- 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

---

Downloading Anaconda (see Python section)
https://gwu-libraries.github.io/2020-10-12-gwu/#setup

Gapminder Data Set
Please use this link to download the data to your own computer.
https://go.gwu.edu/gapminderdata

To learn more about the Gapminder Project, visit https://www.gapminder.org/

For viewing the GapMinder data in GitHub
https://github.com/gwu-libraries/gwlibraries-workshops/blob/master/python-datacarpentry/data/gapminder_all_cleaned.csv

GapMinder documentation: https://www.gapminder.org/data/documentation/

Introductions and instructions for breakout room.
Introduce yourselves!
Exploring the data

1. What does each row represent?
2. What does each column represent?
3. What kinds of analytical questions does this dataset let us pose?

Code from Day 1

```
country = 'Angola'
gdpPercap_1952 = 3520.610273
gdpPercap_2007 = 4797.231267
percent_change = (gdpPercap_2007 - gdpPercap_1952) / gdpPercap_1952 * 100
print('Percent change from 1952 to 2007 =', percent_change)

print(country == 'Algeria')
```

```
print(gdpPercap_1952 == 3520.61)

type(country)
type(gdpPercap_2007)

newValue = country + gdpPercap_1952

len(country)
len(gdpPercap_1952)

label = 'GDP per capita for ' + country + ' in 2007: ' + str(gdpPercap_1952)
print(label)
```

Exercise #1

Exercise on variables and calculations

1. Using variables and expressions, calculate the percentage change in life expectancy between 1952 and 2007 for another country in the CSV file.

2. Print a statement that summarizes your findings, including the name of the country.

Hint: Percent change between quantities x and y is calculated as
(y - x) / x.

Answers

```
country2 = "Ireland"
lifeExp1952 = 66.91
lifeExp2007 = 78.885

pChange = (lifeExp2007 - lifeExp1952) / lifeExp1952 * 100

statement = "The % change in life expectancy for " + country2 + "is: " + str(pChange)
print (statement)

amount = (lifeExp_2007-lifeExp_1952)/lifeExp_1952
statement = "the change in life expectancy is "+ str(round(amount, 3))
```

Loading the CSV file

```
with open('data/gapminder_all_cleaned.csv', 'r') as f:
    gap_data = f.read()

print(gap_data)
type(gap_data)

# To see the first character in the string assigned to the gap_data variable
print(gap_data[0])

# To see the third through fourth characters
print(country[2:4])
```

```
# To see the first 100 characters of gap_data
print(gap_data[:100])

# To break our long string into separate lines of text
# The split() method produces a list of strings out of a larger string
lines = gap_data.split('\n')

# Get the length of lines
len(lines)

# Print the first row
print('Header row:', lines[0])

# Print the next three rows
print('Next three rows:', lines[1:4])
```

Exercise on split() and lists

1. The first row of our dataset is a comma-separated string of column headers. Split this string on the commas to create a list of headers and assign it to a variable.

2. Print just the column headers in the header row that contain the column names for GDP per capita (the ones starting with gdpPercap).

```
headers = lines[0].split(',')
print(headers[2:14])
```

Looping through a list

```
for h in headers:
    print(h)

# To create a new list of lists
gap_tbl = []
for line in lines:
    row = line.split(',')
    gap_tbl.append(row)

# To access the first row
print('First row:', gap_tbl[0])

# To access a single element at a particular row, column position
print('Second row, third element:', gap_tbl[1][2])
```

Exercise on slicing and loops

1. Create a new list of just the rows for the countries in Africa. (There are 52 countries in Africa in the file.)
2. Using a for loop, create another list with just the GDP per capita values for African countries for the year 2007.

Hint: Assigning a slice of a list to a variable creates a new list.

```
gap_table=[]
for line in lines:
   row = line.split(",")
   if row[0] == "Africa":
      gap_table.append(row)

gap_Afrida_2007=[]
for line in gap_table:
         gap_Afrida_2007.append(line[13])
```

Lists vs Dictionaries

```
countries = {'country': 'Angola',
         'gdpPercap_1952': 3520.61}

countries['country']
countries['gdpPercap_1952']

# will not work:
countries[0]
countries['Angola']

countries['gdpPercap_2007'] = gdpPercap_2007
print(countries)

print(gap_tbl[0])
print(gap_tbl[100])

headers = gap_tbl[0]

for element in zip(headers, gap_tbl[1]):
   print(element)

row1_dict = dict(zip(headers, gap_tbl[1]))
row1_dict['continent']
gap_tbl[1][12]
```

Exercise #4 using zip() and dict()

Play around with using zip and dict to create dictionaries out of two lists.
For example, zip together a list of state abbreviations with the states' full names. Or zip together a list of months and a list of average temperatures.
Try the zip() procedure on two strings -- can you tell what happened?

```
gap_records = []
headers = gap_tbl[0]
for row in gap_tbl[1:]:
```

```
    record = zip(headers, row)
    record = dict(record)
    gap_records.append(record)

print("Country: ", gap_records[25]["country"], "Life Expectancy 2007", gap_records[25]['lifeExp_2007'])

record1 = gap_records[0]
print(record1['continent'] == 'Europe')

if record1['continent'] == 'Africa':
    print(record1['country'], 'is in Africa.')

gap_records = []
headers = gap_tbl[0]
for row in gap_tbl[1:]:
    if row[0]:
        record = zip(headers, row)
        record = dict(record)
        gap_records.append(record)

len(gap_records)
```

Help us make adjustments for tomorrow. Three questions:
https://forms.gle/98gkqeDJjU1oS9Py9

Day 2: Pandas and Plotting

Working with pandas and DataFrames

```
import pandas as pd
pd.read_csv("data/gapminder_all_cleaned.csv")
data = pd.read_csv("data/gapminder_all_cleaned.csv", index_col="country")
```

DataFrame functions, columns, and types

```
data.head()
type(data)
data.dtypes
data.columns
data.describe()
```

Indexing and slicing a DataFrame

```
data['lifeExp_2007']
data.lifeExp_2007

#data.lifeExp_2007.values
```

Looking at Data by Row

```
data.loc['Algeria']
data.iloc[10]  # providing row numeric index/position
data.loc['Algeria':'Zimbabwe']
data.loc['Angola', 'gdpPercap_2007']
data.iloc[0, 5]
```

Exercise #1

Create a new variable for Life Expectancy 1992

```
lifeExp1992 = data["lifeExp_1992"]
```

or

```
lifeExp1992 = data.lifeExp_1992
```

Get Data of New Zealand GDP per capital in 1962

```
gdpCap1962_NewZealand = data["gdpPercap_1962"]["New Zealand"]
```

or

```
gdpCap1962_NewZealand = data.gdpPercap_1962["New Zealand"]
```

or

```
gdpCap1962_NewZealand = data.loc["New Zealand", "gdpPercap_1962"]
```

or

```
data.tail(1)['gdpPercap_1962']
```

Subsetting both rows and columns

```
#data.loc [ row, column]

#Get data of Life expectancy from 1992 to 2007 for Algeria Zimbabwe
data.loc['Algeria':'Zimbabwe', 'lifeExp_1992':'lifeExp_2007']

#Get data for  Life expectancy from 1952 to 2007  only for Benin, Turkey, and Afghansitan
data.loc[['Benin', 'Turkey', 'Afghanistan'], 'lifeExp_1952':'lifeExp_2007']

rows=['Benin', 'Turkey', 'Afghanistan']
columns = ['lifeExp_1952', 'lifeExp_2007']
data.loc[rows, 'lifeExp_1952':'lifeExp_2007']
data.loc[rows, columns]
```

Using Boolean expressions to subset data

```
data['lifeExp_2007'] > 80

#Conditional subset only for countries with life expectancy higher than 80
high_lifeExp = data['lifeExp_2007'] > 80
data.loc[high_lifeExp]

#Conditional subset only for countries in Europe
data.loc[data['continent'] == 'Europe']
```

Exercise #2

```
data.loc[data.gdpPercap_2007 < 1000]
```

#Part 2

```
europe_life = data[data.continent=="Europe"].loc[:, 'lifeExp_1952':'lifeExp_2007'])
europe_life
```

or

```
europe_life = data.loc['Albania':'United Kingdom'].loc[:, 'lifeExp_1952':'lifeExp_2007'].copy()
```

- #Get value of maximum life expectancy from Europe 2007 subset

```
lifeExp_max = europe_life.max()
print(lifeExp_max)
```

- #Get country of maximum life expectancy from Europe 2007 subset

```
lifeExp_max_country = europe_life.idxmax()
print(lifeExp_max_country)
```

- #Merge them together

```
lifeExp_df = pd.concat([lifeExp_max,lifeExp_max_country],axis=1)
lifeExp_df
type(lifeExp_df)
```

- #Create subset of data just by GDPpercap

```
gdp_data = data.loc[:,'continent':'gdpPercap_2007'].copy()
```

- #Aggregate data

```
grouped = gdp_data.groupby('continent')
gdp_means = grouped.mean()
gdp_means
```

- #Strip out the gdpPercap from columns name

```
columns= gdp_means.columns.str.strip('gdpPercap_')
columns=columns.astype('int')
gdp_means.columns =  columns
gdp_means
```

Exercise 2

```
continent_data = data.continent
lifeExp_data = data.loc[:,'lifeExp_1952':'lifeExp_2007'].copy()
lifeExp_df = pd.concat([continent_data,lifeExp_data],axis=1)
lifeExp_grouped = lifeExp_df.groupby('continent')
```

or

```
grouped2 = data.groupby('continent')
ax = grouped2[['lifeExp_1952','lifeExp_2007']].std().plot(kind='bar')
ax.set_title('StdDev of Life Expectancy, 1952 and 2007')
```

```
ax.set_ylabel('stddev in years');

#Plotting with seaborn
import seaborn as sns
ax = sns.scatterplot(data=data,
                     x=data.gdpPercap_2007,
                     y=data.lifeExp_2007,
                     hue=data.continent,
                     size=data.pop_2007/1e6,
                     sizes=(100,1000),
                     alpha=0.75)
ax.legend(labelspacing=2)
ax.set_title('GDP per capita vs. life expectancy by country')
plt.rcParams['figure.figsize']=(10,10)


data.loc['Albania':'United Kingdom'].loc[:, 'lifeExp_1952':'lifeExp_2007'].copy()
```

printing three DataFrames from one cell:

```
dfs = [lifeExp_data, lifeExp_df, lifeExp_grouped]
for df in dfs:
```

- print(df

Resources

Safari Tech Books/O'Reilly Online: https://www.safaribooksonline.com/library/view/temporary-access

Link to Consultation: https://academiccommons.gwu.edu/writing-research-help

Shell. bash lesson
http://swcarpentry.github.io/shell-novice/
Git lesson
http://swcarpentry.github.io/git-novice/

Complete Python lessons (with additional content):
https://github.com/gwu-libraries/gwlibraries-workshops/tree/master/python-datacarpentry