STCA Traffic Simulation Specification

- Lane Condition list format
  - A list of integers and `None`s will represent cars with velocities and empty spaces respectively
  - Here is an example list:

    ```
    >>> [2,None,None,1,None,1,0,0,0,None,None,None]
    ```

    The list above represents a 12-space lane with 6 cars at positions 0, 3, 5, 6, 7, and 8:
    0. **Car** with velocity 2
    1. Space
    2. Space
    3. **Car** with velocity 1
    4. Space
    5. **Car** with velocity 1
    6. **Car** with velocity 0
    7. **Car** with velocity 0
    8. **Car** with velocity 0
    9. Space
    10. Space
    11. Space
  - A list in this form will be passed to the `Lane` constructor to initialize everything
- `Car` class
  - Private instance variables
    - `__vmax`
      - The maximum possible value of `__velocity`
    - `__p`
      - The probability *p* that the car will slow randomly during the right conditions
    - `__g`
      - The current value of *g*
    - `__velocity`
      - The current velocity of the car
    - `__position`
      - The zero-indexed current position of the car
    - `__next_car_ref`
      - A reference to the car ahead of this car
    - `__lane_bit_state_ref`

        - A reference to the car's lane's `__lane_bit_state` list for updating when `move()` gets called (initialized in the `Car` constructor)
    - Private methods
        - `__do_slow_randomly(self)`
            - If *g* is sufficiently small, returns `True __p * 100` percent of the time. Otherwise, returns `False`.
            - If *g* is large enough, return `False`
        - `__calc_new_g(self)`
            - Calculates the new value of *g* using the position of `__next_car_ref`
                - If `__next_car_ref` is `None`, `__g` is set to `sys.maxint`
                - Otherwise, we calculate `__g` by comparing the position of this car to the position of `__next_car_ref`
    - Public methods
        - `get_velocity(self)`
            - returns `__velocity`
        - `get_position(self)`
            - returns `__position`
        - `get_next_car(self)`
            - returns `__next_car_ref`
        - `set_next_car(self,car)`
            - Sets the value of `__next_car_ref` to `car`
        - `calc_new_velocity(self)`
            - First updates the value of *g* by calling `calc_new_g()`. Then calculates the new velocity *v* using the algorithm specified by STCA. When determining whether or not to slow randomly call `__do_slow_randomly()`
        - `move(self)`
            - Calculates new position based on `__velocity` and then updates `__lane_bit_state_ref` and `__position`
    - Constructor
        - `__init__(self,position,velocity,vmax,p,lane_bit_state_ref)`
        - Sets `__position`, `__velocity`, `__vmax`, `__p`, and `__lane_bitstate_ref` to the provided values
- `Lane` class
    - Private instance variables
        - `__lane_bit_state`
            - NumPy array containing Boolean values representing the state of each "space" in the lane. `True` indicates a car at that position; `False` indicates a space at that position.

        - Example using 12-space lane from above:
          ```
          >>> array([True,False,False,True,False,True,True,
                     True,True,False,False,False])
          ```
    - `__cars`
        - Linked list containing all of the `Car` objects in the lane
- Private methods
    - `__calc_new_velocities(self)`
        - Loops through each `car` in `__cars`. Tells the car to calculate a new `velocity` by calling `calc_new_velocity()`
    - `__move_cars(self)`
        - Loops through each `car` in `__cars`. Tells the car to move to the next position by calling `move()`
    - `__remove_last_car(self)`
        - Removes the last car object from `__cars`, updates `__lane_bit_state` to reflect the removal, and sets the previous car's `__next_car_ref` reference to `None` by calling `car.set_next_car(None)`
- Public methods
    - `step(self)`
        - Calls `__calc_new_velocities()` and `__move_cars()` to step the simulation through one time step
    - `get_lane_bit_state(self)`
        - returns a copy of the list `__lane_bit_state`
    - `get_lane_velocity_state(self)`
        - Builds a new list in the Lane Condition list format based on the current velocities of each car and `__lane_bit_state`
- Constructor
    - `__init__(self,vmax,p,initial_lane_conditions)`
    - Builds `__lane_bitstate` and `__cars` using `initial_lane_conditions`. Links successive cars like this:
      ```
      >>> previous_car.set_next_car(this_car)
      ```
    - Uses `vmax` and `p` when constructing new cars for `__cars`