Traffic Flow Introduction

STCA (Stochastic Traffic Cellular Automata)

- All of the cars are moved simultaneously
- Each car has a velocity $v_n < v_{max}$ which is an integer representing the number of empty spaces that car *n* will move in the next iteration
- The new velocity of car *n* is calculated by the following:
  - First we find how many empty spaces are in front of car *n*; this value is g_n
  - *If* $v_n > g_n$, set $v_n = g_n$
  - Else if $v_n < g_n$ and $v_n < v_{max}$, v_n += *1*
  - Possibility of random slowing: If $v_n > 0$ there may be a chance that the car slows randomly
    - If we're using the "cruise control" model, random slowing does not occur if $v_n = v_{max}$ and $g > v_n$
    - If we're not using the "cruise control" model or if the condition above is not met, Otherwise, there is a non-zero probability *p* that we will subtract 1 velocity unit from v_n
- Now that we have the velocity, move each car by its v_n

ASEP (Asymmetric Stochastic Exclusion Principle)

- Cars are moved one at a time
- Start by picking a car at random
- The new velocity of the car is calculated by the following:
  - Find *g* as before
  - If $v > g$, set $v = g$
  - Else if $v < g$ and $v < v_{max}$, v += 1
- With the new velocity, move the car forward

We'll start by implementing each model and then move on to analyzing the dynamics of each

- Philip and Taylor will each work on logic for one of STCA or ASEP
- STCALane and ASEPLane classes
  - Constructor with parameters for initial car positions (array of bools), max velocity, random slowing probability, and flag for "cruise control" mode or not
  - Method to step the simulation by one step
  -
  -
  -
- We will represent lanes full of cars in the [STCA|ASEP]Lane objects as arrays of STCACar or ASEPCar objects and Nones
  - We'll use NumPy arrays to hold the cars
  - ```
    >>> from numpy  import *
    >>> a = array( [car1,None,car2] )
    ```
- Cody will work on user interface
- Need to follow up with Rianna
    ```