



OPTIMUS PROJECT

OpenFast Python Automation Manual

Written by: Araz Hamayeli Mehrabani

Wind Energy Engineering
University department 2 – Energy and biotechnology
Hochschule Flensburg

09.02.2024

Table of Contents

1	Introduction	1
2	Purpose.....	1
3	OpenFast	1
3.1	Structure.....	2
3.1.1.1	start_OpenFAST_v3-41.bat.....	3
4	Python Automation	3
5	Python script manual.....	5
5.1	Installation Prerequisites	5
6	Autorun Openfast.....	8
6.1	DLCs.....	8
6.2	Full Automation of a Simulation Loop.....	8
6.3	VSCode Run.....	9
6.4	Extra steps.....	10
7	References.....	L

List of Figures

Figure 1: OpenFAST structure [1].....	2
Figure 2: Structure of Directories and Inputs	2
Figure 3: start_OpenFAST_v3-41.bat.....	3
Figure 4: Example of the Python program for DLC5.1	4
Figure 5: Example of the Python program for running DLCs together.....	4
Figure 6: Checking Python installation on Windows	5
<i>Figure 7: Installing Python Extension in VSCode.....</i>	6
Figure 8: Interpreter Selection in VSCode	6
Figure 9: Pip installation in VSCode	7
Figure 10: PyAutoGUI installation in VSCode.....	8
Figure 11: FullyAutomate.py path definition.....	9
Figure 12: Example of a full Automate python script in VSCode	9

1 Introduction

During your third semester of study for a Master of Wind Energy Engineering at Hochschule Flensburg, you have the opportunity to be a member of a team to develop a wind turbine in a course with the name "Project: Development of a Wind Turbine".

The project is separated into different sections as different parts of a wind turbine, in each team students work and search literature to gain knowledge and expertise in their specific chosen team.

2 Purpose

As a member of the Loads and Dynamics team, you would need several tools for calculating, processing, and demonstrating your findings which are going to be used in the design and evaluation process of every other team in this project. Therefore, having a good knowledge of working and understanding these tools is a must.

This Handbook would cast light upon your journey, and guide you through the path of achieving new skills for your future career.

3 OpenFast

OpenFAST is a versatile tool designed for simulating the interconnected dynamic responses of wind turbines, encompassing various physics and fidelities. In practical terms, it functions as a framework or "glue code," integrating computational modules for aerodynamics, offshore structure hydrodynamics, control and electrical system dynamics (servo), and structural dynamics. This integration facilitates a comprehensive simulation of coupled nonlinear aero-hydro-servo-elastic behavior in the time domain. OpenFAST empowers the analysis of diverse wind turbine configurations, including options like two- or three-blade horizontal-axis rotors, pitch or stall regulation, rigid or teetering hubs, upwind or downwind rotors, and lattice or tubular towers. Moreover, it accommodates modeling scenarios on land or offshore, considering fixed-bottom or floating substructures [1]. In this project version 3.4.1 of OpenFast were used for simulation.

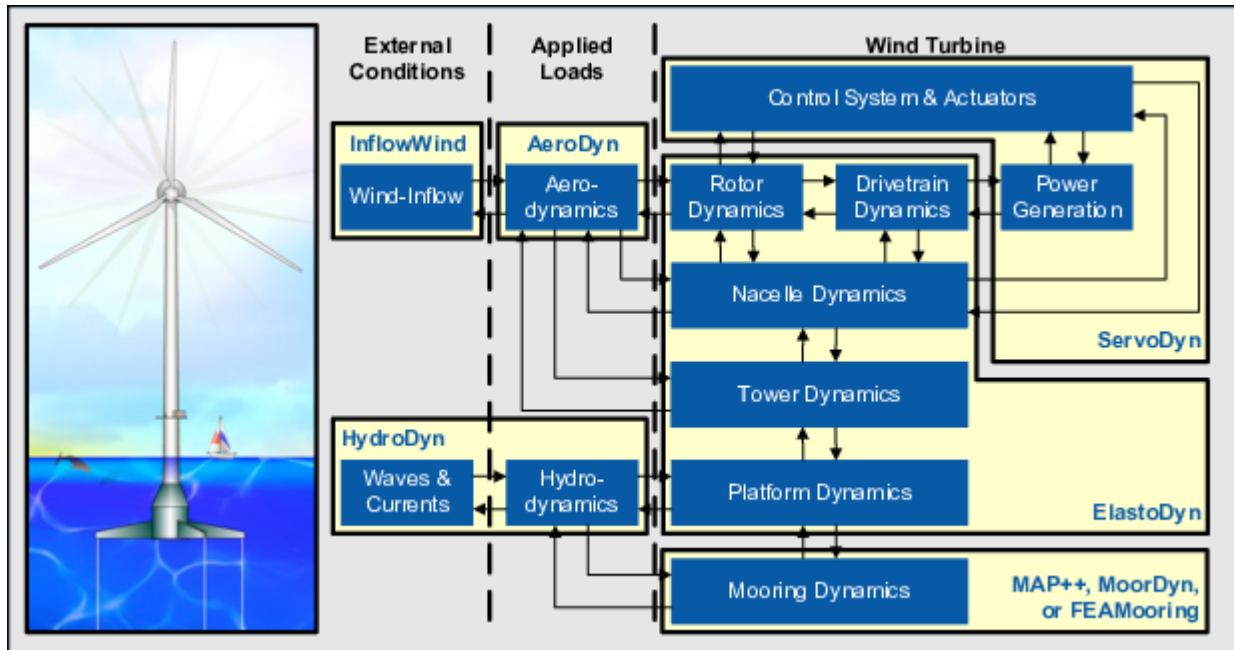


Figure 1: OpenFAST structure [1]

Probably, by now you have seen this diagram several times. This diagram is the simple version of what is happening during an OpenFast simulation.

3.1 Structure

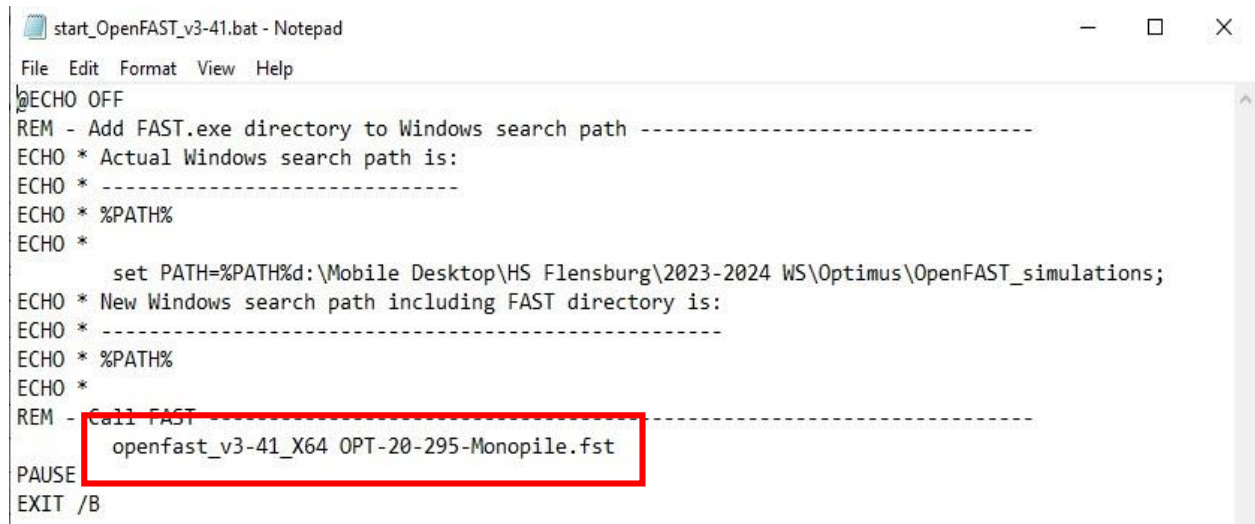
For this project, a set of directories was defined to Classify each file in its related area and prevent confusion. As you can see these files and folders are: AeroData, ServoData, StrucData, Openfast.exe, (name of project).fst, ElastoDyn.dat, HydroDyn.dat, Inflowfile.dat, and start_openfast.bat.

Big P > Final Report > Example >				
Search Example				
Name	Date modified	Type	Size	
AeroData	1/28/2024 10:09 AM	File folder		
ServoData	1/28/2024 10:09 AM	File folder		
StrucData	1/28/2024 10:09 AM	File folder		
openfast_v3-41_X64.exe	10/23/2023 1:23 PM	Application	58,309 KB	
OPT-20-295-Monopile.fst	12/17/2023 2:22 AM	FST File	8 KB	
OPT-20-295-Monopile_ElastoDyn.dat	12/16/2023 11:59 PM	DAT File	19 KB	
OPT-20-295-Monopile_HydroDyn28.dat	12/17/2023 2:22 AM	DAT File	22 KB	
OPT-20-295-Monopile_InflowFile.dat	12/17/2023 7:45 AM	DAT File	6 KB	
OPT-20-295-Monopile_ServoDyn.dat	12/16/2023 10:35 PM	DAT File	13 KB	
start_OpenFAST_v3-41.bat	12/16/2023 5:25 PM	Windows Batch File	1 KB	

Figure 2: Structure of Directories and Inputs

3.1.1 start_OpenFAST_v3-41.bat

Since the OpenFast.exe doesn't have the GUI to be able to run separately, this batch file is necessary for running it in CMD (Terminal). It is not a complicated program inside you just edit it based on the name(version) of your OpenFast and store it next to the openfast.exe, that's it.



```

start_OpenFAST_v3-41.bat - Notepad
File Edit Format View Help
ECHO OFF
REM - Add FAST.exe directory to Windows search path -----
ECHO * Actual Windows search path is:
ECHO * -----
ECHO * %PATH%
ECHO *
set PATH=%PATH%;d:\Mobile Desktop\HS Flensburg\2023-2024 WS\Optimus\OpenFAST_simulations;
ECHO * New Windows search path including FAST directory is:
ECHO * -----
ECHO * %PATH%
ECHO *
REM - Call FAST -----
openfast_v3-41_X64 OPT-20-295-Monopile.fst
PAUSE
EXIT /B

```

Figure 3: start_OpenFAST_v3-41.bat

To understand it better a picture of its details is provided above. It is going to run openfast.exe which is going to read "OPT-20-295-Monopile.fst", where other input paths and conditions are defined.

4 Python Automation

During this project, the number of Design Load Cases (DLCs) and also Wind and Hydrodynamic inputs need numerous times of editing and simulation which increases as the project goes further, for instance, the number of simulations in a loop may be more than 300 different simulations. thus changing parameters for each wind speed and turbulence and hydrodynamic changes by hand would be a time-consuming job that inevitably may cause an error during this process. Also for the sake of post-processing, the output data from OpenFast may need to be renamed as an input to MExtremes, consequently, automation of this process is necessary.

During this project, several versions of Python programs were developed and tested to automate this process. At this stage, two final versions are developed to automate the simulation process.

✓ The goal of the first program is to run each DLC separately, currently, the final version of this program is capable of reading all the Wind and Hydrodyn files from their directory, and for each simulation, it can replace the input files and run the simulation and store the outputs to a separate folder and rename them based on your preferences, then with the help of Pyautogui

```

1  # Developed by Araz Hamayeli Mehrabani 2023 Github: https://github.com/Araz-m
2
3  import os
4  import shutil
5  import subprocess
6  import pyautogui
7  import time
8  # from pyfiglet import Figlet
9
10 # Define the directory where your wind files are located
11 wind_directory = '..\Wind' # Update this path to the actual location
12 hydrodyn_directory = '..\Wave' # Update this path to the actual location
13 main_directory = r'C:\Users\Araz\Desktop\Flensburg\Big P\295\VS\Opt_20_295\DLC5.1\DLC51' # Update this path
14
15 # List all .bts files in the wind directory
16 wind_files = [file for file in os.listdir(wind_directory) if file.endswith('.bts')]
17
18 # Sort wind files based on wind speed and seed number
19 wind_files.sort(key=lambda x: (int(x.split('_')[0][1:]), int(x.split('Seed')[1].split('.bts')[0])))
20
21 for wind_file in wind_files:
22     # Load the current OPT-20-295-Monopile_InflowFile.dat file
23     inflow_file_path = os.path.join(main_directory, 'OPT-20-295-Monopile_InflowFile.dat')
24
25     with open(inflow_file_path, 'r') as inflow_file:
26         inflow_lines = inflow_file.readlines()
27
28     # Find the line containing FileName_BTS and update it
29     for i, line in enumerate(inflow_lines):
30         if 'FileName_BTS' in line:
31             inflow_lines[i] = f'"{os.path.join(wind_directory, wind_file)}" FileName_BTS - Name of the Full field wind file to use (.bts)\n'
32

```

- ✓ The Second one is necessary for running all the DLCs together which is a great help for the loops with more DLCs.

The screenshot shows a Windows File Explorer window. The address bar displays the path: C:\Users\Arax\Desktop\Flensburg\Big P\295\VS\Opt_20_295\Autorun4Samaan\Autorun4Samaan_VII. The main pane shows a directory named 'FullyAutomate.py'. Inside this directory, there is a file named 'DLC12.py'. The left sidebar shows the navigation pane with 'This PC', 'HomeGroup', 'Network', and 'This PC' (repeated) visible.

Figure 5: Example of the Python program for running DLCs together

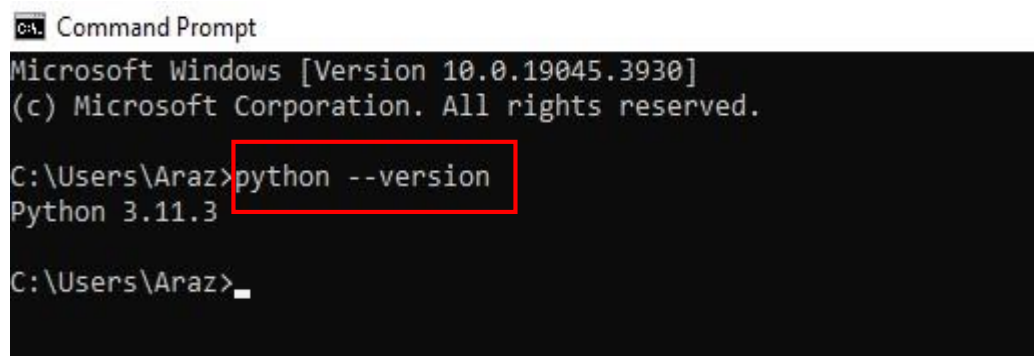
The Python program would be on the developer's Github repository [2], which you can download, also your participation in improving it would be appreciated.

5 Python script manual

5.1 Installation Prerequisites

There are a few requirements for running the Python program on Windows that are going to be described in detail:

- 1- Download and install the latest version of Python from python.org [3].
- 2- during installation fill in the check box to Set Python to the path
- 3- Then open CMD and check if Python is installed on your system by typing:
"python --version"



```
Command Prompt
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Araz>python --version
Python 3.11.3

C:\Users\Araz>
```

Figure 6: Checking Python installation on Windows

Note: *If you prefer to run the program in py.exe, skip Steps 4, 5, 6 and 7 of this subsection

*The issue with this approach is that you can't scroll up to check previous simulations, changes, errors, or warnings

*I recommend working with a text editor rather than py.exe and Python original IDLE for more controllability

- 4- Download and install your text editor for example VSCode (lightweight, Open Source) or Pycharm (CPU Killer but less buggy) [4].
- 5- Inside VSCode, you have to define the main interpreter as the Python you installed but before that, it's better to install the Python extension in the VSCode extension section.

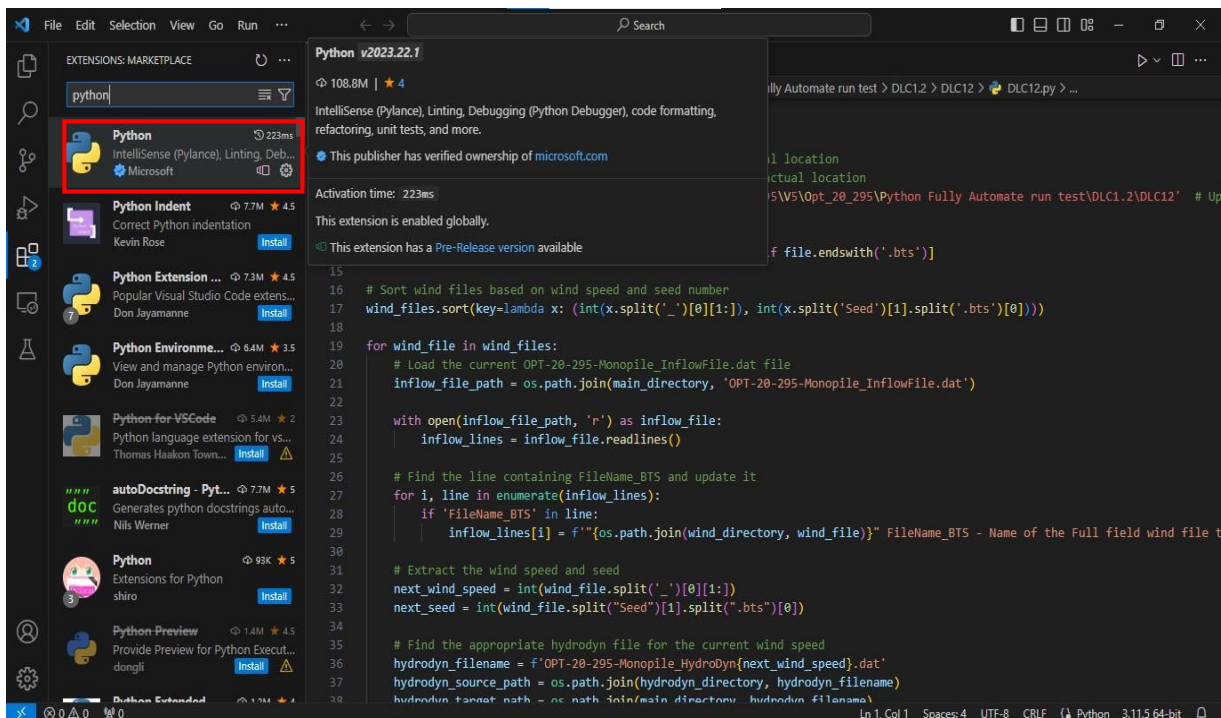


Figure 7: Installing Python Extension in VSCode

6- Now it's necessary to close and reopen VSCode to finalize installations

7- Then press and hold Ctrl + Shift + p, and a search bar opens, type: "select interpreter" → click on it → choose the Python version you downloaded and installed from python.org

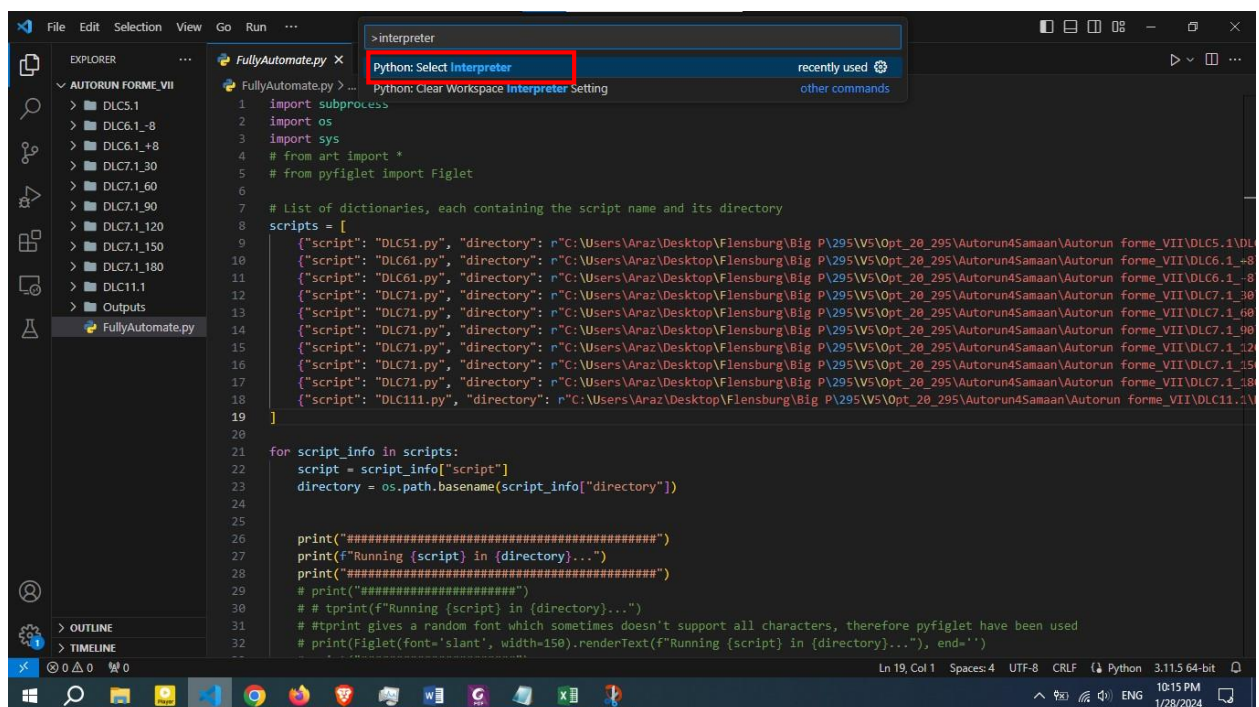
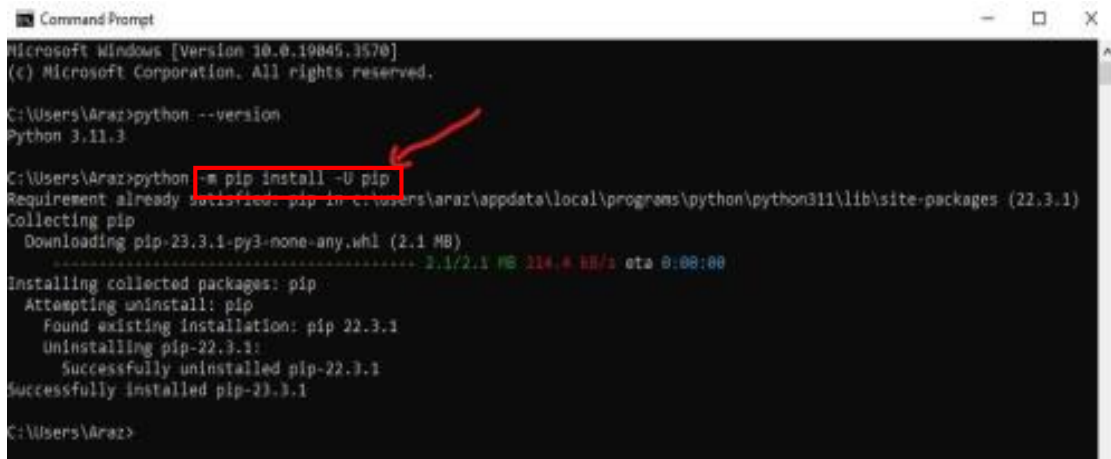


Figure 8: Interpreter Selection in VSCode

- 8- Reopen CMD and type: “python -m pip install -U pip” and press enter or it may need to be updated. Thus carefully read the messages in cmd.

This part is sometimes necessary because when you want to install packages if the pip is not updated, it's not going to work.



```
Microsoft Windows [Version 10.0.19045.3570]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Arar>python --version
Python 3.11.3

C:\Users\Arar>python -m pip install -U pip
Requirement already satisfied: pip in c:\users\arar\appdata\local\programs\python\python311\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
    ----- 2.1/2.1 MB 214.4 KB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
  Successfully installed pip-23.3.1

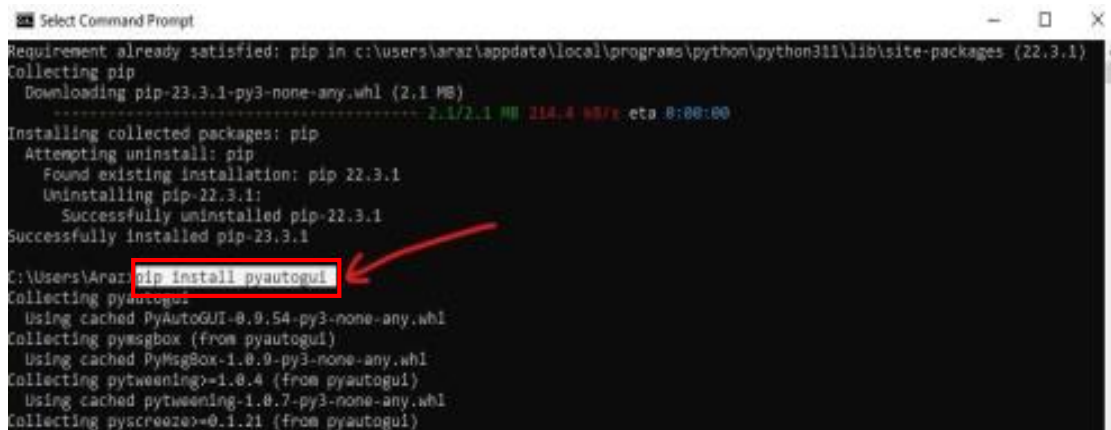
C:\Users\Arar>
```

Figure 9: Pip installation in VSCode

- 9- Now in cmd type: “pip install pyautogui” and enter. PyAutoGUI lets your Python scripts control the mouse and keyboard to automate interactions with other applications [5]. This module would work as an artificial pressing “Enter” or any other key, but in the final version this keystroke changed to “Ctrl” due to some strange behavior with enter.

When you run Openfast after finishing the simulation it says: “Press any key to continue”

This package would help to prevent that problem and it would make the program think that you pressed a key on your keyboard! (the timing can be changed, here it is considered to be 5 Seconds), it means when the message pops up after 5 seconds, it is going to press the Ctrl automatically, why Ctrl? Because as it is virtually clicking or pressing a key on your key board, if for example you choose a function key like Enter during simulation loop it may behave abnormally.



```
Select Command Prompt

Requirement already satisfied: pip in c:\users\arar\appdata\local\programs\python\python311\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
    ----- 2.1/2.1 MB 214.4 KB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
  Successfully installed pip-23.3.1

C:\Users\Arar>pip install pyautogui
Collecting pyautogui
  Using cached PyAutoGUI-0.9.54-py3-none-any.whl
Collecting pymsgbox (from pyautogui)
  Using cached PyMsgBox-1.0.9-py3-none-any.whl
Collecting pytwineing>=1.0.4 (from pyautogui)
  Using cached pytwineing-1.0.7-py3-none-any.whl
Collecting pyscreezer>=0.1.21 (from pyautogui)
```

Figure 10: PyAutoGUI installation in VSCode

6 Autorun Openfast

6.1 DLCs

- 1- It is necessary to put the Python script (here the name is Mainrun.py) in the main folder next to Elastodyn.dat and start_OpenFAST_v3-41.bat and other input files based on the current sorting of files and directories in the main folder.
- 2- The Wind and Wave directory is out of the main directory as decided for this project
- 3- Put the wind.bts files in the Wind folder and let the names be as they are for example:
V3_NTM_IEC1CkaiD320Seed1.bts
- 4- Put all the Hydrodyn files in the Wave Folder but their naming must be related to each wind speed, like: OPT-20-295-Monopile_HydroDyn3.dat where 3 at the end means this is for $V = 3$ m/s.
- 5- Remove the hydodyn.dat from the main folder the program is not going to read it anyway!
- 6- This process should be done for each DLC, in this way it's easier to manage the processes and prevent confusion.

6.2 Full Automation of a Simulation Loop

As described before for running a loop of simulation which may contain 300 or more sets of OpenFast runs, a different version of the Python program was developed that makes the process of simulation more convenient.

For this program also there are a few necessities that need to be prepared:

1. After defining and preparing each DLC separately it is better to rename the mainrun.py in that DLC to the name of that intended DLC for instance: DLC12.py
2. In VSCode add the folder where all DLCs are located and put the "FullyAutomate.py" next to them.
3. Open the Python file and set the path of each DLC main folder where DLC12.py is located.

```
4. scripts = [  
5.     {"script": "DLC12.py", "directory": r"DLC12 path"},  
6.     {"script": "DLC13.py", "directory": r"DLC13 path"},  
7.     {"script": "DLC111.py", "directory": r"DLC23 path"},  
8.     {"script": "DLC111.py", "directory": r"DLC111 path"}  
9. ]
```

Figure 11: FullyAutomate.py path definition

- Here Pyfiglet module was added to the program, it takes ASCII text and renders it in ASCII art fonts that would be a useful addition during a loop simulation [6].

This module will help to know which DLC and wind speed running at the moment, but in this version, it's commented so it can be activated based on your needs.

- By running this Python file (which is going to be described in the next section) you will be able to run all DLCs just by a few clicks.

6.3 VSCode Run

- ✓ Now open VSCode then click on the file → open folder → find the main folder and open. It should look like this:

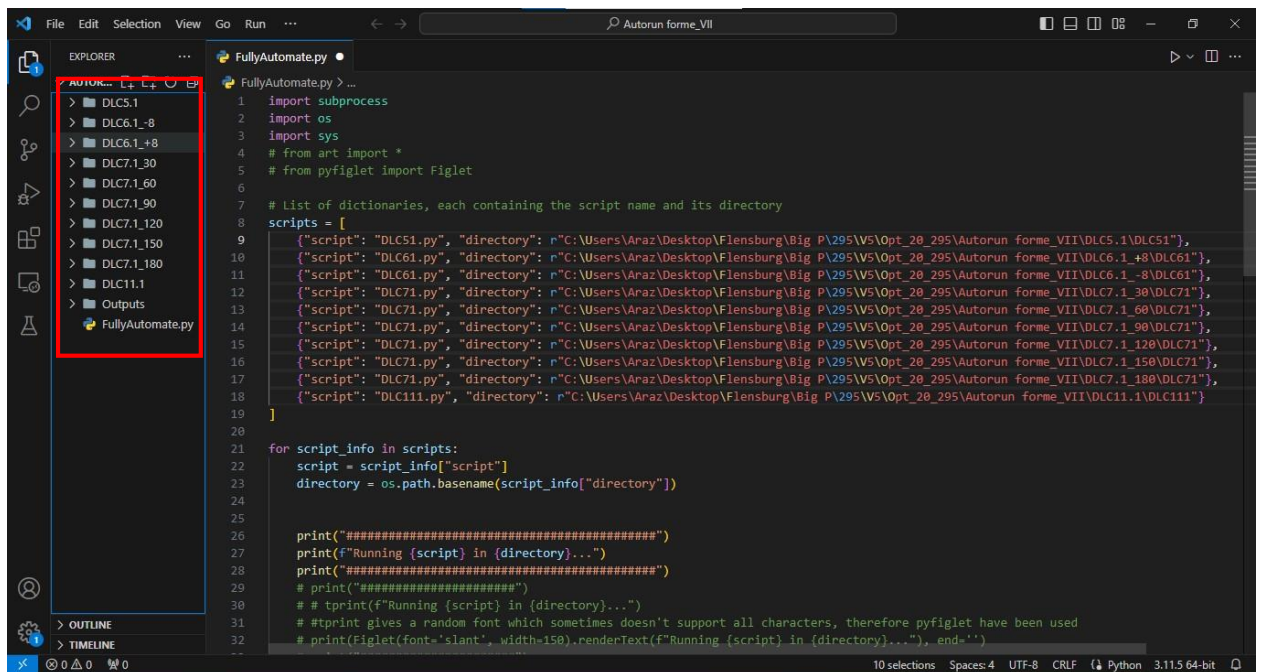


Figure 12: Example of a full Automate python script in VSCode

- ✓ Click on Mainrun.py: Inside you have to change a few names (Strings)

Update this path:

```
main_directory = r'path' # Update this path  
to your main folder Path (just the part between quotation ")
```

- ✓ In line 95 change the name of the DLC you are working on (it will create a folder with that name)

```
dlc_directory = os.path.join(output_directory, 'DLC12')  
again (just the part between quotation ")
```

- ✓ Save the changes

- ✓ Right-click on the Mainrun.py and click on “Run Python File in Terminal”
- ✓ That’s it. It will run all the wind speeds and seeds and waves you put in the Wind and Wave directory and store the output files in a folder named “Outputs” within the DLC folder you defined in line 95
- ✓ The name of the files would be like: OPT_MP_V3_S1.out, they are abbreviated because for MExtremes they were too long and confusing

6.4 Extra steps

- 1- You can also install the “Material Icon Theme” extension which will help you to know in which language you are coding or which type of file you are editing, for example for Python, it shows a Python icon, and for batch files, it has a different icon
- 2- Usually, it’s better to have an environment manager because you may have scripts that may not work with newer versions of Python or some other library. You can create different environments and install different versions of Python or libraries or packages, then they are isolated from other environments and you can manage them based on your needs. I recommend MiniConda, over Venv or Anaconda, MiniConda is a free minimal installer for conda. It is a small bootstrap version of Anaconda that includes only conda, Python, the packages they both depend on, and a small number of other useful packages (like pip, and a few others) [7], but it depends on your preference. But for this Automation, neither of them is necessary.

7 References

- [1] „OpenFAST Documentation, Version: v3.5.2,,“ Jan 18, 2024.
- [2] „<https://github.com/Araz-m>,“ [Online]. Available: <https://github.com>.
- [3] „Python Documentation,“ [Online]. Available: <https://www.python.org/downloads/>.
- [4] „Visual Studio Code documentation,“ [Online]. Available: <https://code.visualstudio.com/docs>.
- [5] „PyAutoGUI’s documentation,“ [Online]. Available: <https://pyautogui.readthedocs.io/en/latest/>.
- [6] „Project description and documentation,“ [Online]. Available: <https://pypi.org/project/pyfiglet/>.
- [7] „MiniConda documentation,“ [Online]. Available: <https://docs.conda.io/projects/miniconda/en/latest/>.