PHILOSOPHY AND HISTORY OF SCIENCE WITH COMPUTATIONAL MEANS

PROF. DR. GERD GRAßHOFF

Introduction to Pandas

We start by importing NumPy, which is the fundamental package for scientific computing. Then we import Pandas so that we can create Series and DataFrames (think of them as spreadsheets). So that we do not have to write each time "Series." and "pandas.DataFrame" we also import Series and DataFrame.

```
In [1]: import numpy as np
        import pandas as pd

        from pandas import Series, DataFrame
```

```
In [2]: presocratics = Series(['Thales of Miletus','Anaximander','Anaximenes','Pythagoras','Xenophanes',
                               'Heraclitus','Parmenides','Empedocles','Anaxagoras','Democritus'])
        presocratics
```

```
Out[2]: 0    Thales of Miletus
        1          Anaximander
        2           Anaximenes
        3           Pythagoras
        4           Xenophanes
        5           Heraclitus
        6           Parmenides
        7           Empedocles
        8           Anaxagoras
        9           Democritus
        dtype: object
```

Note that each value is indexed. We can change the index name (creating another series), and we can select series values.

```
In [3]: arche = Series(['water','apeiron','air','numbers','earth','logos','to be','four elements','seeds','atoms'],index=[presocratics])
        arche
```

```
Out[3]: Thales of Miletus            water
        Anaximander                apeiron
        Anaximenes                     air
        Pythagoras                 numbers
        Xenophanes                   earth
        Heraclitus                   logos
        Parmenides                   to be
        Empedocles           four elements
        Anaxagoras                   seeds
        Democritus                   atoms
        dtype: object
```

```
In [4]: arche['Democritus']
```

```
Out[4]: Democritus    atoms
        dtype: object
```

To create a DataFrame from scratch

The number points of each list must match (each column must have the same number of rows).

```
In [48]: presocratics_arche = {'Philosopher':['Thales of Miletus','Anaximander','Anaximenes','Pythagoras','Xenophanes','Heraclitus',
                                              'Parmenides','Empedocles','Anaxagoras','Democritus'],'Arche':['water','apeiron','air',
                                              'numbers','earth','logos','to be','four elements','seeds','atoms']}

In [49]: presocratics_frame = DataFrame(presocratics_arche)

In [50]: presocratics_frame
```

```
Out[50]:
          Philosopher          Arche
   0  Thales of Miletus        water
   1        Anaximander      apeiron
   2         Anaximenes          air
   3         Pythagoras      numbers
   4         Xenophanes        earth
   5         Heraclitus        logos
   6         Parmenides        to be
   7         Empedocles  four elements
   8         Anaxagoras        seeds
   9         Democritus        atoms
```

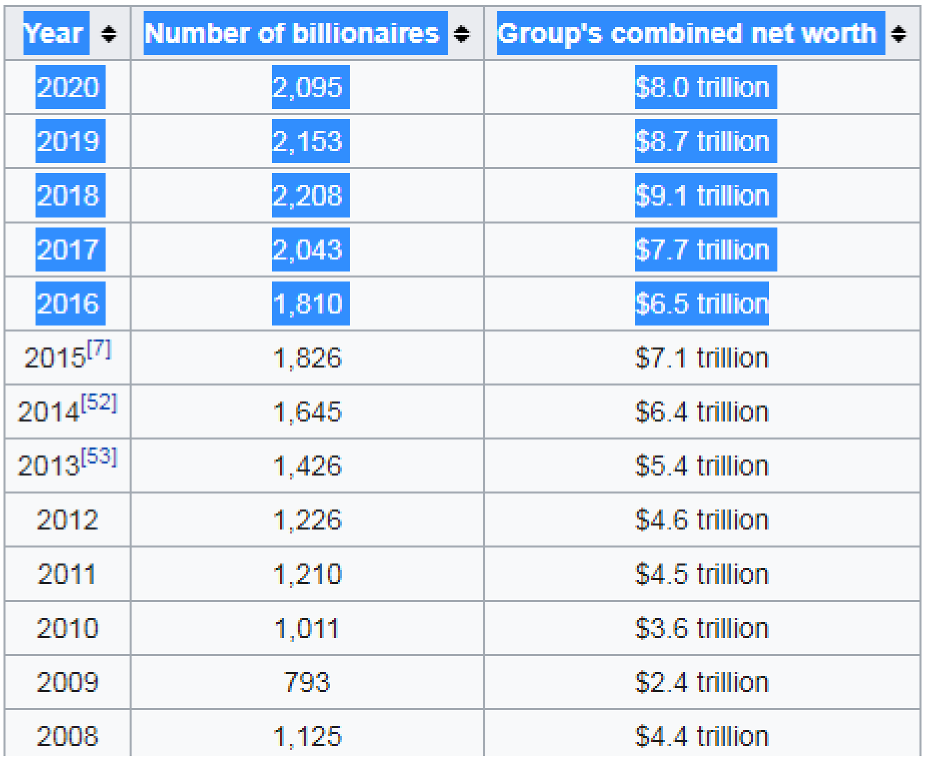
Description: ChooseName equal to open curly braces in quotes NameFirstColumn colon open square brackets in quotes the list of the names of the rows separated with a comma close square brackets comma in quotes NameSecondColumn colon open square brackets in quotes the list of the names of the rows separated with a comma close square brackets close curly braces.

To grab a DataFrame from a web page

1. You have to import webbrowser
2. Then create a name (in this case it was "website") equal to and in quotes your link
3. Use the function webbrowser.open(your_name)
4. Press Shift-Enter and the web page will open

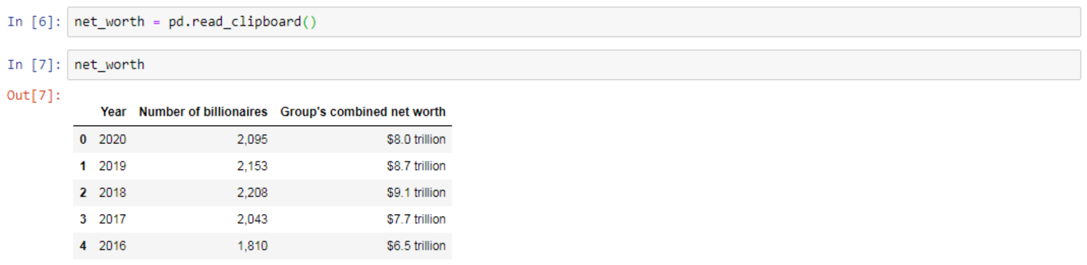
```
In [5]: import webbrowser
        website='https://en.wikipedia.org/wiki/The_World%27s_Billionaires'
        webbrowser.open(website)

Out[5]: True
```

5. Copy the frame (a table with rows and columns): highlight them and select Copy



Year	Number of billionaires	Group's combined net worth
2020	2,095	$8.0 trillion
2019	2,153	$8.7 trillion
2018	2,208	$9.1 trillion
2017	2,043	$7.7 trillion
2016	1,810	$6.5 trillion
2015[7]	1,826	$7.1 trillion
2014[52]	1,645	$6.4 trillion
2013[53]	1,426	$5.4 trillion
2012	1,226	$4.6 trillion
2011	1,210	$4.5 trillion
2010	1,011	$3.6 trillion
2009	793	$2.4 trillion
2008	1,125	$4.4 trillion

6. Choose another name for the DataFrame (in this case it was "net_worth") equal to pd.read_clipboard()
7. Press Shift-Enter to run the cell, write your chosen name and run the cell again.



```
In [6]: net_worth = pd.read_clipboard()

In [7]: net_worth
Out[7]:
```

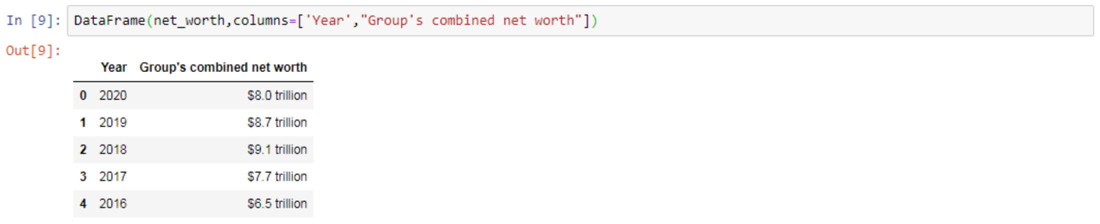
	Year	Number of billionaires	Group's combined net worth
0	2020	2,095	$8.0 trillion
1	2019	2,153	$8.7 trillion
2	2018	2,208	$9.1 trillion
3	2017	2,043	$7.7 trillion
4	2016	1,810	$6.5 trillion

Other functions

- To see the columns

```
NameOfDataFrame.columns
```

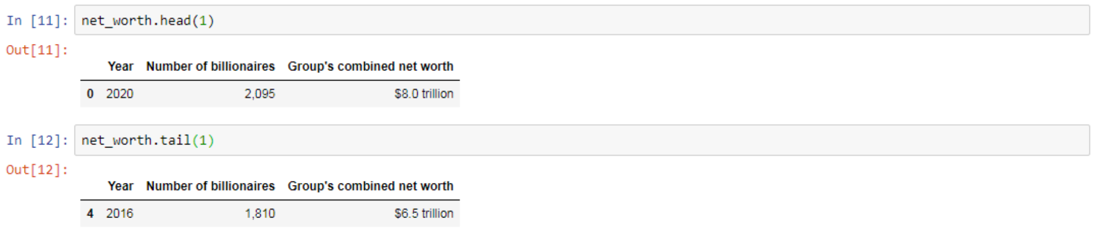
- To create DataFrame with selected columns



```
In [9]: DataFrame(net_worth,columns=['Year',"Group's combined net worth"])
Out[9]:
```

	Year	Group's combined net worth
0	2020	$8.0 trillion
1	2019	$8.7 trillion
2	2018	$9.1 trillion
3	2017	$7.7 trillion
4	2016	$6.5 trillion

- To get the first or the last row. If you want the first or last two, change the number 1 to 2 and so on



```
In [11]: net_worth.head(1)
Out[11]:
```

	Year	Number of billionaires	Group's combined net worth
0	2020	2,095	$8.0 trillion

```
In [12]: net_worth.tail(1)
Out[12]:
```

	Year	Number of billionaires	Group's combined net worth
4	2016	1,810	$6.5 trillion

- To get a specific row, for example, the first one:

```
NameOfDataFrame.ix[0]
```

  - 0 for the first row, 1 for the second, 2 for the third one and so on
- To see all the functions check the documentation:
  - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

Important to note

- In Python, indexing starts at 0.

```
Character: S c i e n c e
Index:     0 1 2 3 4 5 6
```

- To show a pop up of the methods on an object press on Tab
- To show help pop up for docstring press Shift-Tab
- To ignore code, put a hashtag at the beginning (in Code cell):

  ```
  # print("This will be ignored because of the hashtag")
  ```

- To comment, change to a Markdown cell
- We can mix single and double quotes, so that Python does not get confused:

  ```
  "I know that I don't know"
  ```

-