- - - - - --- -
WEEK 4 - JULY 19
Collaboratory
Instructor: Vania
Lesson: Pandas DataFrames

How do you see previous commands in notebook? history

If you know what you're looking for, how would you find the index number (given a name how do you find the number?)
https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Index.get_loc.html: This gives you the indeix for the columns
data.columns.get_loc("gdpPercap_1952")

Given that you know the actual value of the min GDP is there a way to reference the column name?
https://stackoverflow.com/a/41403912

What does "<bound method DataFrame.(command) of (data)" mean? Any command given (eg. subset.median) will only return the actual data set.
Did this show up as an error message? What specific command produced this output.
This showed up in the rop right corner of the data set.

command: print(subset.median())
output:
```
<bound method DataFrame.median of             gdpPercap_1952  gdpPercap_1957
gdpPercap_1962 country                                       Italy          4931.404155
6248.656232     8243.582340 Montenegro      2647.585601     3682.259903     4649.593785
Netherlands     8941.571858    11276.193440    12790.849560 Norway        10095.421720
11653.973040    13450.401510 Poland         4029.329699     4734.253019     5338.752143>
```

When do you know to use () vs [] in a command or vice versa?
Think of the square brackets [ ] as a way to access specific values within a dataset, whereas the parameters within the parentheses ( ) specify additional parameters or inputs for functions.
Example: both iloc and loc access values within a DataFrame, and use square [ ] brackets.
Functions like .aggregate or .groupby use parentheses, because the specs you put inside do not specifically refer to data values.

Challenge question 2
Do the two statements below produce the same output?
2. Based on this, what rule governs what is included (or not) in numerical slices and named slices in Pandas?
[NAME]: RESPONSE
Carla: the .iloc will give a 2x2, the .loc will give a 3x3, this is because with lists, the value after the column is the number of data points out, not the location of the last data point
Salva iloc[0:2] counts the index up to 2 and if 2 is needed we should go one above
loc["name":"name" includes all the raw and columns. in numerical slices the upperbound is not included but it is inculuded in the string slices.
Michelle : The first does not include Belgium or 1962, while the second line does, due to the difference in indexing by label (inclusive) vs. by index (not inclusive).
My: upperbounds are exclusive in numerical slices, inclusive in named slices
Li: not the same. first give 2*2 matrix, the seond give 3*3 matrix. data.iloc[0:3,0:3] will result the same as the second.
Allison: No, the first one does not include the upper bound but the second one does

Menglin:not the same, first one doesnt include the upper bound

Nicole C: 1. two countries, 2 years. 2. 3 countries, 3 years. iloc includes the last.
Shambhavi: The first statement inclused only the first two rows and columns however the second statement includes three rows and columns. The numerical value 0:2 excludes the final index.
Cheyenne: No because when indexing numerically using iloc, you only index up to, but not including, the last value. To get the same thing, you would need to use iloc[0:3,0:3]
Zoe I: They won't be the same. the iloc statement would return values from Albania to Austria, as the rows 0:2 include "country, Albania and Austria", and the columns 0:2 include only "country, 1952,1957". The loc statement would return data drom Albania to Bulgaria, inclusive, which is one more row thatn above.

to use the indices you need to go one beyond what you're looking for (ie n+1 if you're looking for the value in row or column n)

Glenn: (1) they are not the same since (2) using numerical indexing is non incluse of the final index, but when using titles/names as indexes, they are inclusive kelly t: the first statement does not include belgium or 1962

Marcela: Not the same. With numbers the value from the upper bound is not included. Calling the row name does include the upper bound.

Challenge question 3
Please describe, line-by-line, what the following script does

```
first = pd.read_csv('data/gapminder_all.csv', index_col='country')
second = first[first['continent'] == 'Americas']
third = second.drop('Puerto Rico')
fourth = third.drop('continent', axis = 1)
fourth.to_csv('result.csv')
```

Michelle
(1)
```
first = pd.read_csv('data/gapminder_all.csv', index_col='country')
```
load the data to the variable first and gives the column data the label country
(2)
```
second = first[first['continent'] == 'Americas']
```
creates a subset (new data set) that only includes those with the Americas as their continent
(3)
```
third = second.drop('Puerto Rico')
```
drops the row of data from Puerto Rico from the data set
(4)
```
fourth = third.drop('continent', axis = 1)
```
drops the whole col continent from the data set
(5)
```
fourth.to_csv('result.csv')
```
writes the resultant data set (without continent and Puerto Rico) to a new csv called result

Carla
```
first = pd.read_csv('data/gapminder_all.csv', index_col='country')
```
make the variable "first" contain all the GDP data indexed by country
```
second = first[first['continent'] == 'Americas']
```
show only the American countries (only those with the "Americas" result in the "continent" column)
```
third = second.drop('Puerto Rico')
```
remove Puerto Rico from the data
```
fourth = third.drop('continent', axis = 1)
```
remove the "continent" column
```
fourth.to_csv('result.csv')
```

make a csv file of the new data frame in the working directory, titled result.csv

Marcela
first = pd.read_csv('data/gapminder_all.csv', index_col='country') #Data is loaded in the variable "first" and indexed by country
second = first[first['continent'] == 'Americas'] #Creates a subset "second" for countries in Americas
third = second.drop('Puerto Rico') #creates a subset "third" for countries in Americas exluding Puerto Rico
fourth = third.drop('continent', axis = 1)fourth.to_csv('result.csv')

Zoe:

(1) imports "all" Data, defined as variable 'first'
(2) selects rows whose continent is labeled "Americas" , defined as variable 'second'
(3) removes data from row with label "Puerto Rico", defined as variable "third"
(4) removed column displaying the continent of each country, defined as 'fourth'
(5) Writes data into a csv file.

LI:
first = pd.read_csv('data/gapminder_all.csv', index_col='country') # import the data set
second = first[first['continent'] == 'Americas'] # extract all the rows which the continent = Americas dimesion:25*37
third = second.drop('Puerto Rico') # remove the Puerto Rico row from second dimension 24*37
fourth = third.drop('continent', axis = 1) # remove continent column from third dimension 24*36
fourth.to_csv('result.csv') # save the fourth data set.

Allison:
first = pd.read_csv('data/gapminder_all.csv', index_col='country') #import dataset, index by country, assign all of it to the variable "first"
second = first[first['continent'] == 'Americas'] #creates subset of countries within the continent "Americas" within the "first" dataset and assigns the variable "second"
third = second.drop('Puerto Rico') #removes "Puerto Rico" data from the "second" subset and assigns this to "third"
fourth = third.drop('continent', axis = 1) #removes data listing the continent and save data as "fourth"
fourth.to_csv('result.csv') #save data in a csv file

Menglin
first import dataset including all countries and label country as first column
second select all country that belongs to continent which is Americas
third delete a row for puerto Rico

fourth delete the continent column
fifth save data into result.csv
kelly t
first = pd.read_csv('data/gapminder_all.csv', index_col='country') --- makes a variable called first that has imported the gapminder_all.csv data, changes the index of the row to the country name
second = first[first['continent'] == 'Americas']   -- makes a variable called second with countries from first that are all under Americas continent
third = second.drop('Puerto Rico') -- makes a variable called third that is the variable second with Puerto Rico removed
fourth = third.drop('continent', axis = 1 -- makes a variable called fourth is the variable third with continent removed
)fourth.to_csv('result.csv') -- creates a csv file of the variable fourth

Ashley
first = pd.read_csv('gapminder_all.csv', index_col='country') # reads in data from csv
second = first[first['continent'] == 'Americas'] # finds all data w/ continent = Americas
third = second.drop('Puerto Rico') # removes row "Puerto Rico" from dataset
fourth = third.drop('continent', axis = 1) # removes "continent" column
fourth.to_csv('result.csv') # saves modified data to csv file

Nicole
first = pd.read_csv('gapminder_all.csv', index_col='country') # organize all by country
second = first[first['continent'] == 'Americas'] # organize by country then by continent, specifically americas
third = second.drop('Puerto Rico') # puerto rico has been deleted from the data set produced in second fourth = third.drop('continent', axis = 1) # the continent label has been dropped from data produced in third fourth.to_csv('result.csv') # make a new data file after all the organization called "result.csv"

Cheyenne
# first = pd.read_csv('data/gapminder_all.csv', index_col='country') - read the csv file and index the columns as countries
# second = first[first['continent'] == 'Americas'] - add a continent column and populate it with countries fromt the Americas
# third = second.drop('Puerto Rico') - remove Puerto Rico row
# fourth = third.drop('continent', axis = 1) - remove continent column
# fourth.to_csv('result.csv') - save the new data set to a csv

Alyssa
First - imports the data so you can use it
Second - pulls only the data from countries that are from america

third - gets rid of puerto rico in the data set
fourth - gets rid of the contient column
fourth.to_cvs - turns the new set of data into a csv file

First: inports all the data with column country
Second: Can print the data only from America
Third: this one gets rid of Porto-rico
Fourth: This one get rid of continent column

second:

kelly t
1. data.loc[:,"gdpPercap_1982"]
2. data.loc["Denmark",:]