VehicleManager.py：车辆的检测、分组和关系查找

遍历所有车辆，按 lane\_index[0] 分类为主线（a）或匝道（"j"）车辆。返回主线车辆列表、匝道车辆列表和 ego 车辆对象。

查找与本车同车道（或匝道车查主路车）的所有车辆，找到前方最近的那一辆，作为leader。

找出主路上距离汇入区起点前后50米范围内的车辆，作为与合流相关的主路车辆。

对主路车辆按与汇入点的距离排序；找到距离汇入点最近的、在 ramp 车前方的主路车（leader），以及最近的、在 ramp 车后方的主路车（follower）；可判断匝道车能否安全插入主路。

planner.py：主要是对每辆车进行延迟补偿，规划目标速度和安全间距，计算合流加速度。

首先根据通信延迟预测当前车的速度和位置；根据最小安全距离（设定值）、速度补偿项和跟车反应距离计算动态安全车间距；

控制距离的逻辑（手动控制）主要是：如果实际距离小于安全距离的 0.8 倍，说明太近，目标速度设为前车速度的 0.95 倍（减速）如果实际距离大于安全距离的 1.2 倍，说明太远，目标速度设为前车速度的 1.05 倍（加速，但不超过 35）。否则，目标速度等于前车速度（跟驰）。目标速度限制在 [25, 35] m/s。

最后控制理想插入点

executor.py：

先检查是否要紧急避撞，如果与前车距离小于安全距离的0.7倍，强制大减速，防止追尾。否则根据规划速度调整目标速度，速度变化大时平滑加减速，变化小则直接设定。

控制匝道车，计算与主路前后车的间隙和所需安全间隙。如果间隙够，加速准备汇入，距离汇入点近时用最优加速度插入，否则跟随后车略加速。如果不够，减速等待，距离很近时大幅减速，防止强行插入。

控制主路车辆，如果即将到达汇入点（距离<30米），提前降低目标速度，为匝道车让出空间，提升合流安全性。

delay\_model.py：

记录更新每辆车的发送时间、接收时间和单程延迟；

判断车辆是否在指定的延迟估计区间内；、

查询某辆车的当前延迟估计，若无历史则返回默认值0.05秒

history.py：

记录误差：速度误差：对每辆车，计算实际速度与计划速度的相对误差，累加后取均值；间距误差：对每对前后车，计算实际间距与计划间距的相对误差，累加后取均值；历史记录：每步都将当前误差、速度、间距等写入历史字典。

计算当前步的平均速度误差和间距误差（相对误差）。

统计整个仿真过程的平均速度误差和间距误差

（这个地方用ai改了很多次越改越复杂，但是省略了经常会撞车……）

controller：调用模块的集成部分

main：运行这个代码

调用controller.update()，完成感知-决策-执行-记录；获取ego车辆状态，判断是否接近汇入点，决定变道或保持；执行环境env.step(action)；渲染当前帧并保存；检查是否有车辆发生碰撞，若有则警告；若仿真结束，跳出循环。

使用matplotlib绘制三幅图：仿真场景最后一帧；各车辆目标速度与实际速度对比曲线；速度误差与间距误差随时间变化曲线，并标注15%阈值线。

打印仿真帧数、平均速度误差和平均间距误差。