HORIZON-CL5-2022-D3-01-11
Demonstration of innovative forms of storage and their successful operation and integration into innovative energy system and grid architectures

AGiSTIN

Advanced Grid Interfaces for innovative STorage INtegration

Battery Model Insights

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Climate, Infrastructure and Environment Executive Agency (CINEA). Neither the European Union nor the granting authority can be held responsible for them.

Agenda

- Project Overview
- Project Directory Structure
- Key Battery Model Files
- Batteries.py - Parameters
- Batteries.py - Variables
- Batteries.py - Constraints
- ExampleBatteryPV_Case_1_131124.py
- Optimization Workflow
- Data Handling & Solver
- Conclusion

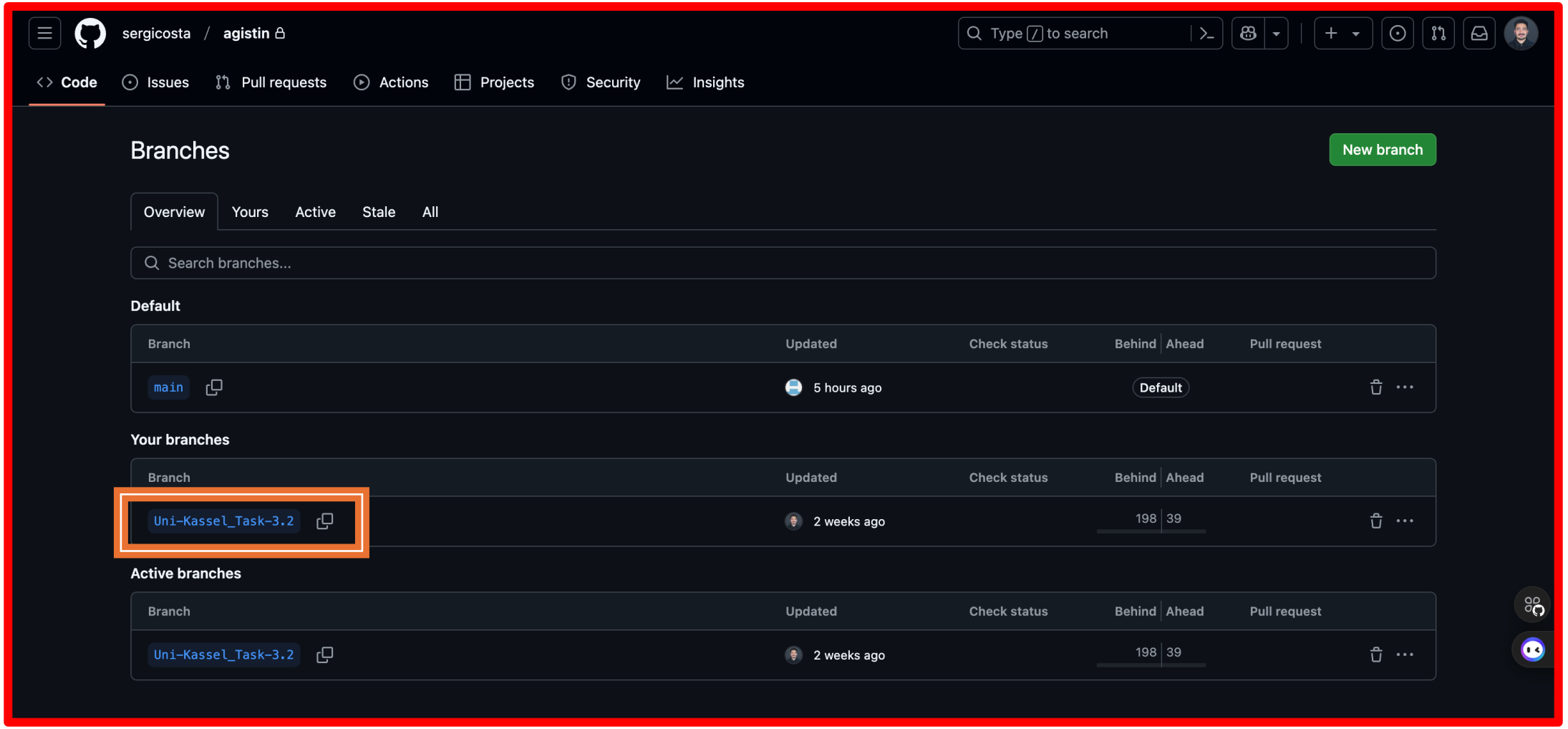
Project Introduction

- **Project Overview:** The Battery Model Project aims to enhance synergy between battery storage, photovoltaic systems, and grid performance.
- **Focus Areas:** Key focus includes optimization techniques for maximizing efficiency in energy storage and generation processes.
- **Integration Importance:** A critical aspect is understanding how optimizing both components leads to improved overall energy management.

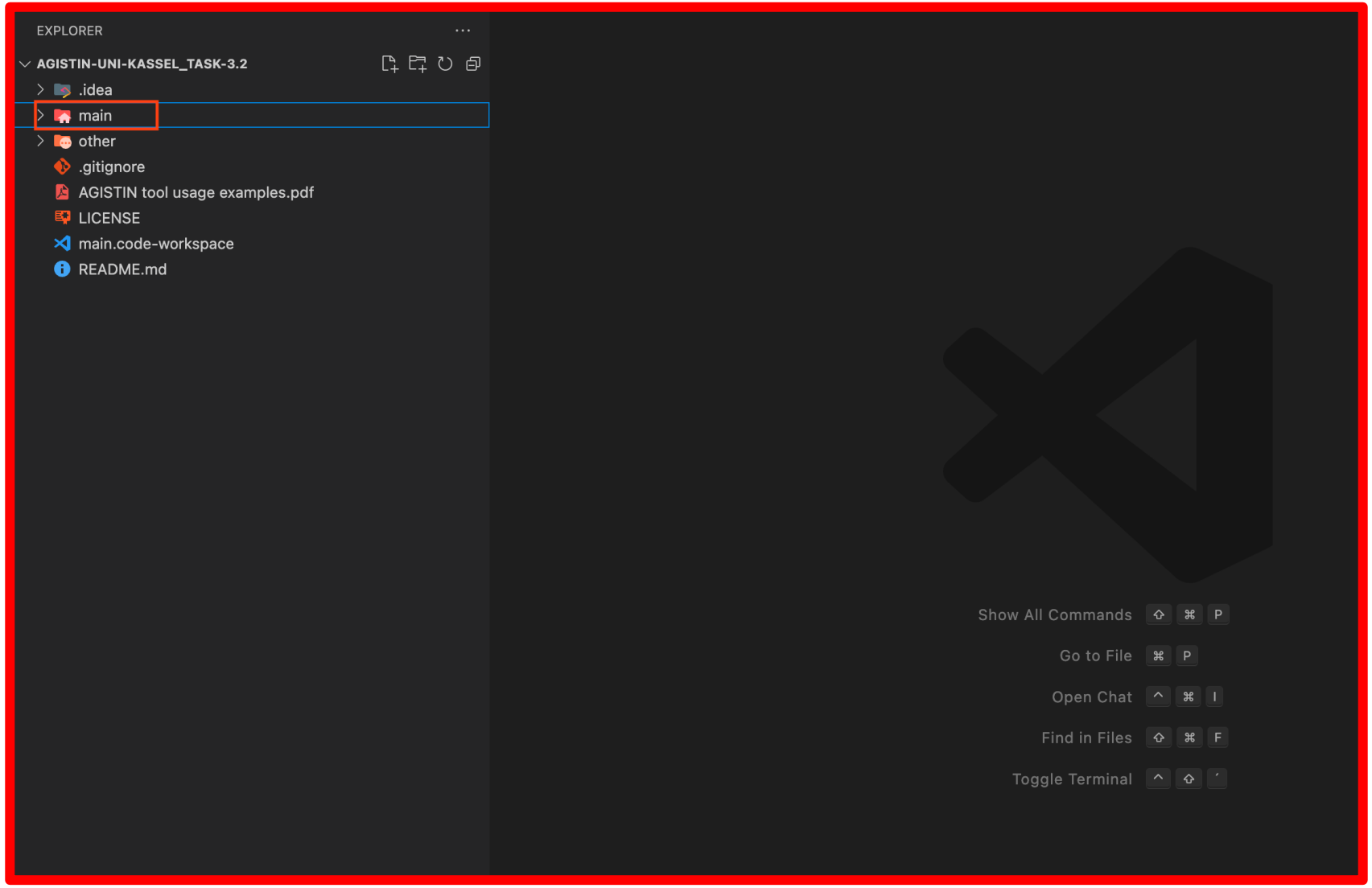
Project Directory Structure

- **Directory Structure:** The directory contains organized folders for efficient project management and accessibility of files.
- **Main Folder:** 'agistin-Uni-Kassel_Task-3.2' serves as the root, centralizing all project-related documentation and substructures.
- **Subfolders Overview:** Subfolders include 'main' for primary files, 'Cases' for data, and 'Devices' for Python scripts.

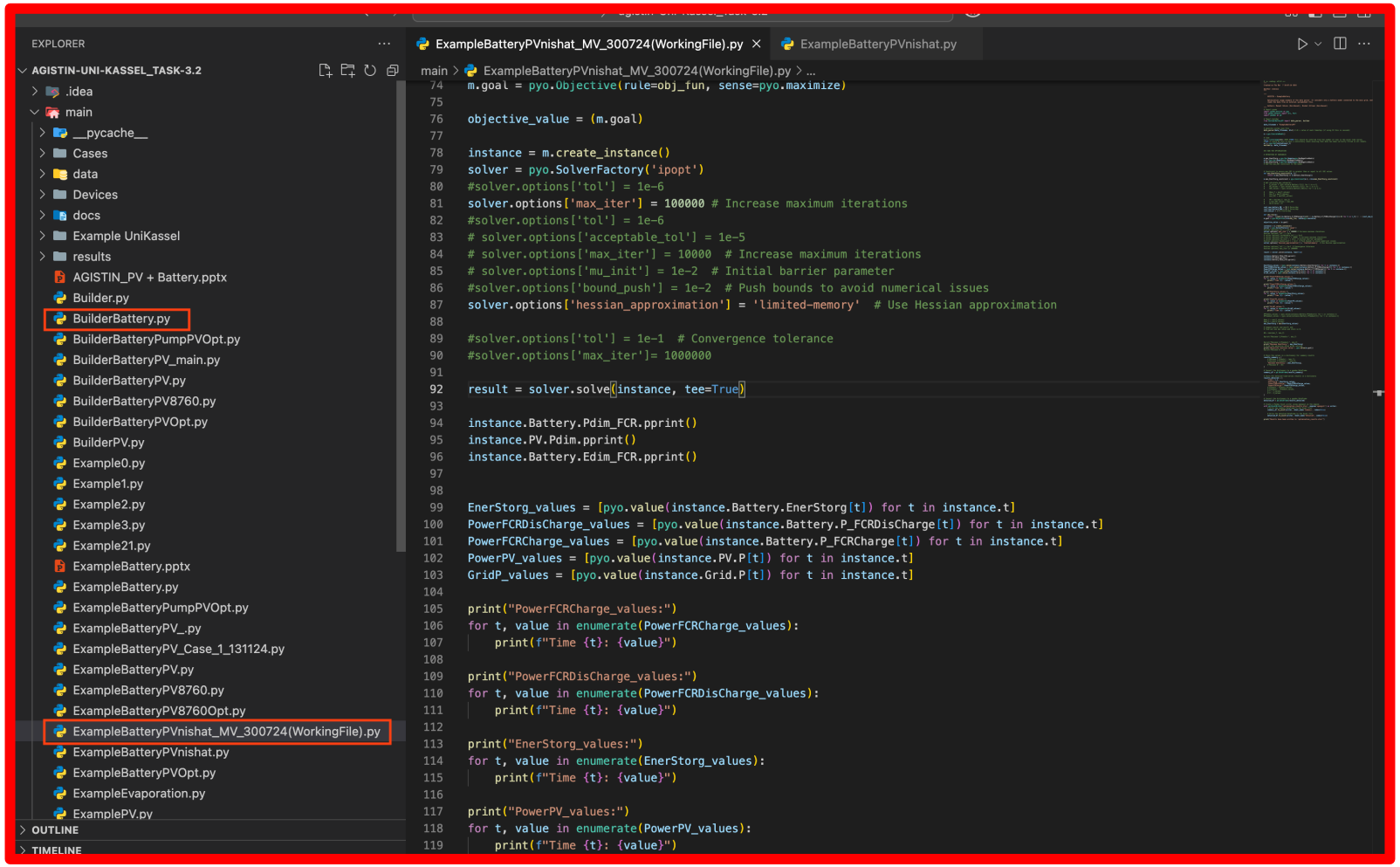
1- This is our GitHub repository.[https://github.com/sergicosta/agistin/branches]
2- It contains a main file where all official work is updated.
3- Additionally, we have a copy of the file where we work, which I have highlighted.



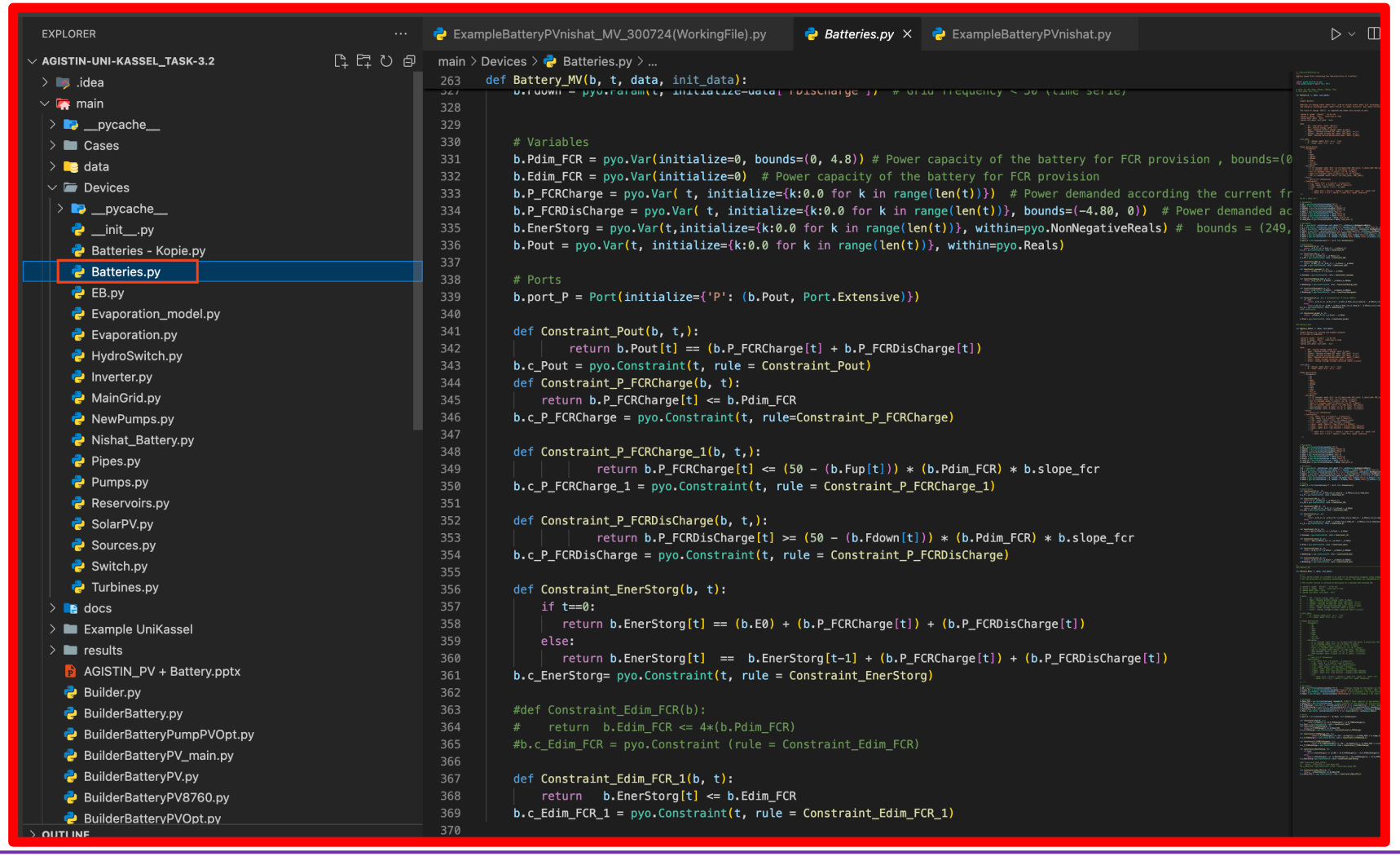
1- In this, we first have a main folder that contains subfolders and files inside it.



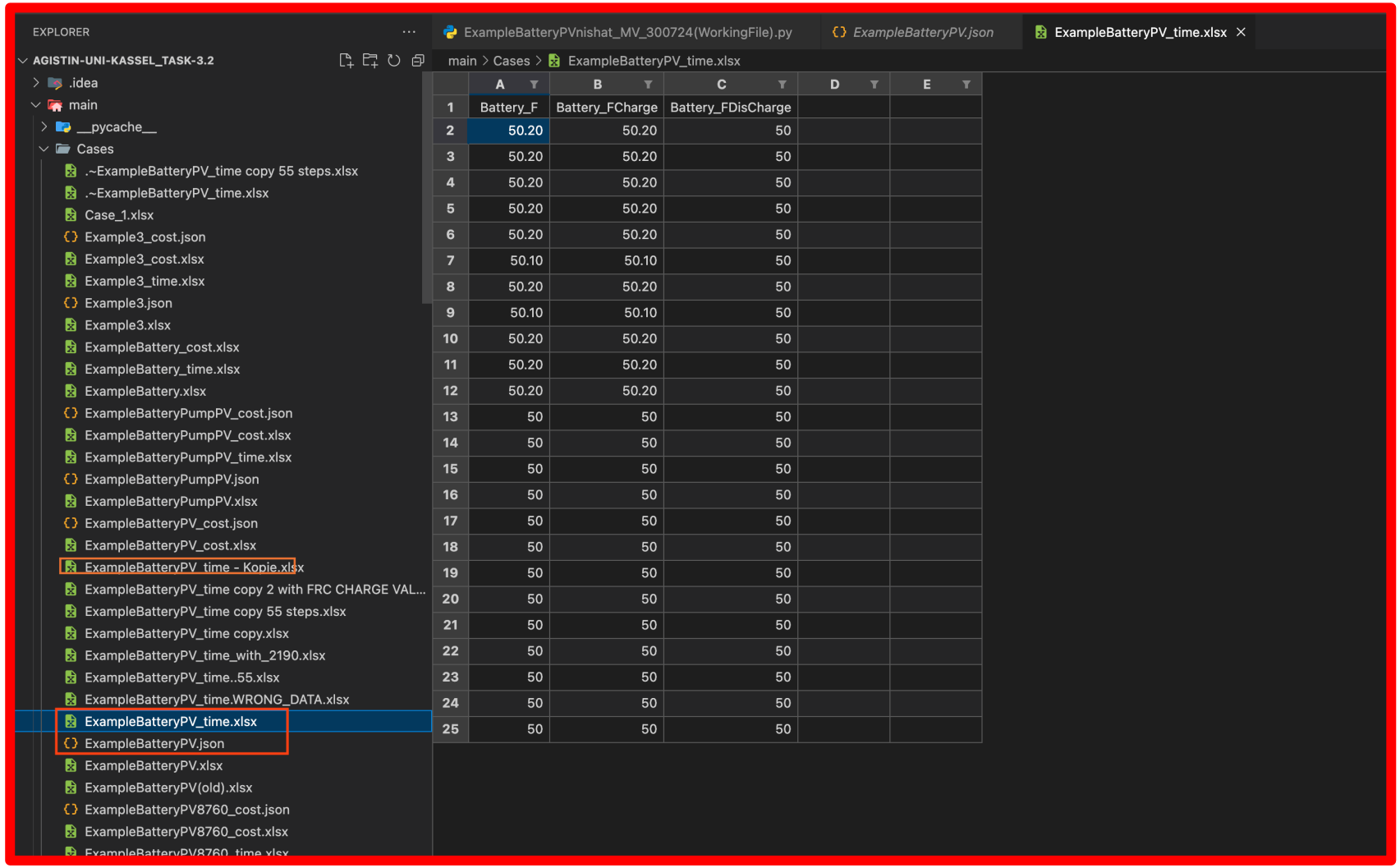
1- In this, we have a main file: **ExampleBatteryPVnishat_MV_300724(WorkingFile)**.
2- This file is where we declare the data points, and we can custom-select the ones present in the Excel file.
3- There's also a **builderbattery.py** file, which helps in calling our JSON and Excel files. We don't make many changes to this file.



1- In this, we have a **devices folder** containing the **batteries.py** file.
2- In this file, we declare different variables, constraints, and functions that we want to print or use.



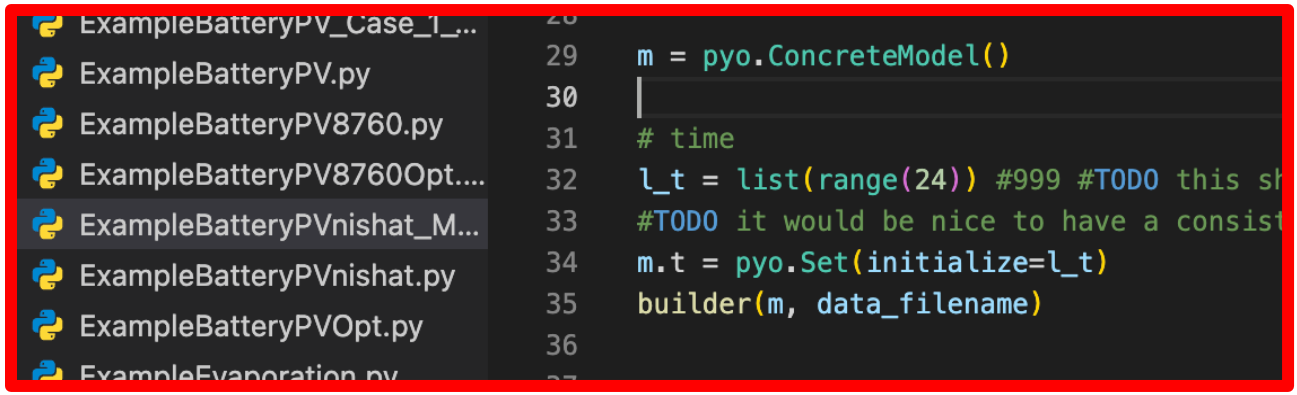
1- In this, we have a **cases folder** that contains 3 files, which are marked.
2- The important file is **examplebatteryPV_time**, where we declare our data points.
3- To run different cases, these 3 files are used.



	A	B	C	D	E
1	Battery_F	Battery_FCharge	Battery_FDisCharge		
2	50.20	50.20	50		
3	50.20	50.20	50		
4	50.20	50.20	50		
5	50.20	50.20	50		
6	50.20	50.20	50		
7	50.10	50.10	50		
8	50.20	50.20	50		
9	50.10	50.10	50		
10	50.20	50.20	50		
11	50.20	50.20	50		
12	50.20	50.20	50		
13	50	50	50		
14	50	50	50		
15	50	50	50		
16	50	50	50		
17	50	50	50		
18	50	50	50		
19	50	50	50		
20	50	50	50		
21	50	50	50		
22	50	50	50		
23	50	50	50		
24	50	50	50		
25	50	50	50		

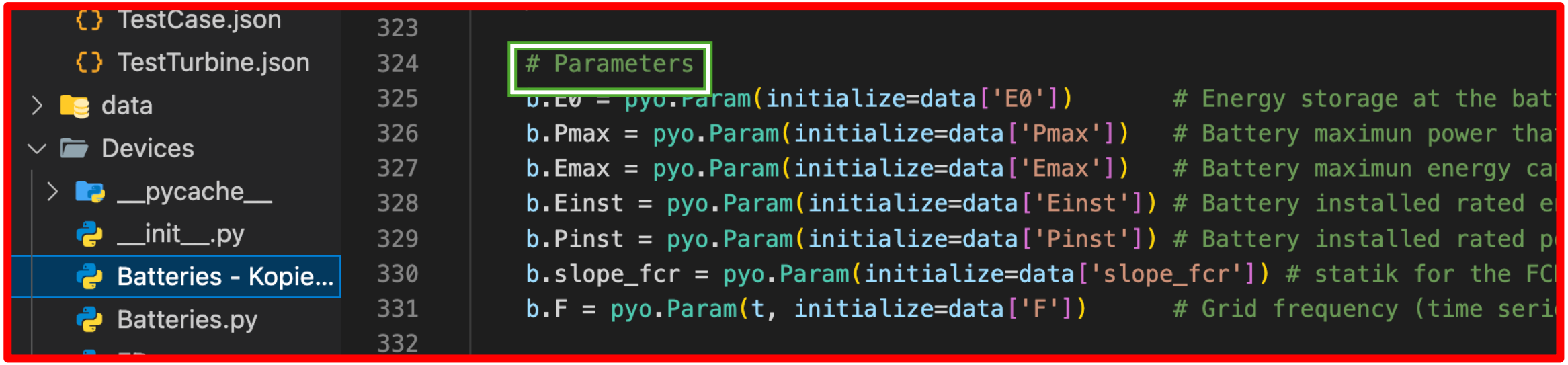
Key Battery Model Files

- **Core Model File:** 'BuilderBatteryPV' is crucial for assembling and developing the comprehensive battery model architecture efficiently.
- **Test Case Definition:** 'ExampleBatteryPVnishat_MV_300724' outlines specific test conditions to ensure model robustness and reliability.
- **File Integration:** Integration of these files enhances project modularity, facilitating easier adjustments and future developments.



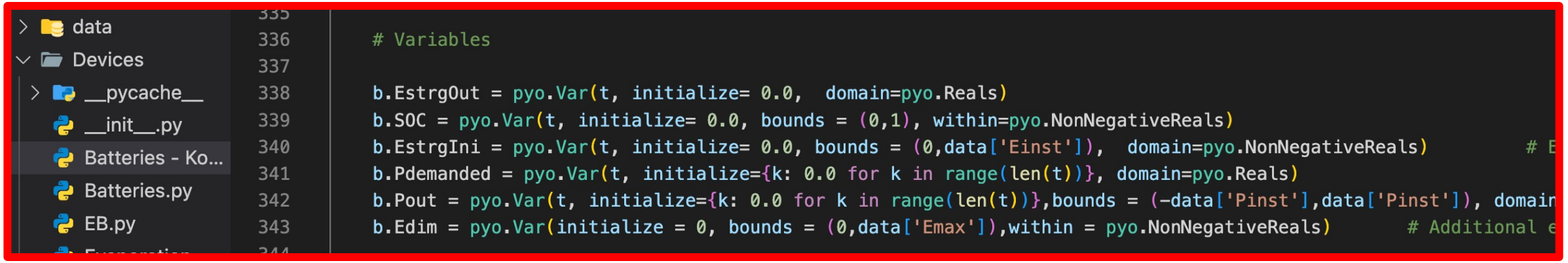
Batteries.py - Parameters

- **Initial Energy Storage:** Parameter b.E0 indicates the initial stored energy within the battery, facilitating mathematical optimization.
- **FCR Performance Slope:** The variable b.slope_fcr represents the effective change in frequency response during charging and discharging phases.
- **Grid Frequency Deviations:** Parameters b.Fup and b.Fdown define acceptable grid frequency thresholds for efficient battery operation management.



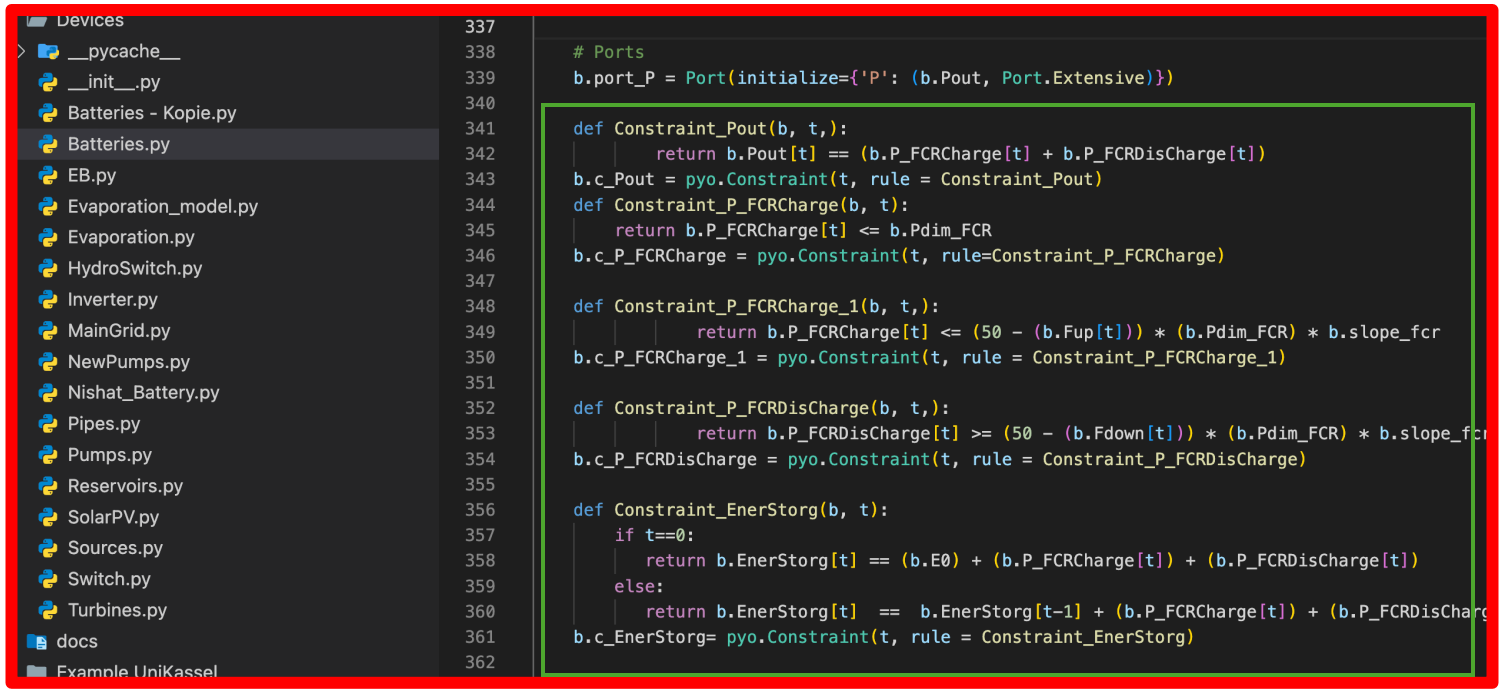
Batteries.py - Variables

- **Max Power Capacity:** Variable b.Pdim_FCR defines the battery's maximum power output during operational scenarios and peak demands.
- **Energy Capacity:** The energy storage capacity, represented by b.Edim_FCR, determines total energy storage available for discharge cycles.
- **Net Battery Output:** b.Pout indicates the actual output power from the battery, factoring in charging and discharging events.



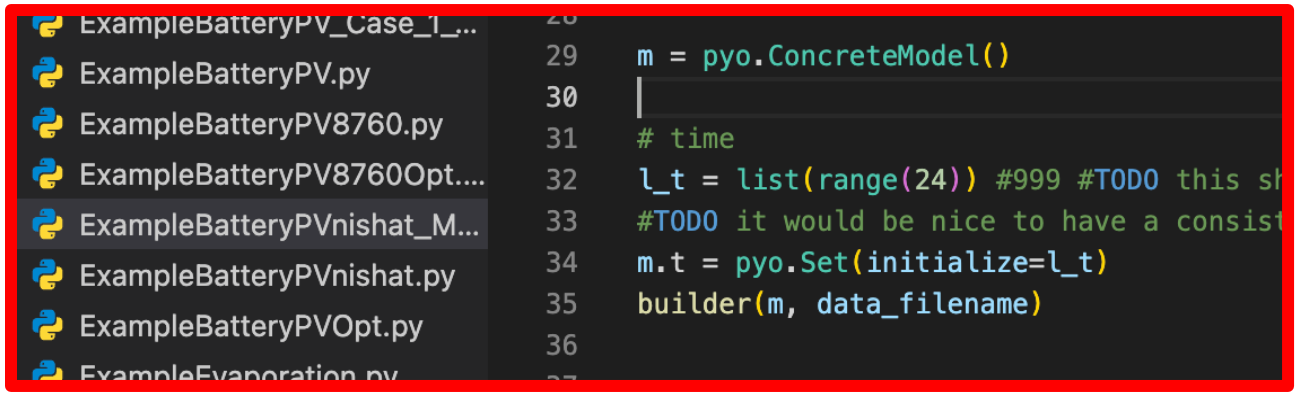
Batteries.py - Constraints

- **Power Output Constraint:** Constraint_Pout ensures balance between charging and discharging power, maintaining operational stability and efficiency.
- **State of Charge Tracking:** Constraint_EnerStorg monitors the State of Charge, aligning with initial energy levels for optimal performance.
- **Energy Capacity Limits:** Constraint_Edim_FCR_1 defines the maximum and minimum energy capacities, ensuring the battery operates within safe thresholds.



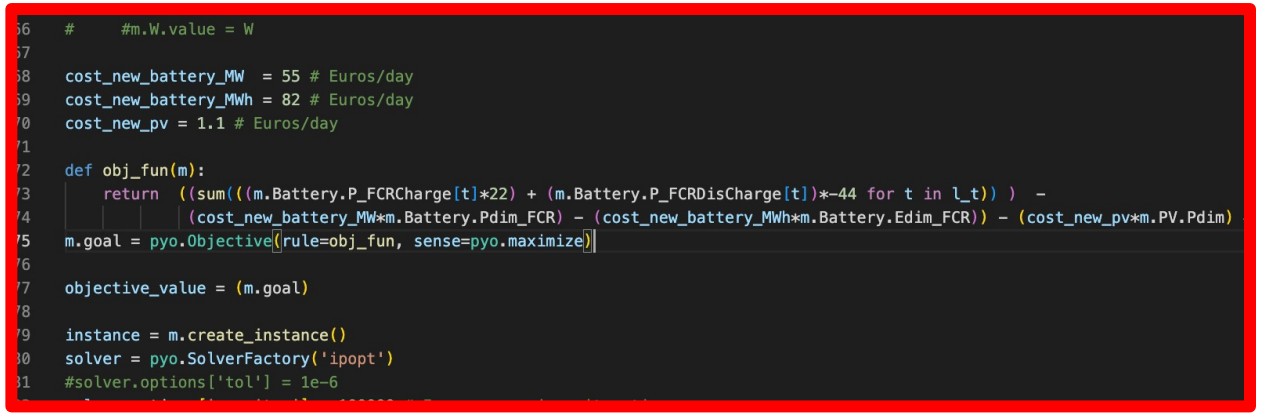
ExampleBatteryPV_Case_1_131124.py

- **Cost Parameters:** Fixed daily costs for capacities are crucial to evaluate financial feasibility of the battery system.
- **Time Steps:** We can customize them in the Excel file, for example, in **ExampleBatteryPV_time**, and then declare them in the main file: **ExampleBatteryPVnishat_MV_300724(Working File)**.
- **Decision Variables:** Key decision variables **Power** and **Energy Capacity** of the battery for optimization.



Optimization Workflow

- **Input Data Parsing:** This initial phase involves organizing and cleansing datasets for accurate model input during optimization.
- **Model Building with Pyomo:** Utilizing Pyomo, models integrate various components, defining relationships crucial for effective energy management solutions.
- **Results Extraction and Analysis:** Post-optimization analysis evaluates outputs, assessing model performance against predefined benchmarks for continuous improvement.



```
#     #m.W.value = W

cost_new_battery_MW  = 55 # Euros/day
cost_new_battery_MWh = 82 # Euros/day
cost_new_pv = 1.1 # Euros/day

def obj_fun(m):
    return  ((sum(((m.Battery.P_FCRCharge[t]*22) + (m.Battery.P_FCRDisCharge[t])*-44 for t in l_t)) )  -
             (cost_new_battery_MW*m.Battery.Pdim_FCR) - (cost_new_battery_MWh*m.Battery.Edim_FCR)) - (cost_new_pv*m.PV.Pdim)
m.goal = pyo.Objective(rule=obj_fun, sense=pyo.maximize)

objective_value = (m.goal)

instance = m.create_instance()
solver = pyo.SolverFactory('ipopt')
#solver.options['tol'] = 1e-6
```

Data Handling & Solver

- **Data Handling Sources:** The project utilizes external JSON files for structured data input, enhancing accessibility and integration.
- **Solver Configuration:** Ipopt settings include customized max iterations.
- **Execution of optimization:** Execute the constraints and the main file.

Conclusion

- **Integrated Model Overview:** The comprehensive framework combines battery storage, PV systems, and grid interaction for optimized energy management.
- **Benefits of Integration:** Enhanced efficiency, cost reduction, and improved reliability are key benefits observed from the integrated model implementation.
- **Next Steps:** Focus on further testing and optimization will ensure sustained improvements in model performance and scalability.

Fraunhofer IGD

Thank you for listening