CODE INSTRUCTIONS

- First open the iPython Notebook titled “arrival_data_final”. Once opened, run all cells. This will automatically create the arrival data .csv files that are used throughout the rest of the python notebooks. It will insert them into the empty data folder included in the zip, which is the correct location for the rest of the python notebooks to access them.
- Once this is done, the notebooks can be opened and run in any order, to simulate specific models or view the EDA/Regression code. However, listed below is the most logical order to run the notebooks in:

    1) ***eda_final.ipynb***
    2) ***regression_final.ipynb***
    3) ***model_1_final.ipynb***
    4) ***model_2_final.ipynb***
    5) ***model_3_final.ipynb***
    6) ***model_4a_final.ipynb***
    7) ***model_4b_final.ipynb***

SUPPLEMENTAL WRITE UP

While the majority of the questions asked were answered in our true final report, this document is to answer some of the questions that weren’t fully answered:

- **What did you learn through implementing the project? Any major takeaways?**
    - The biggest takeaway from this project has been in how to practically implement simulation to solve real-world problems. Simulation inherently requires replication, so it is extremely important to design code efficiently, otherwise models can take hours to run. We faced the issue throughout the course of the project and repeatedly had to redesign code so that unnecessary processes were eliminated. The resulting final versions of our code reflect efficient simulation approaches that all run in reasonable amounts of time.
- **What are the next steps in the project (if you were to continue working on it)?**

    - We plan to further our relationship with Len Testa and TouringPlans.com to gather more data. Their team seemed really impressed with our models, and actually said they modeled the time-inhomogeneous arrivals the same way when they had limited data in the beginning of the company.
- **How do the software packages you used work? Are there other approaches that might work? What are the trade-offs?**
    - We had initially used Python only to perform the regression, data generation and rejection sampling, and planned to use SIMIO for the actual modeling because of the ease of just plugging in numbers and pressing run. However, when it came to the time-inhomogeneous model, it became more difficult to import the data into SIMIO, and certain statistics we wanted to output required extra work for SIMIO. We decided to switch fully to Python for the customization capabilities and ease of export into Pandas DataFrames for plotting using Seaborn. In the end, though SIMIO might be easier and definitely a cleaner approach, we had existing codebases from homeworks in Python, and it let our inputs, outputs, and analysis mesh all in one file.
- **What were the results of your experiments**
    - Answered in report.
- **Do these results seem reasonable? Do they match intuition?**
    - Answered in report.
- **How does your project fit into the larger context of the class & IEOR field?**
    - Applying mathematical concepts to real world operations problems seems to be the perfecting application of this class and IEOR as a whole. It's easy to get bogged down in the purely mathematical side of IEOR or the theory of queuing systems, but it's only upon running the models against existing data that we can see the intersection between the two. It seems to be a balancing act: on one hand we need mathematical bases for our decisions, assumptions, and parameters in a simulation model (such as what distribution to use to model a process based on characteristics of that real world process) and can't pick these randomly. However, the flip side is also true...the theory of these systems don't seem to work in a vacuum due to our lack of arrival data or inability to model customer decision making well. Only when we bridge the two did we get realistic AND mathematically sound results.