券商客户端自动化测试库

一、简介

该项目是一个实验项目，旨在深度学习和挖掘 python pywinauto 库的功能和潜力，实现对中国境内券商客户端的自动化操作，完成自动化交易。该项目采用“客户端驱动型工厂模式”设计，所以从理论上讲，该项目具备了支持所有券商客户端及其未来版本的能力。

二、困难

编写这种 GUI 自动化测试是困难的，原因在于不确定，主要表现在以下三方面：

1、环境不确定。客户端所处的运行环境不确定，如有一些软件经常会遮挡客户端，客户端运行的电脑性能会影响客户端的稳定。

2、自身不确定。客户端运行本身很不稳定，控件或隐或现，弹窗类型不确定，行为不固定。

3、未来不确定。券商的客户端很明显会升级，这会带来最大的不确定性，往往是调试好一个版本时，客户端就升级了。

三、实盘

正是由于上述不确定性，所以不建议将该软件应用于实盘交易，可以用来学习参考 pywinauto 库的应用技巧。

四、版本

券商客户端自动化测试库（pytradecn0.0.1）

谁的谁（41715399@qq.com）

五、声明

本软件遵守“MIT License”开源协议开源，仅供学习和参考。您可以自由使用或修改源代码或二进制文件，但必须保留上述版权声明。该软件旨在深度学习和挖掘 python pywinauto 库的功能和潜力，由于环境的不确定性和该软件的不可靠性，请不要将该软件应用于实盘交易。如您确需量化交易实盘功能，请使用券商提供的量化交易平台，否则由于您使用该软件实盘交易所造成的账户损失或政策风险，开源软件提供者或插件提供者均不承担任何责任。同时，无论是直接的、间接的、偶然的、潜在的因使用该软件所造成的账号安全损失、数据安全损失、账户资产损失或其他任何责任事故，开源软件提供者或插件提供者均不承担任何责任。请不要将该软件应用于商业活动，否则由于把该软件应用于商业活动所造成的一切损失或法律责任，开源软件提供者或插件提供者均不承担任何责任。

六、项目特色

该项目是始于 2023 年初的一个实验项目，由于个人的原因而编写，该项目是笔者第一个 python 项目，错误在所难免。该项目通过使用 python 第三方库 pywinauto，实现对券商客户端的自动化操作测试，包括自动登录、验证码识别、买卖、撤单、查询等功能。该项目深度应用 pywinauto 库，代码中有许多 pywinauto 库的应用技巧，包括该库存在的 BUG 也已在代码中标明。该项目具有以下特点：

1、该项目采用一种“客户端驱动型工厂模式”设计，或者叫“客

户定制式工厂模式”。所以，从理论上讲，该项目具备了支持所有券商客户端及其未来版本的能力。

2、由于交易的严谨性和严肃性以及客户端行为的不确定性，所有的操作均有返回值，要么成功要么失败，不会因运行时错误而“卡”在半路。

3、自动化软件测试受内外环境的影响较大，该项目以最大的可能减少内外环境的变化对软件自动化的影响。

4、由于采用“客户端驱动型工厂模式”，所以项目可扩展性高、可根据不同的券商版本制作不同的交易模型也可以制作不同的部件适应不同的场景，如制作不同的登录引擎以适应不同的登录方式等。

5、调用接口简单，一行代码就能完成对客户端的调用，同时能够完全隐藏您的客户端信息，实现隐式调用。

七、安装

1、第一步：安装 pytradecn

运行 pip install pytradecn

该项目需要以下依赖库：

pywinauto-0.6.8

pillow

2、第二步：安装 Tesseract

拷贝Tesseract文件夹到pytradecn.utils包内或修改ocr.path

下载地址：https://github.com/tesseract-ocr/tesseract

注意：不要忘记下载简体中文支持包。

八、使用之前

使用之前应该尽量净化您的客户端运行环境，以保证您的券商客户端能够稳定运行。应先手动登录您的客户端保证其能正常运行。

九、插件支持

pytradecn 内部集成了银河双子星的客户端支持，但可喜的是 pytradecn 允许以插件的形式支持不同的券商客户端。同花顺软件自带一个交易客户端，如您需要同花顺客户端的插件，请发邮件索取或在 test 文件夹中。41715399@qq.com

十、开始使用

我们以同花顺的客户端为例，将同花顺客户端的插件保存在您项目的任何位置，例如保存在包 mypacket 中。

十一、创建交易对象

首先应该创建交易对象，使用以下代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader(client=THS92030)  # 创建交易对象
print(trader.query('当日委托')[1])
trader.close()  # 关闭交易是一个好的习惯
```

事实上，Trader 方法不需要显式输入参数，pytradecn 可以自动识别您定义的客户端，如下方法：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030
```

```
trader = Trader()  # pytradecn 会自动识别您的客户端 THS92030
print(trader.query('当日委托')[1])
trader.close()  # 关闭交易是一个好的习惯
```

或者，彻底隐藏客户端的导入，如把导入客户端放在包 mypacket 的__init__.py 中，如下：

```
# 在 mypacket.__init__.py 中
from . import ths92030
```

在您的主代码中：

```
# 导入 pytradecn
from pytradecn import Trader

trader = Trader()  # pytradecn 会自动识别您的客户端 THS92030
print(trader.query('当日委托')[1])
trader.close()  # 关闭交易是一个好的习惯
```

十二、交易功能

默认状态下，pytradecn 内部集成的银河双子星或同花顺客户端只提供了买（buy）、卖（sell）、撤单（cancel）、查询（query）四个功能，可喜的是您可以修改或编写插件实现任何您想要的功能。

十三、处理错误

由于 GUI 程序运行很不稳定，鉴于此，pytradecn 设计成完成任何功能均会有输出，不会因运行时错误卡死代码，所以上述四个功能或者您自定义的功能的返回值永远是一个二元组，元组的第一项是标

识成功与否的布尔值，元组的第二项是真正的返回值，如发生错误，第二项为发生错误的原因。所以在您的代码中一定要有容错处理。

十四、买入

创建了交易对象后就可以执行买入操作，使用下列代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader()  # pytradecn 会自动识别您的客户端 THS92030

success, text = trader.buy(code='601002', price='4.18', count='100')
if success is True:
    print('委托编号：', text)
else:
    print('错误：', text)

trader.close()  # 关闭交易是一个好的习惯
```

注意：由于计算机的浮点数危机，如果采用 float 型，在金融和财务领域是致命的，所以 pytradecn 对所有的输入和输出均为字符串，特别是价格和数量等，这样便于在您的代码中使用 Decimal 类进行精确计算。

十五、卖出

同买入一致。

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader()
```

```
success, text = trader.sell(code='601002', price='5.18', count='100')
if success is True:
    print('委托编号: ', text)
else:
    print('错误: ', text)

trader.close()
```

十六、撤单

先参考如下代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader()

success, text = trader.sell(code='601002', price='5.18', count='100')
if success is True:
    print('委托编号: ', text)
    trader.cancel(合同编号=text)  # 撤销掉刚才的委托单
else:
    print('错误: ', text)

trader.close()
```

事实上，cancel 功能非常强大，它可以撤销掉您任何想要的组合单。以下是可能的使用场景：

```
# 1、撤销全部委托单
trader.cancel()
# 2、使用一个关键字参数过滤委托单
trader.cancel(证券名称='农业银行')  # 将证券名称为农业银行的委托单撤销
```

```
trader.cancel(操作='买入')  # 将所有的买入委托单撤销
trader.cancel(合同编号='123456')  # 将合同编号为 123456 的委托单撤销
# 3、使用多个关键字参数过滤委托单
trader.cancel(证券名称='农业银行', 操作='买入')  # 将农业银行的买入单撤销
# 4、使用一个关键字参数，多值过滤委托单
trader.cancel(证券名称=('农业银行', '平安银行'))  # 将农业银行和平安银行的委托单撤销
trader.cancel(合同编号=('123456', '654321'))  # 将合同编号为‘123456’和‘654321’的委托单撤销
# 5、使用多关键字参数，多值过滤委托单
trader.cancel(证券名称=('农业银行', '平安银行'), 操作='买入')  # 将农业银行和平安银行的买入单撤销
```

十七、查询

客户端查询功能中的所有查询项，pytradecn 均可以完成查询，注意，pytradecn 默认未提供日期选择功能，您可以自行编写代码添加。参考以下代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader()

success, table = trader.query('当日委托')
if success is True:
    print(table)
else:
    print('错误: ', table)

trader.close()
```

如果查询成功，则会返回表格对象，table 对象是一个只读列表，您可以像操作列表一样操作它，包括遍历、索引以及一些有关列表的方法，但这个列表是只读的，不可以修改。列表中的每一个项是一个只读字典，可以像字典一样去遍历、索引它，但不可以修改。

表格对象有一个重要的方法 items()，它可以按条件过滤表格，返回一个新表格，可能的应用场景如下：

```
# 1、返回表格副本
table.items()
# 2、使用一个关键字参数过滤表格
table.items(证券名称='农业银行') # 过滤出证券名称为农业银行的数据
table.items(操作='买入') # 过滤出所有的买入数据
table.items(合同编号='123456') # 过滤出合同编号为 123456 的数据
# 3、使用多个关键字参数过滤表格
table.items(证券名称='农业银行', 操作='买入') # 过滤出将农业银行的买入数据
# 4、使用一个关键字参数，多值过滤表格
table.items(证券名称=('农业银行', '平安银行')) # 将农业银行和平安银行的数据过滤出来
table.items(合同编号=('123456', '654321')) # 将合同编号为‘123456’和‘654321’的数据过滤出来
# 5、使用多关键字参数，多值过滤表格
table.items(证券名称=('农业银行','平安银行'), 操作='买入') # 将农业银行和平安银行的买入单过滤出来
```

表格对象还有一个 item()方法，与 items()一样的参数，只不过 item()方法只返回唯一的一个项（字典类型），如果为空或有多个项则会报错。

十八、关闭交易

在您完成交易后，关闭交易是一个好的习惯，关闭的方法有两种

一种是使用 close()方法关闭，如下代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

trader = Trader() # pytradecn 会自动识别您的客户端 THS92030
# 您的代码
trader.close()  # 关闭交易
```

一种是使用 with 关键字，如下代码：

```
# 导入 pytradecn 和同花顺客户端插件，导入顺序不分先后
from pytradecn import Trader
from mypacket.ths92030 import THS92030

with Trader() as trader:  # 使用 with 关键字会自动关闭交易
    # 您的代码
```

十九、编写插件

往往会编写插件以支持某一券商客户端或者某一特定的版本,需要编写的内容包括客户端配置、登录引擎、交易模板、交易模型，或许可能还有一些特殊的自定义控件。pytradecn 已经内置了三种登录引擎，分别是不需要验证码的普通登录（DEFAULT）、验证码登录（VERIFYCODE）、主动刷新验证码登录（VERIFYCODEPLUS），交易模板内置了一种默认模板（DEFAULT），分别提供 buy、sell、cancel、query 四种功能，内置的默认模型（DEFAULT）实现了这四种功能。如果内置的登录引擎、交易模板、交易模型或者一些控件无法满足要求，则需要新建这些登录引擎、交易模板、交易模型或者控件。最好

的方法是针对某一券商的某一版本新建全套的插件，ths92030 是一个案例，它是同花顺 9.20.30 版本的支持插件。插件的组织方法有两种，一种是集中模式，一种是分散模式。ths92030 采用的是集中模式。

二十、集中模式

集中模式指将客户端配置、登录引擎、交易模板、交易模型，或自定义控件集中编写在一个 py 文件中，在您项目的任何位置将该 py 文件引入，pytradecn 即可自动设别。这样做的好处是便于理解和传播，缺点是比较凌乱。

二十一、分散模式

分散模式即将客户端配置、登录引擎、交易模板、交易模型，或自定义控件分别编写在不同的 py 文件中，放在同一个包内（事实上不同的包也是可以的），在包的__init__.py 文件中将这些 py 文件引入即可。