# How to run the code:

**Step 1:**

Unzip the folder on your local machine.

You could also pull the code using the below link:

<https://github.com/neilrohra14/SmartestEnergy.git>

The code is in the master branch.

**Step2:**

Open terminal on your system and change your directory to the SmartestEnergy folder.

**Step 3:**

Run the below command in the terminal to install all the pre-requisites needed for running the code. This includes creation of any folders if needed as well as installation of packages.

*pip install -r requirements.txt*

**Step 4:**

To run the code, use the below command,

*python -m src.pipeline.process*

or we can also access the entire report generation process using the API. To run the server, use the below command.

*python application.py*

use this command to access this API from the web browser or use an API interface like postman. Change the date as needed. If no date is passed, the solution runs for the previous day.

*http://127.0.0.1:5000/api/daily-report?date=2024-10-10*

This process will write the report to an excel file and open it automatically once completed.

**Step 5:**

To run all the tests in the solution, run the command

*python -m unittest discover -s tests*

To run a specific test in the solution

*python -m unittest tests/test\_external.py*

### Details of the Assignment:

* The code functions both as a server and a standalone Python script.
* Initial API analysis was conducted in a Jupyter notebook, including tests to confirm that the sell and buy prices are identical to over 40 decimal points.

### Process Followed:

1. **API Call**: External API calls are encapsulated in a dedicated file to ensure separation of third-party integrations.
2. **Data Cleaning**: The response data is cleaned using a configuration file to extract required columns, along with quality checks (e.g., datatype validation, handling missing values).
3. **Data Storage**: Cleaned data is saved in CSV format.
4. **Time Series Analysis**: Time series for systemSellPrice, systemBuyPrice, and netImbalanceVolume are generated, both individually and combined.
5. **Imbalance Summary**: Key metrics are calculated and saved in different sheets of an Excel file.
6. **Automation**: The Excel file opens automatically upon completion of the process.

### Additional Future Analysis:

* **Price Sensitivity to Imbalance Volume**: Plotting imbalance volume against buy/sell prices to explore the relationship.
* **Hourly Price Volatility**: Analysing hourly price fluctuations (effective when buy and sell prices differ).
* **Daily Imbalance Trends**: Monitoring the daily net imbalance (deficit or surplus).
* **Price Moving Average**: Calculating 3/4/5-hour moving averages to assist in predicting future trends.

### Pending tasks:

* Generating tests for all functions within the project.   
  Currently tests are present for only external.py