In-Class Work: Traffic Flow

1. Random Numbers

This task is to remind you of the commands in python to get different types of random numbers. (You did a similar task on Feb.7 when our class topic were random walks.) Copy into your working directory

`~kvollmay/share.dir/inclass2023.dir/traffic1.py`

make it executable and run this python-program. Look at the program and confirm what it is doing. You may look at the first half of the data with

```
./traffic1.py  | gawk 'NR<=100{print $1,$2}'   | xmgrace -pipe
```

and the second half of the data with one of the following two commands:

```
./traffic1.py  | gawk 'NR>100{print $1,$2}'   | xmgrace -pipe
./traffic1.py  | gawk 'NR>100{print $1,$3}'   | xmgrace -pipe
```

2. Initialize Car Positions

Copy `traffic1.py` into `traffic2.py` and initialize an array of name `road` and of length `ROADLENGTH=20` such that each lattice site is occupied by a car (for now value 1) with probability `PCAR=0.3` and otherwise the site is unoccupied (value -1). Check your code by printing the road.

To look up python syntax you may use the example files

`~kvollmay/share.dir/pythonsamples.dir/sample_*.py`

and any previous inclass work or of your main project.

3. Initialize Road

Copy `traffic2.py` into `traffic3.py` and modify the program to put on each site of the `road` array a car with probability `PCAR` and give each car on the road a random velocity $v \in \{0, 1, \ldots,$ `VMAX` $\}$ (use `VMAX=5`). Each of these velocities is equally likely.

4. Distance (if time)

Please get me before you start working on the following so that I can explain the next steps.

4a. Next we work on finding the distance between a car and the car in front of it. To get this distance we will avoid in the following to have to check each site in front of a car if it is empty or not. Instead we define an additional array: `carpos` which stores for each car on the road its position on the road. So if the first car (index 0) on the road is on site 3 then `carpos[0]=3`, if the second car is on site 5 then `carpos[1]=5`, etc.. Add to your program of 3. the array `carpos`, initialize `carpos` in the same loop when you initialize the `road` array, and check your program by printing out both the complete `road` and the `carpos` array.

4b. Now add to your program (after the initialization of `road` and `carpos`) a loop over cars on the street determine (and print out) for each car the distance to the car in front of it. Take into accout the periodic boundary conditions both to determine the car in front and to determine the distance.

Comment: This task is more difficult than it might seem at first glance. Please get me after having thought about this task.