Instructions to use the Infinity Train simulation code

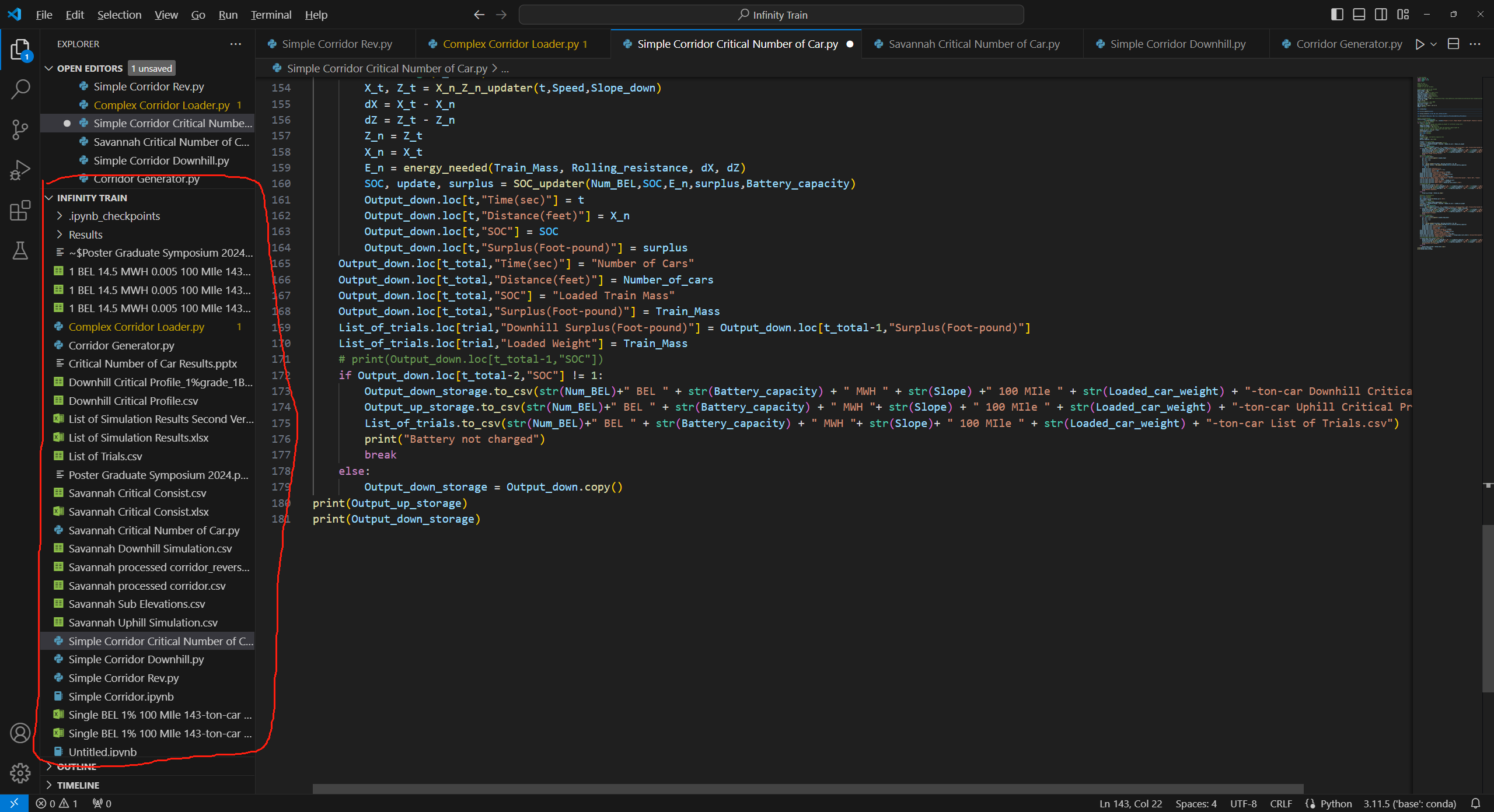
General Setups

VS Code or Jupyter Notebook need to be installed and running

If you are in **VS Code**:

Do open folder-> click on the Infinity Train folder (without opening it and enter lower levels) -> Click Select Folder

This will lead you to this working space, make sure you can see these files circled in red on your left



Click on the play button “Run Python File” to run the script “**Simple Corridor Critical Number of Car.py**”, with different setup, this should get you results of critical number of cars your particular setup can operate without charging.

If you are in **Jupyter Notebook**:

Activate your Jupyter Notebook and go to the location you placed the infinity train folder. Then you can click on this folder circled and access all the files within Jupyter Notebook.

Note: You cannot put these in hard drives other than C, downloads and desktop should work.

Open “**Simple Corridor Critical Number of Car.ipynb**” to use the code, click on Run button to run the code, the results should be the same as what VS Code would provide.

Instructions for using the **Simple Corridor Critical Number of Car:**

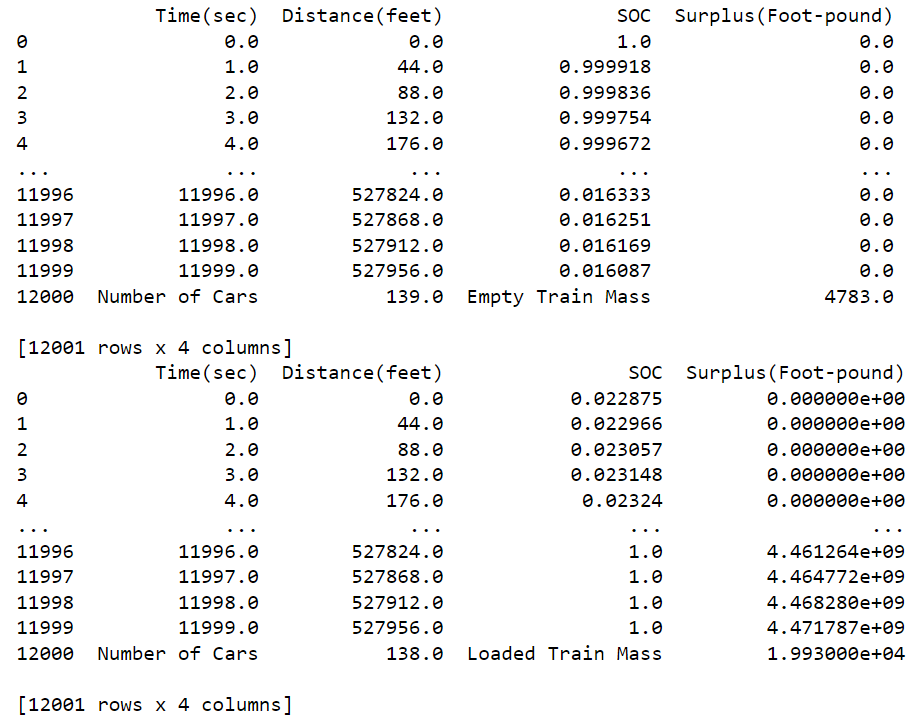
This is a simulation code that automatically build a uniform grade corridor by specified length and grade. The current default setup is 100 miles in length with three different grade selections: 0.5%; 1.0%; 1.5%. There are three different empty & loaded car weight standards:

|  |  |
| --- | --- |
| Empty Car Weight (Tons) | Loaded Car Weight (Tons) |
| 31.5 | 131.5 |
| 33 | 143 |
| 32.5 | 157.5 |

There are four different BEL battery capacities proposed: 2.4 MWh; 4.8 MWh; 9.6 MWh; 14.5 MWh.

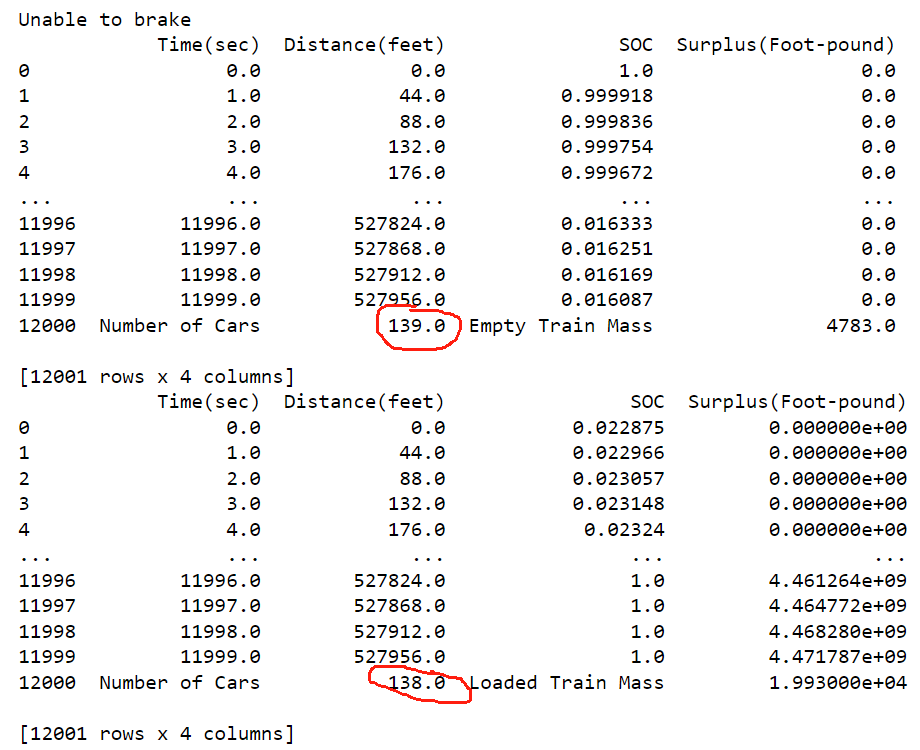
Therefore, user need to specify the setup with one of these numbers for each variable mentioned above. User need to specify these parameters under the “#Uphill Initialization “, parameters need to be changed are “**Empty\_car\_weight**”, “**Loaded\_car\_weight**”, “**Slope**”, “**Battery\_capacity**”. To get the critical number of cars for each setup, user need to run a simulation for upper limit (maximum number of cars) and a lower limit (minimum number of cars). The upper limit is achieved by setting the operating sign in this line as plus; the lower limit is achieved by setting the operating sign to be minus.

After each simulation run, a pair of recording data frames will be printed at the end:

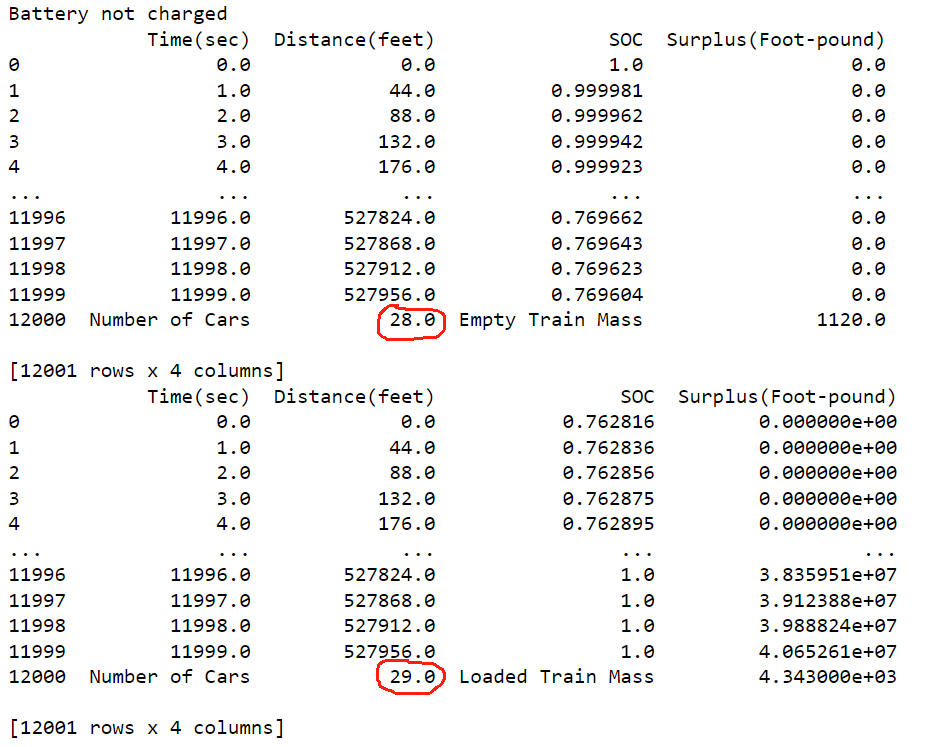


First one for uphill simulation and the other for downhill. If you don’t see one of the error message printed directly before these two, this means all trials for this range of cars are feasible, therefore a larger range is needed by changing the 120 number in this line 

The expected printed messages for finding the critical number of cars are: “Unable to brake” or “Battery not charged” or “Battery depleted”. For the upper limit, the limit is the lower number of the two numbers listed in the two dataframes printed at the end.



Likely, for lower limit, the minimal number of cars is the larger number of cars of the two dataframes.



Make sure you record these numbers in a separate file on your own to document the critical number of cars for each setup. Note that there could be setups without a feasible range, which will result in NA for both maximum and minimum number of cars.

At the end of day, a summary of critical number of cars is needed in the following format:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grade (%) | Loaded Car Weight (Tons) | Battery Capacity (MWh) | Maximum Number of Cars | Minimum Number of Cars |
| 1 | 143 | 14.5 | 138 | 29 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |