
Sign in here:

Hi NSBE! Please click on the link below and click REQUEST TO JOIN so that you're on our mailing list. This is where we'll share opportunities for FREE workshops and how to sign up for instructor training. We want you to teach with us! https://carpentries.topicbox.com/groups/equity-cohort

- - Jason, williams@cshl.edu
- Sarah
- Chris Etienne, christopher.etienne@prattwhitney.com, https://www.linkedin.com/in/christopher-etienne/
- Akin aadegoke@nd.edux
- Trey Lane, hlane2@nd.edux
- Darrel, JohnsonDarrelD@johndeere.comx
- Cameryn Battle, cameryn@gatech.edux
- Edashall(EJ),egk1806@hsutx.edux
- Santiago Rodriguez, srodri25@nd.edux
- Joshua Legis, joshua1.legis@famu.edux
- Tariq Hopkins, tariq1.hopkins@famu.edux
- Leigh Phan, leighphan@ucla.edu x
- Yaseen Anderson, yanders1@binghamton.edux
- John Akudike; jakudike@ttu.edux
- Jote Jinfessa, jyj5@pitt.edux
- Baye Gaspard bgaspard@umassd.edux
- Nia Wilson: wilsonni18@students.ecu.edux
- Marcel Amazan marcelamazan18@gmail.comx
- Kevin Paul kpaul3@umassd.edu, kpaulo949@gmail.com x
- Alice Abidoye aliceabid28@gmail.comx
- Selina Kwarteng kwarts@rpi.edux

Thank you for attending the Python workshop at NSBE! This mailing list is for receiving announcements of opportunities.

TODAY'S LESSON LINKS

WORKSHOP INFO PAGE: https://brownsarahm.github.io/2022-03-23-nsbe/

ALL SWC: https://software-carpentry.org/lessons/
TODAY'S PYTHON LESSON: https://swcarpentry.github.io/python-novice-gapminder/index.html
UNIX SHELL: https://swcarpentry.github.io/shell-novice/

UNIX LESSON

Setup data: https://swcarpentry.github.io/shell-novice/data/shell-lesson-data.zip
Using the command line to communicate with the computer

Commands:

# How to shorten command line path
```
$ export PS1 = '$ '
```

# To clear previous lines in command line
```
$ clear
```

# **Print Working Directory (pwd)** - your current directory/folder location
```
$ pwd
```

# **Change Directory (cd)**
```
$ cd
```

# **Change directory to the Desktop**
```
$ cd ~/Desktop
```

- Should look like '/Users/Williams/Desktop'

# **List Directory (ls)** - lists all items in your current directory
```
$ ls
```

# Change to the workshop data directory: 'shell-lesson-data'
```
$ cd ~/Desktop/shell-lesson-data
```

# Check to see if you're in the shell lesson folder
```
pwd
```

Exercise: What's the difference between the results for the following?

```
$ ls
$ ls -F
```

- exercise-data/
- north-pacific-gyre/

- -F adds a '/' at end of each pathname that is a directory, indicating that these are folders/directories
- Using a dash ('-') indicates an option

Getting help in the command line using built-in help pages

```
$ ls -help
```

- # Gives you the help page for the 'ls' command

```
$ man ls
```

- # lists the built-in help manual

# To exit the help -- 'q' to quit

```
$ q
```

# Exercises

Exploring Other Directories

```
$ ls -F Desktop
$ ls -F ~/Desktop/
```

Absolute & Relative Paths

```
$ pwd
/Users/williams/Desktop/shell-lesson-data
```

- Example of an Absolute Path

```
$ cd ~/Desktop/shell-lesson-data
```

- Example of a Relative Path
- ~ stands for your home directory

# To return to the directory level above your current location

```
$ cd ..
```

- This path is **relative** to your current location

How to see all commands you entered previously

```
$ history
```

More practice with the Shell

https://explainshell.com/

Python

Download the data:
lesson data: http://swcarpentry.github.io/python-novice-gapminder/index.html
Lesson ref: http://swcarpentry.github.io/python-novice-gapminder/reference

```
# Move the data folder to your Documents directory
$ mv /Downloads/python-novice-gapminder-data.zip /Documents

$ cd /Documents

# Unzip the data folder
$ unzip python python-novice-gapminder-data.zip
```

Installing Python using Anaconda
https://swcarpentry.github.io/python-novice-gapminder/setup.html#installing-python-using-anaconda

- Anaconda installs a collection of scientific packages that support scientific computing

Python does math

```
4+5
4 + 5   # gives same result
```

Creating variables in Python

```
name = 'sarah'
venue = "NSBE"    # Python accepts single or double spaces
```

To exit Python in the Command Prompt:

```
exit()    # works on all OS's
```

- short-cut in MacOS: ctrl+d

Working with Jupyter Notebook

Markdown cheatsheet: https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

Examples of interesting Jupyter Notebooks!

- https://gist.github.com/yuanzhaoYZ/b84db082be5e42acb65765c68c22b858
- **Introduction to Chemical Engineering Analysis** (using Jupyter Notebooks): https://jckantor.github.io/CBE20255/

- **Kerr nonlinearities**: https://nbviewer.ipython.org/github/jrjohansson/qutip-lectures/blob/master/Lecture-14-Kerr-nonlinearities.ipynb
- **The Sound of Hydrogen**: https://nbviewer.ipython.org/github/Carreau/posts/blob/master/07-the-sound-of-hydrogen.ipynb

- Create new notebook in upper-right hand corner -- select Python 3
- In your folder, a green icon will indicate that the notebook is currently running/open

In the notebook

- An active cell has a blue line on left side
- shift+enter to run a cell
- 'b' creates a new cell below
- Each time you run a cell, the count of a cell [1] will be tracked to the left of the cell
- When you re-start a notebook, the count will restart

Change format in the cells

- From drop-down menu: **Code**, **Markdown**
- Change cell to Markdown to type plain text along with your code. Helps make your notebook easier to read

```
# Python Basics
The first notebook of the day

Today's lesson covers:
- notebooks
- data analysis
- plotting
```

Include a link in your notebook:
[The name of the link**](**https://swcarpentry.github.io/python-novice-gapminder/02-variables/index.html**)**

https://swcarpentry.github.io/python-novice-gapminder/02-variables/index.html

Variables & Assignment

** NOTE: Avoid naming variables using the names of Python operators

```
name = 'sarah'
sentence = 'sarah's name is'
last_name = 'Brown'
```

Pro Tip: ** Use TAB to auto-complete **

```
print(name)
age = 34
age + 5

full_name = name + last_name
'sarahBrown'

[1,3,4]*4
```

- [1, 3, 4, 1, 3, 4, 1, 3, 4, 1, 3, 4]

```
name[1:3]
name[0:4]
'sara'
name[-1]
'h'

my_string = 'NSBE 48 is in Anaheim'
```

Exercise: How can you print out only the word 'Anaheim' (7 characters)?

```
my_string[-7:]
```

# this tells Python to start from the 7th letter from the end : to the end of the string

Why Jupyter Notebook?

* Increasingly helpful when working with data
* Can save & re-open the notebook and re-run
* All code and results will remain in notebook

Adding notes to your notebook using Markdown

## Notes so far

Working with the built-in Python print function

```
print('before')
print()
print('after)
```

Getting Help in Jupyter Notebooks

While in a cell, **shift+tab** will display a help window

```
print()
```

In a cell with 'print()', hit shift+tab

print?   <-- adding a '?' at end of a function will return help page

Exercise: Print out what we saw above using only one print call

```
print('before', 'after', sep ='\n')
before
after

print('before', 'after', sep ='\n\n')
before

after
```

```
my_string = 'Hello world!'

len(my_string)
12

my_string.upper()
'HELLO WORLD!'

my_string.swapcase()
'hELLO WORLD!'

my_string.isupper()
False

my_string.upper().isupper()    # assigns uppercase to the string, then queries whether the string is in uppercase
True

remaing = 100-age
remaing
66
```

Python Libraries

To load libraries in Python:

```
import math
math.cos(2*math.pi)
```

Can import parts of a library

- * Saves time to avoid loading huge libraries
- * Helpful to load all libraries at top of the notebook
- * In Python its helpful to abbreviate the name of a library

```
from math import cos, pi
import math as m
```

Put the following lines in order, then fill in blanks:

```
bases="ACTTGCTTGAC"
import math
import random
___ = random.randrange(n_bases)
___ = len(bases)
print("random base ", bases[___], "base index", ___)
```

Python can only give you more information about a library if you import it first.

Working with Tabular Data using Pandas

```
import pandas as pd
data = pd.read_csv('gapminder_gdp_oceania.csv')
data = pd.read_csv('gapminder_gdp_oceania.csv', index_col='country')
```

- # setting the index_col to 'country' will display the data by years based on the country

```
print(data)
```

Getting more info on the data frame

```
data.info()

print(data.columns)
```

Use ( ) for actions
Use [ ] for peeking inside a dataframe

# transpose the data frame

```
print(data.T)
```

print is a built-in Python function which is designed to turn results into text. Entering the function into Jupyter directly will display the data. See the difference below:

# describe the data

```
print(data.describe())
data.describe()
```

Exercises

1. Read the data in gapminder_gdp_americas.csv (which should be in the same directory as gapminder_gdp_oceania.csv) into a variable called americas and display its summary statistics.

2. After reading the data for the Americas, use help(americas.head) and help(americas.tail) to find out what DataFrame.head and DataFrame.tail do.

- What method call will display the first three rows of this data?
- What method call will display the last three columns of this data? (Hint: you may need to change your view of the data.)

Pandas DataFrames: Indexing & Slicing

We'll go over how to select individual values in a dataframe

Read in the Europe data file
** Install all libraries you plan to use at beginning of notebook

```
import pandas as pd
```

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

# **iloc** = **'index location'** -- lets you get the value of a specific cell in a dataframe
```
print(data.iloc[0,0])
```

# get data by name of column
```
print(data.loc["Albania", "gdpPercap_1952"])
```

# To verify Albania is first on our list, use 'head' to see the first 10 rows of the dataframe:
```
data.head()
```

# Get all columns
```
print(data.loc["Albania",:])     # Print all columns, from Albania to end
```

# select multiple columns or rows
```
print(data.loc['Italy':'Poland', 'gdpPercap_1962':'gdpPercap_1972'])
```

- List values for countries between Italy and Poland, between 1962-1972
- When indexing, Pandas is inclusive of the last item (e.g. includes Poland) vs. Python which would exclude the last item

# get the maximum GDP for a subset of countries year
```
print(data.loc["Italy":"Poland", "gdpPercap_1962":"gdpPercap_1972"].max())
```

# Creating a boolean mask
```
subset = data.loc["Italy":"Poland", "gdpPercap_1962":"gdpPercap_1962"]

mask = subset > 10000
(print(subset[mask]))
```

# print only the values that fulfill a condition
```
print('\nWhere are values large?\n', subset > 10000)
```

# do math using masked data
```
print(subset(subset > 10000).describe())
```

Group By: split-apply-combine

Grouping data and applying a specific method to different groups in the dataset.

```
mask_higher = data > data.mean()
wealth_score = mask_higher.aggregate('sum', axis=1) /len(data.columns) #axis=1
specificies by row
```

Exercises

1. Assume Pandas has been imported into your notebook and the Gapminder GDP data for Europe has been loaded:

- import pandas as pd
- df = pd.read_csv('data/gapminder_gdp_europe.csv', index_col='country')

Write an expression to find the Per Capita GDP of Serbia in 2007.

2. Extent of Slicing

======

Notes for Python workshop

https://brownsarahm.github.io/2022-03-23-nsbe-notes/nsbepython.html

Plotting

```
import pandas as pd
import matplotlib.pyplot as plt

time = [0,1,2,3]
position = [0,100,200,300]

plt.plot(time, position)

data = pd.read_csv('gapminder_gdp_ocean.csv', index_col = 'country')
data.head()
years = data.columns.str.strip('gdpPercap_')
years
# years are still in string format

# convert the data type of years to integers
data.columns = years.astype(int)
```

Exercise

Fill in the blanks to plot the minimum GDP per country in Europe:

```
data_europe = pd.read_csv('data/gapminder_gdp_europe.csv', index_col='country')
data_europe._____.plot(label='min')
data_europe._____
plt.legend(loc='best')
plt.xticks(rotation=90)
```

WHICH EPISODES WOULD YOU LIKE NEXT?
For Loops - x1
Conditionals - x1
Looping Over Data Sets x5
Writing Functions x1
Variable Scope

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

Looping Over Multiple Datasets

```
import pandas as pd
import glob
```

# create a list in our loop and iterate over the loop
Print out minimum GDP for each Africa and Asia

```
for filename in ['gapminder_gdp_africa.csv', 'gapminder_gdp_asia.csv']:
    data = pd.read_csv(filename, index_col = 'country')
    print(filename, data.min)


glob.glob('*.csv')
```

# make a list of the files except the `all` one
```
continent_files = glob.glob('*gdp*')
continent_files
```

# list the minimum GDP for each continent in 1952
```
for continent in continent_files:
    data = pd.read_csv(continent)
    print(continent, data['gdpPercap_1952'].min())
```

FEEDBACK: 1 UP & 1 DOWN

+ Learning curve was formatted; felt that went from knowing little about Python to being able to directly apply it
- Technology: my own technology was limiting; and a lot to need to install

+ Liked that everything was documented
+ Like the examples
- A lot to cover in 1 day; squeezing a lot into a day, would be better if into a few days; suggestion: sharing background/locations beforehand
+ Liked that everything is hands-on and in real-time
- Came in late; if there is a visual list of procedures to catch-up
+ Instructors and learners learn from mistakes
+ Learned about glob - very efficient way to import info
- Amount of material was a bit too much for one day; used UNIX shell in beginning but didn't return to it much during the Python lesson
+ Enjoyed learning about using glob library and now have Jupyter installed
- Since came later, it was hard to catch up
+ Joined late, but team was helpful in helping catch up with software, etc.