Countries Repos: https://github.com/OSU-HPCC/countries

Algeria
Libya

My Country:
Venezuela
United_States.md
PERU
Greece
Korea
Australia
Norway
India
Ireland
Andorra
Afghanistan
Nepal
Nigeria
Canada
China

Welcome to Etherpad!

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

Get involved with Etherpad at http://etherpad.org

*Attendance (Day 02):
https://docs.google.com/forms/d/e/1FAIpQLSeb9Oe5ozaSnhZoR9s3w56uOzXqUp4bjCDVVDGmNx0AvU1v7A/viewform?usp=sf_link

*
***Lesson 1: The Bash Shell**

*File Download: http://swcarpentry.github.io/shell-novice/data/data-shell.zip

*Case Sensitivity in the Bash Shell

Generally most things in the Bash shell and other unix-like command line environments are case sensitive. Some of this is built into the programs themselves. Some of this is because of the computer's file system.

Some computer filesystems are case sensitive. On a case sensitive file system, you could have two files named "File.txt" and "file.txt" in the same directory. On a filesystem that is not case sensitive, you would not be able to create two files named "File.txt" and "file.txt" in the same directory.

Windows systems **almost never** use case sensitive filesystems.
Mac systems may use case sensitive filesystems. (The output of the command "diskutil info /" will show "case-sensitive" in its output if your filesystem is case sensitive.)

Linux systems **almost always** use case sensitive filesystems.

Most of the commands you enter in the bash shell are the names of programs. These are called "external commands". When you run the "ls" command, bash calls the program file named "ls" (or "ls.exe" on windows). Some commands (for example "cd") are built into bash. These are called "internal commands".

On a computer with a non-case sensitive filesystem, the names of external commands and any command line arguments that represent files or directories will not be case sensitive (because it's impossible to have separate files named "ls", "Ls", "lS", and "LS" all in the same place).

Internal commands and command line arguments are usually case sensitive (depending on the individual program) no matter what type of filesystem is used.

Case sensitivity is simpler from a computational perspective, but unfortunately more potentially confusing on the human side. Computers are fundamentally case sensitive. The characters "A" and "a" are completely different things, but they have roughly equivalent meaning; humans are great at dealing with this sort of thing, but computers need explicit rules to define this sort of thing (thus more complicated code, and longer processing time).

*Lesson 2: Introduction to Python

At the python prompt:

- x="hello"
- y=3
- type(x)
- # Should show "str"
- type(y)
- # Should show "int"
- y*2
- # Should show "6"
- x*2
- # Should show "hellohello"
- x*3.14159
- # Should result in an error
- x
- # Should show "hello"
- x+1
- # Shows an error
- y
- # Should be "3"
- y+1
- # Should be "4"
- x*y
- # Should be "hellohellohello"
-

Create file "hello.py" with contents:

```
x="hello"
y="world"
print(x,y)
```

python hello.py

Source material for today's lesson:
https://swcarpentry.github.io/python-novice-gapminder/

please download and unzip the file python-novice-gapminder-data.zip.
https://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

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

***01/11/2019**
*
***Morning - Python (in Jupyter)**
*
***Instructor: Nathalia**

Code below includes everything we did in class, and code that I didn't have time to show.
ENJOY!

---If you use a Mac:

- - Open Terminal, go to Desktop, create a folder '1-11-2019_Python2', go into 1-11-2019_Python2, and launch jupyter. The bash code will look like this (don't type the $ when you are in the terminal. Here $ is to indicate your bash prompt):
        - $ cd Desktop
        - $ mkdir 1-11-2019_Python2
        - $ cd 1-11-2019_Python2
        - $ jupyter notebook
    - - Click on 'New' > 'Python3'

---If you use Windows:

- - Launch Anaconda prompt, type 'jupyter notebook' and press enter. The browser will launch. The click on Desktop.
- - Create a folder by clicking on 'New' > 'Folder'. Notice a folder called 'untitled' is created.
- - Click on the check box in front of 'untitled', and select 'Rename'. Then, rename it to '1-11-2019_Python2'.
- - Click on the name of the folder to go in.
- - Click on 'New' > 'Python3'

*
***Quick RECAP**

- Markdown:

Copy the following lines within the ____ (...) ____ to a Markdown cell, then run it to see what it renders:

____

Who doesn't love a bullet point?!

+ Great way to keep track of your code/analysis
+ Reproducibility

- Easily re-run analysis
- Write definitions, descriptions, meanings, ... what the code does so you can remember what you did...

1. Hello!
2. Bye!

```
# Level 1
## Level 2
### Level 3

*This is italic*
**This is bold**
***Wow, this is italic AND bold***
```

- Python:

```
a=3
b='string'
c=4
d=a+c
print(d)
a=10
a
print(d)
fruits=['orange','strawberry','banana','mango']
fruits[1]
fruits[0]
fruits[-1]
print('I am going to the supermarket, and will buy',a, fruits[-1],'LOL')
print('I am going to the supermarket, and will buy',a, fruits[-1]+'s','LOL')
type(a)
type(b)
type(fruits)
len(fruits)
numbers=[1,3,6,44,55]
print('min =', min(numbers),'max =',max(numbers))
fruits.append('coconut')
fruits
del fruits[1]
fruits
fruits.insert(1,'strawberry')
fruits
fruits.insert(0,'strawberry')
fruits
for item in fruits:
    print(item)
for banana in fruits:
    print(item)
for banana in fruits:
    print(banana)
for item in fruits:
    print(item)
for i in range(1,55,4):
for i in range(1,55,4):
    print(i)
```

- Working with real data set:

- **1. Download data:**

```
## The commands  (%%bash) below might not work.
## In case they don't work, download and extract the file manually to the folder you opened your notebook (Desktop/1-11-2019_Python2/)
## This is the link http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip if you need to download it manually.

%%bash
curl -O http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

%%bash
unzip python-novice-gapminder-data.zip

ls

## During the workshop, some people had the data sets extracted in this path: python-novice-gapminder-data/data/
## others had data/
## Do `ls` to see how your files were extracted.
## It is IMPORTANT to provide the correct path to the gapminder*.csv files, otherwise none of the code below will work.

cd python-novice-gapminder-data/data/          OR            cd data/
```

- **2. Importing data into DataFrame**
-

```
import pandas as pd

data=pd.read_csv('gapminder_gdp_oceania.csv')

data.head()

data=pd.read_csv('gapminder_gdp_oceania.csv', index_col='country')

data.head()
data.columns
data.info()
```

- **3. Summary Statistics**

data.describe()

- **4. Save summary statistics to a file**

data.describe().to_csv('oceania_summ_stats.csv')

# enter a new name that is unique

ls

- Exercise:
- # 1.
- ## A. Import `gapminder_gdp_americas.csv` as `americas`,
- ## B. display summary statistics, and
- ## C. save to `americas_summ_stats.csv`

- Solution:

```
americas=pd.read_csv('gapminder_gdp_americas.csv', index_col='country')
americas.head()
americas.describe()
americas.describe().to_csv('americas_summ_stats.csv')
americas
```

- **5. Transpose**
-

```
americas.head()
americas.T
americas.describe()
americas.T.describe()
americas.head()
americas_t=americas.T
americas_t
americas
americas.head()
```

- **6. Slicing, and selecting values**

```
americas.loc["Brazil",:]
americas.loc[:,"gdpPercap_1952"]
americas['Ecuador':'Jamaica',:]
americas.loc['Ecuador':'Jamaica',:]
americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972']
americas.loc['Ecuador':'jamaica','gdpPercap_1957':'gdpPercap_1972']
americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972']
americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972'].max()
americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972'].T.max()
americas.loc['Ecuador':'Jamaica',:].T
americas.loc['Ecuador':'Jamaica',:]
americas.loc['Ecuador':'Jamaica',:]
americas.loc['Ecuador':'Jamaica',:].max()
americas.loc['Ecuador':'Jamaica',:].T.max()
americas.loc['Ecuador':'Jamaica',:].T

americas.iloc[9:16,2:6]

americas.iloc[:,1:].head() # omits the 'continent' column
americas.iloc[0:3,1:3] # 1:3 -- 1 omits the 'continent' column

# Observation:
```

- 1:3, omits the final index (i.e. index 3) in the range provided, while a named slice, ‘gdpPercap_1952’:’gdpPercap_1962’, includes the final element.

- **7. Selecting a subset of data**
-

```
subset_am=americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972'].T
subset_am
subset_am.T
subset_am

# Type these next two lines in one code cell:
print('\t--> Subset of Americas:\n\n', subset_americas,'\n\n')
print('\t--> Where are values larger than 3,500 \n\n',subset_americas >=3500)
```

- **8. Masking**

```
mask=subset_am>3500
subset_am[mask]

#Another way to mask is:
   subset_americas[subset_americas >=3500]

#Compare summary statistics
subset_americas[mask].describe()
americas.loc['Ecuador':'Jamaica','gdpPercap_1957':'gdpPercap_1972'].describe()
```

- Exercise:
- # 2.
- ## A. Import `gapminder_gdp_europe.csv` as `europe`,
- ## B. slice to keep `Italy` to `Poland`, and
- ## C. also slice `1962` to `1972`
- ## D. Mask values less than `8000`
- Solution
- europe=pd.read_csv('gapminder_gdp_europe.csv', index_col='country')
- subset_europe=europe.loc['Italy':'Poland','gdpPercap_1962':'gdpPercap_1972'] subset_europe
- masker=subset_europe>8000
- subset_europe[masker]
- **9. Plotting!**

```
%matplotlib inline
import matplotlib.pyplot as plt

# For the plots to work, type the each block below into one code cell

time=[0,1,2,3,4]
position=[0,100,200,300,400]
plt.plot(time,position)
plt.xlabel('Time h')
plt.ylabel('Position km')


data.loc['Australia'].plot()


data.loc['Australia'].plot()
plt.ylabel('Years')


years=data.columns.str.strip('gdpPercap_')
data.columns=years.astype(int)


data.loc['Australia'].plot()
plt.ylabel('Years')
plt.xlabel('GDP')


data.loc['Australia'].plot()
plt.ylabel('GDP')
plt.xlabel('Years')


gdp_aus=data.loc['Australia']
gdp_aus


plt.plot()


plt.plot(years, gdp_aus)


plt.plot(years, gdp_aus,'g--')


plt.plot(years, gdp_aus, 'k:')


plt.plot(years, gdp_aus,'rD')


plt.plot(years, gdp_aus, 'rD-')
plt.plot?  # A window will open with information on how to customize your plots
```

```
data.plot()
```

```
data.T
```

```
data_transposed=data.T
plt.ylabel('GDP per capita')
plt.xlabel('Years')
```

```
gdp_nz=data.loc['New Zealand']
gdp_nz
```

```
plt.plot(years, gdp_aus, 'b-', label='Australia')
plt.plot(years, gdp_nz, 'g-', label='New Zealand')
plt.legend(loc='upper left')
plt.xlabel('Years')
plt.ylabel('GDP per capita')
```

```
plt.scatter(gdp_aus, gdp_nz)
plt.xlabel('Australia')
plt.ylabel('New Zealand')
```

```
# another way to scatter plot
data.T.plot.scatter(x='Australia',y='New Zealand')
```

```
# another way to scatter plot
data.T.plot.scatter(x='Australia',y='New Zealand', c='green', marker='D')
```

```
plt.scatter?
```

```
plt.style.use('ggplot')
data.T.plot(kind='bar')
plt.ylabel('GDP per capita')
```

- Exercise:
- # 3.
- ## A. Use `europe`
- ## B. Plot a line graph with the `min`, `mean`, and `max` GDP per capita over time for all countries

- Solution:
- europe.min().plot(label='min')

- europe.max().plot(label='max')
- europe.mean().plot(label='mean')
- plt.legend(loc='best')
- plt.xlabel('Years')
-

More plotting: Shows the relationship between min and max GDP in Asian countries for each year. No particular correlations can be seen between the minimum and maximum gdp values year on year. It seems the fortunes of asian countries do not rise and fall together.

```
asia = pd.read_csv('gapminder_gdp_asia.csv', index_col='country') asia.describe().T.plot(kind='scatter', x='min', y='max')
```

```
asia.max().plot()
print(asia.idxmax())
print(asia.idxmin())
```

```
data_all = pandas.read_csv('gapminder_all.csv', index_col='country')
data_all.head()
data_all.plot(kind='scatter', x='gdpPercap_2007', y='lifeExp_2007', s=data_all['pop_2007']/1e6)
```

```
help(data_all.plot)
```

- **10. Saving your plots**
-

```
# Add `plt.savefig('my_figure.png')`  to the end of the cell in which you have the plotting code.
# MAKE SURE YOU CHANGE NAME OF THE FIGURES
```

-

```
data_all.plot(kind='scatter', x='gdpPercap_2007', y='lifeExp_2007', s=data_all['pop_2007']/1e6)
plt.savefig('my_figure.png')
```

FROM THIS PART ON, WE DID NOT COVER. BUT IT MIGHT SERVE SOME OF YOU.

- **11. Looping over datasets**
-

```
import glob
```

-

```
# glob matches files with a pattern, aka globbing

print(glob.glob('*.csv')) # we have extra files with summs... remove those

print(glob.glob('gap*.csv'))

for file in glob.glob('gap*.csv'):
    print(file)
```

```
for file in glob.glob('gap*.csv'):
        dataframe=pd.read_csv(file,index_col='country')
```

- print(file, '\n\n',dataframe.head(),'\n\n')

```
print('{}\t{}\t{}\t{}'.format('Filename','Minimum','Mean','Maxima'))
for file in glob.glob('gap*.csv'):
        dataframe=pd.read_csv(file,index_col='country')
        subset=dataframe.loc[:,"gdpPercap_1952"]
        print('{}\t{}\t{}\t{}'.format(file.strip('.csv'),subset.min(),subset.mean(),subset.max()))
```

```
with open('loop_minTOmax.txt','w') as output:
       output.write('{}\t{}\t{}\t{}\n'.format('Filename','Minimum','Mean','Maxima'))
       for file in glob.glob('gap*.csv'):
       dataframe=pd.read_csv(file,index_col='country')
       subset=dataframe.loc[:,"gdpPercap_1952"]
       output.write('{}\t{}\t{}\t{}\n'.format(file.strip('.csv'),subset.min(),subset.mean(),subset.max()))

output.close()
```

-

- **12. Functions**

```
# Type the following block of code into one code cell, and run it.

def temp_converter(a, b):
    if a == 'C':
        fahr=9/5*b+32
        print('Converting from Celsius to Fahrenheit')
        print('{}C is {}F'.format(b,int(fahr)))
    elif a == 'F':
        celsius=(b-32)*5/9
        print('Converting from Fahrenheit to Celsius')
        print('{}F is {}C'.format(b,int(celsius)))


temp_converter('C',25)


temp_converter('F',101)
```

Further Resources:
Data Carpentry: https://datacarpentry.org/
Software Carpentry: https://software-carpentry.org/

Contact Info:

Phillip Doehle: doehle@okstate.edu

Kay Bjornen: kay.bjornen@okstate.edu