

---

深圳天软科技



# 天软平台和 Python 的交互

第三方交互 07



## 1 更新日志

### 1.1 更新日志

| 更新日期       | 更新说明   |
|------------|--|
| 2014-06-23 | 文档创建和发布  |
| 2015-09-14 | TSLPy2.pyd; TSLPy3.pyd 模块支持  |
| 2016-02-03 | 添加说明   |
| 2018-01-05 | 添加 byte 类型转换为字符串方法到附录  |
| 2019-05-08 | 添加 python 调用天软配置说明<br>添加 pythonpath 配置<br>添加字符串转换的 python2 的版本限制   |
| 2019-07-15 | 新增 TSL 通过 do Python 的方法执行 python 脚本  |
| 2019-07-25 | 新增 python 调用天软的两个取数范例：<br>2.4.3 两种方式取截面数据包括基本面数据，财务数据，行情数据<br>2.4.4 两种方式取行情数据的时间序列数据<br>新增天软调用 python 时重载 python 模块，清除变量的方法及范例 |
| 2019-08-19 | 新增 TSL 调用 Python 的函数 PyErr()说明及范例<br>新增注意事项本地 Python 版本切换注意系统环境变量 Pythonhome 的更改<br>新增落地服务器开启 Python 服务的配置方式及范例                |

## 1.2 摘要

1. Tsl 调用 python  
推荐使用 `pyputvar`(送入数据);`pyrun`(执行脚本);`pygetvar`(获取数据);`pyerrore`(获取报错内容)模式
2. Python 调用 tsl  
`RemoteCallFunc` 调用天软函数  
`RemoteExecute` 执行天软脚本
3. 注意事项
  - 1) 默认 `TSLPy3.PYD` 支持 `python3.4` ;`TSLPy2.PYD` 支持 `python 2.7`;如果使用 `Python3.5` 请将 `TSLPy35.PYD` 复制成为 `TSLPy3.PYD`; 用 `Python3.4` 请 `TSLPy34.PYD` 复制为 `TSLPy3.PYD`;
  - 2) 注意模块名称的大小写,如果目录下的 `TSLPY3.PYD` 请修改为 `TSLPy3.PYD`
  - 3) 当 `TSLPy` 模块存在更新后, 建议使用最新版本的 `TSLPy` 模块, 否则会导致部分 `TSL` 调用 `Python` 的函数出错, 例如 `PyError()`。
  - 4) `tsl` 和 `python` 的位数应该对应,不可混用。
  - 5) 当安装多个版本的 `Python` 时, 如果要进行 `Python` 版本的切换, 除了修改 `TSLPy` 模块外, 还需修改环境变量 `PYTHONHOME`, 将环境变量中的 `Python` 路径修改为当前版本的 `Python` 路径。
  - 6) `Python` 远程调用天软取数据的时候请设置系统参数
  - 7) 基础类型自动转换,自定义类型,需要用户自行将其转换为基础类型.(附录中有一个程序可以将 `list,tuple,dict` 转换为字符串,克服其部分单元格为中文的显示问题)
  - 8) `Python`,天软的安装目录都请添加到系统环境变量中
  - 9) 如果远程调用无法执行,重启 `python` 试一下
  - 10) 某些 `python` 编辑器(如 `pycharm`)可能会有缓存导致调用不成功可以清除一下缓存
  - 11) 在客户端编辑 `.py` 文件, 将自动以 `UTF8` 的格式存贮, 以吻合目前 `python` 的默认编码为 `UTF8` 格式。

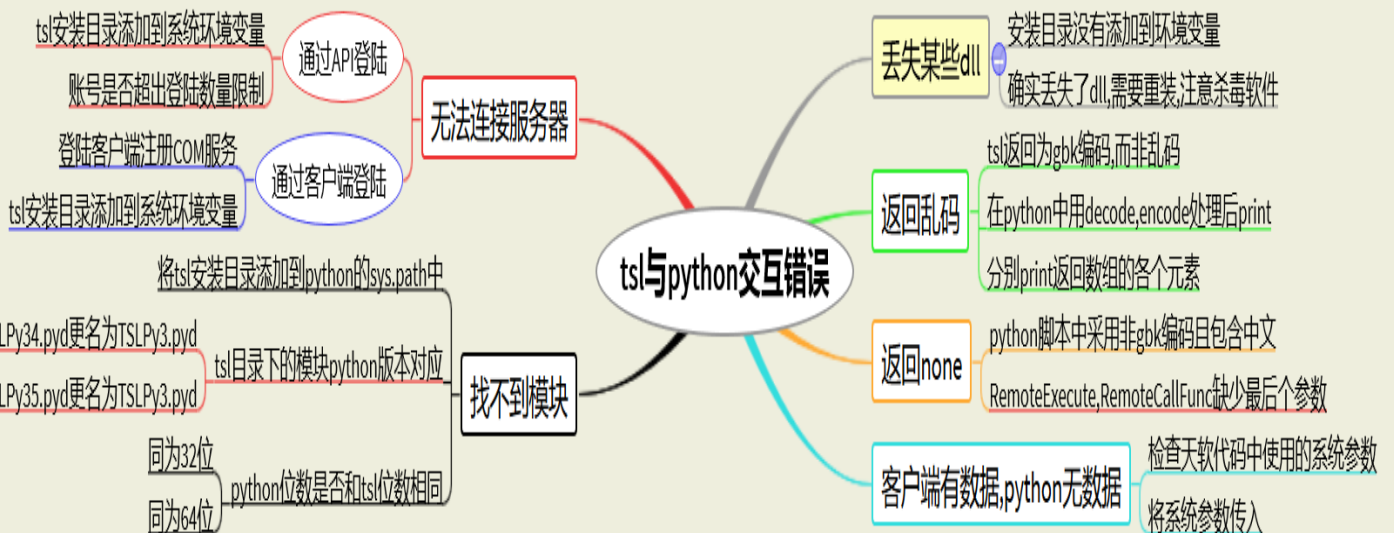
### 1.2.1 对应模块支持版本

| Python 版本 | 天软目录下的 python 模块         | 操作                              |
|-----------|--------------------------|---------------------------------|
| 2.7       | <code>TSLPy2.PYD</code>  |                                 |
| 3         | <code>TSLPy3.PYD</code>  |                                 |
| 3.4       | <code>TSLPy34.PYD</code> | 请将模块更名为 <code>TSLPy3.PYD</code> |
| 3.5       | <code>TSLPy35.PYD</code> | 请将模块更名为 <code>TSLPy3.PYD</code> |
| 3.6       | <code>TSLPy36.PYD</code> | 请将模块更名为 <code>TSLPy3.PYD</code> |
| 3.7       | <code>TSLPy37.PYD</code> | 请将模块更名为 <code>TSLPy3.PYD</code> |

- 1) 确保天软目录下 `python` 模块名的**太小写**同上表格一致;如果不一致请修改模块名。
- 2) 当 `TSLPy` 模块存在更新后, 建议使用最新版本的 `TSLPy` 模块, 否则会导致部分 `TSL` 调用 `Python` 的函数出错, 例如 `PyError()`。

### 1.3 错误处理

| 错误描述                 | 解决方案  |
|----------------------|---|
| Python 调用返回为 none    | 1.登陆是否成功  |
|                      | 2.Python 脚本文件请采用 gbk 编码   |
|                      | 3.客户端信任配置   |
|                      | 4.调用接口少送入了一个参数  |
| Python 调用登陆失败        | 1.客户端信任配置设置   |
|                      | 2.天软客户端是否添加到系统环境变量(最好放最前面)  |
| Python 调用找不到模块       | 将天软安装目录添加到 python 的 sys.path 中,需要重启 python                        |
| Python 调用不是有效的 win32 | 确认 python 和天软位数相同   |
| Python 调用返回乱码        | 天软返回是是 gbk 编码如果是中文,会出现乱码,如果返回一个数组,可以分别 print 数组的各个元素,可以参考附录中的两个范例 |



2 Python 调用 tsl

2.1 Python 调用天软配置步骤说明

- 1) 检查 Python 和天软位数是否一致，同为 64 位或者同为 32 位，不可混用。
- 2) 修改天软主目录下的对应模块名。

具体方法：打开天软客户端》左上角系统》系统设置》打开运行/存贮目录

默认 TSLPy3.PYD 支持 python3.4 ;TSLPy2.PYD 支持 python 2.7;

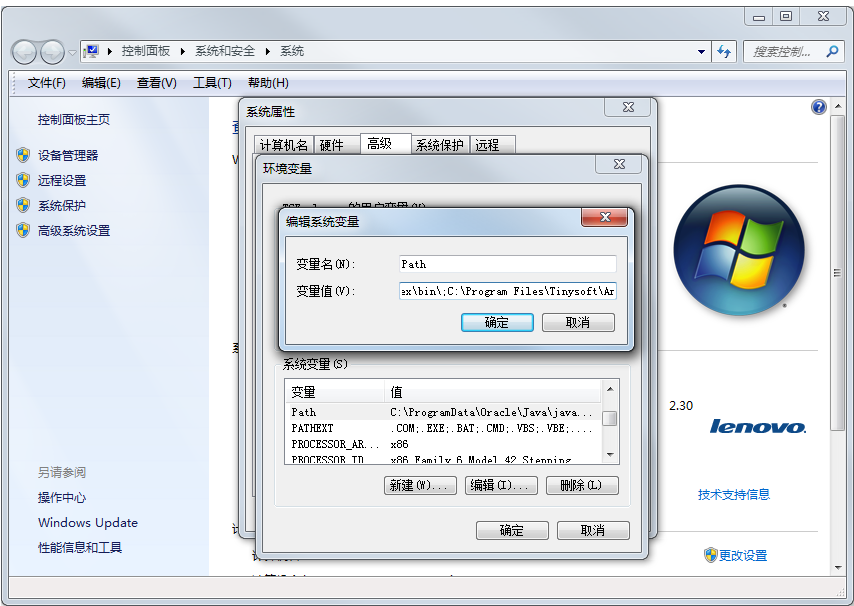
如果使用 Python3.5 请将 TSLPy35.PYD 复制成为 TSLPy3.PYD; 用 Python3.4 请 TSLPy34.PYD 复制为 TSLPy3.PYD;

注意模块名称的大小写,如目录下的 TSLPY3.PYD 请修改为 TSLPy3.PYD,确保大小写与表中一致。

| Python 版本 | 天软目录下的 python 模块 | 操作                 |
|-----------|------------------|--------------------|
| 2.7       | TSLPy2.PYD       |                    |
| 3         | TSLPy3.PYD       |                    |
| 3.4       | TSLPy34.PYD      | 请将模块更名为 TSLPy3.PYD |
| 3.5       | TSLPy35.PYD      | 请将模块更名为 TSLPy3.PYD |
| 3.6       | TSLPy36.PYD      | 请将模块更名为 TSLPy3.PYD |
| 3.7       | TSLPy37.PYD      | 请将模块更名为 TSLPy3.PYD |

- 3) 将天软客户端所在路径设置成系统路径

具体方法：右击计算机，属性->高级系统设置->环境变量->系统变量中找到 path,点击编辑，加入天软客户端所在路径，这里最好是将天软的路径加在 path 中的第一条。



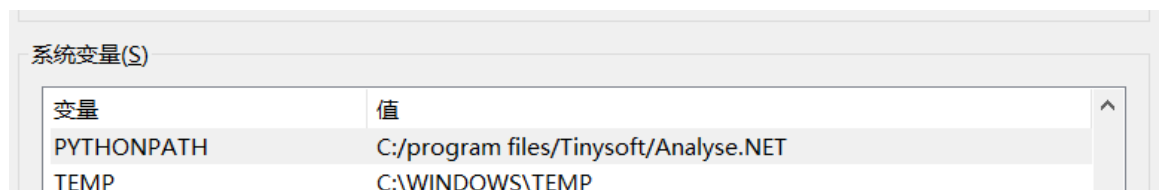
**注意：修改系统环境变量后需要重启电脑，重启后修改的系统环境变量才能生效**

- 4) 将天软支持模块添加到 python 目录。具体操作方法：假设天软所在安装路径是：C:/program files/Tinysoft/Analyse.NET;

一种方式是：此方式要求在每个 python 脚本中都要写上下列语句，导入系统模块，拓展路径

```
import sys
sys.path.append("C:/program files/Tinysoft/Analyse.NET")#
import TSLPy2 #导入模块
```

另一种方式是：此方式是在系统环境变量中新建名为 **PYTHONPATH** 的路径，在其中添加天软所在的安装路径 C:/program files/Tinysoft/Analyse.NET;，运用此方法则不需要在每次编写调用天软的脚本时加上 import sys sys.path.append(.....)，直接就能 import TSLPy2/ TSLPy3  
运用此方式时，设置系统环境变量后要重启电脑。



- 5) 这里用户需要确认，是否采用启动天软客户端的形式登录天软服务器，若选择启动天软客户端形式登录，则只需要打开并登录天软客户端即可；否则则需要在 Python 中配置相应登录参数，配置参数参见：[在 Python 中远程登录天软服务器](#)；
- 6) 在 Python 调用天软时，com 必须是注册成功的，用户需要先检查这一点，注意 com 注册中需要关闭 360 等安全软件，防止其阻拦 com 注册。

com 注册参考：<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=15144>

检查 com 注册是否成功参考：<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=15124>

- 7) 执行一段语句，确认配置成功：

```
import sys
sys.path.append(天软安装目录) #指天软的安装路径如 D:\tinysoft\Analyse.NET
import TSLPy2 #导入模块
data = TSLPy2.RemoteExecute("return 1;",{ }) #执行语句
结果:(0, 1, ' ', 0) ;#data[0]==0 表示成功,data[1] 运行结果,
Data[2] 如果错误的错误信息
```

## 2.2 在 Python 中远程登录天软服务器（使用登录客户端形式调用天软不要考虑这一项）

在 Python 要登录天软服务器中，有两种方法：

- (1) [#通过配置文件登陆天软服务器](#)
- (2) [#通过函数设置登陆信息登陆](#)
- (3) 客户端登陆的情况情况下可以直接使用

我们建议客户端登陆天软服务器,因为在程序中不需要其他额外设置,就可执行,如果采用其他方式登录请将

tsl 按照目录添加到系统环境路径中(如果系统路径太多,请放在最前面)

### 2.2.1 通过配置文件登陆天软服务器

在 Plugin 目录(天软安装目录的 plugin 目录)下 添加 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=**

**注: 配置完成后重启 python 和天软生效**

登陆范例:

```
import sys
sys.path.append(天软安装目录)
import TSLPy2 #导入模块
TSLPy2.DefaultConnectAndLogin("test") #调用函数登陆
```

### 2.2.2 通过函数设置登陆信息登陆执行 tsl 语句

```
#coding=gbk
import sys
sys.path.append(天软安装目录)
import TSLPy2 as ts
ts.ConnectServer("tsl.tinysoft.com.cn",443)
dl = ts.LoginServer("用户名","密码") #Tuple(errno,errmsg) 登陆用户
```

```
if dl[0]==0 :
    print "登陆成功"
    print "服务器设置:",ts.GetService()
    ts.SetComputeBitsOption(64) #设置计算单位
    print "计算位数设置:",ts.GetComputeBitsOption()
    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]
```

## 2.3 TSLPy2/TSLPy3 模块介绍

天软将一些常用功能封装到 TSLPy2.pyd 或者 TSLPy3.pyd 中这些函数包括

### 2.3.1 服务器登陆相关

- DefaultConnectAndLogin(Alias:String):tuple(ErrNo,ErrMsg)  
功能:通过配置文件登陆
- ConnectServer(Addr:String;port:Integer[;list|tuple(proxyaddr,proxyport[,proxyuser,proxypass]))  
功能:连接服务器,参数一次是地址,端口,代理服务器;返回为 0 表示连接成功
- LoginServer(User:String;Pass:String):Tuple(ErrNo,ErrMsg)  
功能:登陆用户
- Disconnect():NONE  
功能:断开连接
- Logined():Boolean;  
功能:判断是否连接
- SetService(Service:String)  
功能:设置服务器类型;新功能测试版或者正式版
- SetComputeBitsOption(Option:Integer)  
功能:设置计算服务器位数; 32 或者 64
- GetComputeBitsOption():Integer  
功能:获得计算服务器位数
- GetService():String  
功能:获得服务器类型

### 2.3.2 执行相关

- RemoteExecute(ExecString:String;SysParam:dict):  
函数说明:远程执行 ts 语句  
参数说明:



ExecString 天软语句的字符串;

SysParam: 天软执行的系统参数,字典类型,参考[系统参数简介](#)

返回说明: Tuple(ErrNo,Result,ErrMsg)

ErrNo : 错误标志 0 成功,

Result 结果,

ErrMsg 如果错误的时候的信息

- **RemoteCallFunc(Func:String;Args:list|tuple;SysParam:dict):Tuple(ErrNo,Result,ErrMsg);**

函数说明:远程执行 ts 函数

参数说明:

Func 天软函数名;

Args:天软函数的参数列表,类型为 python 的数组,每个元素对应天软函数的一个参数

**注意:如果天软函数需要的参数为表达式类型,传递会失败,需要另外想办法:比如重新封装函数;然后传入字符串,将字符串转换为天软的表达式类型**

SysParam: 天软执行的系统参数,字典类型,参考[系统参数简介](#)

返回说明: Tuple(ErrNo,Result,ErrMsg)

ErrNo : 错误标志 0 成功,

Result 结果,

ErrMsg 如果错误的时候的信息

- **SetSysParam(Name:String;Value:Any):Boolean ;**

功能:设置系统参数

注意: 该函数只能在本地调用,无法将此设置的参数传到天软服务器,即其设置的参数无法影响 RemoteExecute 和 RemoteCallFunc 中设置的参数

- **GetSysParam(Name:String):Any**

功能:获得系统参数

注意: 该函数只能在本地调用,无法用此函数获取天软服务器上的参数,即其无法获取 RemoteExecute 和 RemoteCallFunc 中设置的参数

### 2.3.3 日期转换相关

- **EncodeDate(Y,M,D):double**  
功能:在 python 中构造 tsl 日期
- **EncodeTime(h,m,s,ss): double**  
功能:在 python 中构造 tsl 的时间
- **EncodeDateTime(Y,M,D,h,m,s,ss):double**  
功能:在 python 中构造 tsl 的日期时间
- **DecodeDate(date):Tuple(Y,M,D)**  
功能: 将 tsl 的日期转换到 python 数组
- **DecodeTime(time):Tuple(h,m,s,ss)**  
功能:将 tsl 的时间转换到 python 的数组
- **DecodeDateTime(datetime):Tuple(Y,M,D,h,m,s,ss)**  
功能:将天软的日期时间转换到 python 的数组

## 2.4 Python 中调用天软范例

### 2.4.1 例 1: 取个股收盘价

*#打开并登陆了客户端的情况*

```
import sys
sys.path.append(天软安装目录)
import TSLPy2 #导入模块
Close = TSLPy2.RemoteExecute("return close();", {"StockID": "SH000001"})
#注意第二个参数为系统参数不可省略,但是可以为空{}
Stockname = TSLPy2.RemoteCallFunc("stockname", ["SH000001"], {});
#注意第三个参数为系统参数,不可省略,但是可以为空{}
```

**备注:** *RemoteExecute* 和 *RemoteCallFunc* 函数说明参考:[执行相关](#)

### 2.4.2 例 2: 交易策略验证

本范例旨在演示如何进行交易策略验证,在一段时间内,周期为日线,一篮子股票在 10 日线上穿 20 日线时买入,20 日线上穿 10 日先时卖出。

```
import sys
sys.path.append(天软安装目录)
import TSLPy2 #导入模块
tsstr="""
Stocks:=
"SZ000001;SZ000002;SZ000004;SZ000005;SZ000006;SZ000007;SZ000008;SZ000009;SZ
000010;SZ000011;SZ000012;SZ000014;SZ000016;SZ000017;SZ000018;SZ000019;SZ000
020";
BegT := 20151001T;
EndT := 20160531T;
cy := '日线';
BuyCond := @Cross(MA(Close(),10),MA(Close(),20))=1;
SellCond := @Cross(MA(Close(),20),MA(Close(),10))=1;
return TSFL_PB_TechnicalIndicator(Stocks,BegT,EndT,cy,BuyCond,SellCond);
"""
data = TSLPy2.RemoteExecute(tsstr, {})
print(tostry(data[1]))
```

**注:** *tostry* 函数代码参考附录,相关的编码也可以参考, *tostry* 的代码需要复制在此函数下,或封装附录中 *tostry* 的代码为 *python* 的函数

### 2.4.3 例 3: 取多个个股指定日指标

取浦发银行 SH600000, 万科 A SZ000002, 在指定日的截面数据, 返回: 代码, 名称, 日期, 上市日, 指定日总股本, 指定日最新报告期, 最新报告期营业收入, 最新报告期归属母公司的净利润, 指定日开

盘价, 指定日最高价, 指定日最低价, 指定日收盘价, 指定日成交量, 指定日成交金额。

### 2.4.3.1 方式一: 调用天软语句串

调用天软语句串, 使用函数的方式取高开低收等行情数据

```
import sys
import tsbytestostr
sys.path.append(天软安装目录)
import TSLPy3 as ts
#日期可以通过参数的方式输入 字符串形式
endt='20190722'
ts_str="""stockarr:=array('SH600000','SZ000002');
    endt:="""+endt+"""+T;
    a:=array();
    for i:=0 to length(stockarr)-1 do
    begin
        stk:=stockarr[i];
        setsysparam(pn_stock(),stk);
        SetSysParam(pn_date(),endt);
        SetSysParam(PN_Cycle(),cy_day());
        //指定日最新报告期
        RDate:=NewReportDateOfEndT2(endt);
        a[i]['代码']:=stk;
        a[i]['名称']:=stockname(stk);
        a[i]['指定日']:=datetostr(endt);
        a[i]['上市日']:=base(12017);
        a[i]['总股本']:=StockTotalShares(endt);
        a[i]['指定日最新报告期']:=rdate;
        a[i]['营业收入']:=report(46002,RDate);
        a[i]['归属于母公司所有者净利润']:=report(46078,RDate);
        a[i]['开盘价']:=open();
        a[i]['最高价']:=high();
        a[i]['最低价']:=low();
        a[i]['收盘价']:=close();
        a[i]['成交量']:=vol();
        a[i]['成交金额']:=amount();
    end
    return a;"""
data=ts.RemoteExecute(ts_str,{})
#tsbytestostr 见附录 4.3.1
data=tsbytestostr.tsbytestostr(data)
print(data)
```

### 2.4.3.2 方式二：调用天软函数

调用事先封装好的天软函数，在 python 和天软交互中推荐事先封装好天软函数然后在 python 中调用该函数，效率高且稳定。

在天软平台中封装函数

GetData\_01(StockArr,Endt)

定义：GetData\_01(StockArr:Array(),Endt:TDatetime):Array;

参数：

StockArr: 股票列表

Endt: 指定日

返回：表格数据

函数实现代码：

```
Function GetData_01(Stockarr,Endt);
Begin
    a:=array();
    for i:=0 to length(stockarr)-1 do
    begin
        stk:=stockarr[i];
        setsysparam(pn_stock(),stk);
        SetSysParam(pn_date(),endt);
        SetSysParam(PN_Cycle(),cy_day());
        //指定日最新报告期
        RDate:=NewReportDateOfEndT2(endt);
        a[i]['代码']:=stk;
        a[i]['名称']:=stockname(stk);
        a[i]['指定日']:=datetostr(endt);
        a[i]['上市日']:=base(12017);
        a[i]['总股本']:=StockTotalShares(endt);
        a[i]['指定日最新报告期']:=rdate;
        a[i]['营业收入']:=report(46002,RDate);
        a[i]['归属于母公司所有者净利润']:=report(46078,RDate);
        a[i]['开盘价']:=open();
        a[i]['最高价']:=high();
        a[i]['最低价']:=low();
        a[i]['收盘价']:=close();
        a[i]['成交量']:=vol();
        a[i]['成交金额']:=amount();
    end
    return a;
end
```

在 python 中调用该函数，实现代码

```
import sys
import tsbytestostr
sys.path.append(天软安装目录)
import TSLPy3 as ts
stockarr=['SH600000','SZ000002']
endt=ts.EncodeDate(2019,7,22)
data=ts.RemoteCallFunc("GetData_01",[stockarr,endt],{})
#tsbytestostr 见附录 4.3.1
data=tsbytestostr.tsbytestostr(data)
print(d) print(data)
```

#### 2.4.4 例 4：取个股时间序列行情

取 浦发银行 SH600000 在一段时间内的日线数据

##### 2.4.4.1 方式一：调用天软语句串

```
import sys
import tsbytestostr
sys.path.append(天软安装目录)
import TSLPy3 as ts
stockid='SH600000'
begt=20190701
endt=20190710
ts_str=""" StockId:='%s';
        begt:=%dT;
        ENDT:=%dT;
        setsysparam(PN_Cycle(),cy_day());
        return select *,datetostr(['date']) as 'date' from markettable
        datekey Begt to Endt+0.99999 of StockId end;
        """%(stockid,begt,endt)
data=ts.RemoteExecute(ts_str,{})
#tsbytestostr 见附录 4.3.1
data=tsbytestostr.tsbytestostr(data)
print(data)
```

##### 2.4.4.2 方式二：调用天软函数

调用事先封装好的天软函数，取浦发银行在一段时间内行情数据

在天软平台中封装函数

**GetData\_02(StockID,Begt,Endt)**

定义：**GetData\_02(StockID:String,Begt:Tdatetime,Endt:Tdatetime):Array;**

参数：**StockID**: 股票列表 **Begt**: 开始日 **Endt**: 截止日

返回：表格数据

函数实现代码：

```
Function GetData_02(stockID,Begt,endt);
Begin
    setsysparam(PN_Cycle(),cy_day());
    return select *,datetostr(['date']) as 'date' from markettable datekey
    Begt to Endt+0.99999 of StockId end;
end
```

在 python 中调用该函数，实现代码

```
import sys
import tsbytestostr
sys.path.append(天软安装目录)
import TSLPy3 as ts
stockid='SH600000'
begt=ts.EncodeDate(2019,7,1)
endt=ts.EncodeDate(2019,7,10)
data=ts.RemoteCallFunc("GetData",[stockid,begt,endt],{})
#tsbytestostr 见附录 4.3.1
data=tsbytestostr.tsbytestostr(data)
print(data)
```

在以上例子中，演示了取截面数据和时间序列数据的代码，既包括基本面数据，财务数据，也包括截面的行情数据和时间序列的行情数据。在交互过程中，建议减少调用次数，提高交互效率，建议先在天软平台封装好函数后在 python 调用，这样可以减少天软语言错误，提高取数效率和稳定性。当提取的数据量过大时，建议分批提取以提高取数效率。

除此之外，我们也准备了很多取数的范例，此处篇幅有限，详情请转：

#### **Python 调天软取数范例01：行情数据的提取**

<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=17211>

#### **Python 调天软取数范例02：基本面数据的提取**

<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=17212>

#### **Python 调天软取数范例03：财务数据的提取**

<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=17213>

## 3 TSL 调用 Python

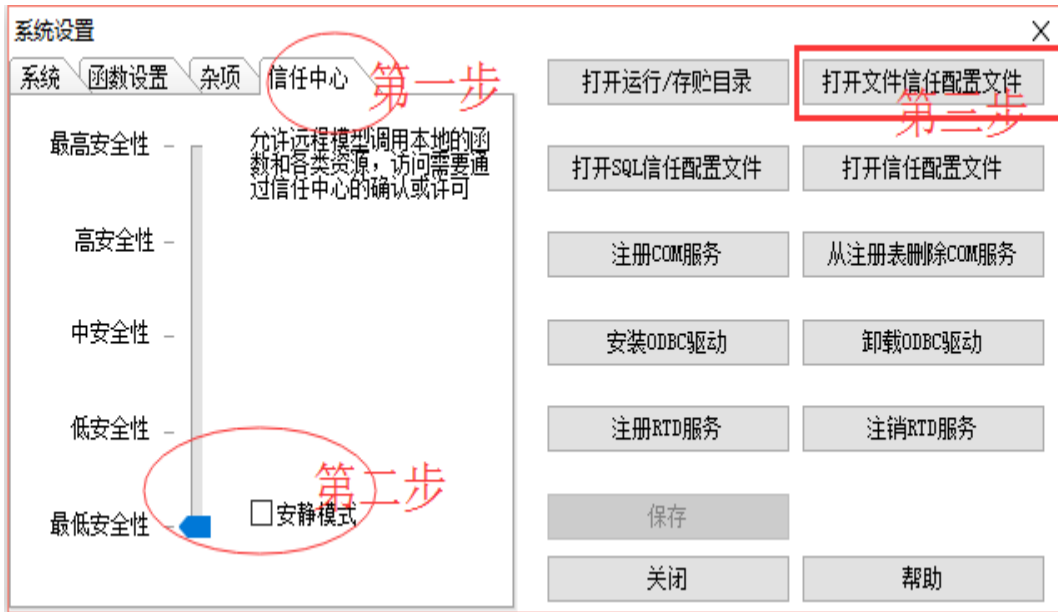
### 3.1 配置

1. 注意 Python\dlls 应该在系统的路径中。

相关设置方法请参考 <http://jingyan.baidu.com/article/d5a880eb6aca7213f047cc6c.html>

**注：配置完成需要重启电脑，系统路径才能生效。**

2. 在天软平台增加信任配置：登陆天软客户端→系统(左上角菜单)→系统设置→打开文件信任配置文件



3. 在打开的 FileMgr.INI 文件中，添加如下内容：

[Python Config]

Enabled=1

version=3 //如果版本为 2.7 则 version=2

**注：配置完成后重启 python 和天软生效**

4. 如果是在服务器端配置与 python 的交互信任配置文件添加如下内容

[Python Config]

Enabled=1

version=3 //如果版本为 2.7 则 version=2

EnginePermit=1 //这表明所有用户均有权调用 Python

EnginePermitUser=User1;User2 //这表明 User1 和 User2 用户有权调用 Python

**注：配置完成后重启 python 和天软生效**

## 3.2 函数支持

- PyPutVar(VarName:String;VarValue:Any):Integer;  
功能:送入 python 变量
- PyGetVar(VarName:String):Any;  
功能:从 python 获得变量
- PyRun(PythonCode:String[;flag:Integer]):Integer;  
功能:简单运行 Python 的代码  
参数:  
flag 默认不被指定，该含义由 PyCompileFlags 指定组合的状态，处理 UTF-8 源代码等问题。具体请参考 Python 手册。

注意:

每次只能运行一条 python 语句;

- PyCall([callFlag:Integer;]ModuleName:String;FuncName:String;Param1..ParamN):Any;

功能:调用 python 模块中的函数

参数:

callFlag 是一个按位的状态组合字

- 1) 设置为 1 则直接返回结果、找不到 ModuleName、FuncName 则函数报错。
- 2) 如果不设置状态 1, 则返回值为一个数组,第 0 项目为 boolean 型表明是否调用成功, 第 1 项为错误字符串。
- 3) 设置状态 2 则函数参数以一个数组送入。否则送入的参数以参数的方式一个个送入。
- 4) callFlag 的默认值为 1, 即默认直接返回结果, 参数一个个送入。

ModuleName 模块名

FuncName 模块中的函数名

- PyErr():String;  
功能:获取当前运行 python 语句时的报错信息
- PyRelease():Boolean;  
功能:释放 python 资源,多线程的时候可以提高效率
- PyEnsure():Boolean;  
功能:锁定 python 资源; 由于 pyrun 和 pyPutVar,pyGetVar 会智能锁定, 所以用户一般来说不需要调用该函数。

### 3.3 调用范例

例 1:

```
a:=2;
rdo2 pyputvar("a",a); //送入变量
rdo2 pyrun("a = a+1"); //计算
b := rdo2 pygetvar("a"); //获取变量 返回结果为 3
```

例 2:

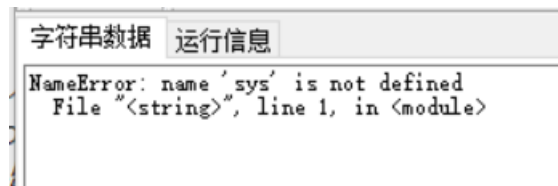
```
rdo2 PyRun("import math"); //执行 python 语句 :导入 python 的 math 模块
a :=rdo2PyCall("math","sqrt",2) ; //a = 1.4142135623731
//调用 math 模块下的 sqrt(开方)函数,参数为 2;
b := rdo2 PyCall("math","pow",2,3) ; //b = 8;
//调用 math 模块下的 pow 函数,参数为 2,3 ;
Return C := a*b;
```

例 3:

推荐将 pyrun 等调用 python 的语句封装在一个函数中, 然后 RDO2 执行这个函数, 减少 rdo2 的次数以提升函数运行效率



```
Function pyrtest();
Begin
    return rdo2 PyErrortest();
end
function PyErrortest()
begin
    pyrun("a=sys.path"); //不导入 sys, 此处 Python 应该会有报错
    c:=pyerror();//返回此时 Python 的报错
    return c;//返回 Python 报错信息
end
返回:
```



## 3.4 多线程调用 python 注意事项

### 3.4.1 Python 中变量的全局性

Python 中的变量对 tsl 调用为全局变量;

例如:

```
function test1()
begin
    pyputvar("a",'b');//修改 python 中的变量 a 的值
end
function test2();
begin
    pyputvar("a",'c');//修改 python 中的变量 a 的值
end

pyputvar("a",'a');
test1();
test2();
return pygetvar("a");//结果为 'c';
```

### 3.4.2 多线程注意事项

当多线程调用 python 的时候调用相关的 python 调用函数会对 python 资源进行锁定直到该线程结束. 为提高效率,当某个线程不需要调用 python 的时候可以使用 pyRelease()手动对 python 资源释放;与 pyRelease() 函数对应的还有 pyEnsure()该函数锁定 python 解释器资源。由于 pyrun 和 pyPutVar,pyGetVar 会智能锁定, 所以用户一般来说不需要调用该函数。

## 4 TSL 通过 do python 的方式运行 python 脚本

在实现天软调用 python 后，客户端可以通过 do python 的方式本地运行 python 语句

### 4.1 语法

```
do extlanguage [paramlist] <scriptstring> [by config]
```

其中 **extlanguage** 指的是拓展的第三方语言，目前已支持 python ,matlab,R, 今后可能更多的语言会被扩展支持，底层用户也可以通过接口规范来扩展支持其他语言。本文中主要演示 do python。

### 4.2 关键字

假如有变量 A，在 do Python 时需要在 python 中调用

| 序号 | 关键字  | 说明     | 示例         |
|----|------|--------|------------|
| 1  | In   | 只送入变量  | In A       |
| 2  | Out  | 只送出变量  | Out A      |
| 3  | Var  | 送入送出变量 | Var A      |
| 4  | Name | 重命名变量  | A name "a" |

注：do python 只能送入送出变量，执行完 python 语句后并不能直接返回结果，因为 python 并没有直接返回结果的 return 语句，所以需要出参来送出 python 执行完的结果，然后在天软中使用 return 返回送出的变量。

如果使用变量名，会默认将同名变量名送入到外部语言中，但由于 TSL 的变量名大小写无关，因而送入的都是小写，假如目标语言大小写相关，可以用 name 关键字。

### 4.3 本地交互示例

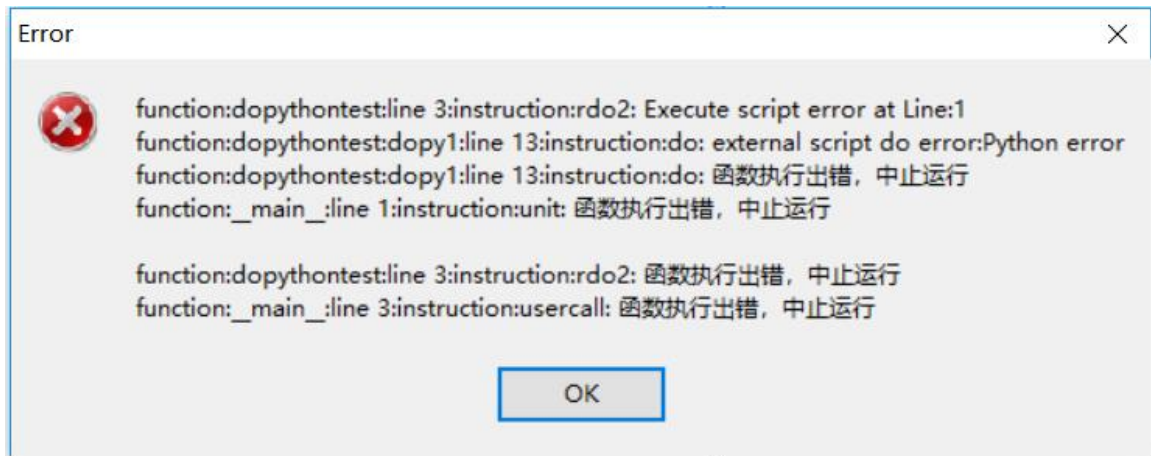
#### 4.3.1 例 1: 基本语法演示

```
Function dopythontest();
Begin
//本地执行 dopy1() 实现 do python 执行 python 脚本
    return rdo2 dopy1();
end

function dopy1()
begin
//do 的方式实现第三方交互
    a:=10;
    b:=100;
    c:=2;
    d:=35;
```

```
do python in a,in b name "E",out c,var d
"c=a+E
d=d*2";
return array(c,d);
End;
//返回[110,70]
```

**注意：** 引号中为python 语法，须靠编辑器最左，有空格的话，会报python 错误



#### 4.3.2 例 2: 导入模块计算

此示例展示 tsl 通过 do python 调用 python 执行数学函数，须在 python 中事先安装好需要的模块，例如 Scipy, math, numpy 等。**未安装也会报例 1 中的错误。**

```
Function KMV_DOPY(EV,St,sigma_St,F,r);
Begin
//KMV 波动率约束法
    EV:=1672130729;    //资产价值
    St:=172330000;    //股票市值
    sigma_St:=0.6179; //波动率
    F:=1499800729;    //债务
    r:=0.0425;        //无风险利率
    return rdo2 py_sub(EV,St,sigma_St,F,r);
end;

function py_sub(EV,St,sigma_St,F,r);
begin
mtic;
do python in EV name 'EV',in St name 'St',
    in sigma_St name 'sigma_St',
    in F name 'F',
    in r name 'r', out ret
%%
from __future__ import division
```

```

from scipy.optimize import fsolve
from scipy.stats import norm
from math import pow, e, log
def KMV(EV,St,sigma_St,F,r):
    def f(x):
        #定义 fsolve 函数所需要的参数 f

        Vt = float(x[0])
        sigma = float(x[1])
        dt = float(x[2])

        def N(y):
            # 定义标准正态分布的累积函数
            return norm.cdf(y)

        return [Vt/St*N(dt) - pow(e, -r)*(F/St)*N(dt - sigma) - 1,
                sigma*(Vt/St)*N(dt) - sigma_St,
                (log(Vt/F) + r + pow(sigma, 2)/2)- sigma*dt]
    x0 = [EV,0.5,1]
    result = fsolve(f,x0,epsfcn=0.5)
    result=[result[0],result[1],result[2],result[2]-result[1]]#DTD 结果为
dt-sigma
    return result
ret = KMV(EV,St,sigma_St,F,r)
%%;
echo mtoc;
    return ret;
end
//返回[1607259056.45737, 0.0698146136874408, 1.63483060267175, 1.56501598898431]

```

## 5 落地服务器开启 Python 服务

### 5.1 工具

update(X64-yyyymmdd).rar

**注：1) yyyymmdd 指更新包版本日期，开启此服务需要20190624 以后版本的更新包**

**2) 需安装有PYTHON**

### 5.2 升级天软平台

1) 将 Update(x64-yyyymmdd).rar 复制到目标盘根目录下，解压 Update(x64-yyyymmdd).rar 到目标盘根目录下生成一个名字为 update-x64 文件夹。

- 2) 在 tinysoft\backup\rar 中查看是否有备份文件 Tinysoft\_YYYYMMDD\_backup.rar 日期为最新一天的文件, 如果没有就运行 Tinysoft\backup\tinysoft\_backup.cmd 进行备份。
- 3) 用管理员身份运行命令提示符 (cmd), 在命令行下, 执行 net stop tssvc, 停止服务
- 4) 用管理员身份运行命令提示符 (cmd), 在命令行下, 在 \update-x64 目录下执行 update-x64-Level1.cmd。

### 5.3 服务开启配置

**注:** 新建的 Python 文件会在 tinysoft\bin\pythonext\用户名\ 中生成, 如:

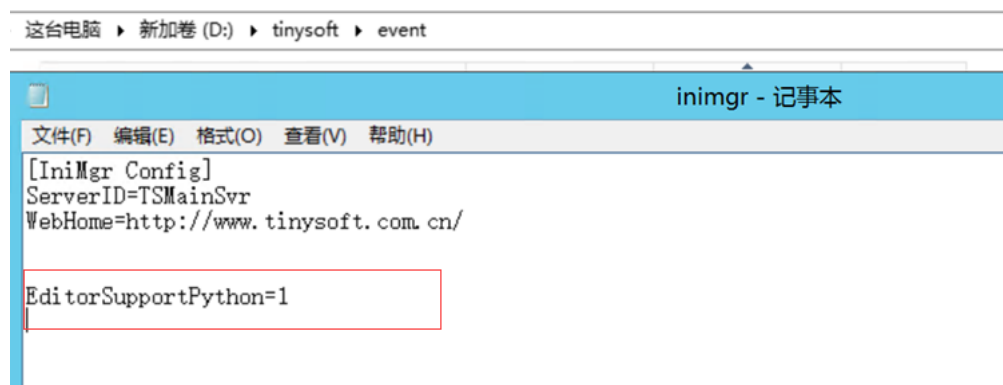
D:\tinysoft\bin\PythonExt\testpy\nonamepy1.py

- 1) 升级天软平台, 详情见 2.2
- 2) 将 python 模块版本文件复制到 tinysoft\bin 目录下

|            |                 |      |
|------------|-----------------|------|
| TSLPy2.dll | 2015/9/11 11:44 | 应用   |
| TSLPy2     | 2019/5/9 19:29  | Pytl |
| TSLPy3.dll | 2015/9/11 11:33 | 应用   |
| TSLPy3     | 2019/6/21 18:17 | Pytl |
| TSLPy34    | 2015/9/17 10:31 | Pytl |
| TSLPy35    | 2019/6/21 18:16 | Pytl |
| TSLPy37    | 2019/6/21 18:16 | Pytl |

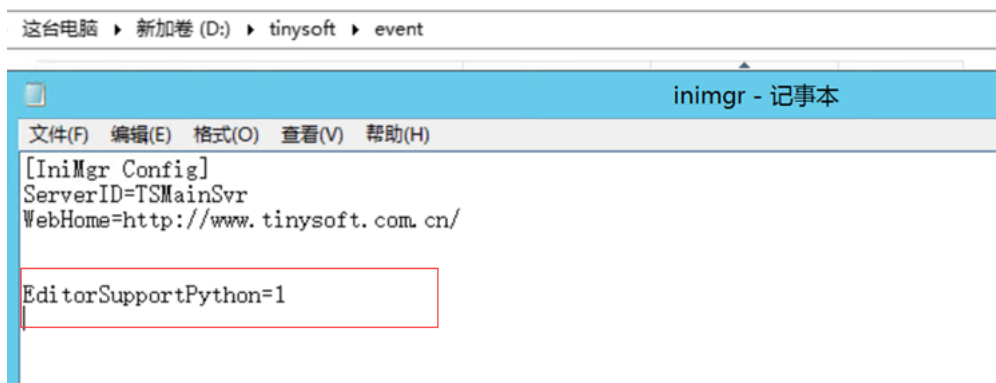
**注:** 根据 python 版本号和子版本号选择复制对应 TSLpy.pyd 文件和 DLL 文件, 如: python3 版本只需复制 TSLPy3.pyd 和 TSLPy3.dll, 如果是 python3.7 需要复制 TSLPy37 和 TSLPy3.dll

- 3) 在 tinysoft\event\inimgr.ini 文件中, 添加如下语句, EditorSupportPython=1, 此语句是开启能将 Python 文件直接拖入天软客户端编辑器中的功能。



- 4) 在 tinysoft\bin\plugin\filemgr.ini 文件中, 添加如下语句:

|                         |                              |
|-------------------------|------------------------------|
| [Python Config]         |                              |
| Enabled=1               | <b>注:</b> 允许打开模块             |
| EnginePermitUser=testpy | <b>注:</b> 授权给某一用户            |
| storeenabled=1          | <b>注:</b> 允许保存 Python 文件到服务器 |
| version=3               | <b>注:</b> Python 版本号         |
| SubVersion=7            | <b>注:</b> Python 子版本号        |



## 5.4 使用范例

### 5.4.1 例 1: 直接在天软客户端新建 Python 模块



注: 同时支持将 Python 文件直接拖入天软客户端并打开,函数名须加 .py

文件所在路径: D:\tinysoft\bin\PythonExt\testpy\nonamepy1.py

在客户端编辑.py 文件, 将自动以 UTF8 的格式存贮, 以吻合目前 python 的默认编码为 UTF8 格式。

#### 5.4.2 例 2: 天软平台通过 pyrun()等函数直接调用 Python

```
Function pytest_01();
Begin
    a:=2;
    pyputvar('a',a);
    pyrun('b=a+1');
    b:=pygetvar('b');
    return b;
End;
```

#### 5.4.3 例 3: 天软平台通过 do Python 语法直接调用 Python

```
Function pytest_02();
Begin
    //do 的方式实现第三方交互
    a:=10;
    b:=100;
    c:=2;
    d:=35;
    do python in a,in b name "E",out c,var d
    "c=a+E
    d=d*3";
    return array(c,d);
end
```

```
Function pytest_03();
Begin
    //约束法计算 KMV 波动率
    //天软取数, Python 计算
    EV:=1672130729;    //资产价值
    St:=172330000;    //股票市值
    sigma_St:=0.6179; //波动率
    F:=1499800729;    //债务
    r:=0.0425;        //无风险利率
    do python in EV name 'EV',in St name 'St',
        in sigma_St name 'sigma_St',
        in F name 'F',
        in r name 'r', out ret
    %%
    from __future__ import division
    from scipy.optimize import fsolve
    from scipy.stats import norm
    from math import pow, e, log
```

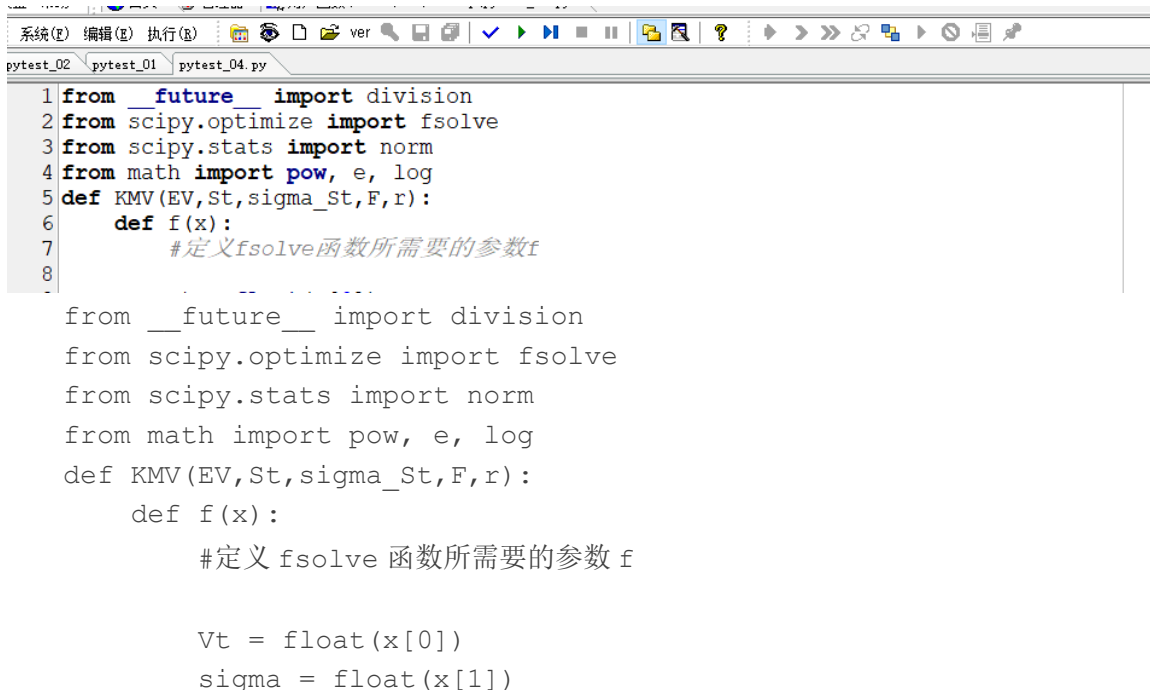
```
def KMV(EV,St,sigma_St,F,r):
    def f(x):
        #定义 fsolve 函数所需要的参数 f

        Vt = float(x[0])
        sigma = float(x[1])
        dt = float(x[2])
        def N(y):
            # 定义标准正态分布的累积函数
            return norm.cdf(y)

        return [Vt/St*N(dt) - pow(e, -r)*(F/St)*N(dt - sigma) - 1,
                sigma*(Vt/St)*N(dt) - sigma_St,
                (log(Vt/F) + r + pow(sigma, 2)/2)- sigma*dt]
    x0 = [EV,0.5,1]
    result = fsolve(f,x0,epsfcn=0.5)
    result=[result[0],result[1],result[2],result[2]-result[1]]#DTD 结果为
dt-sigma
    return result
ret = KMV(EV,St,sigma_St,F,r)
%%;
return ret;
end
```

#### 5.4.4 例 4: 直接调用在天软平台编辑或拖拽进入天软平台的.py 文件

1、先编辑.py 文件或者是直接将已经编辑好的.py 文件拖入客户端, 这里在客户端编写 pytest\_04.py 文件





```

dt = float(x[2])

def N(y):
# 定义标准正态分布的累积函数
    return norm.cdf(y)

    return [Vt/St*N(dt) - pow(e, -r)*(F/St)*N(dt - sigma) - 1,
            sigma*(Vt/St)*N(dt) - sigma_St,
            (log(Vt/F) + r + pow(sigma, 2)/2) - sigma*dt]
x0 = [EV, 0.5, 1]
result = fsolve(f, x0, epsfcn=0.5)
result=[result[0], result[1], result[2], result[2]-result[1]]#DTD 结果为
dt-sigma
    return result
ret = KMV(EV, St, sigma_St, F, r)

```

## 2.调用平台中编辑好的.Py 文件

```

Function pytest_05()
//约束法计算 KMV 波动率
//天软取数, Python 计算, pycall 方式
    EV:=1672130729;    //资产价值
    St:=172330000;     //股票市值
    sigma_St:=0.6179;  //波动率
    F:=1499800729;     //债务
    r:=0.0425;         //无风险利率
return pycall('pytest_04', 'KMV', EV, St, sigma_St, F, r);

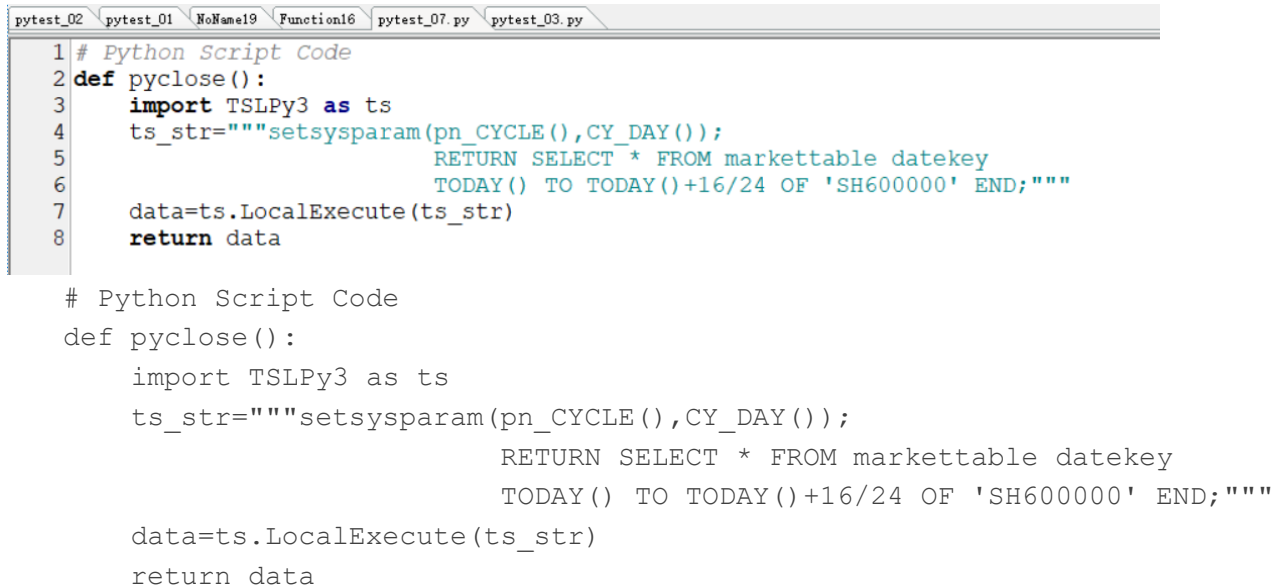
Function pytest_06();
Begin
//约束法计算 KMV 波动率
//天软取数, Python 计算, do python 方式
    EV:=1672130729;    //资产价值
    St:=172330000;     //股票市值
    sigma_St:=0.6179;  //波动率
    F:=1499800729;     //债务
    r:=0.0425;         //无风险利率
do python in EV name 'EV', in St name 'St',
    in sigma_St name 'sigma_St',
    in F name 'F',
    in r name 'r', out ret
%%
import pytest_04
ret=pytest_04.KMV(EV, St, sigma_St, F, r)
%%;

```

```
return ret;
End
```

### 5.4.5 例 5: 在天软平台编辑.py 文件调用天软平台

1. 编辑.py 文件，取 SH600000 今日日线，pytest\_07.py



```
1 # Python Script Code
2 def pyclose():
3     import TSLPy3 as ts
4     ts_str="""setsysparam(pn_CYCLE(),CY_DAY());
5                                     RETURN SELECT * FROM markettable datekey
6                                     TODAY() TO TODAY()+16/24 OF 'SH600000' END;"""
7     data=ts.LocalExecute(ts_str)
8     return data

# Python Script Code
def pyclose():
    import TSLPy3 as ts
    ts_str="""setsysparam(pn_CYCLE(),CY_DAY());
                                     RETURN SELECT * FROM markettable datekey
                                     TODAY() TO TODAY()+16/24 OF 'SH600000' END;"""

    data=ts.LocalExecute(ts_str)
    return data
```

**注：**此处需要使用 LocalExecute()，如果使用 RemoteExecute() 会出错，无返回结果

如果调用天软函数，例如 close()，也需使用 LocalCallFunc().[函数使用方法点此链接](#)

2. 在天软平台调用 pytest\_07.py 文件

```
Function NoName19();
Begin
    pyrun('import sys');
    //须新增服务器上保存账户下新建的.py 文件的路径，否则会报找不到模块的错误
    //账户指以客户账户名命名的文件夹
    pyrun("sys.path.append('D:/tinysoft/bin/PythonExt/账户')");
    b:=pycall('pytest_07','pyclose');
    return b;
end
```

返回：返回的四个参数中，第二个为返回结果

| 数组数据 |            |
|------|------------|
| 0    | 0          |
| 1    | <ARRAY[1]> |
| 2    |            |
| 3    | 0          |

→ 返回结果

| 数组数据 |                |                 |               |                  |                       |                  |        |                |       |       |       |           |       |           |               |
|------|----------------|-----------------|---------------|------------------|-----------------------|------------------|--------|----------------|-------|-------|-------|-----------|-------|-----------|---------------|
|      | sectional_open | sectional_close | buy_vol       | sectional_yclose | sectional_sale_amount | sectional_amount | yclose | amount         | sale5 | sale2 | sale3 | StockName | sale1 | date      | sectional_buy |
| 0    | 11.22          | 11.38           | 20,944,107.50 | 11.22            | 194,617,120.50        | 430,744,321.00   | 11.22  | 430,744,321.00 | 11.42 | 11.39 | 11.40 | 浦发银行      | 11.38 | 43,696.00 | 236,127,      |

## 6 附录

### 6.1 数据类型转换

基础类型自动转换规则

| TSL   | Python |
|-------|--------|
| NIL   | None   |
| NAN   | nan    |
| INF   | inf    |
| array | List   |
| array | tuple  |
| array | dict   |

由于其他自定义类型的复杂性,非基础类型请用户自行转换.例如 python 中的时间类型送入到 tsl 中,在天软 TSLPy2\TSLPy3 模块中提供的相应的转换函数.

### 6.2 系统参数简介

| 系统参数名称      | 含义     | 例子                           |
|-------------|--------|------------------------------|
| StockID     | 当前股票代码 | "SH000001"                   |
| CurrentDate | 当前日期   | TSLPy2.EncodeDate(2015,9,25) |
| Cycle       | 周期     | "日线"                         |
| bRate       | 复权方式   | 0                            |
| RateDay     | 复权基准日  | TSLPy2.EncodeDate(2015,9,25) |
| Precision   | 显示精度   | 4                            |

相关的系统参数可以在天软首页找到

<http://www.tinysoft.com.cn/tsdn/helpdoc/display.tsl?id=1863>

### 6.3 数据转换为字符串(克服部分中文在 python 中的显示问题)

首先,python 返回的不是乱码!举个栗子:

```

Type "help", "copyright", "credits" or "license" for more information.
>>> a = "你好";
>>> b= "hello"
File "<stdin>", line 1
    b= "hello"
IndentationError: unexpected indent
>>> b= "hello"
>>> a
'\xc4\xe3\xba\xc3'
>>> b
'hello'
>>> data={"你好":1,b"hello":2}
>>> data[b"你好"]
1
>>>

```

参考: <http://www.cnblogs.com/jxzheng/p/5186490.html>

### 6.3.1 结构不变

**注:** 此函数只在 Python3 版本中有效, Python2 中无效

不破坏数组结构, 将数组中的中文解码显示出来

```

def tsbytestostr(data):
    if (isinstance(data, (tuple)) or isinstance(data, (list))):
        lendata = len(data)
        ret = []
        for i in range(lendata):
            ret.append(tsbytestostr(data[i]))
    elif isinstance(data, (dict)):
        lendata = len(data)
        ret = {}
        for i in data:
            ret[tsbytestostr(i)] = tsbytestostr(data[i])
    elif isinstance(data, (bytes)):
        ret = data.decode('gbk')
    else:
        ret = data
    return ret

```

### 6.3.2 转换为字符串

**注:** 此函数只在 Python3 的特定版本中有效, 因为 Python3 的某些版本中需要将数组转换成字符串才能进行解码, 此函数本身无解码功能, Python2 中无效

```

def tostry(data):
    ret = ""

```

```

if isinstance(data, (int, float)):
    ret = "{0}".format(data)
elif isinstance(data, (str)):
    ret = "\"{0}\"".format(data)
elif isinstance(data, (list)):
    lendata = len(data)
    ret += "["
    for i in range(lendata):
        ret += tostry(data[i])
        if i < (lendata-1):
            ret += ","
    ret += "]"
elif isinstance(data, (tuple)):
    lendata = len(data)
    ret += "("
    for i in range(lendata):
        ret += tostry(data[i])
        if i < (lendata-1):
            ret += ","
    ret += ")"
elif isinstance(data, (dict)):
    it=0
    lendata = len(data)
    ret += "{"
    for i in data:
        ret+= tostry(i)+":"+tostry(data[i])
        it+=1
        if it<lendata :
            ret += ","
    ret += "}"
else:
    ret = "{0}".format(data)
return ret;

```

## 6.4 本地调用天软(通常不需要使用)

- LocalExecute(ExecString:String):tuple(ErrNo,Result,ErrMsg,ErrLineNo)  
功能:本地执行天软语句
- LocalCallFunc(Func:String;Args:list|tuple):  
功能:本地执行天软函数

```

import sys
sys.path.append(天软安装目录)

```

```
import TSLPy2  #导入模块
data = TSLPy2.LocalExecute("return rand(5,5);") #执行语句
data = TSLPy2.LocalCallFunc("rand",[5,5]); #执行函数,rand 需要两个实数做参数
可用python中的list 或者 tuple 送入
```

## 6.5 重置加载包，清除变量

在天软与 python 交互的时候，客户端如果不关闭或重启，则 python 的环境不会重启，则此时由于 Python 的缓存机制，上次函数执行的环境并未清除，导致上次与 python 交互加载的包还在，传入 python 的变量也未被清除，则可能导致本次天软与 python 的交互出现错误，或者出现错误的结果，则此时，需要在执行一次天软与 python 交互的时候清除变量，重置加载包。

### 6.5.1 实现代码

```
Unit tspy_unit;
Interface
    function QY_py_reloadMK(modelname);      //重载模块
    function QY_py_cleardiyVar();            //清除自定义变量
    function QY_py_trandate(endt,varname);    //转变天软日期为pyrhon 日期
    function QY_py_getsyspath();             //获得 sys.path
    function QY_py_reset();                  //重置 python 环境
    function QY_py_sns_load_dataset(dsetname);
Implementation
//1._____
//____重载模块_____
    function QY_py_reloadMK(modelname);
    begin
        pyrun("import imp");
        pyrun(format("imp.reload(%s)",modelname));
    end
//2._____
//____删除模块_____
    function QY_py_delMK(modelname);
    begin
        pyrun(format("del %s",modelname));
    end
//3._____
//____清除自定义变量_____
    function QY_py_cleardiyVar();
    begin
        pyrun('[globals().pop(key) if not key.startswith("__") else 1 for
key in list(globals().keys())]');
```

```

    end
//4. _____
//_____ 将天软日期转为 python 日期_____
    function QY_py_trandate(endt,varname);
    begin
        //endt 日期; varname python 中变量名
        pyrun("import datetime");
        pyrun(format("%s = datetime.date(%d,%d,%d)",
                    varname,YearOf(endt)
                    ,MonthOf(endt),dayof(endt)
                    ));
    end;
//5. _____
//_____ 获得 sys.path_____
    function QY_py_getsyspath();
    begin
        pyrun("syspath = sys.path");
        return pygetvar("syspath");
    end
//6. _____
//_____ 重载所有模块_____
    function QY_py_reloadallMK();
    begin
        pyrun("import imp");
        pyrun("import types");
        pyrun('[imp.reload(globals()[key]) if
isinstance(globals()[key],types.ModuleType) else 0 for key in
list(globals().keys())]');
    end
//7. _____
//_____ 重置 python 环境_____
    function QY_py_reset();
    begin
        QY_py_reloadallMK();//重载所有模块
        QY_py_cleardiyVar();//清理所有自变量
    end
//8. _____
//_____ 导入 seaborn 数据集_____
    function QY_py_sns_load_dataset(dsetname);
    begin
        pyrun("__temp__ = None");
        pyrun("import seaborn as sns");
        pyrun(format("__temp__
sns.load_dataset('%s').to_dict('record')",dsetname));
    end

```

```

        return pygetvar("__temp__");
    end;
Initialization
Finalization
    //这里调用 QY_py_reset, 行不通 2018-08-16, 默认这里的调用是本地执行
End.

```

## 6.5.2 示例

### 6.5.2.1 方式一: pyrun 调用

```

//导入系统模块, 获取系统路径
rdo2 pyrun("import sys");
rdo2 pyrun("a=sys.path");
s1:=rdo2 pygetvar("a");
//执行清除变量, 重载模块前能获取到系统路径
Rdo2 unit(tspy_unit).QY_py_reset();
//执行之后则不会获取到系统路径
rdo2 pyrun("b=sys.path");
s2:=rdo2 pygetvar("b");
s3:=rdo2 pygetvar("a");
return array(s1,s2,s3);

```

### 6.5.2.2 方式二: do python 调用

```

Function dopy();
Begin
//导入系统模块, 获取系统路径
    rdo2 py_test();
    s1:=rdo2 py_test1();
    Rdo2 unit(tspy_unit).QY_py_reset();
// s2:=rdo2 py_test1();
//此处如果再次执行 rdo2 py_test1();则会 python 报错,
//原因是 sys 模块被清除, python 会报 sys 未定义的错误
    return s1;
end;

function py_test();
begin
//导入系统模块
do python
%%
import sys%%;
end

```



```
function py_test1();  
begin  
//获取系统路径  
do python out a  
%%  
a=sys.path%%;  
return a;  
end
```