#4 可达性评估

数据文件：

accessibility.csv——从对应 o_zone 到目标 d_zone 在需求时间内按畅通旅行时间所能走行的距离.
accessibility_aggregated.csv——不同 agent 种类从 zone 的质心出发，在给定时间内所能达到的最远点.

名词解释：
可达性——假设一点为起始点，在给定交通条件下，给定时间内所能到达的最远距离所组成的范围.
Centroids——zone_id 映射到其质心坐标后的结果.
TT——Travel Time.
AT——Agent Type.

代码文件：

accessibility.py——内含评估可达性与输出对应结果的方法.

主函数：

evaluate_accessibility(ui,multimodal=True,mode='p',time_dependent=False,demand_period_id=0,output_dir='.')
multimodal——是否开启多模式可达性评估，如果开启，评估可达性时就按照设置文件中所有可能的交通模式进行评估，如果不开启，只按照目标 mode 进行可达性评估.
time_dependent——是否开启时间依赖，开启则使用需求时段的畅通旅行时间评估可达性，关闭则使用 Link 的长度与畅通旅行速度评估可达性.
demand_period_id——需求时间段的 ID，目前只有一个时间段.

_update_min_travel_time()——更新最小旅行时间.其中用到了“更新广义旅行费用”与“求最短路”两个方法，在所有的 min_tt（所有 zone 到 zone 的旅行时间）中筛选出最大的一项.

```
            if min_tt < MAX_LABEL_COST and max_min < min_tt:
                max_min = min_tt

    return max_min
```

```
if multimodal:
    ats = base.get_agent_types()
    for at in ats:
        #在所有agent_type里再筛选出最大的Min_tt
        an.set_target_mode(at.get_name())
        max_min_ = _update_min_travel_time(an,
                                           at,
                                           min_travel_times,
                                           time_dependent,
                                           demand_period_id)
        if max_min_ > max_min:
            max_min = max_min_
```

外层循环从所有 agent_type 中寻找最大的 min_tt.

_get_interval_id(t)——计算要输出的时间间隔区域个数.
（最后输出的['TT_10', 'TT_15', 'TT_20', 'TT_25', 'TT_30', 'TT_35', 'TT_40', 'TT_45', 'TT_50', 'TT_55', 'TT_60', 'TT_65', 'TT_70', 'TT_75', 'TT_80', 'TT_85', 'TT_90']表示的值（在 accessibility_aggregated.csv 中）是从出行开始，到出行时间的第 19 个 Interval 所能

到达的区间个数.“TT_10”表示据出行开始的第 10 个 Interval.）