Input-Output analysis and modelling with MARIO

Hands-on 7 – Installing MARIO

Please, be aware that all the supporting materials required for this hands-on session is available on Zenodo at the following link: https://doi.org/10.5281/zenodo.8308515

Please use the following citation for:

- **MARIO Software**

Mohammad Amin Tahavori, Lorenzo Rinaldi, & Nicolò Golinucci. (2022). SESAM-Polimi/MARIO: MARIO v0.1.0 (v0.1.0). Zenodo. https://doi.org/10.5281/zenodo.5879382

Learning outcomes

By the end of this exercise, you will learn how to:

1) Install MARIO Software

2) Be familiar with Spyder IDE

3) Test MARIO installation

4) Properly target folders and files in Spyder

Important requirement

Please make sure you have Microsoft Excel (or an equivalent alternative) installed on your PC. It is not strictly required for this hands-on but it is for the next ones.

Install MARIO Software

MARIO is a Python library installable via PyPi.

The recommended method to install MARIO is available at the following link in the official MARIO documentation: https://mario-suite.readthedocs.io/en/latest/installation.html

The steps to be followed are:

- To download and install the Anaconda Python distribution

- To open the Anaconda prompt and create a new environment by running the command:
  conda create -n mario python=3.8

  To run a command it is enough to type it and press the "Enter" key

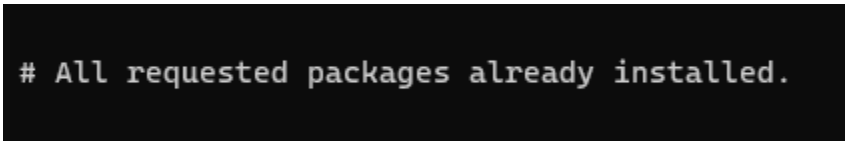
- To activate the new environment by running the command:
  conda activate mario

- To install MARIO by running the command:
  pip install mariopy

For Python beginners, we recommend to use the Spyder IDE. Check whether Spyder is installed in the *mario* environment by running the command:

conda install spyder

In case Spyder is not installed or updated to the latest release, the installation will start, otherwise the following command will be shown:



N.B. Spyder must be installed specifically in the mario environment you just created. Before running 'conda install spyder' be sure you ran the 'conda activate mario' command before

The Spyder IDE

Spyder is one of the many Python IDEs (Integrated Development Environments)

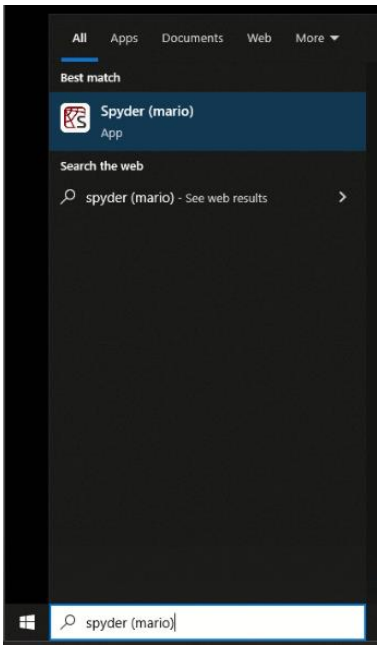
You are not forced to use Spyder if you are not a Python beginner, however all the following hands-on sessions of this course will be developed by using Spyder.

To launch Spyder:

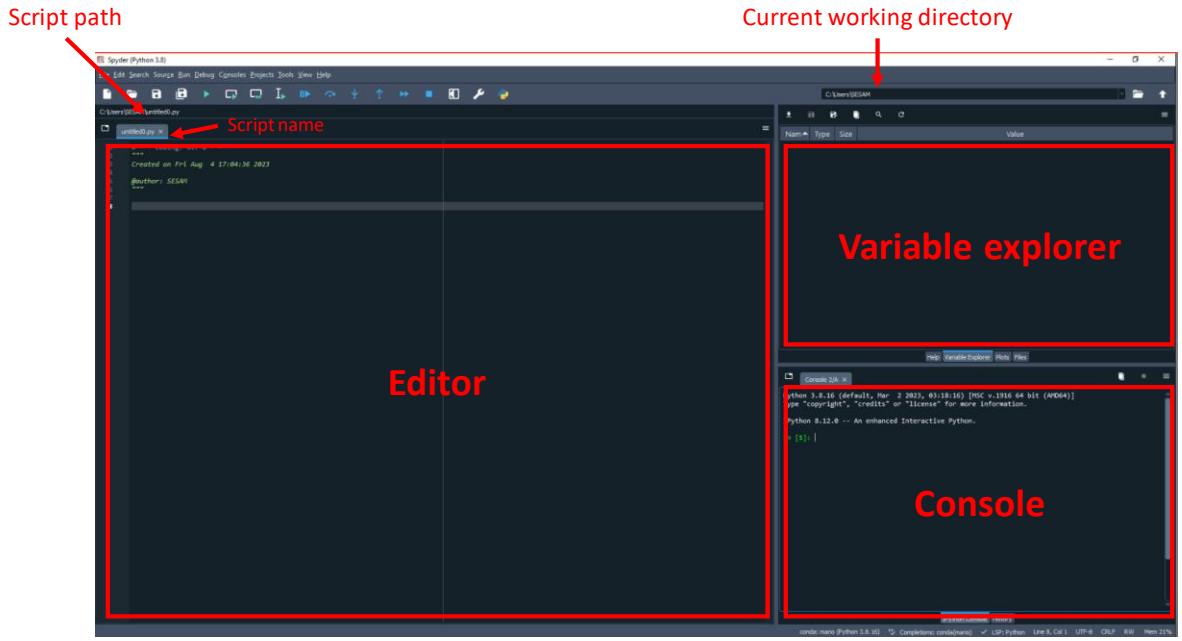
- Open the Anaconda prompt and activate the *mario* environment by running the command:
  conda activate mario

- Now launch Spyder by running the command:
  spyder

  Alternatively, if you are using Microsoft Windows, just type "***Spyder (mario)*"** in the Windows search bar as shown in the figure below.



Once you launched Spyder you can visualize a window similar to the one below.



Let's define each of these components.

Console

In Spyder, the console is an interactive environment where you can execute Python code and see the results immediately. It's similar to an interactive Python shell (REPL - Read-Eval-Print Loop) but integrated into the IDE. You can access the console in Spyder through the "IPython Console" pane.

Current Working Directory

The current working directory (CWD) is the directory on your file system where Spyder will look for files and where any files you create or save will be placed by default. It is the starting point for file operations in your Python code.

Script Path

The script path refers to the location of the Python script file you are currently working on. When you open or create a Python script in Spyder, the IDE automatically sets its path as the current script path. This path is essential for loading and saving files relative to the script's location.

Script Name

The script name is the filename of the currently opened or active script in the editor. It typically ends with the ".py" extension, indicating it is a Python script file. Spyder displays the script name in the title bar of the editor tab.

Script Editor

The script editor is the main workspace in Spyder, where you can write, edit, and execute Python code. It provides features like syntax highlighting, code folding, code completion, and debugging capabilities. Spyder's editor makes it convenient for writing and organizing your code effectively.

Variable Explorer

The variable explorer is a feature in Spyder that displays a table with the current state of all variables in your Python environment. It shows the variable names, their data types, and their values. This tool is helpful for inspecting the values of variables during debugging and for analyzing data when working with datasets. You can find the variable explorer in the "Variable Explorer" pane in the IDE.

Test MARIO installation

To test whether MARIO is properly installed, just run the following command in the Spyder console:

import mario

A warning message about the installation of the *cvxpy* package is raised by default. No worries, *cvxpy* is not required in this course.

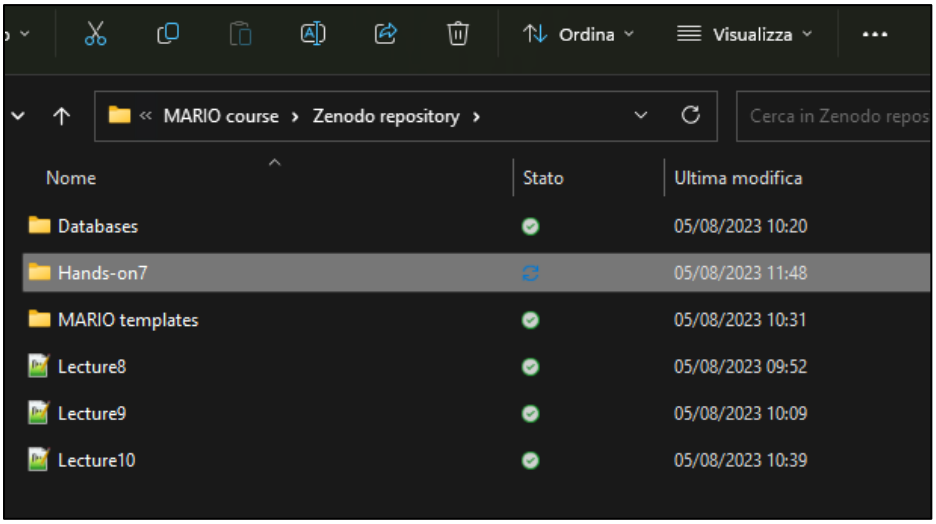
```
In [1]: import mario
cvxpy module is not installed in your system. This will raise problems in some of the abilities
of MARIO
```

In case you get a **ModuleNotFound Error** please repeat the installation process.

Direct to folders and files in Python

Most of the basic Python knowledge required to use MARIO involves the capability of correctly reading files (Excel .xlsx files in particular).

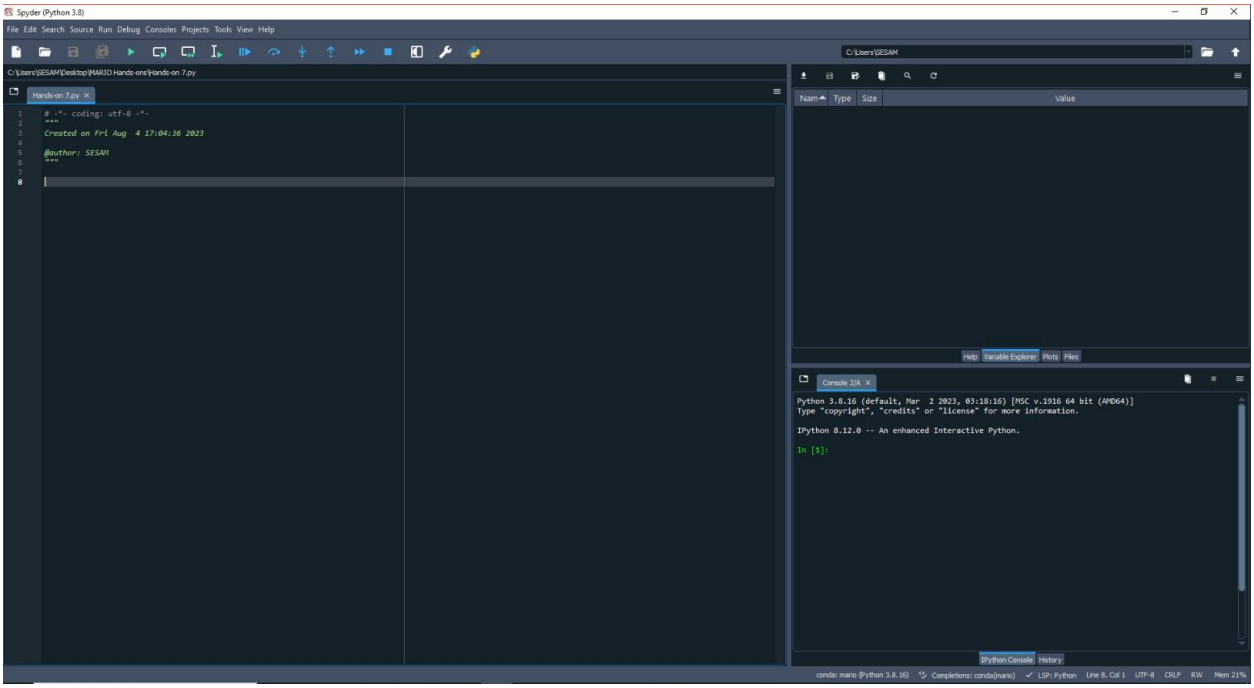
To make an example, let's create a test folder named "Hands-on1" in a desired directory of your PC.



The full path to this folder is **C:\Users\...\\Hands-on7**. Let's save a script from Spyder in this folder. To do so, just save the currently opened script or create a new script from the File menu and, again from the File menu, click "Save as". Then rename the file as "Hands-on7.py" and save it in the Hands-on1 folder.

Let's also create a new Excel file named "test.xlsx" in the same folder.

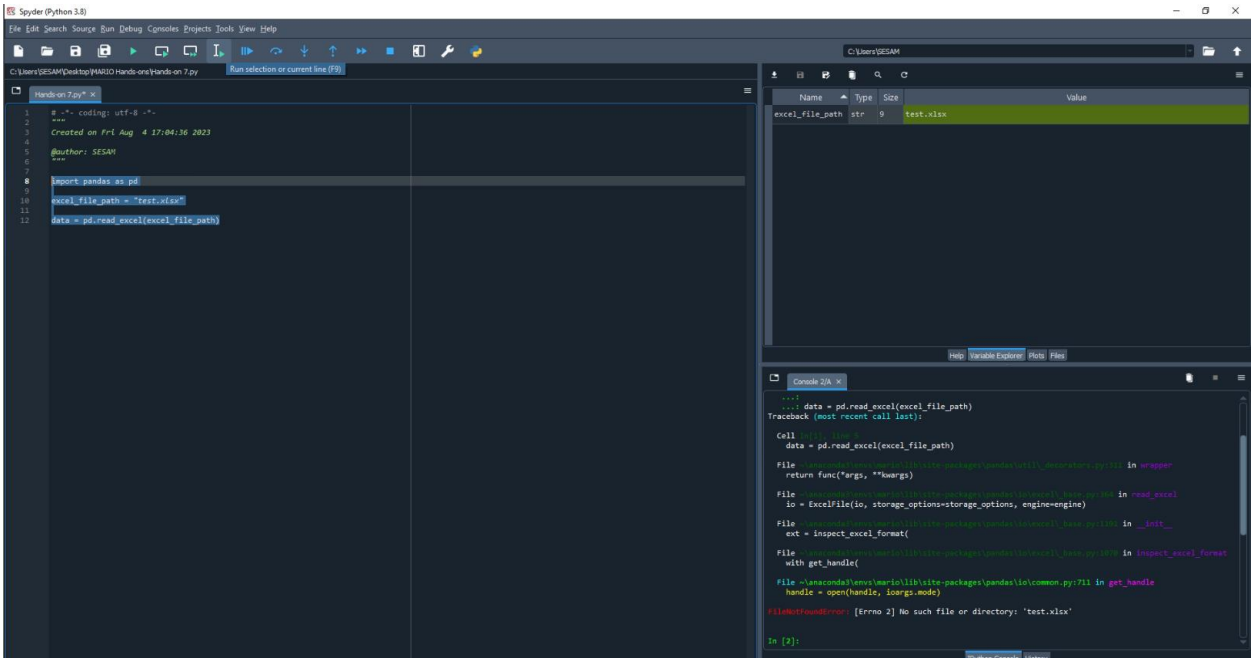
You can see now that the file is saved properly and that the **script path** is consistent with the saving just performed.



Let's say that we want to parse the Excel file just created with Python. To do so, the library we need to use is Pandas. If you are a Python beginner, we definitely recommend you learn how to use the basic Pandas functionalities, it can be very useful also outside of this course.

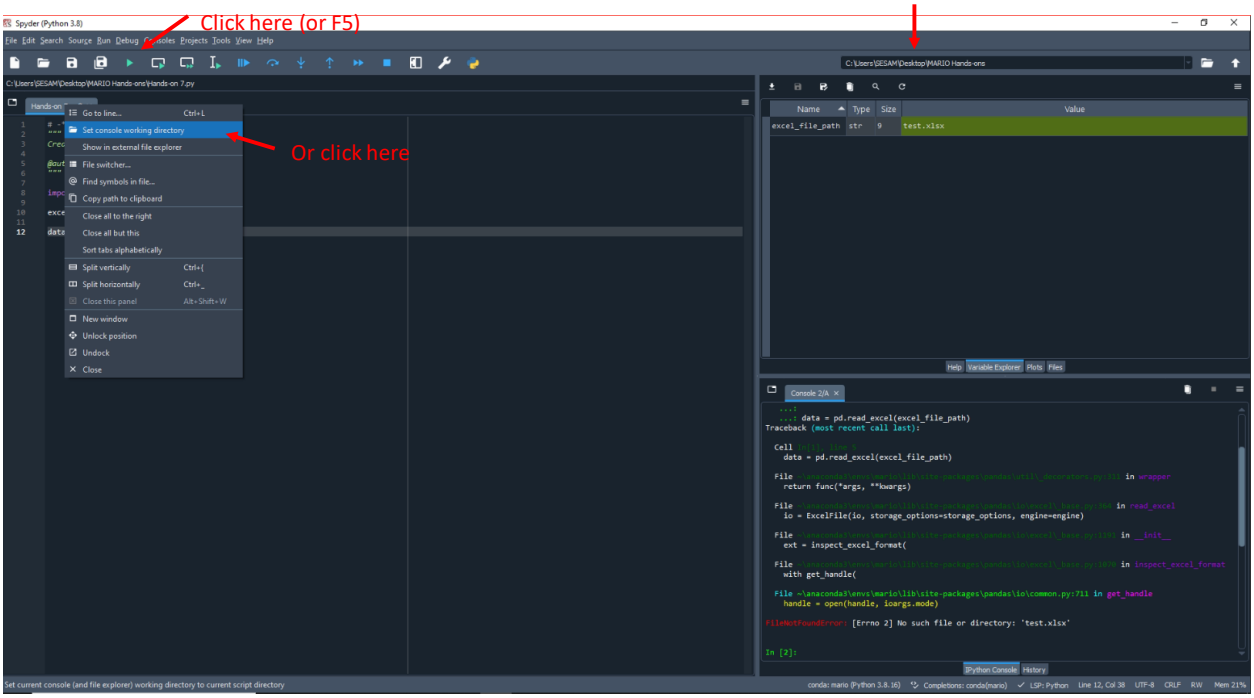
In particular, we need to use the read_excel function.

Let's start filling our script by importing pandas and defining a path variable after the name of your excel file. Then call the read_excel function, highlight all the lines of your script and Run the current selection by clicking on the button highlighted by the pointer in the figure below or by pressing F9. As you can see, we got FileNotFound Error.

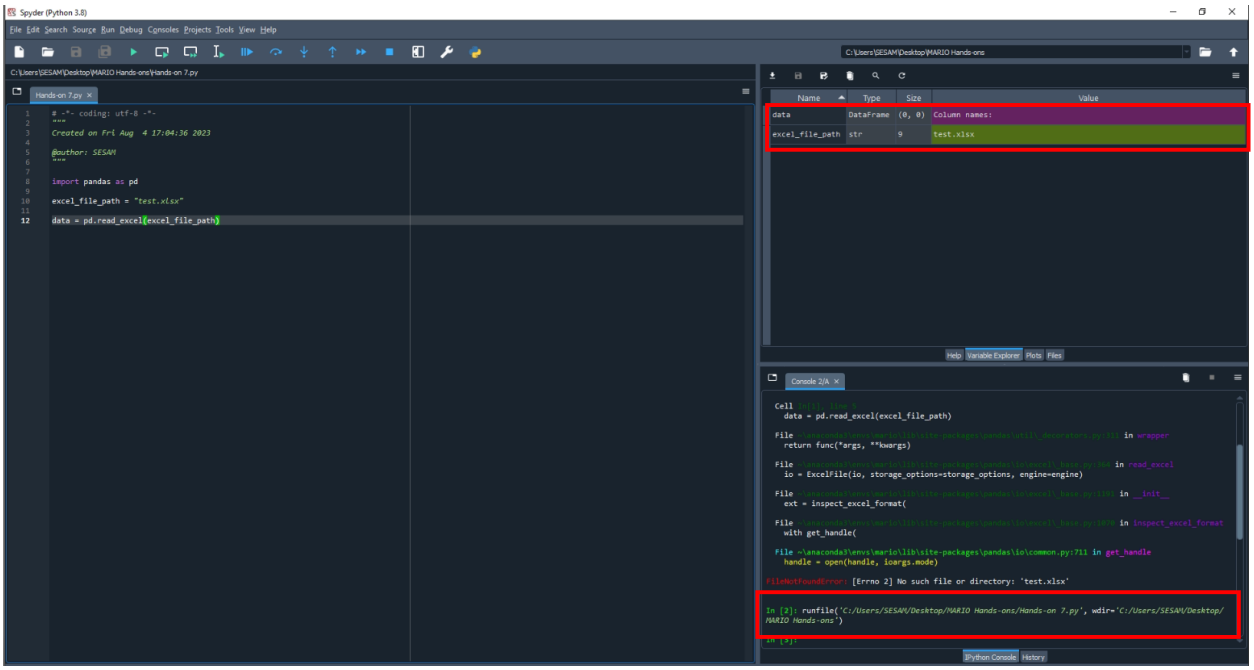


In fact, you need to change the **current working directory** as well to be sure you are targeting the file properly**.** To do it, you can both running the whole script by clicking the green “play” button (or pressing F5) or, to avoid running the script, just right-click on the **script name** and click on “Set console working directory”.

In this way, you can see the **current working directory** switched to the same path as the **script path**



After setting the current working directory properly, run again the script (F5 or select all the lines + F9) and you can see a Dataframe object has now been generated in the **variable explorer**. It's size is (0,0) since the Excel file was empty.



It is also possible to work with files which are not contained in your current working directory. It would be enough to define the variable path (in our case “excel_file_path”) in the following alternative ways

- Using double backslashes: **excel_file_path = "C:\\ANY_PATH\\test.xlsx"**
- Writing an ‘r’ before the path string: **excel_file_path = r"C:\ANY_PATH\test.xlsx"**