In-Class Work: Traffic Flow

2. Initialize Car Positions

In last class you used `traffic1.py` to see how to get random numbers. 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 any of our previous inclass work

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

and your main project, and you may also use the example python scripts

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

3. Initialize Road

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

4. Distance

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 to 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.

5. Update Velocities

5a. For the update of the velocities we will need a function which determines the smallest number of three integers. Python has a function which does this, for example `min(3,1,9)` gives `1` as result. Test this function with a few print commands

5b. Add to the program that it determines all new velocities (and updates them in the `road` array), and prints out the road with the new velocities. Check your program with print commands.

6. Update Positions

Now you are ready to add to your program the update of the positions. To do so use a new loop over all cars and update both the `carpos` array and the `road` array. For the update of `road` make sure to first copy the new velocity into a variable (e.g. `vnew`), then empty the old site and then put the car in road on its new site `xnew = xold + vnew` with the new velocity (the order of these commands matters). Remember to use the periodic boundary condition. After the update of all positions print the complete road and check if it is what you expected.

7. Finish Program (if time)

Now you are set to finish the program for our traffic flow model. Add to your program from 6. the time loop. Print your road for the case of 15 timesteps (`MAXTIMESTEPS = 15`), `PCAR=0.3`, and similarly the other parameters the same as in `traffic4.py`.