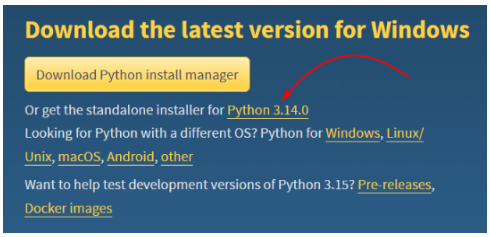
BSB - M1 Data Science : practicle

Setup your environment

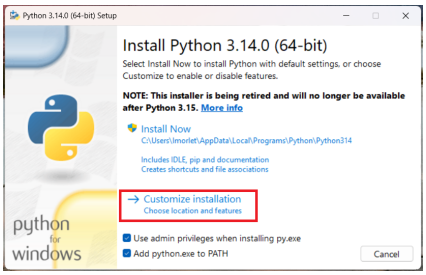
Before starting to use pandas, you will need to install Python on your computer, setup your Visual Studio Code to run it, and then download the required libraries.

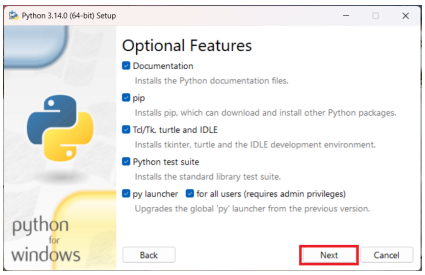
Install Python

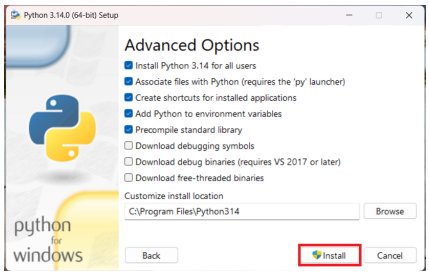
First of all, you will need to download the latest version of Python on your computer. Go to this URL : `https://www.python.org/downloads/` and click on the "standalone installer"



Double-click on the installer you have downloaded. On the following images, tick the checkboxes and then click on the button in the red rectangle





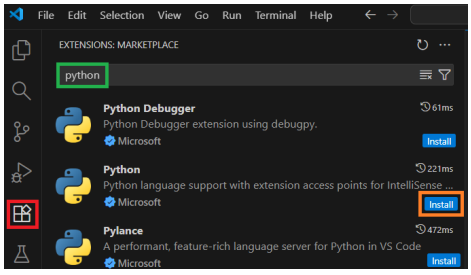


Install Visual Studio Code (VS Code)

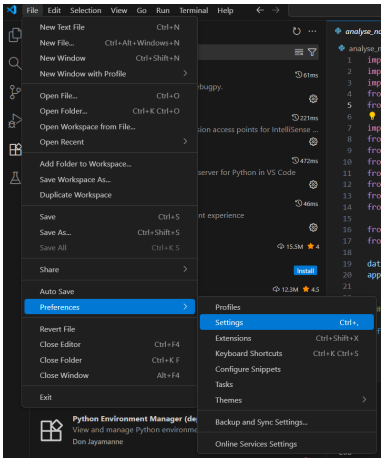
Once your Python interpreter is installed, you can go for the Visual Studio Code installer : `https://code.visualstudio.com/download` No specific instructions for it, just follow the recommended installation.

Link Visual Studio Code to Python

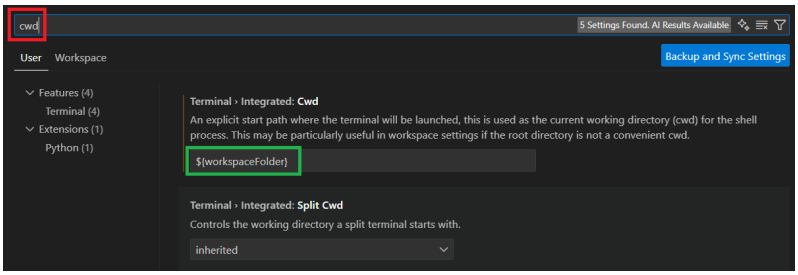
Run your VS Code. On the left panel, click on "Extension" (red rectangle in the following image). Then type "Python" in the search bar (green rectangle). To finish, click on "Install" (orange rectangle) for the Python extension proposed by Microsoft.



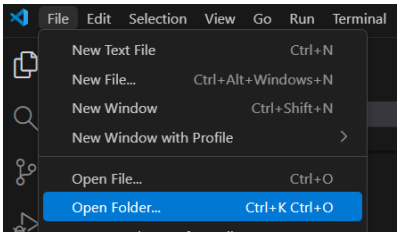
Open the ”Settings” by following the path presented in the following image



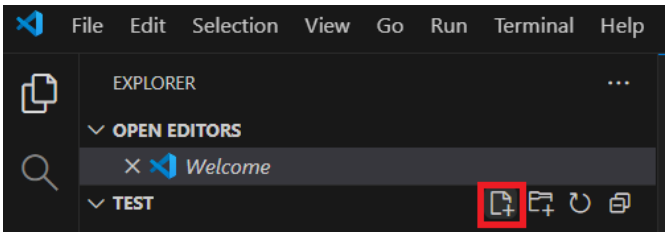
In the settings search bar (red rectangle in the following image), type ”cwd”. Then fill the ”Terminal integrated : Cwd” (green rectangle) with the value **${workspaceFolder}**



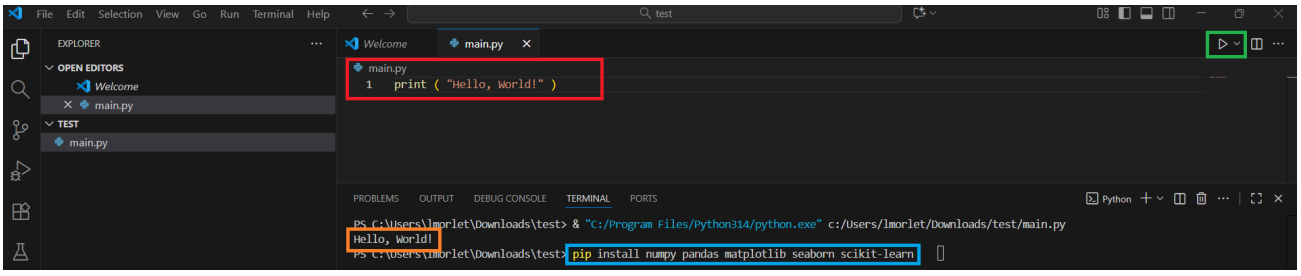
Open the folder where you want to store your script and data by following the path presented in the following image



In the file explorer, click on ”New file” (red rectangle in the following image) and name it ”main.py”



In the text editor (red rectangle in the following image), type **print (”Hello, World!”)**, then click on ”Run” (green rectangle). Your script will be executed and the result will be displayed inside the terminal (orange rectangle). In this terminal, type **pip install numpy pandas matplotlib seaborn scikit-learn** (cyan rectangle) to install all the required libraries.



Your VS Code is now ready!

Session nº 1 : First steps in Pandas

a) Start by downloading the CSV file ”burgundy_2023.csv” and place it in your workspace folder that you have opened in the previous part.

b) At the very beginning of your Python script, import the required libraries with :

```
import pandas as pd
import matplotlib.pyplot as plt
```

c) Then read the CSV and display its basic info

```
df = pd.read_csv('burgundy_2023.csv', sep=';')
print("Number of rows:", len(df))
print("Number of columns:", len(df.columns))
print("Column names:", df.columns)
```

d) How many rows and columns are in your CSV file ?

e) There are too many columns in this CSV, we need to check which ones are truly useful. Display the detailed info to find which columns are empty

```
print("Infos of dataframe")
df.info( verbose=True, show_counts=True)
```

f) Extract the useful columns

```
useful_cols = [ "NUM_POSTE", "NOM_USUEL", "LAT", "LON", "ALTI", "AAAAMMJJHH", "T" ]
df = df[useful_cols]
df.info(True, show_counts=True)
```

g) There are still missing values in your dataframe. To avoid this, you will delete every row that contains at least one missing value

```
df = df.dropna()
df = df.reset_index(drop=True)
df.info( verbose=True, show_counts=True )
```

h) Once your dataframe is ”clean”, let’s check what is inside of it. Display the cities where the data comes from. Why do we use the ”unique” function ?

```
print ( df["NOM_USUEL"].unique() )
```

i) By modifying the following code, create a sub-dataframe for each city.

```
dijon = df.loc[df["NOM_USUEL"] == "DIJON"]
dijon = dijon.reset_index(drop=True)
```

j) By modifying the following code, display the most common statistics about the temperature (”T”) of the studied cities.

```
print("Dijon temperature:")
print( dijon["T"].describe() )
```

k) What can you say about the statistics of the temperatures of the studied cities ?

l) Find a way to display what is inside the ”AAAAMMJJHH” column. Can you identify it ?

m) You will need to convert it to a suitable format before using it. Use the following code to do it.

```
dijon["AAAAMMJJHH"] = pd.to_datetime(dijon["AAAAMMJJHH"], format="%Y%m%d%H")
```

n) Look at the format parameter, can you find what signify the value here ?

o) Plot the evolution of the temperature of Dijon during 2023

```
# Plot temperature over time for each city
plt.plot( dijon["AAAAMMJJHH"], dijon["T"], label="Dijon")
plt.xlabel("Date")
plt.ylabel("Temperature (°C)")
plt.title("Temperature over Time")
plt.legend()
plt.show()
```

p) Modify your code to display the curve of every city in the same figure

Session nº 2 : cleaning

During the previous session, you have written your code in a single file containing everything. This way to create a script is convenient for a "one-shot" code, but when you are doing a project, you will need a more modular code that will be easier to maintain.

a) Start by importing inside your project the two files uploaded on the GitHub : *data_preparation.py* and *data_visualization.py*. These two scripts contain several functions that will help you during the practical and the project.

b) Create a new Python file where you will write the structure of your program. This file is called "main" because it will be the one you will run, and it is in charge of calling every other file and use the functions that are inside them. At the top of the file, write the following lines :

```
# My Python files
import data_preparation as prep
import data_visualization as viz

# Standard libraries
import pandas as pd
```

You already know that the "import pandas as pd" allow you to call pandas function. The two first lines work the same way, they allow you to use functions contained in *data_preparation.py* and *data_visualization.py* files that you have already import inside your project. If you want to use a function "foo" from *data_preparation.py*, you should call it **prep.foo** inside your main file. If you want to use a function "bar" from *data_visualization.py*, you should call it **viz.bar** inside your main file.

c) Make a first try by calling the function **load_dataframe** of *data_preparation.py* and give it the right parameter to load your CSV file *burgundy_2023.csv*.

d) Once this dataframe is loaded, extract the following columns with a call of a already written function :

- NUM_POSTE : the number of the weather station
- NOM_USUEL : the name of the weather station
- LAT : the latitude of the weather station
- LON : the longitude of the weather station
- ALTI : the altitude of the weather station
- AAAAMMJJHH : the date and hour of the record
- T : the mean temperature during this hour
- RR1 : the quantity of rain during this hour
- U : the relative humidity during this hour
- DG : the number of minutes with frost during this hour

e) Check that your dataframe contains every previously listed columns and 35040 entries (the number of lines in your CSV file)

f) Find the name of the cities included in this dataframe (you have already done it in the previous session).

g) For each city, extract the subdataframe that contains only the records concerning this city

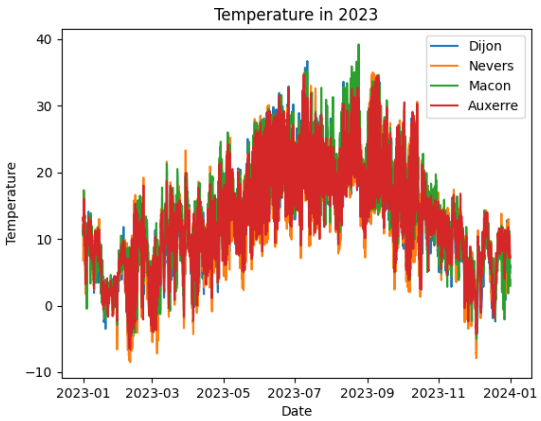
h) For these four dataframes, convert the column "AAAAMMJJHH" to datetime format (you have already done it in the previous session).

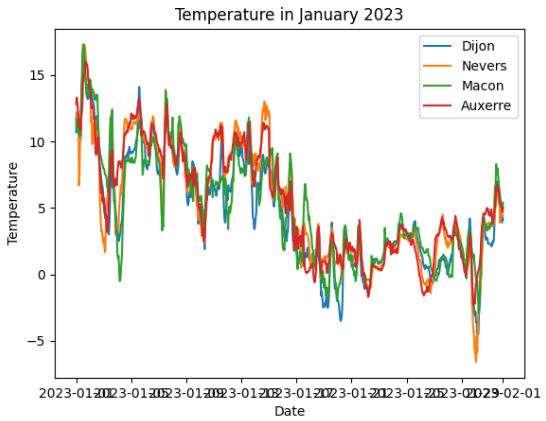
i) Then merge these four dataframes into a unique one by using the "AAAAMMJJHH" column as the matching column (you want all your records concerning the same date and hour to be in a same row). It will be easier to compare the evolution of weather among your cities with this dataframe than with the previous one.

j) Check that you have now a dataframe containing 37 columns and 8760 entries

k) Display the temperature of the four cities during the year 2023

l) Extract the records concerning only January and display the curves





Session nº 3 : correlation between series

Correlation coefficient is a statistical measure between two series that says if these two series tend to evolute in a same way, in an inverted way, or without any dependency. This coefficient is a number between **-1** and **1**

- If the correlation coefficient is closed to 1, the series are correlated : when the first one increase, the second one increase proportionally
- If the correlation coefficient is closed to 0, the series are uncorrelated, the behavior of a serie cannot be deduced from the other
- If the correlation coefficient is closed to -1, the series are inversely correlated : when the first one increase, the second one decrease proportionally

The correlation between two series can be computed only if the two following conditions are respected :

- The two series are numeric and composed of the same number of values
- None of these series is constant

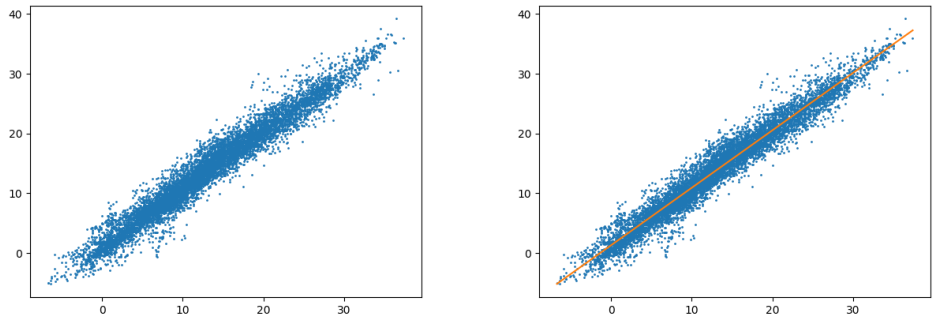
In this session, you will compute correlation between series to find if there are related or not, compute the fitting between two related series, and then learn how to use correlation matrix to find every correlated series among a list.

a) Download the file *correlation.py* from GitHub and place it inside your project

b) Replace the file *data_visualization.py* with the new one uploaded on the GitHub

c) By using the *correlation_coefficient* function from the *correlation.py* file, find two columns of your dataframe that are correlated (correlation coefficient above 0.8)

d) By using the *display_points_cloud* function from the *data_visualization.py* file, display your two series. You should obtain a result that looks like the left figure bellow :



When two series are correlated, you can compute a linear function that convert one of these series to the other. This computation is called a **fitting**

e) By using the *fitting* function from the *correlation.py* file, compute a linear polynomial that convert your first serie to the second one

f) By using again the *display_points_cloud* function from the *data_visualization.py* file, display your two series with the fitting. You should obtain a result that looks like the right figure above

g) By using again the *correlated_subdataframe* function from the *correlation.py* file, find the columns of your dataframe that have at least one of its correlation coefficient above 0.8 (in absolute value), and store them in a variable

h) Display the name of the column in the dataframe you have obtained

i) By using the *display_correlation_matrix* function from the *data_visualization.py* file, display the correlation matrix of the obtained dataframe. You result should looks like this :