## Getting Started

To start using the project it is relatively simple

1. Install [Python](https://www.python.org/), make sure you tick the box to add python to your system path during installation

2. Install [SUMO](https://sumo.dlr.de/wiki/Installing), make sure you tick the box to add SUMO to your system path during installation

3. Clone this repository into a local area.

4. Run any of the scenarios from the scenarios folder either by double clicking them or using the command prompt (ex. python Launch-Intersection-Scenario-1.py)

5. Explore the code base! I've tried to make it quite well documented within the code and I'd recommend reading in this order: scenario file ----> simulation manager ---> vehicle ---> platoon ---> intersection controller

Any Python compatible IDE can be used to edit the project, I used Visual Studio but there are plenty others!

More information about the different scenarios and how they work are below.

## Folder Structure

- Maps: this contains all the maps used in each scenarios, each map has a traffic light version for platooning scenarios and a non-traffic light version for the CIM scenarios. This also contains route information determining how may vehicles spawn in each scenario.

- output: this is where any outputs from running the simulation will be saved, it only contains a file which determines which data is generated within this repository.

- scenarios: this is where the Python files which execute the different scenarios reside

- src: this is where the Python code which deals with platoons, CIM, vehicles and the simulation resides. This code is run to determine how vehicles in each scenario behave.

## Scenarios

- Scenario 1: the control - nothing added just normal SUMO running

- Scenario 2: platooning with normal Traffic Light Systems

- Scenario 3: platooning with CIM

- Scenario 4: platooning with CIM where platoons zip rather than go one by one.

## Purpose of python files

- intersectionController: all functions for controlling vehicles and platoons while they approach a traffic light system (used during CIM only)

- platoon: contains information and functions concerning an individual platoon within the simulation. This contains

information such as all cars in the platoon, who the leader is, current speed, length etc. It also acts as a basis for all the functions needed by

other aspects of the simulation to change a platoon’s behaviour, this could be setting a target speed, merging with another platoon or disbanding it

altogether. Each platoon class also maintains the speed of the platoon ensuring that all vehicles are adhering to the speed set by the platoon leader

who follows the normal SUMO vehicle following model and acceleration models.

- simlib: contains library functions created for this project

- simulationManager: with every loop this class creates platoon and vehicle objects,

places vehicles joining the simulation into any eligible platoons, keeps track of all platoons and vehicles in the simulation and

deactivates any vehicles that leave it. It also calls the update functions of every platoon so that they can update their statuses and speed.

- vehicle: contains getters and setters for a traci vehicle