

天软 pyTSL 接口使用说明

TSDN 组

TSDN

本文档所有内容均属于深圳市天软科技发展有限公司的商业机密，深圳市天软科技发展有限公司亦保留上述文档中阐述的包括而限于技术、方法、思想的所有权。此文档仅仅只许可因本文档所必需传播的范围内使用（仅限于内部），本文档的获得者不得将本文档散发给不相关的其他机构和个人。

深圳天软科技 | 深圳市福田区新闻路 59 号深茂商业中心 19 楼

1 更新日志

1.1 更新日志

更新日志	更新说明
2023-8-29	1、pyTSL.Client 支持代理服务器账号密码验证 2、调整部分范例
2022-8-10	1、pyTSL.Client 类 query 方法 fields 参数支持 tuple 和 list 数据类型 2、pyTSL.Client 类 exec 方法增加 resultname 参数，用于指定委托保存的结果集名称 3、pyTSL.Const 增加 STOCKID 等与行情相关常量 4、修正 TSReturnValue 类 value 方法 parse_date 的说明
2022-7-5	1、新增对 macOS(arm64) Apple M1 支持。 2、修改 register_proc 说明，默认实现了 importfile 和 exportfile 函数，用户不能注册同名函数。 3、调整文章的部分结构。 4、新增常见问题 pyTSL 中返回值编码为 utf-8
2022-6-20	1、新增常见问题： c.logout 后，再次执行 c 的方法会自动重新登陆 。 2、last_error 的报错信息查看源链接。 3、新增 pip install tspsyl 安装方式，使用方法不变 4、新增章节： 7 对于老用户如何使用 pyTSLPy 模块替代 TSLPy3.pyd
2021-09-03	Client 增加了代理服务器的支持 Batch 系列方法加了 key 参数 增加了基于 centos7 的编译包，可以用于 centos7 及以上操作系统 arm64 适配了飞腾 CPU
2021-06-17	新增了 pyTSL.Client 对象的 ini 文件路径异常处理 FAQ、last_error 的常见返回错误代码及解释
2021-01-20	新增了 mac 最新版本交互异常问题的处理 FAQ
2020-12-25	Batch 类增加 query 方法、构造函数增加 reconnect 参数、Task 类增加 key 方法 Client 类增加 server_list 方法、Client.exec 方法增加 stock 等参数设置 增加 Const 模块返回常用的一些常量
2020-12-07	增加了对 MacOS 的支持
2020-10-20	Client 类增加了对加密码的支持 增加了指定、返回默认执行的服务节点的方法 TSReturnValue 类增加返回 TSL 流格式、STN 格式数据的方法
2020-04-03	添加了 TSBatch 类说明、增加了图像数据转换例子
2020-03-09	pyTSL 接口使用说明文档的创建与发布

1.2 摘要

1. Python 通过 pyTSL 接口与天软进行交互
2. 支持 Windows, Linux 与 MacOS 版本
3. 新增 pip install tspytsl (pyTSL 包名不通过, 所以改为 tspytsl) 安装方式, 导入仍然是 import pyTSL。

2 前言

pyTSL 包是一个独立的 python 模块, 不依赖天软客户端, 更符合 python 开发者的编程习惯。

相关功能介绍: <http://py3k.cn/pyTSL/usage.html#>

技术支持: 后期会由天软官方进行支持并且持续维护, 用户可放心使用。如果有任何问题或者建议, 请发电子邮件到 pytsl@py3k.cn 或者上 TSDN 在线咨询。

3 支持与配置说明

现有两种安装方式

- 1、2022-06-20 新增的 pip install tspytsl 安装方式。导入仍然是 import pyTSL。
- 2、原通过 <http://py3k.cn/pyTSL/setup.html#id2> 链接下载 pyTSL 相关文件的方式。

3.1 支持版本

pyTSL 支持 python3.6+ 的交互。

截止 2022-7-5 日, 支持如下版本:

Python	Windows	Linux(x86_64)	Linux(arm64) 适配华为鲲鹏	MacOS(x86_64)	MacOS(arm64) Apple M1
3.6	支持	支持	支持	支持	不支持
3.7	支持	支持	支持	支持	不支持
3.8	支持	支持	支持	支持	支持
3.9	支持	支持	支持	支持	支持
3.10	支持	支持	支持	支持	支持

3.2 pip install tspytsl 安装方式

使用 cmd:输入命令 `pip install tspytsl` 即可开始安装。

提示：这样的安装方法是在 pip 库已经安装好的前提下进行的。如果没有安装或无法确定是否安装 pip 库，可以查看下文连接确定【[pip 库的安装与版本检查](#)】

3.3 pyTSL 文件下载的安装方式

3.3.1 相关包的下载

pyTSL 包的下载地址：<http://py3k.cn/pyTSL/setup.html#id2>

Windows 操作系统下，下载对应 python 版本的包如：pyTSL.xxxx-win-amd64.pyd.zip。

Linux 操作系统下

(1) x86_64 下载对应 python 版本的包如：pyTSL.xxxx-x86_64-linux-gnu.so.zip;

(2) arm64 下载对应 python 版本的包如：pyTSL.xxxx-aarch-linux-gnu.so.zip。

MacOS 操作系统下

(1) x86_64 下载对应 python 版本的包如：pyTSL.xxxx-darwin.so.zip。

3.3.2 配置步骤

解压已下载好的.zip 文件包。

3.3.2.1 Windows 操作系统

1、把对应 Python 版本的 pyTSL*.pyd 复制到 python 安装目录的 Lib\site-packages 目录，或者自己项目的目录。

2、系统还需安装 VS2019 的 C++运行时库（VC_redist.x64.exe），下载：

https://aka.ms/vs/16/release/VC_redist.x64.exe

3.3.2.2 Linux 操作系统

把对应 Python 版本的 pyTSL*.so 复制到 python 安装目录的 site-packages 目录，或者自己项目的目录。

3.3.2.3 MacOS 操作系统

把对应 Python 版本的 pyTSL*.so 复制到 python 安装目录的 site-packages 目录，或者自己项目的目录

3.3.2.4 测试案例

```
import pyTSL
c = pyTSL.Client("user", "password", "ssl.tinysoft.com.cn", 443)
c.login() #成功返回 1，失败返回 0
r = c.exec("return "测试";")
if r.error(): #有错误
    print(r.message())
else:
    print(r.value())
```

返回说明：若测试案例能成功返回‘测试’字符串，则说明配置成功。否则，可能是配置不成功或登陆失败，可根据报错信息进行问题定位解决。

4 使用说明

4.1 接口说明

4.1.1 pyTSL.Client 类

4.1.1.1 构造函数

构造一个 Client，占用一个登陆数。

- Client(user, password, ip, port, proxy_ip, proxy_port, proxy_user, proxy_password)

函数说明：构造连接天软的实例对象接口，支持代理服务器。

参数说明：

参数	数据类型	说明
User	字符串	天软账号
Password	字符串	天软账号对应的密码，可加密，请用天软客户端安装目录下的 CONNECTMAN.exe 来对密码进行加密，加密后的字符串跟原来一样直接赋值。
Ip	字符串	天软服务器机群地址 深圳服务器地址： "ssl.tinysoft.com.cn"（官方默认服务器） 武汉服务器地址： "wh.tinysoft.com.cn"
port	整型	天软服务器访问端口： 443、444 ， 均适用于两个服务器

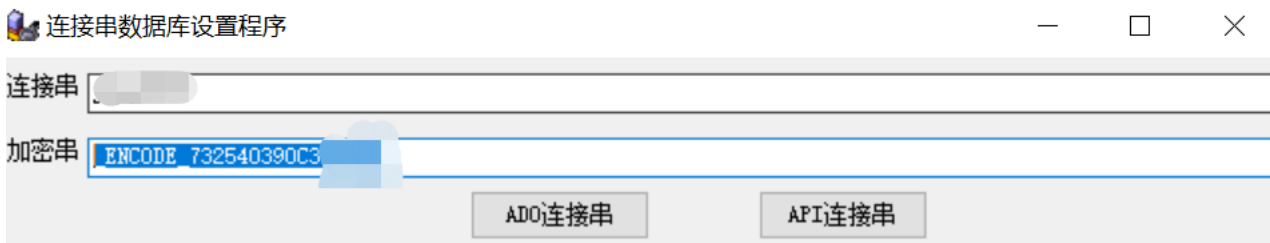
proxy_ip	字符串	代理服务服务器地址，可选参数
proxy_port	整型	代理服务服务器端口，可选参数
proxy_user	字符串	代理服务服务器验证账号，可选参数
proxy_password	字符串	代理服务服务器验证密码，可选参数

调用：

1. `c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)`
2. `c = pyTSL.Client("user", "password", " wh.tinysoft.com.cn", 444)`
3. `c = pyTSL.Client("user", "_ENCODE_732540390C3XXXXX", "tsl.tinysoft.com.cn", 443)#密码加密`

密码加密方法：

1. 打开天软客户端安装目录下的软件 CONNECTMAN.exe
2. 将账号密码输入到连接串的方框中，加密串的方框即显示加密后的密码，如下图所示。



● Client(ini_file)

参数说明：

Ini_file: 字符串，连接服务器相关信息的.ini 文件的绝对路径。

其中.ini 文件配置相关的登录信息的格式如下：

```
user=
password=
ip=
port=
```

其中，user=账号名，password=密码，ip=服务器地址，port=端口，比如：

my.ini - 记事本


文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
user=user
password=password
ip=tsl.tinysoft.com
port=443
```

注意，如果需要使用代理，请在以上的.ini 文件基础上增加配置以下选项：

```
[proxy]
user=
password=
ip=
port=
```

示例如下：

 *my.ini - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
user=xxx
password=xxx
ip=tsl.tinysoft.com
port=443
```

```
[proxy]
user=xxx
password=xxx
ip=218.95.39.41
port=13440|
```

调用：

```
c=pyTSL.Client("E:\\my.ini")
```

注：以后所有方法的调用如 `c.xxxx()` 中的 `c` 均为构造函数的实例，不再一一说明。

使用加密后的密码同原来方式一样，赋值给 `password` 即可。

需要特别注意的是，ini 文件所在的路径若含有中文名，否则会报找不到文件的错误，解决方法可见末尾 FAQ。

4.1.1.2 login

- login()

函数说明：登陆天软服务器

调用：`c.login()`

返回：布尔类型，成功：1，失败：0。

特别地，当返回 0 时候，可以调用 `c.last_error()` 来查看错误详情。

4.1.1.3 last_error

- last_error

函数说明：Client 对象的最近一次错误提示

调用：c.last_error()

返回：长度为二的数组，格式为[错误代码，错误信息]。常见错误信息如下：

错误代码	错误信息	备注
0		无错误
-13	Invalid username or password	账号或密码错误
-1	Invalid User:!	账号为空或不合法
10049	在其上下文中，该请求的地址无效。	端口号不合法
10060	由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败。	端口号错误或网络不稳定
10061	由于目标计算机积极拒绝，无法连接。	服务器地址错误
11001	不知道这样的主机。	服务器地址为空
11004	请求的名称有效，但是找不到请求的类型的数据。	网络错误

其他错误代码可参考一下链接：<https://docs.microsoft.com/zh-cn/windows/win32/debug/system-error-code-lookup-tool>

4.1.1.4 logout

- logout()

函数说明：断开天软服务器

调用：c.logout()

返回：成功：0。

4.1.1.5 exec

- exec(code, **kwargs): TSReturnValue

函数说明：执行天软代码串。

参数说明：

code：字符串，所要执行的代码。

****kwargs:** 执行属性的设置，**可选参数**。包含：

字段	数据类型	说明	默认值
stock	字符串	当前股票代码，比如‘SZ000002’	None
cycle	字符串	当前周期，具体取值请参考：天软周期说明	“日线”
time	datetime 类型、 整型、字符串	当前时间。如 datetime(2019, 10, 1)、43739.0、20191001 或‘2019-10-01 14:03:01’	今天（自然日）
rate	整型	设置除权方式。 0：不复权； 1：为采用交易所数据复权，只考虑比例关系； 2：为采用分红送配数据复权，不仅仅考虑送股的比例关系，还考虑了诸如分红的加减关系	0
rateday	datetime 类型、 整型、字符串	设置除权基准日，0 为最后交易日为基准日，-1 为上市日为基准日，其他为指定的除权基准日	0
precision	整型	浮点数精度，python 默认数据格式打印最多六位小数	-1
viewpoint	datetime 类型、 整型、字符串	设定仿真的时点	0
code	字符串	定义函数代码段，这样不用把函数保存到服务器上就可以直接调用	None
service	字符串	指定执行的服务节点，与登录的服务器有关，具体取值请参考： 天软服务节点 、 server_list	深圳服务器：“正式版”， 武汉服务器：“Default Service”。
timeout	整型	设置超时，单位：毫秒	none，无超时时限
bgrun	整型	设置委托执行，1 是委托执行。	0,不进行委托执行
resultname	字符串	用于指定委托保存的结果集名称，bgrun=1 时有效	none，不保存结果
reportmode	整型	报表数据规则	-1
emptymode	整型	空记录模式	0

返回：一个 TSReturnValue 对象。执行结果通过 TSReturnValue 类的 value()方法进行读取。

范例：指定“新功能测试版”服务器返回上证指数的价格。

[code]

```
import pyTSL
```

```
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码  
c.login()
```

```

r = c.exec("")
return close();
",stock='SH000001',time=20200320,service="新功能测试版",timeout=5000)
if r.error(): #有错误
    print(r.message())
else:
    print(r.value()) #返回 2745.6182
[/code]

```

返回结果：

2745.6182

4.1.1.6 call

● call(funcname, *args, **kwargs)

函数说明：调用函数并返回结果。

参数说明：

funcname：字符串，函数名。

*args：funcname 函数的参数，可以自动转换这些 python 的数据类型到 TSL 的数据类型：

分类	Python	TSL
实数类型、日期时间类型	float、datetime、numpy.float64、numpy.datetime64、numpy.float32	real、tdatetime
整型、布尔型	bool、int、numpy.int32、numpy.int64	int
字符串型	str、bytes	str
数组型	list、dict、tuple、pandas.DataFrame、numpy.ndarray	array

kwargs：执行属性的设置，可选参数**。包含：

字段	数据类型	说明	默认值
stock	字符串	当前股票代码，比如‘SZ000002’	None
cycle	字符串	当前周期，具体取值请参考： 天软周期说明	“日线”
time	datetime 类型、整型、字符串	当前时间。如 datetime(2019, 10, 1)、43739.0、0191001 或‘2019-10-01 14:03:01’	今天（自然日）
rate	整型	设置除权方式。 0：不复权； 1：为采用交易所数据复权，只考虑比例关系； 2：为采用分红送配数据复权，不仅仅考虑送股的比例关系，还考虑了诸如分红的加减关系	0

rateday	datetime 类型、整型、字符串	设置除权基准日，0 为最后交易日为基准日，-1 为上市日为基准日，其他为指定的除权基准日	0
precision	整型	浮点数精度，python 默认数据格式打印最多六位小数	-1
viewpoint	datetime 类型、整型、字符串	设定仿真的时点	0
code	字符串	定义函数代码段，这样不用把函数保存到服务器上就可以直接调用	None
service	字符串	指定执行的服务节点，与登录的服务器有关，具体取值请参考： 天软服务节点 、 server_list	深圳服务器："正式版"， 武汉服务器："Default Service"。
timeout	整型	设置超时，单位：毫秒	none，无超时时限
bgrun	整型	设置委托执行，1 是委托执行。	0,不进行委托执行
reportmode	整型	报表数据规则	-1
emptymode	整型	空记录模式	0

返回：一个 TSReturnValue 对象。执行结果通过 TSReturnValue 类的 value()方法进行读取。

范例 1：取个股时间序列数据

```
[code]
import pyTSL
import pandas
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
test = ""
function get_value(begt,endt);
begin
    n:=tradedays(inttodate(begt),inttodate(endt));//时间段内交易日数
    return nday(n,'date',datetostr(sp_time()),'close',close(),'vol',vol(),'amount',amount());
end;
""

r = c.call('get_value',20190410,20190418,code=test,stock='SH000001',cycle="日线",time=20190418)
if r.error(): #有错误
    print(r.message())
else:
    print(pandas.DataFrame(r.value()))
[/code]
```

返回结果：

	date	close	vol	amount
0	2019-04-10	3241.9299	3.801344e+10	3.956700e+11
1	2019-04-11	3189.9619	3.530319e+10	3.606846e+11
2	2019-04-12	3188.6256	2.897890e+10	2.889090e+11
3	2019-04-15	3177.7866	3.488098e+10	3.556661e+11
4	2019-04-16	3253.5978	3.579188e+10	3.601666e+11
5	2019-04-17	3263.1179	3.575418e+10	3.545099e+11
6	2019-04-18	3250.2012	3.255399e+10	3.235082e+11

范例 2：调用天软公用函数 GetAbkbyDate (BkName,Endt)，取指定日 A 股各板块成份股

[code]

```
import pyTSL
```

```
import pandas
```

```
import datetime
```

```
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
```

```
r = c.call("GetAbkbyDate","A 股",datetime.date(2020,12,7))
```

```
if r.error(): #有错误
```

```
    print(r.message())
```

```
else:
```

```
    print(pandas.DataFrame(r.value()))
```

[/code]

返回结果：

0	SH600000
1	SH600004
2	SH600006
3	SH600007
4	SH600008
...	...
4081	SZ300912
4082	SZ300913
4083	SZ300915
4084	SZ300916
4085	SZ300999

4.1.1.7 query

- query(*kwargs)

函数说明：查询行情数据

参数说明：

*kwargs: 执行属性的设置，用等号进行赋值。

必须的*kwargs 项包含：

字段	数据类型	说明	默认值
stock	字符串或一维数组	设置当前股票代码，可以指定多只股票	None
cycle	字符串	当前周期，具体取值请参考： 天软周期说明	“日线”
begin_time	datetime 类型、整型、字符串	设置数据开始时间。如 datetime(2019, 10, 1)、43739.0、20191001 或 ‘2019-10-01 14:03:01’	None
end_time	datetime 类型、整型、字符串	设置数据结束时间。如 datetime(2019, 10, 1)、43739.0、20191001 或 ‘2019-10-01 14:03:01’	None

可选的*kwargs 项包含：

字段	数据类型	说明	默认值
fields	字符串、元组、列表	设置所要取字段，如：["date","close"]、("date","close"),字符串类型用","分割，例如："date,close"。具体取值请参考： 天软字段说明	默认为所有字段
rate	整型	设置除权方式。 0: 不复权; 1: 为采用交易所数据复权，只考虑比例关系; 2: 为采用分红送配数据复权，不仅仅考虑送股的比例关系，还考虑了诸如分红的加减关系	0
rateday	datetime 类型、整型、字符串	设置除权基准日，0 为最后交易日为基准日，-1 为上市日为基准日，其他为指定的除权基准日	0
precision	整型	浮点数精度，python 默认数据格式打印最多六位小数	-1
viewpoint	datetime 类型、整型、字符串	设置 viewpoint	0
service	字符串	指定执行的服务节点，与登录的服务器有关，具体取值请参考： 天软服务节点 、 server list	深圳服务器："正式版"， 武汉服务器："Default Service"。
timeout	整型	设置超时，单位：毫秒	none，无超时时限

返回：一个 TSReturnValue 对象。执行结果通过 TSReturnValue 类的 value()方法进行读取。

范例：查询个股一段时间收盘价数据

```
[code]
import pyTSL
import pandas as pd
from datetime import datetime
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号， password 为账号密码
c.login()
r=c.query(stock=['SZ000001','SZ000002'],cycle="周线",begin_time=datetime(2020,1,1),
end_time=datetime(2020,3,20),fields='StockID,StockName,date,close')
if r.error(): #有错误
    print(r.message())
else:
    print(pd.DataFrame(r.value(parse_date='date')))
[/code]
```

返回结果：

	StockID	StockName	date	close
0	SZ000001	平安银行	2020-01-03	17.18
1	SZ000001	平安银行	2020-01-10	16.69
2	SZ000001	平安银行	2020-01-17	16.39
3	SZ000001	平安银行	2020-01-23	15.54
4	SZ000001	平安银行	2020-02-07	14.62
5	SZ000001	平安银行	2020-02-14	15.03
6	SZ000001	平安银行	2020-02-21	15.58
7	SZ000001	平安银行	2020-02-28	14.50
8	SZ000001	平安银行	2020-03-06	15.03
9	SZ000001	平安银行	2020-03-13	14.52
10	SZ000001	平安银行	2020-03-20	12.52
11	SZ000002	万 科 A	2020-01-03	32.05
12	SZ000002	万 科 A	2020-01-10	31.46
13	SZ000002	万 科 A	2020-01-17	30.46
14	SZ000002	万 科 A	2020-01-23	28.98
15	SZ000002	万 科 A	2020-02-07	27.50
16	SZ000002	万 科 A	2020-02-14	30.80
17	SZ000002	万 科 A	2020-02-21	29.25
18	SZ000002	万 科 A	2020-02-28	29.59
19	SZ000002	万 科 A	2020-03-06	31.13
20	SZ000002	万 科 A	2020-03-13	29.45
21	SZ000002	万 科 A	2020-03-20	25.80

4.1.1.8 list_bgrun

● list_bgrun()

函数说明：列出所有委托执行。

调用：c.list_bgrun()

返回：格式 [“+OK”, [委托任务 1], ...]

委托任务格式 [handle, 执行的函数名, 开始执行时间, 已运行时间]

```
>>> c.list_bgrun()
['+OK', [660823521.0, 'FUNCTION1', datetime.datetime(2020, 3, 12, 10, 11, 1), 7]]
```

4.1.1.9 stop_bgrun

- stop_bgrun(handle)

函数说明：停止指定的委托执行任务。

参数说明：

handle：浮点型，委托任务编号。

调用：c.stop_bgrun(handle)

返回：成功则返回 1。

```
>>> c.stop_bgrun(660823521.0)
1
```

4.1.1.10 admin

- admin(**kwargs)

函数说明：执行天软客户端任务管理命令。

参数说明：

**kwargs：字符串，天软客户端任务管理命令，如”oa”, ”ou”

返回：成功则返回对应指令的查询信息。

范例：

[code]

```
import pyTSL
```

```
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
```

```
c.login()
```

```
if c.login() == 0:
```

```
    print(c.last_error())
```

```
else:
```

```
    print(c.admin("ou"))
```

[/code]

返回结果：

CLIENTID	ADDR	PORT	RECV(M)	SEND(M)	MINS	USERNAME	PSENDS	PSEND(M)	OLTASK	BGTASK	SUBTASK	#RESERVED
250053	183.236.225.46	53923	0.0	0.0	0		0	0.00	0/ 5	0/ 2	0/ 3	0

1 user(s) online

4.1.1.11 set_callback

- set_callback(function)

函数说明：设置 rdo2 的回调函数。

参数说明：function：字符变量，函数名。

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
def rdo(*args):
    print(*args)
    return 1
c.set_callback(rdo)
c.exec('"a := rdo2 mypycallback('ok'); echo a;"); #rdo 会被调用，打印出 mypycallback "ok"
[/code]
```

返回结果：

```
mypycallback ['ok']
ECHO:1
```

4.1.1.12 default_service

- default_service(service)

函数说明：指定、返回默认执行的服务节点。如果参数为空就返回当前的默认设置，否则进行设置。

参数说明：service：字符串，指定执行的服务节点，与登录的服务器有关，具体取值请参考：[天软服务节点](#)

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
c.default_service("正式版")#设置当前服务器
r = c.exec("return TSAppservername();")
if r.error(): #有错误
    print(r.message())
else:
    print(r.value())
    print(c.default_service())#返回当前服务器
[/code]
```


返回结果：

```
192.168.102.12:z:\server\bin\exec64.exe  
正式版
```

4.1.1.13 server_list

- server_list()

函数说明：返回服务节点列表。

范例：

```
[code]  
import pyTSL  
import pandas as pd  
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码  
c.login()  
if c.login() == 0:  
    print(c.last_error())  
else:  
    print(pd.DataFrame(c.server_list()))  
[/code]
```

返回结果：

```
0          正式版  
1      新功能测试版  
2      下一代全新测试  
3  192.168.102.46:z:\se  
4  192.168.102.48:z:\se  
5  192.168.102.7:z:\ser  
6  192.168.102.49:z:\se  
7  192.168.102.41:z:\se  
8  192.168.102.44:z:\se  
9  192.168.102.45:z:\se  
10 192.168.102.43:z:\se  
11 192.168.102.12:z:\se  
12 192.168.102.9:z:\ser  
13 192.168.102.2:z:\for  
14 192.168.102.49:z:\ti
```

4.1.2 pyTSL 方法

4.1.2.1 DatetimeToDouble

- DatetimeToDouble(datetime)

函数说明：将时间从 Python 的 datetime 类型转换到 TSL 的时间类型，时间精度到毫秒。

参数说明：datetime: datetime 类型，时间。

返回：转换后得到的 TSL 时间类型的时间

调用：pyTSL.DatetimeToDouble(datetime)

4.1.2.2 DoubleToDatetime

- DoubleToDatetime(t)

函数说明：将时间从 TSL 的时间类型转换到 Python 的 datetime 类型，时间精度到毫秒。

参数说明：t: TSL 时间类型 Tdatetime，时间。

返回：转换后得到的 datetime 类型的时间

范例：

[code]

```
import pyTSL
```

```
import pandas as pd
```

```
from datetime import datetime
```

```
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
```

```
c.login()
```

```
r=c.exec("return nday(tradedays(20200401T,20200410T),'date',sp_time(),'close',close());
```

```
","stock='SH000001',cycle='日线')
```

```
df=pd.DataFrame(data=r.value())
```

```
df['date_double'] = df['date'] #转换前后结果对比
```

```
df['date'] = df['date'].apply(lambda x: pyTSL.DoubleToDatetime(x))
```

```
if r.error(): #有错误
```

```
    print(r.message())
```

```
else:
```

```
    print(df)
```

[/code]

返回结果：

	date	close	date_double
0	2023-08-18	3131.9530	45156.0
1	2023-08-21	3092.9777	45159.0
2	2023-08-22	3120.3338	45160.0
3	2023-08-23	3078.4021	45161.0
4	2023-08-24	3082.2439	45162.0
5	2023-08-25	3064.0747	45163.0
6	2023-08-28	3098.6363	45166.0

4.1.2.3 DataFrameDiff

- DataFrameDiff(df1, df2)

函数说明：比较两个 DataFrame 对象，需要安装 pandas

参数说明：两个 DataFrame 对象**结构要一样**

df1: DataFrame 类型，DataFrame 对象 1。

df2: DataFrame 类型，DataFrame 对象 2。

返回：两个对象的不同数据以及分别的行列位置

范例：

```
[code]
import pyTSL

import pandas

r1=((1,2,3),(4,5,6))
r2=((1,2,4),(6,5,4))

df1=pandas.DataFrame(r1)
df2=pandas.DataFrame(r2)

print(pyTSL.DataFrameDiff(df1, df2))
[/code]
```

返回结果：

```
>>> r1=((1, 2, 3), (4, 5, 6))
>>> r2=((1, 2, 4), (6, 5, 4))
>>> df1=pandas.DataFrame(r1)
>>> df2=pandas.DataFrame(r2)
>>> pyTSL.DataFrameDiff(df1, df2)
      Left  Right
0  2      3      4
1  0      4      6
2      6      4
```

4.1.2.4 EncodeStream

- EncodeStream(obj) -> bytes

函数说明：把 python 的变量转换成 TSL 的流格式，方便存储。

参数说明：

obj: python 变量

返回：TSL 的流格式

4.1.2.5 DecodeStream

- DecodeStream(bytes, parse_date=False) ->object

函数说明：把 TSL 的流转换成 python 的变量

参数说明：

bytes: 字符或字符串列表

可选参数：

parse_date: 布尔或字符串，是否转换 TSL 流中的日期类型为 python 的 datetime 类型，其取值如下：

赋值	说明
parse_date=False	默认值，不进行转换，天软中的 Tdatetime 类型实质为实数
parse_date=True	当 TSL 流的值为 TSL 中的 Tdatetime 类型、字符串类型的日期时，将其转换为 python 的 datetime 类型。
parse_date=[**kwargs]	其中，[**kwargs] 为字符串，为表中日期列的列名（如 'date'，若多个列名用 “,” 隔开），将 TSL 流的表中特定字段 **kwargs 的 Tdatetime 类型、字符串类型的日期转换为 datetime 类型。

返回：python 的变量

范例：

```
[code]
import pandas
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
test = ""
function get_value(a);
begin
a:= select ['StockName'],['date'],['vol'],['price'] from markettable datekey
20200323T to 20200324T of 'SH000001' end;
a[0]['dates']:=43193.0;
a[1]['dates']:=43194.0;
return a;
end;
""

r = c.call('get_value',code=test,cycle="1 分钟",stock='SZ000001',timeout=5000)
if r.error(): #有错误
    print(r.message())
else:
    a=pyTSL.EncodeStream(r.value())
    b=pyTSL.DecodeStream(a)
    c=pyTSL.DecodeStream(a,parse_date=['date','dates'])
    print(a)
    print(pandas.DataFrame(b))
    print(pandas.DataFrame(c))
[/code]
```

b'\x05\x02\x00\x00\x00\x00\x00\x00\x00\x05\x05\x00\x00\x00\x06\t\x00\x00\x00StockName\x00\x02\x08\x00\x00\x00\xc9\				
StockName	date	vol	price	dates
0	上证指数	43913.0	2.498202e+10	2660.1674 43193.0
1	上证指数	43914.0	2.570219e+10	2722.4381 43194.0
StockName	date	vol	price	dates
0	上证指数	2020-03-23	2.498202e+10	2660.1674 2018-04-03
1	上证指数	2020-03-24	2.570219e+10	2722.4381 2018-04-04

- `register_proc(funcname1,funcname2)`

参数说明:

function2: 函数名变量，函数名 2。Python 中定义的函数名

```
[code]
```

```
import pyTSL
```

```
def py_exportfile(filename, value):
```

```
print(filename,value)
```

return 1

```
pyTSL.register_proc("py_exportfile", py_exportfile)
```

```
r = c.exec("""a := rdo2 py exportfile('1.csv',array(1,2,3)); echo a; return a+1;""")
```

[/code]

```
1.csv [1, 2, 3]
ECHO:1
```

- DataFrameToTSArray (DataFrame)

参数说明:

DataFrame: DataFrame 类型，python DataFrame 对象。

范例：

```
[code]
import pyTSL
import pandas
r1=((1,2,3),(4,5,6))
df1=pandas.DataFrame(r1)
print(pyTSL.DataFrameToTSErrorArray(df1))
[/code]
```

结果：

```
[{0: 1, 1: 2, 2: 3}, {0: 4, 1: 5, 2: 6}]
```

4.1.3 TSErrorBatch 类

用于并发执行多个天软任务。可利用函数 `key` 来标识每个任务，字符串参数可用 `format` 函数进行配置，用法请参考 [call](#) 中的范例。

4.1.3.1 构造函数

- `Batch(parallel,reconnect)`

函数说明：构造 `Batch` 实例对象接口。

参数说明：

`parallel`：整型，指定执行并行函数的个数，和用户允许的并发任务有关，默认值是 5。

`reconnect`：整型，是否重新连接。0：不重新连接，1：重新连接。默认值是 0。

调用：

```
batch=pyTSL.Batch(5)
```

4.1.3.2 Task 类

`Batch` 类执行任务的标识。使用方法参考 [call](#) 和 [query](#)。

- `id()`

函数说明：返回任务整数类型 ID。

- `key()`

函数说明：设置或返回任务的关键字信息。默认为空，不设置关键字信息。

4.1.3.3 exec

- `exec(client,code,**kwargs)`

函数说明：执行天软代码串。

参数说明：

client: pyTSL.Client 实例。

code: 字符串，所要执行的代码。

kwargs: 执行属性的设置，可选参数**，同：[exec](#)，但不能设置为委托执行。

TSBatch 中**kwargs 特有的参数。

字段	数据类型	说明	默认值
key	字符串	标识每个任务	""

返回：[1, Task 类] 或者 [0, 错误信息]

说明：使用迭代器 iter 来提取 Batch 类方法返回的数据，返回一个 TSReturnValue 对象，结果通过 TSReturnValue 类的 value()方法进行读取。

范例：

[code]

```
def run_batch(c, parallel, job_count):
    assert c.login() == 1
    batch = pyTSL.Batch(parallel)
    for i in range(job_count):
        s = "sleep(100); echo %s ; return %s;" %(i,i)
        batch.exec(c, s)
    res = []
    it = iter(batch)    #使用迭代器来提取返回的结果数据
    for r in it:
        assert r.error() == 0
        assert type(r.value()) == int
        res.append(r.value())
    assert len(res) == job_count
    for i in range(job_count):
        assert i in res
```

```
def test_batch_exec(c):
    run_batch(c, 5, 100)
```

```
import pyTSL
import time
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
start = time.perf_counter()
test_batch_exec(c)
```

```

end = time.perf_counter()
print("Running time: %s Seconds"%(end-start))
[/code]

```

返回结果：

```

ECHO:0      ECHO:85
ECHO:2      ECHO:86
ECHO:3      ECHO:87
ECHO:1      ECHO:88
ECHO:4      ECHO:89
ECHO:6      ECHO:90
ECHO:5      ECHO:91
ECHO:7      ECHO:92
ECHO:8      ECHO:93
ECHO:9      ECHO:95
ECHO:11     ECHO:94
ECHO:12     ECHO:96
ECHO:10     ECHO:97
ECHO:13     ECHO:98
ECHO:14     ECHO:99
ECHO:15     Running time: 2.8783988000359386 Seconds

```

4.1.3.4 call

- `call(client,functionname,*args,**kwargs)`

函数说明：执行天软代码串。

参数说明：

`client`: `pyTSL.Client` 实例。

`functionname`: 字符串，所要执行的代码。

`*args`: `funcname` 函数的参数。

`**kwargs`: 执行属性的设置，**可选参数**，同：[call](#)，但不能设置为委托执行。

`TSBatch` 中 `**kwargs` 特有的参数。

字段	数据类型	说明	默认值
key	字符串	标识每个任务	""

返回：[1, Task 类] 或者 [0, 错误信息]

说明：使用迭代器 `iter` 来提取 `Batch` 类方法返回的数据，返回一个 `TSReturnValue` 对象，结果通过 `TSReturnValue` 类的 `value()` 方法进行读取。

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)
#user 为天软账号， password 为账号密码
assert c.login() == 1
r = c.call("getbk", "A 股")
if r.error(): #有错误
    print(r.message())
else:
    bk = r.value()
    batch = pyTSL.Batch(100)
    cy_day = c.call("cy_day").value()
    cy_10m = c.call("cy_10m").value()
    for s in bk:
        task = batch.call(c, "Stock_RD_VAR", cy_10m, 3, stock=s, time=20200519, cycle=cy_day,
            key="{ }-{:d}".format(s, 20200519))[1] #用 key 来标识每个任务

    for r in iter(batch):
        print(r.key(), r.value()) #返回的结果有对应的任务 key
    c.logout()
[/code]
```

返回结果：

```
NE430017-20200519 2.82468003576532
NE430047-20200519 867.8567907457548
NE430090-20200519 8.116053943773707
NE430139-20200519 61.94868461853527
NE430198-20200519 19.935782475280696
NE831526-20200519 0.0
NE831641-20200519 0.0
NE831906-20200519 0.0
NE833346-20200519 1211.27010172412
NE833580-20200519 0.0
NE834407-20200519 48.786447513381134
NE834682-20200519 0.0
NE834770-20200519 0.0
NE834950-20200519 0.0
NE834765-20200519 1807.5354159780295
```

4.1.3.5 query

- query(*kwargs)

函数说明：查询行情数据

参数说明：

*kwargs: 执行属性的设置，用等号进行赋值，可选参数，同：[query](#)，但不能设置为委托执行。

TSBatch 中**kwargs 特有的参数。

字段	数据类型	说明	默认值
key	字符串	标识每个任务	""

返回：[1, Task 类] 或者 [0, 错误信息]

说明：使用迭代器 iter 来提取 Batch 类方法返回的数据，返回一个 TSReturnValue 对象，结果通过 TSReturnValue 类的 value()方法进行读取。

范例：

```
[code]
import pyTSL
from pyTSL.Const import *
from datetime import datetime
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)
#user 为天软账号，password 为账号密码
assert c.login() == 1
r = c.call("getbk", "上证 A 股")
if r.error(): #有错误
    print(r.message())
else:
    bk = r.value()[0:10]
    batch = pyTSL.Batch(100)
    for s in bk:
        task =
        batch.query(c, stock=s, cycle=cy_day, begin_time=datetime(2020, 5, 19), end_time=datetime(2020, 5, 19), fields='
        StockID, StockName, date, close', key="{ }-{:d}".format(s, 20200519))[1] #用 key 来标识每个任务
    for r in iter(batch):
        print(r.key(), r.value()) #返回的结果有对应的任务 key
    c.logout()
[/code]
```

返回结果：

```
SH600000-20200519 [{'StockID': 'SH600000', 'StockName': '浦发银行', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 10.35}]
SH600004-20200519 [{'StockID': 'SH600004', 'StockName': '白云机场', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 15.17}]
SH600006-20200519 [{'StockID': 'SH600006', 'StockName': '东风汽车', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 3.99}]
SH600007-20200519 [{'StockID': 'SH600007', 'StockName': '中国国贸', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 13.4}]
SH600008-20200519 [{'StockID': 'SH600008', 'StockName': '首创环保', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 3.27}]
SH600015-20200519 [{'StockID': 'SH600015', 'StockName': '华夏银行', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 6.43}]
SH600010-20200519 [{'StockID': 'SH600010', 'StockName': '宝钢股份', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 1.1}]
SH600009-20200519 [{'StockID': 'SH600009', 'StockName': '上海机场', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 69.18}]
SH600011-20200519 [{'StockID': 'SH600011', 'StockName': '华能国际', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 4.34}]
SH600012-20200519 [{'StockID': 'SH600012', 'StockName': '皖通高速', 'date': datetime.datetime(2020, 5, 19, 0, 0), 'close': 5.19}]
```

4.1.4 pyTSL.Const 模块

返回常用的变量，例如周期，和 TSL 的相关方法对应，字段对应参考：[pyTSL.Const 常量列表](#)。

范例：

```
[code]
import pyTSL

from pyTSL.Const import *

print(cy_1m) #打印:1 分钟线，和 TSL 的 cy_1m()对应。
[/code]
```

返回结果：

```
1分钟线
```

4.1.5 TSReturnValue 类

天软返回的数据结果对象类。以下调用中用到的 r.xxxx()中 r 均为 TSReturnValue 类实例。不再一一说明。

4.1.5.1 error

- error()

函数说明：检查返回类是否出错

返回：0 表示没有错误，非 0 表示出错。

调用：r = c.exec("return "测试";")

```
print(r.error())
```

4.1.5.2 message

- message()

函数说明：打印出错信息

返回：有错误时为出错信息，无错误为空

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
r = c.exec("return "测试" ")#缺少分号
print(r.error())
print(r.message())
[/code]
```

返回结果：

```
-1
Statement missing terminator
```

4.1.5.3 value

- value(parse_date=False)

函数说明：返回结果

参数说明：

可选参数：

parse_date：布尔或字符串，是否转换 TSL 流中的日期类型为 python 的 datetime 类型，其取值如下：

赋值	说明
parse_date=False	默认值，不进行转换，天软中的 Tdatetime 类型实质为实数
parse_date=True	当类中值为 TSL 中的 Tdatetime 类型、字符串类型的日期时，将其转换为 python 的 datetime 类型。 注意：1、只对一维数组、单个值有效，二维数组需使用 parse_date=[**kwargs] 方式设置。 2、对于 query 方法返回默认值是 parse_date=True
parse_date=[**kwargs]	Tuple 或者 list, **kwargs 为字符串，为表中日期列的列名，将类的表中特定字段 **kwargs 的 Tdatetime 类型、字符串类型的日期转换为 datetime 类型。例如： parse_date=["date1","date2"]或者 parse_date=("date1","date2")。

范例 1：

```
[code]
import pyTSL
from datetime import datetime
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
r = c.exec('return array(20180901T, 20180902T);')
if r.error(): #有错误
    print(r.message())
else:
    print(r.value(parse_date=True))
    print(datetime(2018,9,1),datetime(2018,9,2))
[/code]
```

返回结果：

```
[datetime.datetime(2018, 9, 1, 0, 0), datetime.datetime(2018, 9, 2, 0, 0)]
2018-09-01 00:00:00 2018-09-02 00:00:00
```

范例 2：

```
[code]
import pyTSL
import pandas as pd
from datetime import datetime
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号， password 为账号密码
c.login()
test=""
function get_data(begT, endT);
begin
    return select ['StockID'],['StockName'],['date'],['close'] from markettable datekey begT to endT of
    DefaultStockID() end;
end;
""

r = c.call('get_data', datetime(2020,1,1), datetime.now(), stock="SZ000001", cycle="日线", code=test);
if r.error(): #有错误
    print(r.message())
else:
    df1=pd.DataFrame(r.value())
    df2=pd.DataFrame(r.value(parse_date=['date']))
    print(df1.head())    #打印前五行数据
    print(df2.head())    #打印前五行数据
[/code]
```

返回结果：

	StockID	StockName	date	close
0	SZ000001	平安银行	43832.0	16.87
1	SZ000001	平安银行	43833.0	17.18
2	SZ000001	平安银行	43836.0	17.07
3	SZ000001	平安银行	43837.0	17.15
4	SZ000001	平安银行	43838.0	16.66

	StockID	StockName	date	close
0	SZ000001	平安银行	2020-01-02	16.87
1	SZ000001	平安银行	2020-01-03	17.18
2	SZ000001	平安银行	2020-01-06	17.07
3	SZ000001	平安银行	2020-01-07	17.15
4	SZ000001	平安银行	2020-01-08	16.66

4.1.5.4 dataframe

- dataframe()

函数说明：输出 pandas 的 DataFrame 对象，需要先安装 pandas。

返回：DataFrame 的表格类型。

调用：r=c.query(stock='SZ000001',cycle="周线",begin_time=datetime(2019,1,1),end_time=datetime.now(),fields='date,close')

```
print(r.dataframe()) #相当于 print(pandas.DataFrame(r.value()))
```

4.1.5.5 stream

- stream()

函数说明：返回结果

返回：TSL 的流格式数据

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
r = c.exec('return array(20180901T, 20180902T);')
if r.error(): #有错误
    print(r.message())
else:
    s = r.stream()#将结果集转为 TSL 的流格式数据
    print(s)
    print(pyTSL.DecodeStream(s))#对 TSL 的流格式数据进行解码
[/code]
```

返回结果：

```
b'\x05\x02\x00\x00\x00\x00\x00\x00\x00\x00P\xa9\x00\x00\x00\x01\x00\x00\x00\x00Q\xa9\x00\x00'
[43344, 43345]
```

4.1.5.6 stn

- stn()

函数说明：返回结果

返回：TSL 的 STN 格式数据

范例：

```
[code]
import pyTSL
c = pyTSL.Client("user", "password", "tsl.tinysoft.com.cn", 443)#user 为天软账号，password 为账号密码
c.login()
```

```

r = c.exec('return array(20180901T, 20180902T);')
if r.error(): #有错误
    print(r.message())
else:
    print(r.stn())
[/code]

```

返回结果：

```
b'\r\narray(43344,43345)'
```

5 附件

5.1 天软周期列表

天软周期为系统参数，系统参数名称为：cycle，可取值范围如下：

频率	
低频	“日线”，“周线”，“月线”，“季线”，“半年线”，“年线”
高频	“1 分钟线”，“2 分钟线”，“3 分钟线”，“5 分钟线”，“10 分钟线”，“15 分钟线”，“20 分钟线”，“30 分钟线”，“40 分钟线”，“60 分钟线”，“120 分钟线”，
超高频	“成交明细”，“1 秒线”，“2 秒线”，“3 秒线”，“4 秒线”，“5 秒线”，“6 秒线”，“10 秒线”，“12 秒线”，“15 秒线”，“20 秒线”，“30 秒线”，“半秒线”
其它周期	“任意周期”，“期货 30 分钟线”，“期货 60 分钟线”

5.2 天软字段列表

天软字段为可选参数，参数名称为：fields，可取值范围如下表中的字段：

1、日线类数据：包括日线、周线、月线、季线、半年线、年线

字段	数据类型	名称	备注
StockID	string	代码	股票 ID
StockName	string	名称	股票名称
date	datetime	日期	当前时间

price	real	最新 价	该周期内最后一笔成交价
open	real	开盘 价	表示交易所公布的开盘价
close	real	收盘 价	表示交易所公布的收盘价
high	real	最高 价	周期内最高价
low	real	最低 价	周期内最低价
vol	real	成交 量	<p>周期内的成交量。</p> <p>单位：对于</p> <ol style="list-style-type: none"> 1、股票，股 2、指数，上交所的日线、深交所的日线单位都是股。 3、基金，份 4、期货，合约数量，如果要建立成交量和成交金额之间的关系，还需考虑合约乘数。 5、期权，合约数量 6、债券，深交所债券历史成交量的单位从 20031208 由手变为了张，上交所为手
amount	real	成交 金额	周期内的成交金额
cjbs	real	成交 笔数	<p>(1) 深交所取值为成交笔数。2017 年 7 月开始，上交所取值为成交笔数，此前上交所不公布该字段，数据为 0。</p> <p>(2) 对于期货/期权而言，是周期内的持仓的变动量，日线的话表示当日持仓量。</p>
yclose	real	系统 昨收	交易所发布的考虑了分红送股后的价格

syl1	real	市盈率 1	期货/期权：如果是日线，表示当日结算价。 其他：0
syl2	real	市盈率 2	期货/期权：昨日结算价 ETF/LOF：时点净值 其他：0

2、其他数据（Level1）：包括交易明细、秒线、分钟线等高频周期。

字段	数据类型	名称	备注
StockID	string	代码	股票 ID
StockName	string	名称	股票名称
date	datetime	时间	当前时间
price	real	价格	交易明细：该时间点最后一笔成交价； 其他：该周期内最后一笔成交价
vol	real	成交量	周期内的成交量。如果周期为分钟，则表示这分钟的成交量，并非从开盘累计。 单位：对于 1、股票，股 2、指数，上交所的高频单位是手，深交所的高频单位都是股。 3、基金，份 4、期货，合约数量，如果要建立成交量和成交金额之间的关系，还需考虑合约乘数。 5、期权，合约数量 6、债券
amount	real	成交金额	周期内的成交金额。如果周期为分钟，则表示这分钟的成交金额，并非从开盘累计。

cjbs	real	成交笔数	(1) 深交所取值为成交笔数。2017 年 7 月开始，上交所取值为成交笔数，此前上交所不公布该字段，数据为 0。 (2) 对于期货/期权而言，是周期内的持仓的变动量
yclose	real	上次价	上一周期的收盘价
syl1	real	市盈率 1	期货/期权： 如果是分钟线或秒线，则在当日结算价存储在当前所有记录的最后一个成交记录中，其他记录的结算价无效； 其他：0
syl2	real	市盈率 2	期货/期权：昨日结算价 ETF/LOF：时点净值 其他：0
buy1	real	买一价	买一价
buy2	real	买二价	买二价（期货没有该价）
buy3	real	买三价	买三价（期货没有该价）
buy4	real	买四价	买四价（期货没有该价）
buy5	real	买五价	买五价（期货没有该价）
sale1	real	卖一价	卖一价
sale2	real	卖二价	卖二价（期货没有该价）
sale3	real	卖三价	卖三价（期货没有该价）
sale4	real	卖四价	卖四价（期货没有该价）
sale5	real	卖五价	卖五价（期货没有该价）
bc1	int	买一量	当前以买一价出价的委买量
bc2	int	买二量	当前以买二价出价的委买量
bc3	int	买三量	当前以买三价出价的委买量
bc4	int	买四量	当前以买四价出价的委买量
bc5	int	买五量	当前以买五价出价的委买量
sc1	int	卖一量	当前以卖一价出价的委卖量

sc2	int	卖二量	当前以卖二价出价的委卖量
sc3	int	卖三量	当前以卖三价出价的委卖量
sc4	int	卖四量	当前以卖四价出价的委卖量
sc5	int	卖五量	当前以卖五价出价的委卖量
wb	real	委比	公式：委买/委卖（w_buy/w_sale）
lb	int	量比	<p>不同的周期，处理方式不同：</p> <p>1、对于秒线：</p> <p>量比取周期内交易明细最后一笔的数据。其中，交易明细中的量比由交易所发布。</p> <p>2、对于 1 分钟线：</p> <p>量比=现成交总手/[(过去 5 个交易日平均每分钟成交量) × 当日累计开市时间(分)]</p> <p>3、对于其他分钟线：量比取周期内最后一分钟的数据。</p>
zmm	int	买卖标识	<p>对交易明细公式：按下面的前后逻辑优先判定</p> <p>没有成交 -> 0</p> <p>当前成交价 > 上一笔买一价 -> 主买 1</p> <p>当前成交价 < 上一笔卖一价 -> 主卖 2</p> <p>否则 -> 3 (一般是集合竞价、涨停、跌停)</p> <p>对于非交易明细（秒线或分钟线等）：没有意义</p>
buy_vol	real	主买量	主动性买盘成交量：周期内买卖标识为 1 的成交量+买卖标识为 3 的成交量/2
buy_amount	real	主买金额	主动性买盘成交金额：周期内买卖标识为 1 的成交金额+买卖标识为 3 的成交金额/2
sale_vol	real	主卖量	主动性卖盘成交量：周期内买卖标识为 2 的成交量之和+买卖标识为 3 的成交量/2
sale_amount	real	主卖金额	主动性卖盘成交金额：周期内买卖标识为 2 的成交金额+买卖标识为 3 的成交金额/2

w_buy	real	委买	一个周期内的委买量
w_sale	real	委卖	一个周期内的委卖量
sectional_buy_vol	real	时点当日 累计主买 量	从开盘到当前时间的主买量之和
sectional_buy_amount	real	时点当日 累计主买 金额	从开盘到当前时间的主买金额之和
sectional_sale_vol	real	时点当日 累计主卖 量	从开盘到当前时间的主卖量之和
sectional_sale_amount	real	时点当日 累计主卖 金额	从开盘到当前时间的主卖金额之和
sectional_w_buy	real	时点当日 累计委买	从开盘到当前时间的委买量之和
sectional_w_sale	real	时点当日 累计委卖	从开盘到当前时间的委卖量之和
sectional_yclose	real	前日收盘	上一交易日(周期是日)的收盘价，与‘上次价’不一样。‘上次价’表示前一个周期的收盘价。
sectional_open	real	时点当日 开盘	今天的开盘价
sectional_high	real	时点当日 最高	从开盘到当前时间中最高价
sectional_low	real	时点当日 最低	从开盘到当前时间中最低价

sectional_vol	real	时点当日 累计成交量	从开盘到当前时间的成交量之和（主买+主卖）
sectional_amount	real	时点当日 累计成交金额	从开盘到当前时间的成交金额之和（主买+主卖）
sectional_cjbs	int	时点当日 累计成交笔数	股票：成交笔数 股指期货/期权：市场现有持仓量
sectional_wb	real	时点当日 累计委比	当日累计委买/当日累计委卖

5.3 天软官方服务器与服务节点列表说明

1、天软服务器说明：

服务器	服务器地址	端口	服务器节点	默认节点
深圳服务器	"tsl.tinysoft.com.cn"	443 或 444	正式版 新功能测试版	正式版
武汉服务器	"wh.tinysoft.com.cn"	443 或 444	Default Service	Default Service

注：本文中的 service 属性的值，默认由当前连接（Client 中的 ip 指定）的服务器群随机分配，分配的服务节点如下表。以下服务节点对应于天软客户端：计算设置-服务选择下的节点信息，若下表部分节点失效请使用最新节点信息。

服务节点（深圳服务器）	服务节点（武汉服务器）
"192.168.102.12:z:\se"	"192.168.89.21:d:\tin"
"192.168.102.44:z:\se"	"192.168.89.22:d:\tin"
"192.168.102.45:z:\se"	"192.168.89.23:d:\tin"
"192.168.102.13:z:\se"	"192.168.89.24:d:\tin"
"192.168.102.41:z:\se"	"192.168.89.25:d:\tin"

"192.168.102.43:z:\\se"	"192.168.89.26:d:\\tin"
"192.168.102.15:z:\\se"	
"192.168.102.2:z:\\ser"	
"192.168.102.4:z:\\ser"	
"192.168.102.9:z:\\ser"	
"192.168.102.2:z:\\for"	

5.4 pyTSL.Const 常量列表

pyTSL.Const 有一系列常量，其中中文为字符串，数字为整形，对应常量如下：

pyTSL.Const	对应常量	pyTSL.Const	对应常量	pyTSL.Const	对应常量
cy_month	“月线”	cy_detail	“成交明细”	ftCSV	0
cy_day	“日线”	cy_1s	“1 秒线”	ftXLS	1
cy_week	“周线”	cy_2s	“2 秒线”	ftStream	2
cy_quarter	“季线”	cy_3s	“3 秒线”	ftString	3
cy_halfyear	“半年线”	cy_4s	“4 秒线”	ftADO	4
cy_1m	“1 分钟线”	cy_5s	“5 秒线”	ftXML	5
cy_2m	“2 分钟线”	cy_6s	“6 秒线”	ftXLS2	6
cy_3m	“3 分钟线”	cy_10s	“10 秒线”	ftXLS3	7
cy_5m	“5 分钟线”	cy_12s	“12 秒线”	ftDBF	8
cy_10m	“10 分钟线”	cy_15s	“15 秒线”	rwByte	0
cy_15m	“15 分钟线”	cy_20s	“20 秒线”	rwInt	1
cy_20m	“20 分钟线”	cy_30s	“30 秒线”	rwReal	2
cy_30m	“30 分钟线”	cy_halfs	“半秒线”	rwStr	3
cy_40m	“40 分钟线”	cy_f30m	“期货 30 分钟线”	rwObj	4
cy_60m	“60 分钟线”	cy_f60m	“期货 60 分钟线”	rwRaw	5
cy_120m	“120 分钟线”			rwBinary	6
AMOUNT	'amount'	OPEN	'open'	SECTIONAL_LOW	'sectional_low'
BC1	'bc1'	PRICE	'price'	SECTIONAL_OPEN	'sectional_open'

BC2	'bc2'	SALE1	'sale1'	SECTIONAL_SALE_AMOUNT	'sectional_sale_amount'
BC3	'bc3'	SALE2	'sale2'	SECTIONAL_SALE_VOL	'sectional_sale_vol'
BC4	'bc4'	SALE3	'sale3'	SECTIONAL_VOL	'sectional_vol'
BC5	'bc5'	SALE4	'sale4'	SECTIONAL_WB	'sectional_wb'
BUY1	'buy1'	SALE5	'sale5'	SECTIONAL_W_BUY	'sectional_w_buy'
BUY2	'buy2'	SALE_AMOUNT	'sale_amount'	SECTIONAL_W_SALE	'sectional_w_sale'
BUY3	'buy3'	SALE_VOL	'sale_vol'	SECTIONAL_YCLOSE	'sectional_yclose'
BUY4	'buy4'	SC1	'sc1'	STOCKID	'StockID'
BUY5	'buy5'	SC2	'sc2'	STOCKNAME	'StockName'
BUY_AMOUNT	'buy_amount'	SC3	'sc3'	SYL1	'syl1'
BUY_VOL	'buy_vol'	SC4	'sc4'	SYL2	'syl2'
CJBS	'cjbs'	SC5	'sc5'	VOL	'vol'
CLOSE	'close'	SECTIONAL_AMOUNT	'sectional_amount'	WB	'wb'
DATE	'date'	SECTIONAL_W_BUY_AMOUNT	'sectional_buy_amount'	W_BUY	'w_buy'
HIGH	'high'	SECTIONAL_W_BUY_VOL	'sectional_buy_vol'	W_SALE	'w_sale'

LB	'lb'	SECTIONAL_ CJBS	'sectional_cjbs'	YCLOSE	'yclose'
LOW	'low'	SECTIONAL_ HIGH	'sectional_high'	ZMM	'zmm'

6 范例

6.1 天软回测框架的使用案例

运行环境：jupyter notebook

代码：

```
[code]
import pyTSL
import pandas as pd
import datetime, os
font_name = "STKaiti"
import matplotlib
matplotlib.rcParams['font.family']=font_name
matplotlib.rcParams['axes.unicode_minus']=False
%matplotlib inline

code=''
Function PYTSL_TSBackTesting(BegT, EndT, IndexID, RateType, InitCash,
PriceType, VolModType, DividendType, AllotmentType);
Begin
    obj := CreateObject('TSMYBackTestingQuantity');
    obj.FBegT:=BegT;
    obj.FEndT:=EndT;
    obj.FGroupType:=1;
    obj.FIndexId:=IndexID;
    obj.FRateType:=RateType;
    obj.FIniCash:=InitCash;
    obj.FPriceType:=PriceType;
```



```
obj.FVolModType:=VolModType;
obj.FDividendType:=DividendType;
obj.FAllotmentType:=AllotmentType;
obj.BackTest();
return array(
    //---组合基础
    '配置明细':obj.GetPortfolioPercent(BegT,EndT),
    '交易明细':obj.GetTradeData(BegT,EndT),
    "资产配置":obj.GetAssetData(BegT,EndT),
    "持仓明细":obj.GetHoldData(BegT,EndT),
    "行业配置":obj.GetSectorAllocation(BegT),

    //---组合盈亏、交易
    "组合盈亏":obj.GetGainandLoss(BegT,EndT),
    "交易汇总":obj.GetTradingAmount(BegT,EndT),
    "组合盈亏（按证券）":obj.GetGainandLossBySecurity(BegT,EndT),
    "交易汇总（按证券）":obj.GetTradingAmountBySecurity(BegT,EndT),

    //---组合收益
    '区间组合收益率': obj.GetPortfolioReturn(BegT,EndT),
    '组合和基准收益率序列':obj.GetPortfolioReturn2(BegT,EndT),
    '阶段收益':obj.GetTrailingReturn(EndT),
    '滚动收益':obj.GetRollingReturn(BegT,EndT,cy_month()),

    //----组合评价
    '风险回报':obj.GetReturnandRisk(BegT,EndT),
    '相对回报':obj.GetRelativePerformance(BegT,EndT),
);
End;
Type TSMYBackTestingQuantity=class(TSBackTesting)
public
    function GetTimeSeries();override;
    function GetTradeOrder(vEndT);override;
end;
```

```
function TSMYBackTestingQuantity.GetTimeSeries();override;
begin
    return rdo2 timeseries();
end;

function TSMYBackTestingQuantity.GetTradeOrder(vEndT);override;
begin
    return rdo2 tradeorder(vEndT);
end;

'''

class TSBacktesting:
    def __init__(self, df):
        self.portfolio = df

    def run(self, c, begin_time, end_time, index='SH000300', ratetype = -1,
cash=10000000, pricetype=2, volmodetype=0, dividendtype=0, allotmenttype=0):
        def timeseries():
            return sorted(set(self.portfolio["截止日"].to_list()))

        def tradeorder(dt):
            dt = pyTSL.DoubleToDatetime(dt)
            return self.portfolio[self.portfolio["截止日"] == dt]

        pyTSL.register_proc("timeseries", timeseries)
        pyTSL.register_proc("tradeorder", tradeorder)
        assert c.login() == 1

        r = c.call('PYTSL_TSBackTesting', begin_time, end_time, index, ratetype,
cash, pricetype, volmodetype, dividendtype, allotmenttype, code=code)
        return r
```

df = pd.read_csv("E:\protfolio.CSV", parse_dates=[0]) #导入组合目标持仓数据，路径为protfolio.CSV 文件位置

df.head() #打印的 protfolio.CSV 前五行，结果如下图所示：

	截止日	代码	名称	方向	比例(%)	乘数	保证金比例(%)	开仓费率(%)	平仓费率(%)
0	2010-12-31	SH600000	浦发银行	1	11.25	1	100	0.16500	0.06500
1	2010-12-31	SH600004	白云机场	1	11.25	1	100	0.16500	0.06500
2	2010-12-31	SZ000001	深发展A	1	11.25	1	100	0.11975	0.01975
3	2010-12-31	SZ000002	万科A	1	11.25	1	100	0.11975	0.01975
4	2010-12-31	IF01	股指期货	0	45.00	300	20	0.11975	0.01975

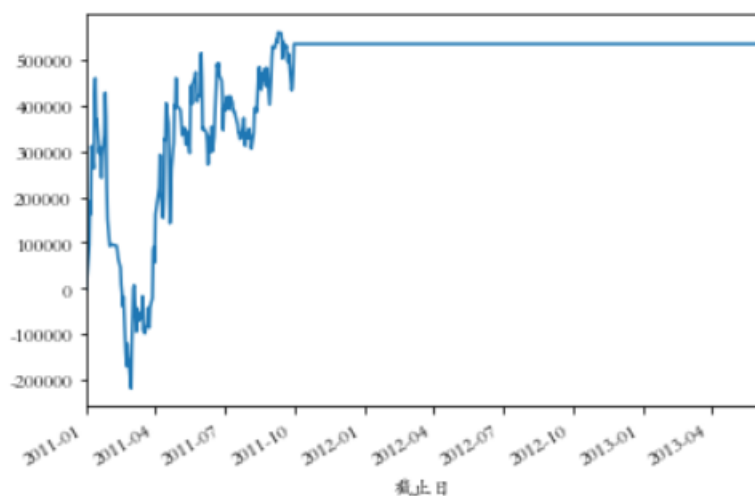
```
c = pyTSL.Client("E:\\my.ini") #登录信息
ts = TSBacktesting(df)
r=ts.run(c, datetime.datetime(2010,12,31), datetime.datetime(2013,5,30))
report = r.value(parse_date=['截止日'])
report.keys() #打印回测结果的下标如下所示：
```

dict_keys(['区间组合收益率', '交易汇总(按证券)', '相对回报', '风险回报', '滚动收益', '阶段收益', '组合盈亏', '交易明细', '资产配置', '持仓明细', '行业配置', '组合和基准收益率序列', '组合盈亏(按证券)', '交易汇总', '配置明细'])

```
df = pd.DataFrame(report["组合盈亏"])
df.index = df['截止日']
df.head() #打印前五组合盈亏数据
```

截止日	债券盈亏	截止日	其他资产累计盈亏	基金盈亏	手续费	当日盈亏	基金累计盈亏	金融衍生品累计盈亏	债券累计盈亏	金融衍生品盈亏	股票盈亏	当日盈亏(净额)	累计盈亏	股票累计盈亏	其他资产盈亏
2010-12-31	0.0	2010-12-31	0.0	0.0	18396.7575	-4613.5609	0.0	5378.7647	0.0	5378.7647	-9992.3256	13783.1966	-4613.5609	-9992.3256	0.0
2011-01-04	0.0	2011-01-04	0.0	0.0	0.0000	99387.5147	0.0	-111556.0967	0.0	-116934.8614	216322.3761	99387.5147	94773.9538	206330.0505	0.0
2011-01-05	0.0	2011-01-05	0.0	0.0	0.0000	95421.9317	0.0	-76431.1764	0.0	35124.9203	60297.0114	95421.9317	190195.8855	266627.0619	0.0
2011-01-06	0.0	2011-01-06	0.0	0.0	0.0000	-27305.8238	0.0	-47086.3062	0.0	29344.8702	-56650.6940	-27305.8238	162890.0617	209976.3679	0.0
2011-01-07	0.0	2011-01-07	0.0	0.0	0.0000	148149.3298	0.0	-25744.5824	0.0	21341.7238	126807.6060	148149.3298	311039.3915	336783.9739	0.0

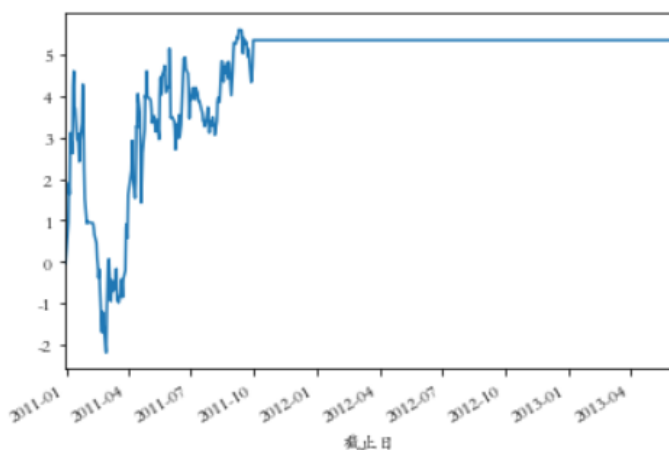
```
df['累计盈亏'].plot()      #累计盈亏画图
<matplotlib.axes._subplots.AxesSubplot at 0x21dcab6f848>
```



```
zhshy = pd.DataFrame(report['组合和基准收益率序列'])
zhshy.index = zhshy['截止日']
zhshy.head()      #打印前五行组合和基准收益率序列数据
```

截止日	累计超额收益率 (%)	截止日	日超额收益率 (%)	基准收益率 (%)	日期	复合指数	净值收益率 (%)	沪深300	基准累计收益率 (%)	累计净值收益率 (%)
2010-12-30	0.000000	2010-12-30	0.000000	0.000000	20101230	1000.000000	0.000000	1000.000000	0.000000	0.000000
2010-12-31	-2.140161	2010-12-31	-2.140161	2.094026	20101231	999.538644	-0.046136	1020.940257	2.094026	-0.046136
2011-01-04	-3.150824	2011-01-04	-0.969089	1.963423	20110104	1009.477395	0.994334	1040.985634	4.098563	0.947740
2011-01-05	-1.739047	2011-01-05	1.384803	-0.439542	20110105	1019.019589	0.945261	1036.410063	3.641006	1.901959
2011-01-06	-1.489309	2011-01-06	0.236469	-0.504430	20110106	1016.289006	-0.267962	1031.182097	3.118210	1.628901

```
zhshy['累计净值收益率 (%)'].plot()      #累计净值收益率 (%) 画图
<matplotlib.axes._subplots.AxesSubplot at 0x21dce70b9c8>
```

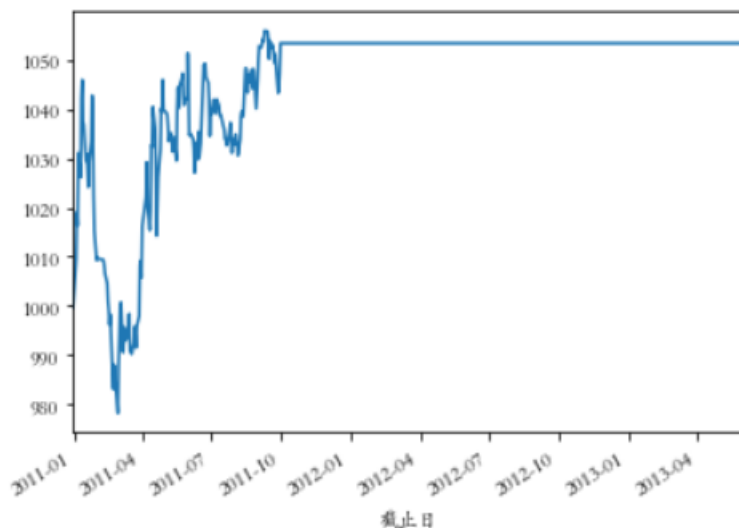


```

zhsy['复合指数'].plot()          #复合指数画图
[/code]

<matplotlib.axes._subplots.AxesSubplot at 0x21dce78b988>

```



6.2 图形数据的转换处理

运行环境：IDLE(Python 3.7 64-bit)

代码：

[code]

```

import pyTSL
import datetime
import pandas as pd
import plotly.graph_objects as go

c = pyTSL.Client("C:\\xxx\\my.ini") #ini 文件所在的路径
assert c.login() == 1

r = c.call('Kline', 5, 10, 20, 60, stock="SZ000001", time=datetime.datetime(2020,3,23),nday=180)

if r.error(): #有错误
    print(r.message())
else:
    kf = pd.DataFrame(r.value()['data'])
    df = kf['data'][0]
    df2 = pd.DataFrame.from_dict(df)

```

```
fig = go.Figure(data=[go.Candlestick(x=pd.to_datetime(df2['time']), format='%Y%m%d'),
                                open=df2['open'],
                                high=df2['high'],
                                low=df2['low'],
                                close=df2['close'])])

fig.show()

[/code]
```

返回结果：K 线图



7 对于老用户如何使用 pyTSLPy 模块替代 TSLPy3.pyd

天软对于 python 与 TSL 交互的技术逐渐完善，提供了些功能更为强大的 python 交互的模块，而对于老用户可能已经早已熟悉了使用 TSLPy3.pyd 来编写代码，所以提供了 pyTSLPy 模块，让用户尽可能少的修改原来的代码，做到既能兼容旧的 python 交互模块，又能使用到新模块的功能。

7.1 pyTSLPy 介绍

pyTSLPy 是使用 pytsl 模块实现了 TSLPy3.pyd 的功能的一个 python 交互模块。

7.2 安装 pyTSLPy

使用 cmd:输入命令 `pip install pyTSLPy` 即可开始安装。

提示：这样的安装方法是在 pip 库已经安装好的前提下进行的。如果没有安装或无法确定是否安装 pip 库，可以查看下文连接确定 [【pip 库的安装与版本检查】](#)

注意：pyTSLPy 只支持 python3.6+ 的交互，同 pyTSL。

截止 2022-6-20 日，支持如下版本：

Python	Windows	Linux(x86_64)	Linux(arm64) 适配华为鲲鹏	MacOS(x86_64)	MacOS(arm64) Apple M1
3.6	支持	支持	支持	支持	不支持
3.7	支持	支持	支持	支持	不支持
3.8	支持	支持	支持	支持	不支持
3.9	支持	支持	支持	支持	不支持
3.10	支持	支持	支持	支持	不支持

7.3 pyTSLPy 与 TSLPy 接口的对照说明

pyTSLPy 对 TSLPy3.pyd 的函数支持如下（接口与用法保持不变）：

模型	模型功能	pyTSLPy 对函数支持
DefaultConnectAndLogin	通过配置文件登陆	支持
ConnectServer	连接服务器	支持
LoginServer	登陆用户	支持
Disconnect	断开连接	支持
LoggedIn	判断是否连接	支持
SetService	设置服务器类型	支持
SetComputeBitsOption	设置计算服务器位数	暂不支持
GetComputeBitsOption	获得计算服务器位数	暂不支持
GetService	获得服务器类型	支持
RemoteExecute	远程执行 ts 语句	不完全支持（不支持本地交互）
RemoteCallFunc	远程执行 ts 函数	不完全支持（不支持本地交互）
SetSysParam	设置系统参数	暂不支持
GetSysParam	获得系统参数	暂不支持
EncodeDate	在 python 中构造 tsl 日期	支持
EncodeTime	在 python 中构造 tsl 时间	支持

EncodeDateTime	在 python 中构造 tsl 日期时间	支持
DecodeDate	将 tsl 的日期转换到 python 数组	支持
DecodeTime	将 tsl 的时间转换到 python 数组	支持
DecodeDateTime	将 tsl 的日期时间转换到 python 数组	支持

7.3.1 RemoteExecute、RemoteCallFunc 修改样例

RemoteExecute、RemoteCallFunc 接口与用法保持不变，由于是不依赖客户端平台，所以不能与本地交互，现只实现了 importfile 和 exportfile，其余的就是原需要加 rdo 和 rdo2 的函数，需要用户使用 python 重新实现或使用 pyTSL.register_proc 实现。

pyTSL.register_proc 实现如下：

- register_proc(funcname1,funcname2)
- 函数说明：注册 echo、rdo 类的方法，方便调用。
- 参数说明：

function1：字符串，函数名 1。

function2：函数名变量，函数名 2。

- 范例：

```
[code]
import pyTSLPy as ts
from pyTSLPy import pyTSL as pyTSL
ts.ConnectServer('tsl.tinysoft.com.cn', 443)
dl = ts.LoginServer('user', 'password')
def exportfile2(filename, value):
    print(filename,value)
    return 1

pyTSL.register_proc("exportfile2",exportfile2)#用 pyTSL.register_proc 重新
注册 TSL 中的 exportfile2 函数
r = ts.RemoteExecute(''''a := rdo2 exportfile2('1.csv',array(1,2,3));
return a+1;''',{})
[/code]
```


返回结果：

```

1 import pyTSLPy as ts
2 from pyTSLPy import pyTSL as pyTSL
3
4 ts.ConnectServer('tsl.tinysoft.com.cn', 443)
5 dl = ts.LoginServer('user', 'password')
6
7 def exportfile2(filename, value):
8     print(filename, value)
9     return 1
10 pyTSL.register_proc("exportfile2", exportfile2)
11 r = ts.RemoteExecute('a := rdo2 exportfile2('1.csv', array(1,2,3)); return a+1;', {})
12
1.csv [1, 2, 3]

```

7.4 如何修改代码来兼容 TSLPy3.pyd

修改步骤：

- 1、加载 pyTSLPy 包。
- 2、修改登陆配置。
- 3、看 tsl 代码中是否有使用本地交互函数，若有，则需修改此部分代码。

7.4.1 import pyTSLPy as ts

import pyTSLPy as ts 来替代掉原来的 import TSLPy3 as ts（一般都是重命名为 ts，若不是改为同名即可）

例如：加入第 2 行 import pyTSLPy as ts，注释或删除第 1 行的 import TSLPy3 as ts。

```

1 #import TSLPy3 as ts
2 import pyTSLPy as ts
3
4 ts.SetConnectConfig('tslclient.ini')
5 dl = ts.DefaultConnectAndLogin('test')
6 if dl[0] == 0:
7     print("登陆成功")
8     data = ts.RemoteExecute("return 'return a string';", {}) # 执行一条语句
9     print("数据:", data)
10    Close = ts.RemoteExecute("return close();", {"StockID": "SH000001"})
11    print("Close:", Close)
12    ts.Disconnect() # 断开连接
13 else:
14    print(dl[1])

```

7.4.2 修改登陆配置

由于 pyTSLPy 是不依赖客户端的，所以根据 TSLPy3 模块中登录天软服务器方式不同，修改方式也会有所不同。

7.4.2.1 使用登录客户端 com 形式调用天软

由于 pyTSLPy 是不依赖客户端的，所以登录客户端 com 方式的，需要改为配置相应登录参数，通过输入账号密码参数登录天软服务器。具体可看参考下文：[远程登录天软服务器](#)

7.4.2.2 远程登录天软服务器

7.4.2.2.1 通过配置文件登陆天软服务器

只需要原配置文件设置到代码中即可。

具体操作：在执行代码前加上 `ts.SetConnectConfig('C:/Program Files/Tinysoft/Analyse.NET/Plugin/tslclient.ini')`，参数为配置文件名路径。

```
1 #import TSLPy3 as ts
2 import pyTSLPy as ts
3
4 ts.SetConnectConfig('C:/Program Files/Tinysoft/Analyse.NET/Plugin/tslclient.ini')
5 dl = ts.DefaultConnectAndLogin('test')
6 if dl[0] == 0:
7     print("登陆成功")
8     data = ts.RemoteExecute("return 'return a string';", {}) # 执行一条语句
9     print("数据:", data)
10    Close = ts.RemoteExecute("return close();", {"StockID": "SH000001"})
11    print("Close:", Close)
12    ts.Disconnect() # 断开连接
13 else:
14    print(dl[1])
```

```
登陆成功
数据: (0, 'return a string', '')
Close: (0, 3287.6156, '')
```

注意：原配置文件在 Plugin 目录(天软安装目录的 plugin 目录)下的 `tslclient.ini`。或者自行重新配置。

`tslclient.ini` 配置文件内容如下

[TSLClient Config]

;调用是登陆的别名

[test]

;本地许可

Permit=local

;用户名

LoginName=

;密码

LoginPass=

;服务器 地址

Address=tsl.tinysoft.com.cn

;端口

Port=443

;以下为代理服务器

ProxyPort=

ProxyAddress=

ProxyUser=

ProxyPass=

proxy_user=

proxy_password=

代码范例：

```
#import TSLPy3 as ts
```

```
import pyTSLPy as ts
```

```
ts.SetConnectConfig('C:/Program Files/Tinysoft/Analyse.NET/Plugin/tslclient.ini')
```

```
dl = ts.DefaultConnectAndLogin('test')
```

```
if dl[0] == 0:
```

```
    print("登陆成功")
```

```
    data = ts.RemoteExecute("return 'return a string';", {}) # 执行一条语句
```

```
    print("数据:", data)
```

```
    Close = ts.RemoteExecute("return close();", {"StockID": "SH000001"})
```

```
    print("Close:", Close)
```

```
    ts.Disconnect() # 断开连接
```

```
else:
```

```
    print(dl[1])
```

7.4.2.2.2 通过函数设置登陆信息登陆

通过函数设置登陆信息登陆，只需完成上述的第一步的 [import pyTSLPy as ts](#) 即可。

```

1 #import TSLPy3 as ts
2 import pyTSLPy as ts
3
4 ts.ConnectServer('tsl.tinysoft.com.cn', 443)
5 dl = ts.LoginServer('user', 'password')
6 if dl[0] == 0:
7     print("登陆成功")
8     data = ts.RemoteExecute("return 'return a string';", {}) # 执行一条语句
9     print("数据:", data)
10    ts.Disconnect() # 断开连接
11 else:
12    print(dl[1])

```

代码范例：

```
#import TSLPy3 as ts
```

```
import pyTSLPy as ts
```

```
ts.ConnectServer('tsl.tinysoft.com.cn', 443)
```

```
dl = ts.LoginServer('user', 'password')
```

```
if dl[0] == 0:
```

```
    print("登陆成功")
```

```
    data = ts.RemoteExecute("return 'return a string';", {}) # 执行一条语句
```

```
    print("数据:", data)
```

```
    ts.Disconnect() # 断开连接
```

```
else:
```

```
    print(dl[1])
```

7.4.3 tsl 代码中是否有使用本地交互函数

如果 tsl 代码中是否有使用本地交互函数，可查看 [7.3.1 RemoteExecute、RemoteCallFunc 修改样例](#)

7.5 如何使用 pyTSLPy 其他功能

pyTSLPy 还有其他的功能主要是用 pytsl 模块实现，可以直接 import pyTSL，使用 pyTSL 的功能，具体 pyTSL 的用法参考上文

8 常见问题

8.1 使用 pyTSL 时出错 RuntimeError: Could not allocate string object!

解决方法：旧版本 pyTSL 模块有 bug，请用最新版本：<http://py3k.cn/pyTSL/setup.html#id2>

8.2 import pyTSL 时出错 ImportError: DLL load failed: 找不到指定的模块。

解决方法：系统还需要安装 VS2019 的 C++运行时库，请按照步骤配置：<http://py3k.cn/pyTSL/setup.html#>

8.3 最新版本的 mac 交互时 import pyTSL 遇到报错

A：比如报错如下：

```
ImportError: dlopen(/Users/Bo/opt/anaconda3/envs/prophet/lib/python3.7/site-packages/pyTSL.cpython-37m-darwin.so, 2): no suitable image found. Did find:
/Users/Bo/opt/anaconda3/envs/prophet/lib/python3.7/site-packages/pyTSL.cpython-37m-darwin.so: code signature in (/Users/Bo/opt/anaconda3/envs/prophet/lib/python3.7/site-packages/pyTSL.cpython-37m-darwin.so) not valid for use in process using Library Validation: library load disallowed by system policy
/Users/Bo/opt/anaconda3/envs/prophet/lib/python3.7/site-packages/pyTSL.cpython-37m-darwin.so: stat() failed with errno=35
```

原因：这是因为 mac 系统版本不同，授权签名不同，并且改过签名后还是需要开放权限。

解决方案：可以用命令 `sudo spctl --master-disable` 解决

8.4 使用 Client(ini_file)时报找不到文件错误

解决方法：使用 GBK 编码，例如

```
ini_file="D:\\测试\\a.ini"
```

```
c=pyTSL.Clinet(ini_file.encode('gbk'))
```

8.5 c.logout 后，再次执行 c 的方法会自动重新登陆

如果在 `c.logout()` 返回 0，退出登陆再执行 `c`（即 `pyTSL.Client` 类）的方法时会自动重新登陆。所以，退出登陆后不要再执行相关调用，会导致登陆数被占用，当不需要再使用时应该直接删除 `c`，比如 `del c`。

8.6 pyTSL 中返回值编码为 utf-8

天软客户端与服务器中是 gbk ， pyTSL 中返回值编码为 utf-8，用户使用 pytsl 时无需再进行编码的转换，模块内进行自动进行转换。