COMP4030 - Lab 1
Introduction to Jupyter Notebooks and Python for data manipulation

Johann Benerradi
Jeremie Clos

February 2, 2023

Because this is not a programming course, we expect that you have some programming background. Experience with Python is not strictly necessary because we will be using relatively basic features of the language, the most complex of which would be something like list comprehensions. The first lab is intentionally left easy so that students with little experience of Python can take the week to get familiar with the language.

There are many ways of learning Python, depending on your current background and your familiarity with C-type programming languages. There wouldn't be much interest in me copying a Python tutorial into this document, so I will provide useful links instead.

My personal recommendations are the following:

- Hackingscience.org provides a set of 50+ exercises that will get you up to speed on some basic programming in Python.
- The official Python tutorial is excellent and touches on most if not all features. The essentials sections are 1, 2, 3, 4, 5, 7, 10.5, 10.7
- The official documentation is extremely instructive and a good webpage to bookmark for future use.
- The LearnPython tutorial.
- Take COMP4008-PRG (MSc students only) or buy the book Conceptual Programming with Python by its lecturers.

Jupyter Notebooks, Anaconda, and the CS Virtual Desktop client If you are using your own machine but do not want to use Jupyter Notebooks locally you can make use of the university infrastructure to do the labs, provided you have a solid Internet connection. Follow the steps described on the University's website, for whichever OS your machine is running. Once the setup is complete you can launch the WVD ”Computer Science Desktop” from the list of workspaces.

If you want to run Jupyter Notebooks on your own local machine the easiest way is to download and install Anaconda at Anaconda.com, a scientific Python distribution.

1 Getting started with Jupyter Notebooks

Jupyter Notebooks are great for teaching and learning Python. They enable to run a Python environment with a graphical interface in your web browser to execute code blocks and write notes in

Markdown language.

You have two main options to run and create Jupyter Notebooks: - I like it simple → use the classic **Jupyter Notebook**: can be found in the CS Virtual Desktop in Anaconda, or can be launched directly from the Windows Start Menu. - I like to customise my developer tools → use **JupyterLab**: can be found in the CS Virtual Desktop in Anaconda

Open the lab's Jupyter Notebook file, **from this point you should be reading the rest of this lab sheet with Jupyter…**

When clicking in the left margin of the content you should notice that the file is composed of many cells, double click on them or hit return when a cell is selected to edit them.

Task 1.1: Get familiar with the interface and find the keyboard shortcuts to: - Create a cell below an existing cell - Create a cell above an existing cell - Delete a cell - Change the type of a cell into a Markdown cell (for note taking) - Change the type of a cell back into a code cell (to execute Python code) - Run a cell (you can run Markdown cells for formatting or code cells for execution)

(Feel free to edit the lab sheet to take notes and write your answers)

```
[ ]:
```

Jupyter Notebooks use the Markdown syntax to format text cells, giving you the opportunity to merge code and documentation into a single executable document. This is an extremely valuable tool for the data and machine learning scientist as it allows you to produce self-documenting computational experiments. You do not have to separately write and code, as you can mix both together. However, Jupyter Notebooks can also be dangerous because they allow you to execute code in a different order that the traditional linear fashion that traditional computer programs follow.

Task 1.2: Get familiar with Markdown and find the syntax to: - Create headers - Format text (bold/italic) - Insert links - Create block quotes - Create tables

```
[ ]:
```

Most of the libraries useful for this module should already be installed on the CS Virtual Desktop, however there is a way to install a library from a notebook should you need to:

```
[ ]: !pip install pandas
```

> **Note**: *Actually the Python interpreter runs within the command prompt, it is a command line interface within a command line interface. In Jupyter we interact with level -1 by default (the Python interpreter) but the* `!` *enables us to interact with the level above, level 0 (so the Windows Command Prompt).*

Important: when submitting work as Jupyter Notebook, always make sure to restart the kernel and re-execute all cells (fast forward button) to be sure that no variables from deleted cells were used.

2 Load data with Pandas

Obviously we first want to…

[2]:

```
import pandas as pd
```

Create a DataFrame:

[3]:

```
d = {'col1': [1, 2], 'col2': [3, 4]}
df = pd.DataFrame(data=d)
df
```

[3]:

```
   col1  col2
0     1     3
1     2     4
```

Create a DataFrame from a file:

[4]:

```
with open('countries.csv', 'r') as countries_file:
    headers = countries_file.readline().strip().split(',')
    d = {header: [] for header in headers}
    for line in countries_file:
        l = line.strip().split(',')
        for i, header in enumerate(headers):
            d[header].append(l[i])
print(d)
df = pd.DataFrame(data=d)
df
```

```
{'COUNTRY': ['China', 'India', 'US', 'Indonesia', 'Brazil', 'Pakistan',
'Nigeria', 'Bangladesh', 'Russia', 'Mexico', 'Japan', 'Germany', 'France', 'UK',
'Italy', 'Argentina', 'Algeria', 'Canada', 'Australia', 'Kazakhstan'], 'POP':
```

```
['1398.72', '1351.16', '329.74', '268.07', '210.32', '205.71', '200.96',
'167.09', '146.79', '126.58', '126.22', '83.02', '67.02', '66.44', '60.36',
'44.94', '43.38', '37.59', '25.47', '18.53'], 'AREA': ['9596.96', '3287.26',
'9833.52', '1910.93', '8515.77', '881.91', '923.77', '147.57', '17098.25',
'1964.38', '377.97', '357.11', '640.68', '242.5', '301.34', '2780.4', '2381.74',
'9984.67', '7692.02', '2724.9'], 'GDP': ['12234.78', '2575.67', '19485.39',
'1015.54', '2055.51', '302.14', '375.77', '245.63', '1530.75', '1158.23',
'4872.42', '3693.2', '2582.49', '2631.23', '1943.84', '637.49', '167.56',
'1647.12', '1408.68', '159.41'], 'CONT': ['Asia', 'Asia', 'N.America', 'Asia',
'S.America', 'Asia', 'Africa', 'Asia', '', 'N.America', 'Asia', 'Europe',
'Europe', 'Europe', 'Europe', 'S.America', 'Africa', 'N.America', 'Oceania',
'Asia']}
```

[4]:

```
       COUNTRY      POP      AREA       GDP       CONT
0        China  1398.72   9596.96  12234.78       Asia
1        India  1351.16   3287.26   2575.67       Asia
2           US   329.74   9833.52  19485.39  N.America
3    Indonesia   268.07   1910.93   1015.54       Asia
4       Brazil   210.32   8515.77   2055.51  S.America
5     Pakistan   205.71    881.91    302.14       Asia
6      Nigeria   200.96    923.77    375.77     Africa
7   Bangladesh   167.09    147.57    245.63       Asia
8       Russia   146.79  17098.25   1530.75
9       Mexico   126.58   1964.38   1158.23  N.America
10       Japan   126.22    377.97   4872.42       Asia
11     Germany    83.02    357.11    3693.2     Europe
12      France    67.02    640.68   2582.49     Europe
13          UK    66.44     242.5   2631.23     Europe
14       Italy    60.36    301.34   1943.84     Europe
15   Argentina    44.94    2780.4    637.49  S.America
16     Algeria    43.38   2381.74    167.56     Africa
17      Canada    37.59   9984.67   1647.12  N.America
18   Australia    25.47   7692.02   1408.68    Oceania
19  Kazakhstan    18.53    2724.9    159.41       Asia
```

Actually reinventing the wheel is maybe not that smart…

Task 2.1: There is a Pandas function to create a DataFrame from CSV files, find it in the online documentation and use it to create our DataFrame variable `df_countries` (the table should look the similar to the one above, you should see however that by default missing values are replaced by *NaN*). You may want to have a look at its different arguments for future reference (how to handle headers, different delimiters, …). Most Python packages have examples in their online documentation which can come in handy.

[ ]:

```

```

[6]:

```
df_countries
```

[6]:

```
       COUNTRY      POP      AREA       GDP       CONT
0        China  1398.72   9596.96  12234.78       Asia
1        India  1351.16   3287.26   2575.67       Asia
2           US   329.74   9833.52  19485.39  N.America
3    Indonesia   268.07   1910.93   1015.54       Asia
4       Brazil   210.32   8515.77   2055.51  S.America
5     Pakistan   205.71    881.91    302.14       Asia
6      Nigeria   200.96    923.77    375.77     Africa
7   Bangladesh   167.09    147.57    245.63       Asia
8       Russia   146.79  17098.25   1530.75        NaN
9       Mexico   126.58   1964.38   1158.23  N.America
10       Japan   126.22    377.97   4872.42       Asia
11     Germany    83.02    357.11   3693.20     Europe
12      France    67.02    640.68   2582.49     Europe
13          UK    66.44    242.50   2631.23     Europe
14       Italy    60.36    301.34   1943.84     Europe
15   Argentina    44.94   2780.40    637.49  S.America
16     Algeria    43.38   2381.74    167.56     Africa
17      Canada    37.59   9984.67   1647.12  N.America
18   Australia    25.47   7692.02   1408.68    Oceania
19  Kazakhstan    18.53   2724.90    159.41       Asia
```

3 Select data from tables with Pandas

The `head()` function enables to select the top of a table (this can be handy when working with big tables), by default the 5 first rows:

[7]:

```
df_head = df_countries.head()
df_head
```

[7]:

```
     COUNTRY      POP     AREA       GDP       CONT
0      China  1398.72  9596.96  12234.78       Asia
1      India  1351.16  3287.26   2575.67       Asia
2         US   329.74  9833.52  19485.39  N.America
3  Indonesia   268.07  1910.93   1015.54       Asia
4     Brazil   210.32  8515.77   2055.51  S.America
```

Select columns:

[8]:

```
df_head[['COUNTRY', 'GDP']]
```

[8]:

```
     COUNTRY       GDP
0      China  12234.78
1      India   2575.67
2         US  19485.39
3  Indonesia   1015.54
4     Brazil   2055.51
```

Select one column also called a *Series* object (*DataFrame* objects are composed of *Series* objects):

[9]:
```
df_head['POP']
```

[9]:
```
0    1398.72
1    1351.16
2     329.74
3     268.07
4     210.32
Name: POP, dtype: float64
```

Select rows from indices:

[10]:
```
df_countries.iloc[5:10]
```

[10]:
```
     COUNTRY     POP      AREA      GDP        CONT
5   Pakistan  205.71    881.91   302.14        Asia
6    Nigeria  200.96    923.77   375.77      Africa
7 Bangladesh  167.09    147.57   245.63        Asia
8     Russia  146.79  17098.25  1530.75         NaN
9     Mexico  126.58   1964.38  1158.23  N.America
```

Select rows based on a condition:

[11]:
```
df_biggest_countries = df_countries[df_countries['AREA'] > 1000] # rows for which the area is greater than 1000
df_biggest_countries
```

[11]:
```
       COUNTRY      POP      AREA       GDP        CONT
0        China  1398.72   9596.96  12234.78        Asia
1        India  1351.16   3287.26   2575.67        Asia
2           US   329.74   9833.52  19485.39   N.America
3    Indonesia   268.07   1910.93   1015.54        Asia
4       Brazil   210.32   8515.77   2055.51   S.America
8       Russia   146.79  17098.25   1530.75         NaN
9       Mexico   126.58   1964.38   1158.23   N.America
15   Argentina    44.94   2780.40    637.49   S.America
16     Algeria    43.38   2381.74    167.56      Africa
17      Canada    37.59   9984.67   1647.12   N.America
18   Australia    25.47   7692.02   1408.68     Oceania
19  Kazakhstan    18.53   2724.90    159.41        Asia
```

Select rows and columns based on a condition:

[12]:
```
df_countries.loc[df_countries['AREA'] > 1000, ['COUNTRY', 'POP']]
```

[12]:
```
       COUNTRY      POP
0        China  1398.72
1        India  1351.16
```

```
2          US   329.74
3   Indonesia   268.07
4      Brazil   210.32
8      Russia   146.79
9      Mexico   126.58
15  Argentina    44.94
16    Algeria    43.38
17     Canada    37.59
18  Australia    25.47
19 Kazakhstan    18.53
```

Select rows and columns **based on their names**:

```
[13]: df_countries_indices = df_countries.rename(index=df_countries['COUNTRY'])  # use␣
       ↪countries as row names
      df_countries_indices.loc[['UK', 'France'], ['COUNTRY', 'POP']]
```

```
[13]:        COUNTRY    POP
      UK          UK  66.44
      France  France  67.02
```

Select rows and columns **based on indices**:

```
[14]: df_countries.iloc[11:13, 0:2]
```

```
[14]:     COUNTRY    POP
      11  Germany  83.02
      12   France  67.02
```

Task 3.1: Use conditions to store the population of all the European countries of the dataset in a `european_pops` variable.

```
[ ]:
```

```
[16]: list(european_pops)
```

```
[16]: [83.02, 67.02, 66.44, 60.36]
```

4 Arrange tables with Pandas

Task 4.1: Find the Pandas function to sort the `df_countries` DataFrame by descending gross domestic product (GDP), and print only the 10 countries with highest GDP using the `head()` function. Store this sorted table in `df_sort_1`.

```
[ ]:
```

```
[18]: df_sort_1
```

[18]:

```
       COUNTRY      POP     AREA       GDP        CONT
2           US   329.74  9833.52  19485.39   N.America
0        China  1398.72  9596.96  12234.78        Asia
10       Japan   126.22   377.97   4872.42        Asia
11     Germany    83.02   357.11   3693.20      Europe
13          UK    66.44   242.50   2631.23      Europe
12      France    67.02   640.68   2582.49      Europe
1        India  1351.16  3287.26   2575.67        Asia
4       Brazil   210.32  8515.77   2055.51   S.America
14       Italy    60.36   301.34   1943.84      Europe
17      Canada    37.59  9984.67   1647.12   N.America
```

Task 4.2: Use the same Pandas sorting fonction to sort the `df_countries` DataFrame by continent's alphabetical order and descending area. Store this sorted table in `df_sort_2`. Take a note of how *NaN* values are handled, you can change this behaviour with the `na_position` argument.

[ ]:

```

```

[20]:

```
df_sort_2
```

[20]:

```
       COUNTRY      POP      AREA       GDP       CONT
16     Algeria    43.38   2381.74    167.56     Africa
6      Nigeria   200.96    923.77    375.77     Africa
0        China  1398.72   9596.96  12234.78       Asia
1        India  1351.16   3287.26   2575.67       Asia
19  Kazakhstan    18.53   2724.90    159.41       Asia
3    Indonesia   268.07   1910.93   1015.54       Asia
5     Pakistan   205.71    881.91    302.14       Asia
10       Japan   126.22    377.97   4872.42       Asia
7   Bangladesh   167.09    147.57    245.63       Asia
12      France    67.02    640.68   2582.49     Europe
11     Germany    83.02    357.11   3693.20     Europe
14       Italy    60.36    301.34   1943.84     Europe
13          UK    66.44    242.50   2631.23     Europe
17      Canada    37.59   9984.67   1647.12  N.America
2           US   329.74   9833.52  19485.39  N.America
9       Mexico   126.58   1964.38   1158.23  N.America
18   Australia    25.47   7692.02   1408.68    Oceania
4       Brazil   210.32   8515.77   2055.51  S.America
15   Argentina    44.94   2780.40    637.49  S.America
8       Russia   146.79  17098.25   1530.75        NaN
```

Assign new values using indices:

[21]:

```
df_countries_post_2020 = df_countries.copy()
df_countries_post_2020.loc['UK', 'CONT'] = 'Brexit'
df_countries_post_2020
```

```
[21]:        COUNTRY      POP      AREA       GDP       CONT
      0        China  1398.72   9596.96  12234.78       Asia
      1        India  1351.16   3287.26   2575.67       Asia
      2           US   329.74   9833.52  19485.39  N.America
      3    Indonesia   268.07   1910.93   1015.54       Asia
      4       Brazil   210.32   8515.77   2055.51  S.America
      5     Pakistan   205.71    881.91    302.14       Asia
      6      Nigeria   200.96    923.77    375.77     Africa
      7   Bangladesh   167.09    147.57    245.63       Asia
      8       Russia   146.79  17098.25   1530.75        NaN
      9       Mexico   126.58   1964.38   1158.23  N.America
      10       Japan   126.22    377.97   4872.42       Asia
      11     Germany    83.02    357.11   3693.20     Europe
      12      France    67.02    640.68   2582.49     Europe
      13          UK    66.44    242.50   2631.23     Europe
      14       Italy    60.36    301.34   1943.84     Europe
      15   Argentina    44.94   2780.40    637.49  S.America
      16     Algeria    43.38   2381.74    167.56     Africa
      17      Canada    37.59   9984.67   1647.12  N.America
      18   Australia    25.47   7692.02   1408.68    Oceania
      19  Kazakhstan    18.53   2724.90    159.41       Asia
      UK         NaN      NaN       NaN       NaN     Brexit
```

Create a new column from another column:

```
[22]: df_countries['AREA_SQUARE_KM'] = df_countries['AREA'] * 1000
      df_countries
```

```
[22]:        COUNTRY      POP      AREA       GDP       CONT  AREA_SQUARE_KM
      0        China  1398.72   9596.96  12234.78       Asia       9596960.0
      1        India  1351.16   3287.26   2575.67       Asia       3287260.0
      2           US   329.74   9833.52  19485.39  N.America       9833520.0
      3    Indonesia   268.07   1910.93   1015.54       Asia       1910930.0
      4       Brazil   210.32   8515.77   2055.51  S.America       8515770.0
      5     Pakistan   205.71    881.91    302.14       Asia        881910.0
      6      Nigeria   200.96    923.77    375.77     Africa        923770.0
      7   Bangladesh   167.09    147.57    245.63       Asia        147570.0
      8       Russia   146.79  17098.25   1530.75        NaN      17098250.0
      9       Mexico   126.58   1964.38   1158.23  N.America       1964380.0
      10       Japan   126.22    377.97   4872.42       Asia        377970.0
      11     Germany    83.02    357.11   3693.20     Europe        357110.0
      12      France    67.02    640.68   2582.49     Europe        640680.0
      13          UK    66.44    242.50   2631.23     Europe        242500.0
      14       Italy    60.36    301.34   1943.84     Europe        301340.0
      15   Argentina    44.94   2780.40    637.49  S.America       2780400.0
      16     Algeria    43.38   2381.74    167.56     Africa       2381740.0
      17      Canada    37.59   9984.67   1647.12  N.America       9984670.0
      18   Australia    25.47   7692.02   1408.68    Oceania       7692020.0
```

```
19  Kazakhstan    18.53   2724.90    159.41       Asia       2724900.0
```

Delete a column:

```
[23]: df_countries = df_countries.drop(columns=['AREA'])
      df_countries
```

```
[23]:        COUNTRY     POP       GDP       CONT  AREA_SQUARE_KM
      0        China  1398.72  12234.78       Asia       9596960.0
      1        India  1351.16   2575.67       Asia       3287260.0
      2           US   329.74  19485.39  N.America       9833520.0
      3    Indonesia   268.07   1015.54       Asia       1910930.0
      4       Brazil   210.32   2055.51  S.America       8515770.0
      5     Pakistan   205.71    302.14       Asia        881910.0
      6      Nigeria   200.96    375.77     Africa        923770.0
      7   Bangladesh   167.09    245.63       Asia        147570.0
      8       Russia   146.79   1530.75        NaN      17098250.0
      9       Mexico   126.58   1158.23  N.America       1964380.0
      10       Japan   126.22   4872.42       Asia        377970.0
      11     Germany    83.02   3693.20     Europe        357110.0
      12      France    67.02   2582.49     Europe        640680.0
      13          UK    66.44   2631.23     Europe        242500.0
      14       Italy    60.36   1943.84     Europe        301340.0
      15   Argentina    44.94    637.49  S.America       2780400.0
      16     Algeria    43.38    167.56     Africa       2381740.0
      17      Canada    37.59   1647.12  N.America       9984670.0
      18   Australia    25.47   1408.68    Oceania       7692020.0
      19  Kazakhstan    18.53    159.41       Asia       2724900.0
```

Task 4.3: Assigning new values can be useful in cases where we want to discretise quantitative data. Create a new column `SIZE` where the value is `Large` if the area of the country is greater than 1,000,000 square km and `Small` if less.

```
[ ]:
```

```
[25]: df_countries
```

```
[25]:        COUNTRY     POP       GDP       CONT  AREA_SQUARE_KM   SIZE
      0        China  1398.72  12234.78       Asia       9596960.0  Large
      1        India  1351.16   2575.67       Asia       3287260.0  Large
      2           US   329.74  19485.39  N.America       9833520.0  Large
      3    Indonesia   268.07   1015.54       Asia       1910930.0  Large
      4       Brazil   210.32   2055.51  S.America       8515770.0  Large
      5     Pakistan   205.71    302.14       Asia        881910.0  Small
      6      Nigeria   200.96    375.77     Africa        923770.0  Small
      7   Bangladesh   167.09    245.63       Asia        147570.0  Small
      8       Russia   146.79   1530.75        NaN      17098250.0  Large
      9       Mexico   126.58   1158.23  N.America       1964380.0  Large
```

```
10       Japan   126.22   4872.42       Asia       377970.0  Small
11     Germany    83.02   3693.20     Europe       357110.0  Small
12      France    67.02   2582.49     Europe       640680.0  Small
13          UK    66.44   2631.23     Europe       242500.0  Small
14       Italy    60.36   1943.84     Europe       301340.0  Small
15   Argentina    44.94    637.49  S.America      2780400.0  Large
16     Algeria    43.38    167.56     Africa      2381740.0  Large
17      Canada    37.59   1647.12  N.America      9984670.0  Large
18   Australia    25.47   1408.68    Oceania      7692020.0  Large
19  Kazakhstan    18.53    159.41       Asia      2724900.0  Large
```

Write table to a file:

```
[26]: df_countries.to_csv('countries_new.csv', index=False)
```

5 Combine tables with Pandas

Read the file in two halves:

```
[27]: df_1 = pd.read_csv('countries.csv', nrows=15)
      df_2 = pd.read_csv('countries.csv', skiprows=range(1,16))
```

```
[28]: print(df_1.shape)
      df_1
```

```
(15, 5)
```

```
[28]:       COUNTRY      POP      AREA       GDP       CONT
      0       China  1398.72   9596.96  12234.78       Asia
      1       India  1351.16   3287.26   2575.67       Asia
      2          US   329.74   9833.52  19485.39  N.America
      3   Indonesia   268.07   1910.93   1015.54       Asia
      4      Brazil   210.32   8515.77   2055.51  S.America
      5    Pakistan   205.71    881.91    302.14       Asia
      6     Nigeria   200.96    923.77    375.77     Africa
      7  Bangladesh   167.09    147.57    245.63       Asia
      8      Russia   146.79  17098.25   1530.75        NaN
      9      Mexico   126.58   1964.38   1158.23  N.America
      10      Japan   126.22    377.97   4872.42       Asia
      11    Germany    83.02    357.11   3693.20     Europe
      12     France    67.02    640.68   2582.49     Europe
      13         UK    66.44    242.50   2631.23     Europe
      14      Italy    60.36    301.34   1943.84     Europe
```

```
[29]: print(df_2.shape)
      df_2
```

```
(5, 5)
```

```
[29]:      COUNTRY    POP     AREA      GDP       CONT
      0  Argentina  44.94  2780.40   637.49  S.America
      1    Algeria  43.38  2381.74   167.56     Africa
      2     Canada  37.59  9984.67  1647.12  N.America
      3  Australia  25.47  7692.02  1408.68    Oceania
      4 Kazakhstan  18.53  2724.90   159.41       Asia
```

Combine tables vertically (add rows):

```
[30]: df = pd.concat([df_1, df_2], axis=0).reset_index(drop=True)
      df
```

```
[30]:        COUNTRY      POP      AREA       GDP       CONT
      0        China  1398.72   9596.96  12234.78       Asia
      1        India  1351.16   3287.26   2575.67       Asia
      2           US   329.74   9833.52  19485.39  N.America
      3    Indonesia   268.07   1910.93   1015.54       Asia
      4       Brazil   210.32   8515.77   2055.51  S.America
      5     Pakistan   205.71    881.91    302.14       Asia
      6      Nigeria   200.96    923.77    375.77     Africa
      7   Bangladesh   167.09    147.57    245.63       Asia
      8       Russia   146.79  17098.25   1530.75        NaN
      9       Mexico   126.58   1964.38   1158.23  N.America
      10       Japan   126.22    377.97   4872.42       Asia
      11     Germany    83.02    357.11   3693.20     Europe
      12      France    67.02    640.68   2582.49     Europe
      13          UK    66.44    242.50   2631.23     Europe
      14       Italy    60.36    301.34   1943.84     Europe
      15   Argentina    44.94   2780.40    637.49  S.America
      16     Algeria    43.38   2381.74    167.56     Africa
      17      Canada    37.59   9984.67   1647.12  N.America
      18   Australia    25.47   7692.02   1408.68    Oceania
      19  Kazakhstan    18.53   2724.90    159.41       Asia
```

Combine tables horizontally (add columns):

```
[31]: df_A = df[['COUNTRY', 'GDP']].sort_values(by=['GDP']).iloc[2:]
      df_A
```

```
[31]:        COUNTRY      GDP
      7   Bangladesh   245.63
      5     Pakistan   302.14
      6      Nigeria   375.77
      15   Argentina   637.49
      3    Indonesia  1015.54
      9       Mexico  1158.23
      18   Australia  1408.68
      8       Russia  1530.75
```

```
17       Canada   1647.12
14        Italy   1943.84
4        Brazil   2055.51
1         India   2575.67
12       France   2582.49
13           UK   2631.23
11      Germany   3693.20
10        Japan   4872.42
0         China  12234.78
2            US  19485.39
```

```
[32]: df_B = df_2[['COUNTRY', 'AREA', 'POP']]
      df_B
```

```
[32]:      COUNTRY     AREA    POP
      0   Argentina  2780.40  44.94
      1     Algeria  2381.74  43.38
      2      Canada  9984.67  37.59
      3   Australia  7692.02  25.47
      4  Kazakhstan  2724.90  18.53
```

```
[33]: pd.merge(df_A, df_B, on='COUNTRY', how='inner')
```

```
[33]:     COUNTRY      GDP     AREA    POP
      0  Argentina   637.49  2780.40  44.94
      1  Australia  1408.68  7692.02  25.47
      2     Canada  1647.12  9984.67  37.59
```

Task 5.1: have a look at documentation and try to understand the behaviour of the different possible values of the how argument of that merge() function.

```
[ ]:
```

6 Get basic statistics with Pandas

Summary statistics on the table:

```
[34]: df.describe()
```

```
[34]:                POP          AREA          GDP
      count    20.000000     20.000000     20.00000
      mean    248.905500   4082.182500   3036.14250
      std     394.546143   4706.507539   4706.00783
      min      18.530000    147.570000    159.41000
      25%      56.505000    575.002500    572.06000
      50%     126.400000   2173.060000   1588.93500
      75%     206.862500   7897.957500   2594.67500
      max    1398.720000  17098.250000  19485.39000
```

```
[35]: df.median(numeric_only=True)
```

```
[35]: POP       126.400
      AREA     2173.060
      GDP      1588.935
      dtype: float64
```

```
[36]: df['POP'].skew()
```

```
[36]: 2.6479537986199797
```

```
[37]: df['GDP'].kurtosis()
```

```
[37]: 8.356678932041959
```

Task 6.1: do some research on what all those statistics represent if you don't know about them.

```
[ ]:
```

7 Final tasks

Final task 1: Using the dataset of this lab, present in a new column of the table the density of population of every country in people per square meter.

```
[ ]:
```

Final task 2: What continent has the countries with the largest density of population on average?

```
[ ]:
```

Final task 3: Why is it however not scientifically valid to answer this previous research question with our dataset?

```
[ ]:
```

8 Open tasks

Tasks are more time-consuming and open-ended than exercises and allow you to go much deeper in the mastery of the content. We do not expect you to finish all of them in the lab, but they can be good toy problems to practice with.

Open task 1: Load the Penguin dataset and try to perform similar data manipulations as what we have seen on it.

```
[38]: # The Penguin dataset can be loaded with seaborn (data visualisation library for␣
       ↪Python)
      # We will use this library more in depth in later labs

      import seaborn as sns
```

```
df_penguins = sns.load_dataset('penguins')
df_penguins
```

```
[38]:    species     island  bill_length_mm  bill_depth_mm  flipper_length_mm  \
      0     Adelie  Torgersen            39.1           18.7              181.0
      1     Adelie  Torgersen            39.5           17.4              186.0
      2     Adelie  Torgersen            40.3           18.0              195.0
      3     Adelie  Torgersen             NaN            NaN                NaN
      4     Adelie  Torgersen            36.7           19.3              193.0
      ..       ...        ...             ...            ...                ...
      339   Gentoo     Biscoe             NaN            NaN                NaN
      340   Gentoo     Biscoe            46.8           14.3              215.0
      341   Gentoo     Biscoe            50.4           15.7              222.0
      342   Gentoo     Biscoe            45.2           14.8              212.0
      343   Gentoo     Biscoe            49.9           16.1              213.0

           body_mass_g     sex
      0         3750.0    Male
      1         3800.0  Female
      2         3250.0  Female
      3            NaN     NaN
      4         3450.0  Female
      ..           ...     ...
      339          NaN     NaN
      340       4850.0  Female
      341       5750.0    Male
      342       5200.0  Female
      343       5400.0    Male

      [344 rows x 7 columns]
```

[ ]:

Open task 2: Get familiar with the `pivot()` and `melt()` functions from Pandas.

[ ]: