# Explanation for the Starter Code
# And
# Brief Comparison of motion_planing.py and backyard_flyer_solution.py

**Definitions Used:**

A. File 1: motion_planing.py
B. File 2: planing_utils.py,
C. File 3: backyard_flyer_solution.py

File 1 is used to plan motion path and the motion plan for the drone to be programed (In the Stimulator Provided) based on seven states of motions and File 2 (planing_utils.py). The states of motion for the said drone are MANUAL,ARMING, TAKEOFF, WAYPOINT, LANDING, DISARMING, PLANNING. The transition plan for each change in transition based on seven these seven states is defined in File 1. The other things needed for the flight plan like motion path ( as defined as method plan_path())are defined in File 1.

Various utilities like implementation of different algorithms like a-star and heuristic are defined in File 2 along with methods like create_grid to create Grid needed to define motion path and motion plan for the drone to be programmed (In the Stimulator Provided).

File 3 is also used to plan motion path and the motion plan for the drone to be programed (In the Stimulator Provided) but is based on five states of motions and File 2 (planing_utils.py). The states of motion used for the said drone in file 3 are MANUAL,ARMING, TAKEOFF, WAYPOINT, LANDING, DISARMING, implying absence of PLANNING State The transition plan for each change in transition based on these five states is defined in File 3. The other things needed for the flight plan like motion path ( as defined as method plan_path() using A* Algorithms with relevant cost function and activation function)are defined in File 1.

**Q. In the starter code, we assume that the home position is where the drone first initializes, but in reality you need to be able to start planning from anywhere. Modify your code to read the global home location from the first line of the `colliders.csv` file and set that position as global home (`self.set_home_position()`)**

This goal is achieved by adding Line numbers 124to 128 in File 1 and by adding line number 132 in File 1.

**Q. In the starter code, we assume the drone takes off from the map center, but you'll need to be able to takeoff from anywhere. Retrieve your current position in geodetic coordinates from `self._latitude`, `self._longitude` and `self._altitude`. Then use the utility function `global_to_local()` to convert to local position (using `self.global_home` as well, which you just set)**
The goal is achieved by adding Line number 132 and Line Number 139 to File 1.

**Q.In the starter code, the `start` point for planning is hard coded as map center. Change this to be your current local position.**
The goal is achieved by adding Line number 153 to File 1.

**Q. In the starter code, the goal position is hardcoded as some location 10 m north and 10 m east of map center. Modify this to be set as some arbitrary position on the grid given any geodetic coordinates (latitude, longitude)**

This goal is achieved by adding line numbers 162 and 165. In File 1.

```
grid_goal = global_to_local((-122.401247,37.796738,0),self.global_home)
grid_goal = (int(grid_goal[0]+ north_offset),int(grid_goal[1]+ east_offset))
```

The local coordinates are retrieved using global_to_local from File 2. The user can also select their goal from running the script with arguments by adding two arguments in line number 200 and 201:

```
parser.add_argument('--lat', type=float, default=1000, help="latitude")

parser.add_argument('--lon', type=float,  default=1000, help="latitude")
```

**Q. Write your search algorithm. Minimum requirement here is to add diagonal motions to the A* implementation provided, and assign them a cost of sqrt(2). However, you're encouraged to get creative and try other methods from the lessons and beyond!**

The diagonals movements were implemented by adding them to the action enum by adding diagonal directions with cost function of square root 2. The valid_actions method is also modified to reflect the same additional actions.

**Q. Cull waypoints from the path you determine using search.**
The path was pruned using collinearity(collinearity_prune function added in File 2: planing_utils.py).

**Q. Executing the flight**

Flight Can be tested using the command for example:

python motion_planning.py --lat xyz --lon abc
**where ,**
xyz and abc are respective coordinates for the goal.