对于已经书写玩的攻击制造类，我们下面给出文档：

首先给出文档目录：

一、项目文件架构

----> code

---->---->AbnormalCreateClass.py 制造攻击主类

---->---->AbnormalDescriptionClass.py 专门用于记录攻击信息的类

---->---->BasicClass.py 一些功能性的类

---->---->CodingTypeChange.py 二进制和十六进制转换

---->---->LoadingDataClass.py 专门用于数据导入类

---->---->main.py 主函数

----> src

---->---->attack_test 存储生成的攻击文件

---->---->attackDescription 存储对应的攻击描述文件

---->---->learnedRule 存储域分界算法得到的数据字段规则

---->---->testData 用于测试的数据

---->---->webOriginData 模拟真实环境中的数据字段，所有处理字段均来自这里

二、制造攻击类的使用示例

这里默认的都是3s的时间片数据，已经存储为标准格式。

首先导入数据

```
path1 = "../src/testData/0.csv"
path2 = "../src/testData/1.csv"
loadDataExample = LoadDataClass()
loadDataExample.loadDataSnippet(path1, path2)

```

接下来再把数据导入生成攻击类

```
attackCreateExample = AttackCreate()

attackCreateExample.sourceDataSnippet =
loadDataExample.sourceDataSnippet
attackCreateExample.historyNormalDataSnippet =
loadDataExample.historyNormalDataSnippet

```

之后，我们就可以直接开始制造攻击了

```
# def insert_attack(self, id, normal_T, ratio, exist_time)
attackCreateExample.insert_attack('82', 0.01, 2,0.2) #注入攻击
# def erase_attack(self, id, exist_time):
attackCreateExample.erase_attack('82', 0.2) #删除攻击
# def reput_attack_SingleId(self, id, exist_time):
attackCreateExample.reput_attack_SingleId("82", 0.2) #重放单个id的攻击
attackCreateExample.reput_attack_AllData(0.1) #重放所有数据的攻击
attackCreateExample.changedatafield_attack_randomly('82', 0.2) # 随意修
改datafield的攻击

#

# attackCreateExample.changedatafield_attack_const_or_multivalue('565',
0.2)
# attackCreateExample.get_rule(" ")

# 以下是使用攻击定义类来进行sensor修改攻击的实例

```

```
# 更加的细节建议看源代码
valueAttackInfo = DataFieldAttackInformation()
valueAttackInfo.attackChoseType = valueAttackInfo.sensorAttack
valueAttackInfo.relatedThing = valueAttackInfo.max_value_attack
attackCreateExample.change_data_field(valueAttackInfo, '82', 0.02)

```

> src

这两个文档是足够的

三、项目架构

这里的架构是非常体现功力的，代码功力不够会无法掌控。