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 https://etherpad.wikimedia.org).

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

 ----

Python: Plotting and Programming Day 1

Workshop Details:

Zoom link for workshop: https://ucla.zoom.us/j/94365992905
Worshop website: http://swcarpentry.github.io/python-novice-gapminder/
This Etherpad (collaborative notetaking & link sharing document): https://pad.carpentries.org/2023-02-07-python
Etherpad for worshop series (with links to other etherpads):
Please follow the set up instructions for python here: http://swcarpentry.github.io/python-novice-gapminder/setup.html

Data Download:

Download and extract this file to your Desktop:
http://swcarpentry.github.io/python-novice-gapminder/files/python-novice-gapminder-data.zip

Pre-workshop survey:

Please take a minute to complete the pre-workshop survey

☆ Instructor:

Jamie Jamison / jamison@library.ucla.edu

☆ Helpers:

Name / email / social media handle @
Leigh Phan / leighphan@ucla.edu
Geno Sanchez/genosanchez@library.ucla.edu
Tim Dennis /tdennis@library.ucla.edu

✏ Guest Sign In:

Name / email / social media handle @
Kelsey Smith / kmsmith9990@gmail.com
Sahngyoon Rhee / srhee1998@gmail.com
Marvin Browne/ brownegm@gmail.com
Ananya Ravikumar/ananya.ravikumar@gmail.com
Love Patel/love.patel.2024@anderson.ucla.edu
Mohsin Ali / mohsinmalikali@library.ucla.edu

Schedule: Library Carpentry - Python

- **9:00 - Introductions**
- **9:15 - Jupyter Lab**

- http://swcarpentry.github.io/python-novice-gapminder/01-run-quit/index.html

Markdown (if there is time)

- http://swcarpentry.github.io/python-novice-gapminder/01-run-quit/index.html#the-notebook-will-turn-markdown-into-pretty-printed-documentation

- **9:30 - Variables and types**

- http://swcarpentry.github.io/python-novice-gapminder/02-variables/index.html#plotting-and-programming-in-python

- **Challenge**

- Slicing challenge
- given: species_name = "Acacia buxifolia"
- What does this print out:
    - species_name[2:8]
    - species_name[11:] (without a value after the colon)
    - species_name[:4] (without a value before the colon)
    - species_name[:] (just a colon)
    - species_name[11:-3]

        - species_name[-5:-3]
        - What happens when you choose a stop value which is out of range? (i.e., try species_name[0:20] or species_name[:103])

- http://swcarpentry.github.io/python-novice-gapminder/03-types-conversion/index.html#plotting-and-programming-in-python
        - **Discuss**
    - **What type of value (integer, floating point number, or character string) would you use to represent each of the following? Try to come up with more than one good answer for each problem.**
    - For example, in # 1, when would counting days with a floating point variable make more sense than using an integer?
    - Number of days since the start of the year.
    - Time elapsed from the start of the year until now in days.
    - Serial number of a piece of lab equipment.
    - A lab specimen's age
    - Current population of a city.
    - Average population of a city over time.

    - Integer, since the number of days would lie between 1 and 365.
    - Floating point, since fractional days are required
    - Character string if serial number contains letters and numbers, otherwise integer if the serial number consists only of numerals
    - This will vary! How do you define a specimen's age? whole days since collection (integer)? date and time (string)?
    - Choose floating point to represent population as large aggregates (eg millions), or integer to represent population in units of individuals.
    - Floating point number, since an average is likely to have a fractional part.

        - **10:00 - Break**

        - **10:20 - Libraries and built in functions**
    - http://swcarpentry.github.io/python-novice-gapminder/04-built-in/index.html#plotting-and-programming-in-python
    - http://swcarpentry.github.io/python-novice-gapminder/06-libraries/index.html#plotting-and-programming-in-python
        - **10:45 - Break (10 minutes)**
        - **10:55 - Reading Tabular Data into Data Frames**
    - http://swcarpentry.github.io/python-novice-gapminder/07-reading-tabular/index.html#plotting-and-programming-in-python

1. **11:50 - Pandas and Data Frames**

    - http://swcarpentry.github.io/python-novice-gapminder/08-data-frames/index.html#plotting-and-programming-in-python

- **12:15 - Break / Challenge**
- **12:30 - Review / Questions**
- 12:45 - End / Feedback

Notes

Starting

- Anaconda Navigator
- Command line
    - MacOS/Linux open terminal, start from command prompt: jupyter lab
    - Windows using Anaconda Prompt: jupyter lab

NOTE: Where Jupyter lab opens/starts

Bring this up so you will know where to look for your work

1. Jupyter Lab (and Jupyter Noebook) will open up in your home directory.C:\Users\USERNAME.
2. In my case, windows 10:   c:\Users\jamie
3. Or go to the directory that you want to work in and run ‘jupyter lab’ from there
4. for demo: D:\work-related\carpentries-python-day-1, type **jupyter lab**

How to shut down

Log Out or Shut Down
Shut Down actually shuts down the entire service
Log Out shut your Jupyter Lab but the server is still running

Create a new Jupyter Lab notebook:
File > New > Notebook > Select Python3 (ipykernel)

Jupyter Cheatsheet (shortcuts)
https://docs.anaconda.com/_downloads/3b72688842bacce05895ced585bc9f40/Jupyter-cheatsheet.pdf

Getting started with code cells

Set a line or 'cell' in Markdown mode to display text:

- ## markdown
- 1.
- 2.
- 3.
- one
- two
- three
- ### level 2
- #### level 3

Shortcuts

- - While in any code cell, **ESC** to switch between '**m**' and '**c**' for Markdown and Code
    - - Cell border will turn **blue in Edit mode** and **grey in Command mode**
    - - Use ESC to switch between these
- - SHIFT + ENTER to run a code cell

Set a cell in Code mode to display code:

```
2 * 3, 4 + 9
```

Variables and Assignment

```
age = 42
first_name = "Ahmed"

print(first_name, 'is', age, 'years old')

print(last_name)
```

In Python, you can't print anything (e.g. a variable) until it exists; must create it before using it. last_name would not work since we haven't created it.

```
age = age + 3
age
```

Strings are a sequence of characters, and we can access a part of a string.

```
atom_name = 'helium'
atom_name
print(atom_name)
print(atom_name[1])
```

PRO-TIP: use **TAB** to auto-complete in Jupyter.

Python Indexing

In Python counting starts at zero (0).

We can change the value of a variable from 'helium' to 'sodium':

```
atom_name = 'sodium'
```

The following would return the first character, up to, but not including the 3rd character:

```
atom_name[0:3]
atom_name[0]
```

The following returns the last character:

```
atom_name[-1]
```

- **Challenge**
  - Slicing challenge
  - given:
    - species_name = "Acacia buxifolia"
  - What does this print out:
    - species_name[2:8]
      - Acacia buxifolia
    - species_name[11:] (without a value after the colon)
      - Acacia buxifolia
    - species_name[:4] (without a value before the colon)
      - Acacia buxifolia
    - species_name[:] (just a colon)
      - Acacia buxifolia
    - species_name[11:-3]
      - Acacia buxifolia
    - species_name[-5:-3]
      - Acacia buxifolia
    - What happens when you choose a stop value which is out of range? (i.e., try species_name[0:20] or species_name[:103])
      - Acacia buxifolia

```
atom_name = 'sodium'
```

Data types in Python

```
type(atom_name)

print(first_name + " " + age)

print(first_name + " " + str(age))

type(age)
```

Built-in Functions and Getting Help in Python

Built-in functions are also called **Python's standard library**

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

Built-in function example: Get the maximum value in a list of items

```
max('9', 'Q', 'f', 'c')

atom_name_isupper()
atom_name
atom_name_upper
```

Capitalize a variable

```
atom_name.capitalize()
```

Change variable to lower case

```
atom_name_upper.lower()
atom_name_upper.islower()
```

Python Libraries

To use a Python **package, or library**, we need to use the **import** function:

Example: giving a library a shorter alias to avoid needing to repeat typing it out fully:

```
import math as m
```

Pandas Library (stats/tabular data)

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

pwd (Print working directly) will return the current folder you are in on your computer

```
pwd
```

Pandas function: Transpose -- flips the x-y axes of a table

```
data.T

data_transposed = data.T
```

Returns statistical summary of data

```
data.describe()

data.head()
data.tail()
```

Selecting Values

Use DataFrame.iloc[..., ...] to select values by their (entry) position

```
data2 = pd.read_csv('data/gapminder_gdp_europe.csv', index_col='country')
print(data2.iloc[0, 0])
1601.056136
```

You can also read in csv from the web, if you know the url:

data3 = pd.read_csv("https://raw.githubusercontent.com/mpicbg-scicomp/dlbc17-python-intro/master/data/gapminder_gdp_europe.csv", index_col= 'country')

```
data2.iloc[0,0]    # Location by index
1601.056136
```

```
data2.loc['Albania', 'gdpPercap_1952']
```
1601.056136

```
data2.head()
```

Returns all values for the entire row of Albania:
```
data2.loc['Albania',:]
```

Returns all values for GDP per capita for 1952 - all countries:
```
data2.loc[:, 'gdpPercap_1952']
```

Remember that we use colons (' : ') in Python indexing to indicate our range:
```
data2.loc['Italy':'Poland', 'gdpPercap_1962':'gdpPercap_1972']
```

```
data_percap = data2.loc[:, 'gdpPercap_1952']
```
print(data_percap > 10000)

Masking a subset of a large dataset
```
mask_higher = data2 > data2.mean()
```

Using Pandas aggregate function to create a wealth score based on the GDP data
```
wealth_score = mask_higher.aggregate('sum', axis=1) / len(data.columns)
```

```
wealth_score
```

```
country
Albania        0.000000
Austria       0.923077
```

```
Belgium                   0.923077
Bosnia and Herzegovina    0.000000
Bulgaria                  0.000000
dtype: float64
```

```
data.groupby(wealth_score).sum()
```

%whos is a function in Jupyter which displays all variables created in a workspace
can be run without '%' at the beginning too

Questions

can you please explain how max('9','Q','f','c') gave out ‘f’?
Based on Python sorting order or precendence (0-9, A-Z, a-z) 'f' would be the 'max' value.

Day 2 Survey - please let us know how it went! https://forms.gle/B5bbxj7rfpBEKo4AA

Day 2: Plotting, Lists, Loops, Funtions

ZOOM: https://ucla.zoom.us/j/97650739475
Workshop website: https://ucla-data-science-center.github.io/2023-02-06-ucla/

As a land grant institution, the Carpentries at UCLA acknowledges the Gabrielino/Tongva peoples as the traditional land caretakers of Tovaangar To-VAA-ngar (Los Angeles basin, So. Channel Islands).

Sign-in (please sign in)

Name / email / pronouns / favorite current or anytime show/podcast/exhibit (optional)
Tim Dennis / tdennis@library.ucla.edu / he/him / last of us
Leigh Phan / leighphan@library.ucla.edu / she/her
Alexander Farfan/asfarfan1@g.ucla.edu/he/him/ Attack on Titan
Jamie Jamison / jamison@library.ucla.edu / Sylvia Heyden documentry 'Weaverly Life'
Xiaofu Qiao/xiaofuqiao@g.ucla.edu/she/her/Go for Happiness
Geno Sanchez / genosanchez@library.ucla.edu/he:him/Chainsaw Man
Ananya Ravikumar/ananya.ravikumar@gmail.com/Modern Family
Love Patel/lp344@g.ucla.edu
Mohsin Ali / mohsinmalikali@library.ucla.edu

Review

1. What question are there from yesterday?
2. finding out your current directory or folder in Jupyter
3. navigating to your data -- wherever that might be
    1. using unix commands in the Notebook
4. some gotchas with smart quotes, something to be aware of copying and pasting b/t systems

5. reading in data from a url for fun
6. **Uses for Python**
    1. Python has a larger user community and developer community, and in turn has many libraries you can use for your work.
    2. Uses/libraries available for: text processing, data cleaning, data visualization, machine learning, artificial intelligence
    3. https://wesmckinney.com/book/

Notes:

Python for Data Anlysis / Wes McKinney: https://wesmckinney.com/book/

Running OS commands:
MacOS: !pwd
Windows: %pwd   <- sometimes called 'magic commands', to see all of these
%lsmagic

Plotting

Notes: (help us take notes!)

Plotting in Python

```
data2 = pd.read_csv('data/gapminder_gdp_oceania.csv')

data3 = pd.read_csv("https://raw.githubusercontent.com/mpicbg-scicomp/dlbc17-python-intro/master/data/gapminder_gdp_europe.csv", index_col= 'country')


import pandas as pd

import matplotlib.pyplot as plt
```

Once importing a library, can explore its available functions with '.' -- Python will show a preview dropdown:

```
plt.

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

data.columns
```

```
Index(['gdpPercap_1952', 'gdpPercap_1957', 'gdpPercap_1962', 'gdpPercap_1967',
       'gdpPercap_1972', 'gdpPercap_1977', 'gdpPercap_1982', 'gdpPercap_1987',
       'gdpPercap_1992', 'gdpPercap_1997', 'gdpPercap_2002', 'gdpPercap_2007'],
      dtype='object')
```

brackets indicate a list, example [1,2,3]

Rename the columns to the year
Columns are strings so they can be manipulated, remove 'gdpPercap_' and just leave the year
data.columns.str.replace('gdpPercap_', '')

```
data.columns.str.replace('gdpPercap_', '')

years = data.columns.str.replace('gdpPercap_', '')

data.columns = years.astype(int)

data.columns
Int64Index([1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, 2002,
            2007],
           dtype='int64')

data.loc['Australia']

1952    10039.59564
1957    10949.64959
1962    12217.22686
1967    14526.12465
1972    16788.62948
1977    18334.19751
1982    19477.00928
1987    21888.88903
1992    23424.76683
1997    26997.93657
2002    30687.75473
2007    34435.36744
Name: Australia, dtype: float64
```

Plot it!

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

data.T

data.T.plot()
plt.ylabel('GDP per capita')
plt.xlabel('Years')

# changing the plot style and kind
plt.style.use('ggplot')
```

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

* Plotting from matplotlib directly
* command for this is `plt.plot(x.y)`

```
gdp_australia = data.loc['Australia'] # pull out Australia

gdp_australia

1952    10039.59564
1957    10949.64959
1962    12217.22686
1967    14526.12465
1972    16788.62948
1977    18334.19751
1982    19477.00928
1987    21888.88903
1992    23424.76683
1997    26997.93657
2002    30687.75473
2007    34435.36744
Name: Australia, dtype: float64
```

Create a line chart with cross-markers at each year

```
plt.plot(years, gdp_australia, 'y-+')
```

learn more about plt

```
help(plt.plot)


# select two countries

gdp_australia = data.loc['Australia'] # pull out Australia
gdp_nz = data.loc['New Zealand']


# plot with different color markers

plt.plot(years, gdp_australia, 'b--', label = 'Australia')
plt.plot(years, gdp_nz, 'g--', label = 'New Zealand')


# create a legend

plt.legend(loc = 'upper left')
plt.xlabel('Year')
plt.ylabel('GDP per cap')
```

# scatter graph
plt.scatter(gdp_australia, gdp_nz)

Exercise 1: Minima and Maxima

Fill in the blanks below to plot the minimum GDP per capita over time for all the countries in Europe. Modify it again to plot the maximum GDP per capita over time for Europe.

data_europe = pd.read_csv('data/gapminder_gdp_europe.csv', index_col='country')
#read from url
#data_europe =
pd.read_csv('https://raw.githubusercontent.com/swcarpentry/python-novice-gapminder/gh-pages/data/gapminder_gdp_europe.csv', index_col='country')
data_europe.____.plot(label='min')
data_europe.____plt.legend(loc='best')
plt.xticks(rotation=90)

data_europe.min().plot(label='min')
data_europe.max().plot(legend='best')
plt.xticks(rotation=90)

Exercise 2: More Correlations

This short program creates a plot showing the correlation between GDP and life expectancy for 2007, normalizing marker size by population:

data_all = pd.read_csv('data/gapminder_all.csv', index_col='country')
#data_all = pd.read_csv('https://raw.githubusercontent.com/swcarpentry/python-novice-gapminder/gh-pages/data/gapminder_all.csv', index_col='country')
data_all.plot(kind='scatter', x='gdpPercap_2007', y='lifeExp_2007', s=data_all**['pop_2007']/1e6)**
# take pop_2007 and divide by 1,000,000

Using online help and other resources, explain what each argument to plot does.

Saving your plot to a file

plt.savefig('my_figure.png')

Will save the current figure to the file my_figure.png. The file format will automatically be deduced from the file name extension (other formats are pdf, ps, eps and svg).

Making your plots accessible

Whenever you are generating plots to go into a paper or a presentation, there are a few things you can do to make sure that everyone can understand your plots.

- Always make sure your text is large enough to read. Use the fontsize parameter in xlabel, ylabel, title, and legend, and tick_params with labelsize to increase the text size of the numbers on your axes.
- Similarly, you should make your graph elements easy to see. Use s to increase the size of your scatterplot markers and linewidth to increase the sizes of your plot lines.
- Using color (and nothing else) to distinguish between different plot elements will make your plots unreadable to anyone who is colorblind, or who happens to have a black-and-white office printer. For lines, the linestyle parameter lets you use different types of lines. For scatterplots, marker lets you change the shape of your points. If you're unsure about your colors, you can use Coblis https://www.color-blindness.com/coblis-color-blindness-simulator/ or Color Oracle https://colororacle.org/ to simulate what your plots would look like to those with colorblindness.

Examples of plots in matplotlib

- Many styles of plot are available: see the Python Graph Gallery https://python-graph-gallery.com/matplotlib/ for more options.

Break for 10min

Lists and Loops

Notes: (help us take notes!)

```
pressures = [0.273, 0.275, 0.277, 0.276]

pressures
[0.273, 0.275, 0.277, 0.276]

print('length:', len(pressures))
length: 4
```

Lists are mutable objects; can change and add to items in a list.

```
pressures[0:3]
[0.273, 0.275, 0.277]
```

# replace the 0th item with '2':

```
pressures[0] =  2

pressures
[2, 0.275, 0.277, 0.276]
```

# append '5' to the list:

```
pressures.append(5)

pressures
[2, 0.275, 0.277, 0.276, 5]
```

# delete the 4th item:

```
del(pressures[4])

pressures
[2, 0.275, 0.277, 0.276]


pressures[-1]
0.276
```

For Loops

For Loops are a way to handle repetitive tasks.

Structure of a for loop:

- line following a colon in a For Loop needs to be indented

```
for number in [2, 5, 4]:
    print(number)
```

```
2
5
4
```

```
#ranges
for number in range(0,3):
    print(number)
```

```
0
1
```

Accumulator pattern

```
total = 0
for number in range(10):          # counts from 0-10
    total = total + (number + 1)  # new value of total is replaced by total + (number
+ 1)
    print(total)
print(total)
```

```
original = 'tin'
# reverse the string
result = ''            # creates an empty list to store results of the for loop
for char in original:
    result = char + result
    print(char, result)
print(result)
```

Looping Through Datasets

```
import pandas as pd

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

use glob.glob to find a set of files

```
import glob


glob.glob('data/*.csv')

['data/gapminder_gdp_americas.csv',
'data/gapminder_gdp_europe.csv',
'data/gapminder_all.csv',
'data/gapminder_gdp_oceania.csv',
'data/gapminder_gdp_africa.csv',
'data/data_out.csv',
'data/gapminder_gdp_asia.csv']
```

```
# return a list of subsets of the min of each gdpPercap
for filename in glob.glob('data/gapminder_*.csv'):
    data = pd.read_csv(filename)
    print(filename, data['gdpPercap_1952'].min())
```

Exercise 1: Fill in the Blanks

Fill in the blanks so that the program below produces the output shown.

```
values = ____
values.____(1)
values.____(3)
values.____(5)
print('first time:', values)
values = values[____]
print('second time:', values)
```

Exercise 2: From Strings to Lists and Back

Given this:

```
print('string to list:', list('tin'))print('list to string:', ''.join(['g', 'o', 'l', 'd']))
```

1. What does list('some string') do?
2. What does '-'.join(['x', 'y', 'z']) generate?

Loops

Notes: (help us take notes!)

Exercise 1: Reversing a String

Fill in the blanks in the program below so that it prints “nit” (the reverse of the original character string “tin”).

```
original = "tin"
result = ____
```

```
for char in original:
    result = ____
print(result)
```

Exercise 2: Practice Accumulating

Fill in the blanks in each of the programs below to produce the indicated result.

```
# Total length of the strings in the list: ["red", "green", "blue"] => 12
total = 0
for word in ["red", "green", "blue"]:
    ____ = ____ + len(word)
print(total)
```

```
# List of word lengths: ["red", "green", "blue"] => [3, 5, 4]
lengths = ____
for word in ["red", "green", "blue"]:
    lengths.____(____)print(lengths)
```

```
# Concatenate all words: ["red", "green", "blue"] => "redgreenblue"
words = ["red", "green", "blue"]
result = ____
for ____ in ____:
    ____
print(result)
```

Break for 10min

Looping over Datasets

Notes: (help us take notes!)

Exercise 1: Determining Matches

Which of these files is *not* matched by the expression glob.glob('data/*as*.csv')?

1. data/gapminder_gdp_africa.csv
2. data/gapminder_gdp_americas.csv
3. data/gapminder_gdp_asia.csv

Exercise 2: Minimum File Size

Modify this program so that it prints the number of records in the file that has the fewest records.

```
import glob
import pandas as pd
fewest = ____
for filename in glob.glob('data/*.csv'):
    dataframe = pd.____(filename)
    fewest = min(____, dataframe.shape[0])
print('smallest file has', fewest, 'records')
```

Note that the DataFrame.shape() https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.shape.html method returns a tuple (a list like object) with the number of rows and columns of the data frame.

Functions

Notes: (help us take notes!)

A function can store information/capabilities -- until we call the function to use it:

```
def print_greeting():
    print("Hello!")
    print("the weather is cloudy")
    print("IKR?")

def print_date(year, month, day):
```

- joined = str(year) + '/' str(month) + str(day)
- print(joined)

```
print_date(month=3, day=19, year=1871)

def average(values):
    if len(values) == 0:
        return None        # if the length of values is zero, return None
    return sum(values) / len(values)
```

Exercise 1: Identifying Syntax Errors

1. Read the code below and try to identify what the errors are *without* running it.
2. Run the code and read the error message. Is it a SyntaxError or an IndentationError?
3. Fix the error.
4. Repeat steps 2 and 3 until you have fixed all the errors.

Exercise 2: Encapsulation

Fill in the blanks to create a function that takes a single filename as an argument, loads the data in the file named by the argument, and returns the minimum value in that data.

```
import pandas as pd
def min_in_data(____):
    data = ____
    return ____
```

Reflection exercise

Over break, reflect on and discuss the following:

- A common refrain in software engineering is “Don’t Repeat Yourself”. How do the techniques we’ve learned in the last lessons help us avoid repeating ourselves? *Note that in practice there is some nuance to this and should be balanced with doing the simplest thing that could possibly work.*

When would you consider turning a block of code into a function definition?

Resources

- Examples of Jupyter Notebooks: https://github.com/ml-tooling/best-of-jupyter
- Library Data Science Center support: https://www.library.ucla.edu/visit/locations/data-science-center/
- Download JupyterLab Desktop: https://github.com/jupyterlab/jupyterlab-desktop

Feedback form:

https://forms.gle/jyK62Xg6YktwBtnB8