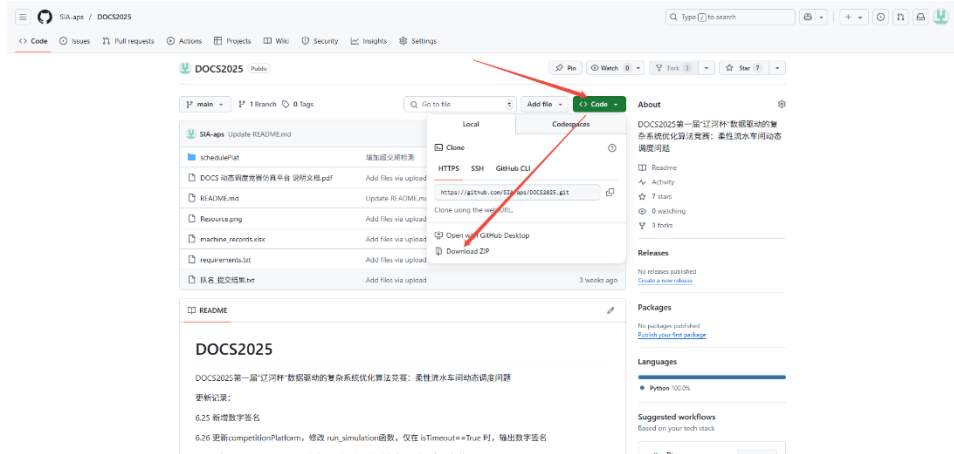
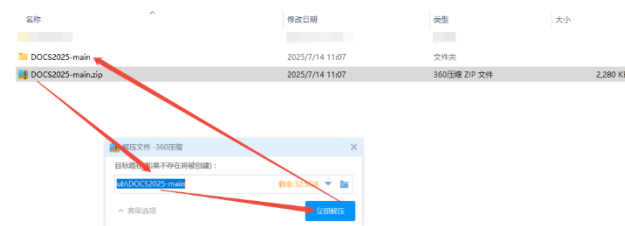


(一) 下载代码

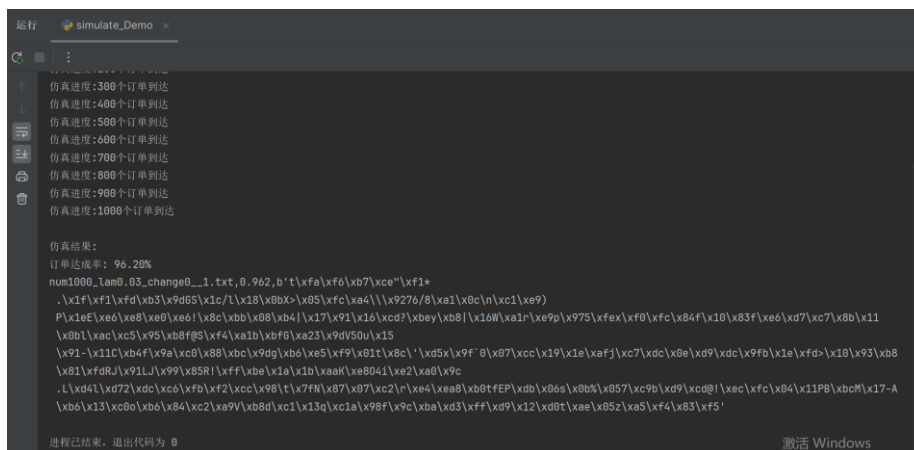
1. 登录 GitHub: <https://github.com/SIA-aps/DOCS2025>
如果您无法登录网站, 请联系赛事人员索要代码。
2. 点击“Code”, 点击“Download ZIP”, 下载代码



3. 解压, 获取完整的工程文件



4. 运行
在您的开发环境中, 运行“schedulePlat”目录下的“simulate_Demo.py”。
如果正常运行, 说明您的代码已经正确下载。



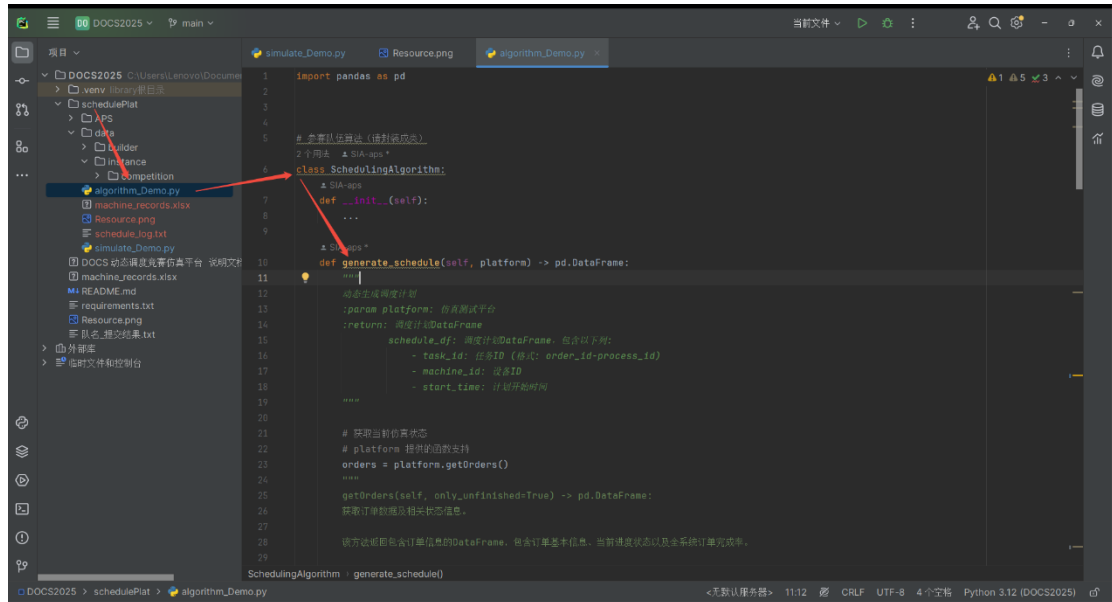
注: 这行代码显示错误是正常情况, 不影响您的运行。



(二) 编写参赛代码

1. 在“schedulePlat”目录下的“algorithm_Demo.py”中编辑您的参赛代码。

注：其他的后缀为.pyc 的代码，均为仿真平台代码，不需要您编辑。



2. 仿真平台原理

仿真平台采用**事件驱动**的重调度架构，其核心流程如下：

- a) 事件触发

当新订单动态到达时，平台将自动生成一个决策点（Decision Point）。

- b) 算法调用

决策点触发后，平台立即调用参赛者算法类 SchedulingAlgorithm 中的 generate_schedule() 函数：

```
def generate_schedule(self, platform) -> pd.DataFrame:
    """
    动态生成调度计划
    :param platform: 仿真测试平台
    :return: 调度计划DataFrame

    schedule_df: 调度计划DataFrame, 包含以下列:
        - task_id: 任务ID (格式: order_id-process_id)
        - machine_id: 设备ID
        - start_time: 计划开始时间
    """
```

- c) 计划生成与输出

该函数需输出决策点后的重调度计划，以结构化 DataFrame 呈现，包含三列关键信息：

****task_id****: 任务唯一标识，格式为 M-order_id-process_id（例如：M-1-0，表示订单 1 的生产阶段 0 的任务）。

****machine_id****: 设备资源分配 ID

****start_time****: 任务的计划开始时间。

3. 可能用到的函数：

在您开发参赛代码的过程中，可能会需要调度下列由仿真平台提供的函数：

a) 当前订单信息:

<code>getOrders(self, only_unfinished=True) -> pd.DataFrame</code>
<p>获取订单数据及相关状态信息。</p> <p>该方法返回包含订单信息的 DataFrame，包含订单基本信息、当前进度状态以及全系统订单完成率。</p> <p>Args:</p> <ul style="list-style-type: none"> <code>only_unfinished (bool, optional)</code>: 是否只返回未完成的订单。默认为 <code>True</code>。 <ul style="list-style-type: none"> <code>True</code>: 仅返回尚未完成的订单 <code>False</code>: 返回所有订单（包括已完成的） <p>Returns:</p> <p><code>pd.DataFrame</code>: 包含订单信息的 DataFrame，包含以下列:</p> <ul style="list-style-type: none"> <code>order_id</code>: 订单唯一标识 <code>product_type</code>: 产品类型 <code>arrival_time</code>: 订单到达时间 <code>due_date</code>: 订单交期 <code>current_stage</code>: 当前处理工序（如果当前没有进行中的工序则为上一个工序的结果。若首工序也未开始，则为 <code>None</code>） <code>assigned_machine</code>: 分配到的机器 ID（如果当前没有分配则为上一个工序的分配结果。若首工序也未开始，则为 <code>None</code>） <code>start_time</code>: 当前工序开始时间（如果未开始则为上一个工序的开始时间。若首工序也未开始，则为 <code>None</code>） <code>end_time</code>: 当前工序结束时间（如果未开始则为上一个工序的结束时间。若首工序也未开始，则为 <code>None</code>） <code>fulfillment_rate</code>: 当前系统订单达成率（已完成订单/所有已到达订单）

b) 当前机器状态:

<code>getCurrentMachineStatus(self) -> pd.DataFrame</code>
<p>获取当前时刻所有机器的状态信息。</p> <p>该方法返回一个 DataFrame，描述在当前时间点各机器的状态（空闲或正在执行的任务信息）。</p> <p>Returns:</p> <p><code>pd.DataFrame</code>: 包含每台机器当前状态的 DataFrame，包含以下列:</p> <ul style="list-style-type: none"> <code>task_id</code>: 当前执行的任务 ID（如果机器空闲则为 <code>None</code>） <code>start_time</code>: 当前任务的开始时间（如果机器空闲则为 <code>None</code>） <code>end_time</code>: 当前任务的结束时间（如果机器空闲则为 <code>None</code>）

c) 产品加工时间:

<code>getMBOM(self) -> pd.DataFrame:</code>
<p>获取制造 BOM(Bill of Materials)信息。</p> <p>该方法从仿真实例中提取产品的工艺路线信息，包括每个产品在各生产阶段可选用的设备和相应处理时间。</p> <p>Returns:</p> <p><code>pd.DataFrame</code>: 包含制造 BOM 信息的 DataFrame，包含以下列:</p> <ul style="list-style-type: none"> <code>product_type</code>: 产品类型标识（格式为"0"、"1"、"2"等） <code>stage</code>: 生产阶段标识（格式为"0"、"1"、"2"等） <code>machine_id</code>: 可用于该工序的设备 ID <code>process_time(s)</code>: 在该设备上完成该工序所需的处理时间（秒）

d) 当前时刻:

<code>getSimulationTime(self) -> float:</code>
<p>获取当前仿真时间（从仿真开始起经过的时间）。</p> <p>该方法计算从仿真基础时间点（通常是仿真启动时间）到当前时刻经过的时间。</p> <p>Returns:</p> <p><code>float</code>: 仿真经过的时间（以秒为单位）</p>

4. Debug

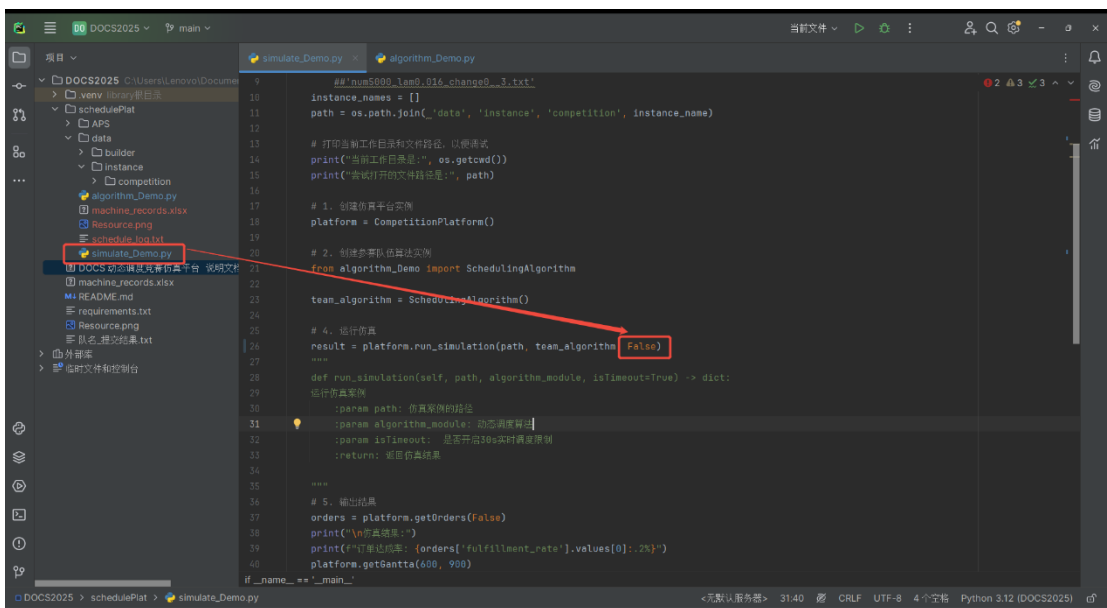
当您完成参赛代码编写后，运行“schedulePlat”目录下的“simulate_Demo.py”。如果未出现报错，并输出订单达成率，则说明算法已经成功通过测试。

```
仿真进度:500个订单到达
仿真进度:600个订单到达
仿真进度:700个订单到达
仿真进度:800个订单到达
仿真进度:900个订单到达
仿真进度:1000个订单到达

仿真结果:
订单达成率: 96.20%
num1000_lam0.03_change0_1.txt,0.962,非评测环境

进程已结束,退出代码为 0
```

注意：在 debug 时，请将“simulate_Demo.py”中的“True”改为“False”，否则可能因为“超时”而报错。



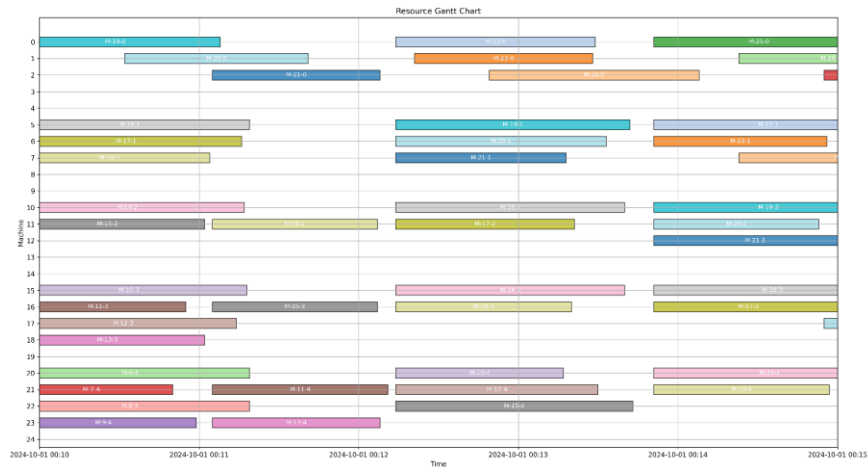
5. 更详细的结果查看

仿真平台输出了完整的作业计划，在“machine_records.xlsx”中。

task_id	start time	end time
M-149-0	4889	4908
M-462-0	15687	15746
M-967-0	32169	32231
M-31-0	986	1065
M-334-0	10700	10762
M-313-0	10078	10140
M-573-0	18622	18697
M-204-0	6404	6460
M-199-0	6324	6403
M-467-0	15263	15338
M-768-0	25372	25451
M-754-0	25274	25336
M-652-0	21645	21727
M-141-0	4637	4699
M-725-0	23987	24066
M-207-0	9868	9947
M-365-0	11937	12016
M-388-0	12852	12931
M-718-0	23748	23823
M-529-0	17195	17287
M-977-0	32504	32579
M-722-0	23900	23975
M-29-0	1135	1197
M-550-0	17854	17943
M-789-0	26568	26643
M-292-0	9440	9515
M-630-0	20846	20921
M-262-0	8277	8339
M-252-0	8008	8087
M-827-0	28241	28316
M-909-0	30407	30486
M-591-0	19248	19323
M-349-0	11327	11406
M-962-0	18160	18222
M-108-0	3355	3417
M-497-0	16045	16107
M-981-0	32839	32901

其中，每个工作表记录了一台机器的完整作业记录。

此外，还可以查看某时段的机器甘特图。



通过调用 `getGantt(self, startTime, endTime)`函数实现。

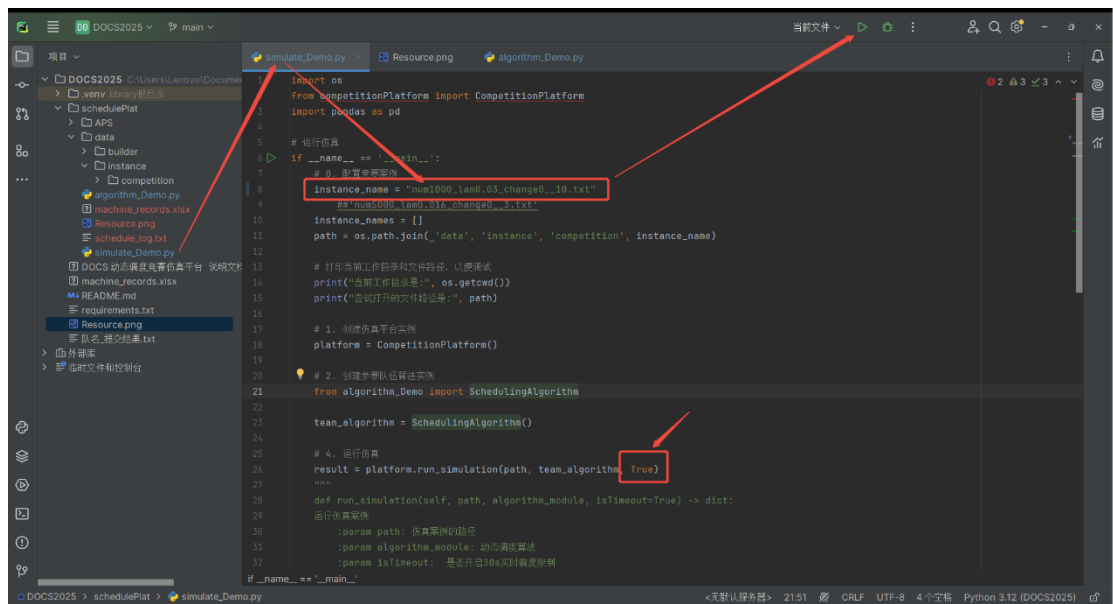
```
def getGantt(self, startTime, endTime)
生成甘特图
:param startTime: 开始时间点
:param endTime: 结束时间点
```

(三) 提交比赛结果

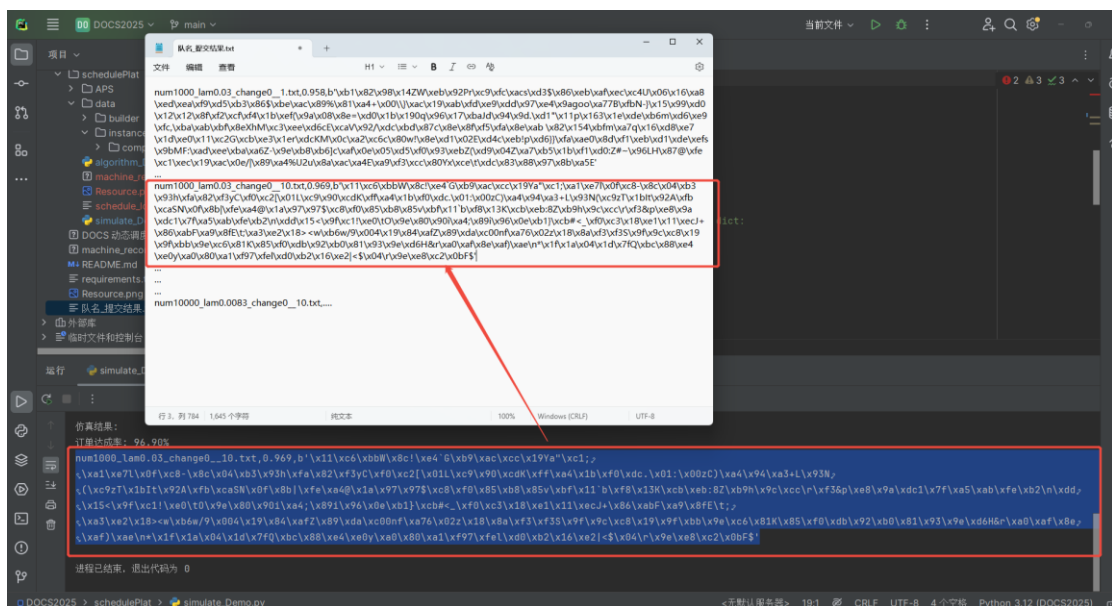
1. 在全部算例下进行仿真

在“simulate_Demo.py”中，修改参数“instance_name”，配置不同的比赛实例，并运行。

注意!!! 此时要将“simulate_Demo.py”中 `platform.run_simulation` 函数的参数“False”改为“True”



2. 将每个 instance 的输出结果，保存在“队名_提交结果.txt”中，将用于初赛的最终评审。



关键要求：

- 必须包含所有测试实例的结果（缺一不可）
- 实例顺序不限，但每行对应一个完整实例
- 不可修改签名内容（任何修改都将导致验证失败）
- 数字签名需保持原始格式（含开头的'b'和结尾的''

3. 提交规范

A. 必交内容

- 主算法文件（队名_Algorithm.py）：必须包含完整的 SchedulingAlgorithm 类实现
- 说明文档（队名_doc.pdf）：简述算法思路和关键函数
- 依赖文件(requirements.txt)：列出非系统级 Python 库
- 初赛仿真结果（队名_提交结果.txt）：由竞赛仿真平台输出的结果，用于初赛评审。

B. 提交方式：

- 邮箱提交：
 - mail: wangrui@sia.cn
 - 提交压缩包命名：团队名_提交日期.zip

(四) 赛事人员联系方式

联系电话：18202411758

邮 箱：wangrui@sia.cn

企业微信：

