachother!
for i in range(100):
    plt.plot(range(100), random_walk(100))

# Challenge: Try to plot where the random walk ended up each of the 100 times and make a histogram.
# Hint: Make a histogram using plt.hist(), for example:
# import matplotlib.pyplot as plt
# import numpy.random as rnd
# plt.hist([rnd.randint(1, 5, 20)])

seaborn
altair -- ggplot-ish
bokeh

end_spots=[]
for i in range(100):
```

- walk = random_walk(100)

```
x=5
if x<=5:
```

- print(x)

```
def my_abs(x):
```

- if x<0:
    - return -x
- else:
    - return x

```
my_abs(-3)
## returns 3
my_abs(7)
## returns 7

# suppose we have a list
# we want to splt the list into a low list and a high list, with low containing everthing < threshold, and high containing everything >= threshold
def split_threshold(threshold,L):
```

- Low = []
- High = []
- for item in L:
    - if item < threshold:
        - Low.append(item)
    - else:
        - High.append(item)
- return Low, High

```
x = split_threshold(2,[-1,1,2,5,7])
x # returns ([-1, 1], [2, 5, 7])
low, high = split_threshold(0,[-1,-4,-7,1,3,5])
low # returns [-1, -4, -7]
high # returns [1, 3, 5]
```

```
for x in 'Jeremy Teitelbaum':
    if x>'r':
    print(x)
```

```
for x in 'Jeremy Teitelbaum':
    if x>'R':
    print(x)
```

```
for x in 'Jeremy Teitelbaum':
    if (x>='r' and x<='u'):
        print(x)
```

```
def format_date(month,day,year):
    """takes integers month, day, year and returns string in form month/day/year"""
    joined = str(month)+'/'+str(day)+'/'+str(year)
    print(joined)
    return joined
```

```
?format_dat ## to get the docstring added to the function
## add a docstring to your functions to remind yourself or others using your function about what it does
format_date() # returns '0/0/0'
```

```
def squares(N):
    """returns a list of i**2 for i from 0 to N-1, N dafault is 10"""
    L=[]
    for i in range(N):
        L.append(i**2)
    return L
```

```
squares()
squares(20)
```

Day 2 - Git - afternoon

I will be sharing my terminal session using "tmate". You can scroll through my terminal history at any time, so, say, if you go to the restroom and fall behind, you can see all my work online to catch up. If you need you can also:

1. Drag to select text in my window and scroll upwards. Copy and paste into your local text editor.
2. At the very end of the day, I will post the full log of commands and output; unfortunately the log doesn't update until I end the session.

Link to tmate session: https://tmate.io/t/ro-pFQzVmp2NQq6CCMFXfL5SgUkD

Socrative room: UCONNSWC

country	pop	gdpPercap
bangladesh.txt	1.356568e+08	1136.390430<-
belgium.txt	1.031197e+07	30485.883750
bolivia.txt	8.445134e+06	3413.262690
botswana.txt	1.630347e+06	11003.605080
cameroon.txt	1.592999e+07	1934.011449
central_african_republic.txt	4.048013e+06	738.690607
chad.txt	8.835739e+06	1156.181860 <- Xiu
chile.txt	1.549705e+07	10778.783850 <- Lei
comoros.txt	6.143820e+05	1075.811558
congo_dem_rep.txt	5.537985e+07	241.165877
ecuador.txt	1.292123e+07	5773.044512
equatorial_guinea.txt	4.956270e+05	7703.495900
eritrea.txt	4.414865e+06	765.350001
gambia.txt	1.457766e+06	660.585600
guatemala.txt	1.117865e+07	4858.347495
honduras.txt	6.677328e+06	3099.728660
hong_kong_china.txt	6.762476e+06	30209.015160<-Huidi
iceland.txt	2.880300e+05	31163.201960 <- Delaina
india.txt	1.034173e+09	1746.769454 <- taken by pariksheet!
italy.txt	5.792700e+07	27968.098170Mahdi
lebanon.txt	3.677780e+06	9313.938830
liberia.txt	2.814651e+06	531.482368
mali.txt	1.058018e+07	951.409752< -- Shaun
mauritania.txt	2.828858e+06	1579.019543
mozambique.txt	1.847378e+07	633.617947
nepal.txt	2.587392e+07	1057.206311
norway.txt	4.535591e+06	44683.975250 <-- Van
pakistan.txt	1.534035e+08	2092.712441
philippines.txt	8.299509e+07	2650.921068
puerto_rico.txt	3.859606e+06	18855.606180
romania.txt	2.240434e+07	7885.360081
saudi_arabia.txt	2.450153e+07	19014.541180
senegal.txt	1.087004e+07	1519.635262
switzerland.txt	7.361757e+06	34480.957710<-Amy
syria.txt	1.715581e+07	4090.925331
thailand.txt	6.280675e+07	5913.187529 <- Becca
tunisia.txt	9.770575e+06	5722.895655
west_bank_and_gaza.txt	3.389578e+06	4515.487575<-Mariya
yemen_rep.txt	1.870126e+07	2234.820827
zambia.txt	1.059581e+07	1071.613938 <- Fengrong

My GitHub repository: https://github.com/omsai/countries

Read about licenses: https://www.gnu.org/licenses/licenses.html

Extra Exercises:

Exercise 0: Create your GitHub account (you will need to do this anyway for the lesson at the very end on "Pull requests")

(1-5 minutes)
If you don't have a GitHub account, create one here using the "Free" plan: https://github.com/join

Exercise 1: Copy SSH Keys on GitHub (saves you from typing your username + password)

(3-10 minutes)

1. Check whether you already have SSH keys created. If the files ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub already exist then continue to the next step. Otherwise in your MacOS terminal or in your Windows Git Bash terminal, type the following:
    1. mkdir -p ~/.ssh
    2. chmod 700 ~/.ssh
    3. ssh-keygen -N "" -f ~/.ssh/id_rsa
2. Now copy your ~/.ssh/id_rsa.pub file to GitHub as explained in https://help.github.com/en/articles/adding-a-new-ssh-key-to-your-github-account
3. Test your connection following the instructions in: https://help.github.com/en/articles/testing-your-ssh-connection

Exercise 2: Install Git plugin for Jupyter Hub

(5-10 minutes)
!!! This will restart Jupyter Hub! Only try this exercise after finishing the python lesson and saving all your Jupyter notebooks !!!
I don't explicit have step-by-step instructions for this; you might do a web search for installing packages using Anaconda:
https://github.com/jupyterlab/jupyterlab-git

Exercise 3: Learn how SSH Keys work

(5-10 minutes)
SSH keys can be understood using the paint mixing analogy where the mixing function is trivial, but reversing the mix is very hard. Your "id_rsa.pub" file is your public key which is directly shared, and "id_rsa" is your secret key which should never be shared. See https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange see also the figures here: https://en.wikipedia.org/wiki/Public-key_cryptography
Public key cryptography is one of the technological breakthroughs that makes e-commerce possible; we would otherwise have to rely on horrible things like symmetric encryption.

Exercise 4: Learn more about the command-line using the Over the Wire "Bandit" game

(5-20 minutes per level)
These command line exercises are how we train the UConn Cyber Security club. The goal is to "steal" passwords from a toy server. Using the knowledge you have, you should be able to get through the first 6 levels without too much difficulty. There is a somewhat linear difficulty progression: https://overthewire.org/wargames/bandit/ ... the tarball unpacking gamea couple of levels in is quite amusing.