python api 文档

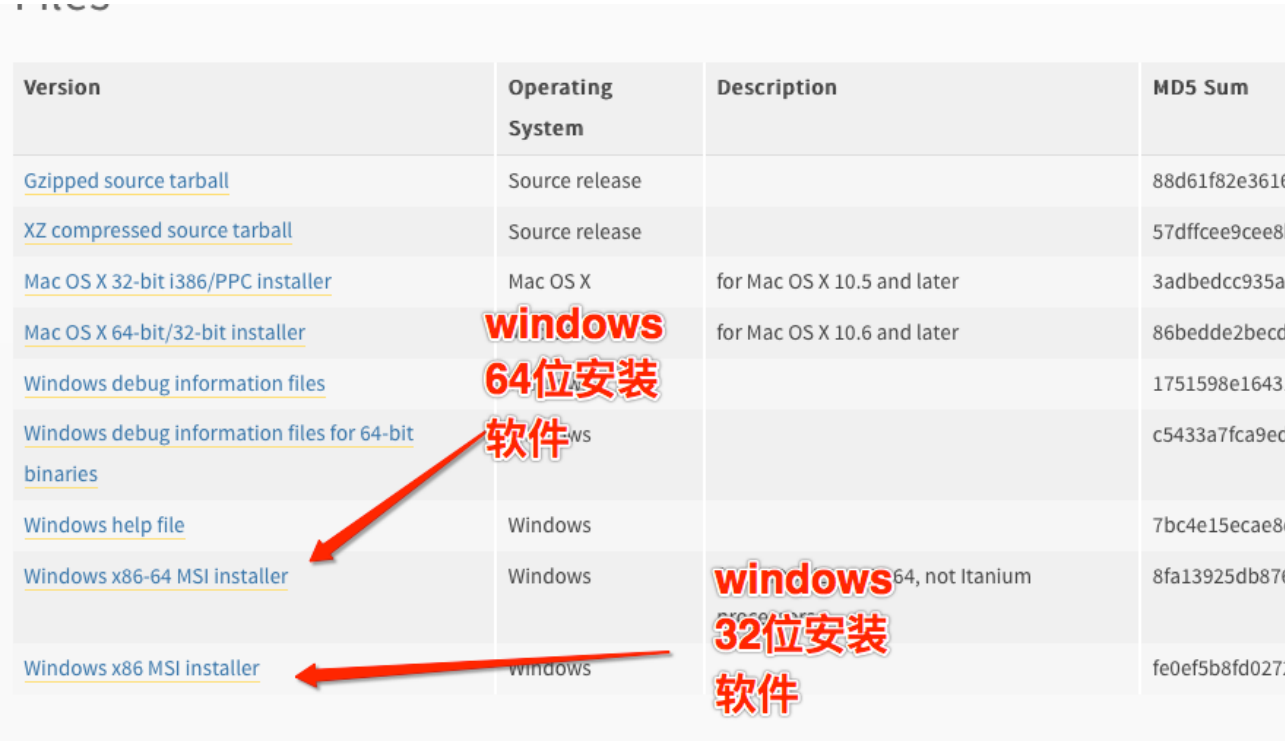
快速开始

安装Python

(下载地址:https://www.python.org/downloads/release/python-2712/)。

Windows

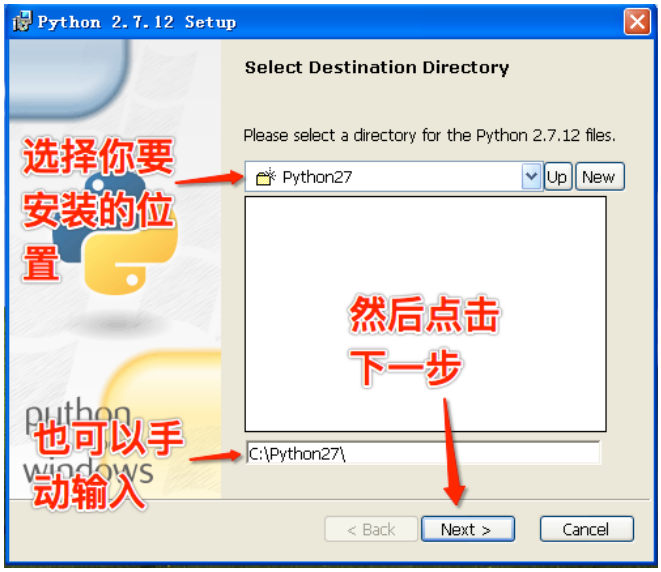
1.使用浏览器登录https://www.python.org/downloads/release/python-2712/
前往下载Windows对应安装软件。

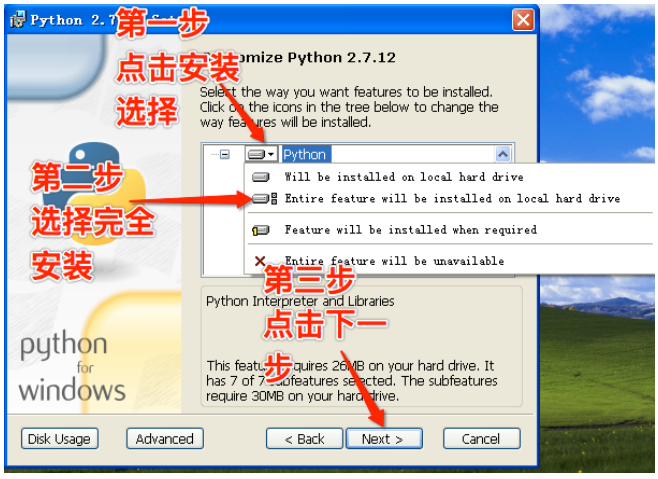


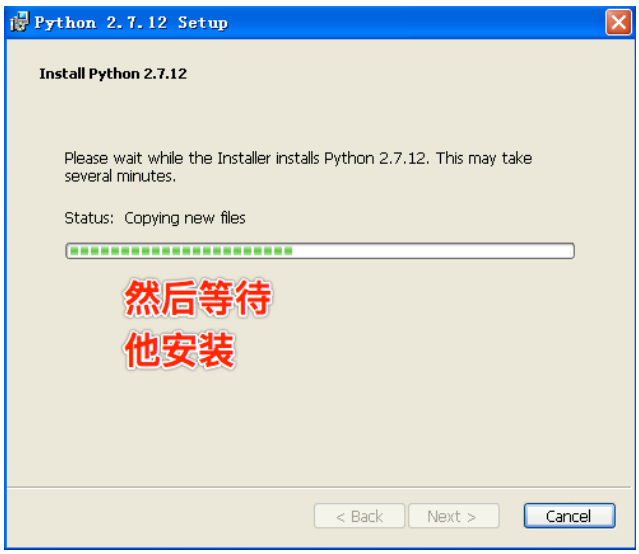
2.点击打开下载好的安装包

3进行如下步骤：

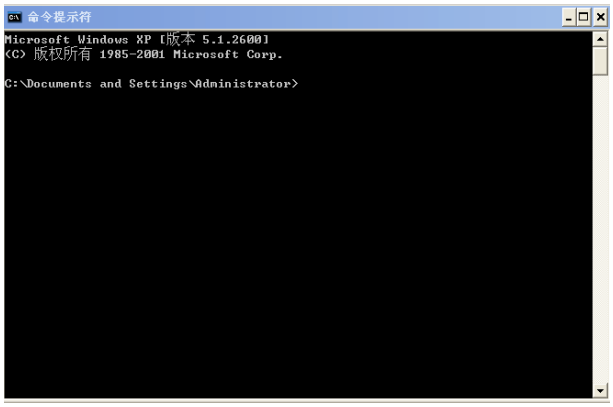

Python 2.7.12 Setup
Select whether to install Python 2.7.12 for all users of this computer.
Install for all users
Install just for me (not available on Windows Vista)
点击
python for windows
Back
Next >
Cancel


Python 2.7.12 Setup
Select Destination Directory
Please select a directory for the Python 2.7.12 files.
选择你要安装的位置
Python27
Up
New
然后点击下一步
也可以手动输入
C:\Python27\
< Back
Next >
Cancel


第一步
点击安装选择
第二步
选择完全安装
第三步
点击下一步
Python 2.7
omize Python 2.7.12
Select the way you want features to be installed.
Click on the icons in the tree below to change the way features will be installed.
Python
Will be installed on local hard drive
Entire feature will be installed on local hard drive
Feature will be installed when required
Entire feature will be unavailable
Python Interpreter and Libraries
python for windows
Disk Usage
Advanced
< Back
Next >
Cancel

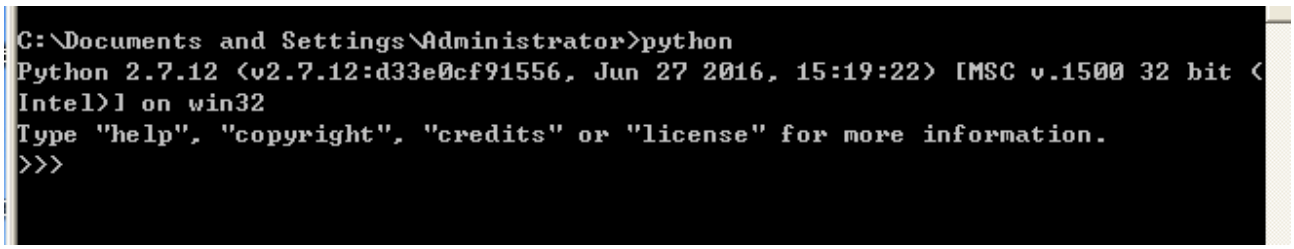


4.验证安装是否成功



使用“Win+R”快捷键召唤出运行窗口，再在运行中输入cmd
出现黑色框体

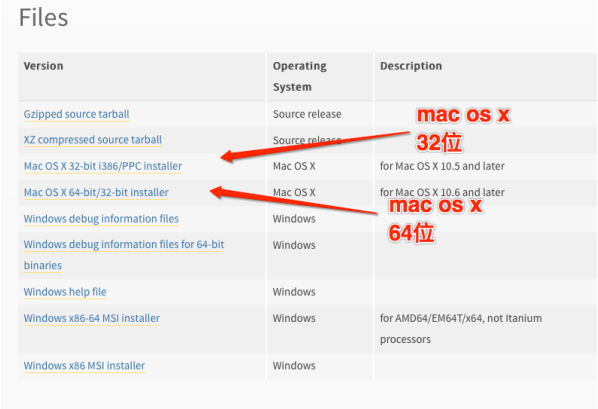
在黑色框体中输入python
出现以下情况安装成功



---

mac

1.使用浏览器登录https://www.python.org/downloads/release/python-2712/
前往下载mac os x 对应安装软件。



2.点击下载后软件

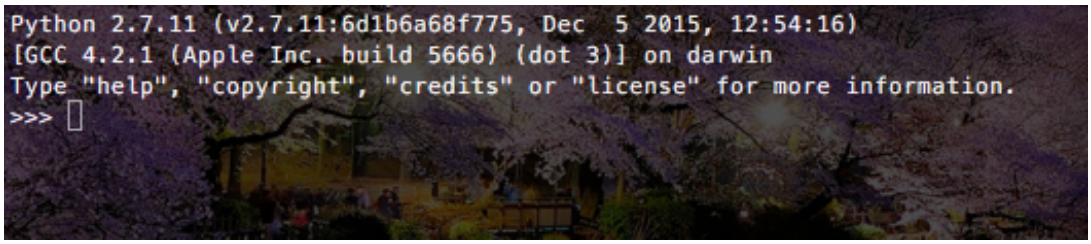

安装“Python”
标准安装将执行于“Macintos”上
介绍
请先阅读
许可
目的宗卷
安装类型
安装
摘要
这将占用您的电脑上的 85.6 MB 空间。
点按“安装”以在磁盘“Macintos”上执行此软件的标准安装。
自定
返回
安装


安装“Python”
重要信息
介绍
请先阅读
许可
目的宗卷
安装类型
安装
摘要
This package will install Python 2.7.12 for Mac OS X 10.6 or later for the following architecture(s): i386, x86_64.
Which installer variant should I use?
Python.org provides two installer variants for download: one that installs a 64-bit/32-bit Intel Python capable of running on Mac OS X 10.6 (Snow Leopard) or later; and one that installs a 32-bit-only (Intel and PPC) Python capable of running on Mac OS X 10.5 (Leopard) or later. This ReadMe was installed with the 10.6 or later variant. Unless you are installing to an 10.5 system or you need to build applications that can run on 10.5 systems, use the 10.6 variant if possible. There are some additional operating system
pported starting with 10.6 and you may see bett
64-bit mode. By default, Python will automatically run in 64-bit mode if your system supports it. Also see Certificate verification and OpenSSL below. The Pythons installed by these installers are built with private copies of some third-party libraries not included with or newer than those in OS X itself. The list of these libraries varies by installer variant and is included at the end of the License.rtf file.
Update your version of Tcl/Tk to use IDLE or other Tk
plications
点击继续
打印...
存储...
返回
继续


安装“Python”
软件许可协议
介绍
请先阅读
许可
目的宗卷
安装类型
安装
摘要
HISTORY AND LICENSE
HISTORY OF THE SOFTWARE
Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.
In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.
In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.
All Python releases are Open Source (see http://www.opensource.org for
打印...
存储...
返回
继续

最后完成安装

3.打开应用程序文件夹,实用工具，终端
输入Python



出现上图所示表示安装成功。

下载sdk

目前提供的sdk支持Windows、mac、linux下Python2.7(下载地址:https://github.com/uxmc/sdk/archive/master.zip)。

安装配置

1. 解压已经下载后的sdk压缩文件，放置到任意位置。
2. 使用快捷键win+R 打开运行窗口 输入cmd 打开dos黑框(windows)

   打开应用程序文件夹,实用工具，终端(mac)
3. 切换到sdk文件夹下py2文件夹中
4. 输入python setup.py install 安装程序。
5. 验证安装，输入Python 进入 python command line 输入import sd 没有报错 则为安装成功

我的第一个策略

首先需前往网站注册账号。

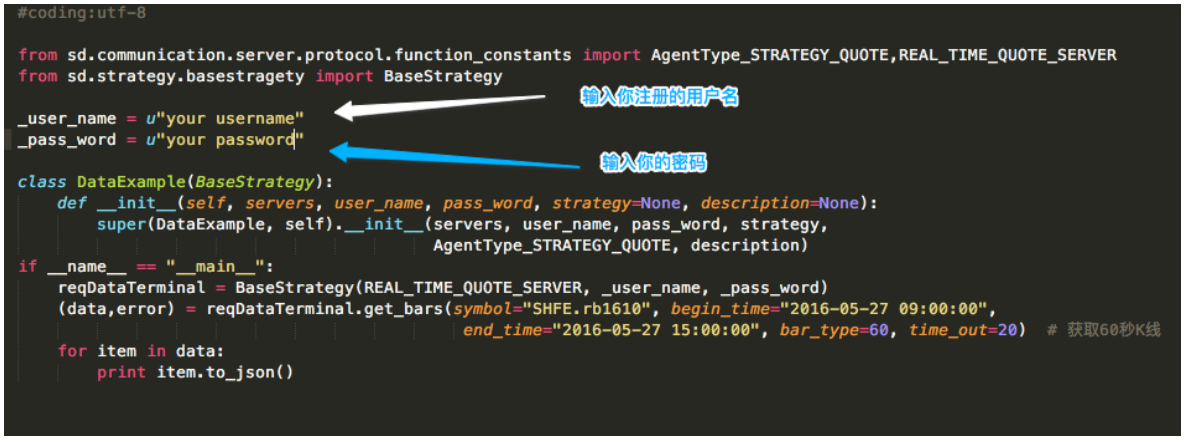
然后进行以下步骤

方式1　直接使用例子运行

使用sdk包中 py2/sd/example中的项目例子。

方式2　创建自己的策略

以下是一个完整的策略代码示例



编译策略并运行

以上代码是获取SHFE.RB1610 2016年5月27日从上午九点到下午3点 60秒K线

参数
symbol 证券代码　begin_time 起始时间　　end_time 结束时间
bar_type bar周期 即 没多少秒的bar　　time_out 错误时退出时间

Python API 范例

数据接口范例

历史数据提取

1. 获取半小时K线

示例：提取上交所浦发银行2016年5月27日上午九点到下午三点半小时K线。

```
from sd.strategy.basestragety import BaseStrategy
from sd.communication.server.protocol.function_constants import REAL_TIME_QUOTE_SERVER

_user_name = u"your username"
_pass_word = u"your password"

reqDataTerminal = BaseStrategy(REAL_TIME_QUOTE_SERVER, _user_name, _pass_word)

(data,error)  = reqDataTerminal.get_bars(symbol="SHSE.600000", begin_time="2016-05-27 09:00:00",
                                         end_time="2016-05-27 15:00:00", bar_type=60 * 30, time_out=20)
for item in data:
    print item.to_json()
```

参数
symbol 证券代码 begin_time 起始时间 end_time 结束时间
bar_type bar周期 即 没多少秒的bar time_out 错误时退出时间

2. 获取60秒K线

示例：提取上交所浦发银行2016年5月27日上午九点到下午三点60秒K线。

```
from sd.strategy.basestragety import BaseStrategy
from sd.communication.server.protocol.function_constants import REAL_TIME_QUOTE_SERVER

_user_name = u"your username"
_pass_word = u"your password"

reqDataTerminal = BaseStrategy(REAL_TIME_QUOTE_SERVER, _user_name, _pass_word)

(data,error)  = reqDataTerminal.get_bars(symbol="SHSE.600000", begin_time="2016-05-27 09:00:00",
                                         end_time="2016-05-27 15:00:00", bar_type=60, time_out=20)
for item in data:
    print item.to_json()
```

参数
symbol 证券代码 begin_time 起始时间 end_time 结束时间
bar_type bar周期 即 每多少秒的bar time_out 错误时退出时间

3. 获取最新的10根半小时K线

示例：提取上交所张江高科最新的10根半小时K线。

```
from sd.strategy.basestragety import BaseStrategy
from sd.communication.server.protocol.function_constants import REAL_TIME_QUOTE_SERVER

_user_name = u"your username"
_pass_word = u"your password"

reqDataTerminal = BaseStrategy(REAL_TIME_QUOTE_SERVER, _user_name, _pass_word)

(data,error) = reqDataTerminal.get_last_n_bars(symbol="SHSE.600895", n=10, bar_type=60 * 30,
                                               time_out=20)
for item in data:
    print item.to_json()
```

参数
symbol 证券代码 n 提取数据条数
bar_type bar周期 即 每多少秒的bar time_out 错误时退出时间

4. 获取指定时间范围的tick数据。

示例：提取上交所浦发银行2016年5月27日上午九点到下午三点tick数据。

```
from sd.strategy.basestragety import BaseStrategy
from sd.communication.server.protocol.function_constants import REAL_TIME_QUOTE_SERVER

_user_name = u"your username"
_pass_word = u"your password"

reqDataTerminal = BaseStrategy(REAL_TIME_QUOTE_SERVER, _user_name, _pass_word)

(data,error) = reqDataTerminal.get_ticks(symbol="SHSE.600000", begin_time="2016-05-27 09:00:00",
                                         end_time="2016-05-27 15:00:00", time_out=20)
for item in data:
    print item.to_json()
```

参数
symbol 证券代码 begin_time 起始时间 end_time 结束时间
time_out 错误时退出时间

行情订阅

订阅浦发银行tick值。
ret = dataExample.req_subscribe(TICK, “SHSE.600000”)

策略接口范例

获取用户持仓

```
from sd.communication.server.protocol.function_constants import *
from sd.configuration.logconfig import configure_log
from sd.strategy.simplestragety import SimpleStrategy

_user_name = u"your username"
_pass_word = u"your password"

class DemoStrategy(SimpleStrategy):
    def __init__(self, trade_servers, quote_servers, user_name, pass_word, strategy=None, description=None):
        super(DemoStrategy, self).__init__(trade_servers, quote_servers, user_name, pass_word,
                                           strategy=strategy,
                                           description=description)

simple_strategy = DemoStrategy(TRADE_SERVER, REAL_TIME_QUOTE_SERVER, _user_name, _pass_word,
                               description="my demo strategy")
err = simple_strategy.init()
if err.errorId != 0:
    print err
    exit(1)
(result, error) = simple_strategy.req_position()
for item in result:
    print item.to_json()
```

req_position() 获取用户持仓

查询用户的委托的订单列表

```
from sd.communication.server.protocol.function_constants import *
from sd.configuration.logconfig import configure_log
from sd.strategy.simplestragety import SimpleStrategy

_user_name = u"your username"
_pass_word = u"your password"

class DemoStrategy(SimpleStrategy):
    def __init__(self, trade_servers, quote_servers, user_name, pass_word, strategy=None, description=None):
        super(DemoStrategy, self).__init__(trade_servers, quote_servers, user_name, pass_word,
                                           strategy=strategy,
                                           description=description)

simple_strategy = DemoStrategy(TRADE_SERVER, REAL_TIME_QUOTE_SERVER, _user_name, _pass_word,
                               description="my demo strategy")
err = simple_strategy.init()
if err.errorId != 0:
    print err
    exit(1)
(result, error) = simple_strategy.req_order_list()

for item in result:
    print item.to_json()
```

req_order_list()查询用户委托订单列表

策略卖出

```
from sd.communication.server.protocol.function_constants import *
from sd.configuration.logconfig import configure_log
from sd.strategy.simplestragety import SimpleStrategy


_user_name = u"your username"
_pass_word = u"your password"


class DemoStrategy(SimpleStrategy):
    def __init__(self, trade_servers, quote_servers, user_name, pass_word, strategy=None, description=None):
        super(DemoStrategy, self).__init__(trade_servers, quote_servers, user_name, pass_word,
                                           strategy=strategy,
                                           description=description)


simple_strategy = DemoStrategy(TRADE_SERVER, REAL_TIME_QUOTE_SERVER, _user_name, _pass_word, _strategy_id,
                               description="my demo strategy")
err = simple_strategy.init()
if err.errorId != 0:
    logging.error("Failed to login,exit:%s", err.to_json())
    exit(1)
(order_id, error) = simple_strategy.send_order("SHFE.rb1701", bs_flag=BSFlag_SELL,
                                               position_effect=PositionEffect_OPEN,
                                               price_type=PRTP_FIX,
                                               price=2089, volume=4)
```

send_order(证券代码,bs_flag = 下单方向,position_offect = 对持仓的影响,price_type = 价格的类型, price = 委托价, volume = 委托量)

策略下单

```
from sd.communication.server.protocol.function_constants import *
from sd.configuration.logconfig import configure_log
from sd.strategy.simplestragety import SimpleStrategy

_user_name = u"your username"
_pass_word = u"your password"

class DemoStrategy(SimpleStrategy):
    def __init__(self, trade_servers, quote_servers, user_name, pass_word, strategy=None, description=None):
        super(DemoStrategy, self).__init__(trade_servers, quote_servers, user_name, pass_word,
                                           strategy=strategy,
                                           description=description)

simple_strategy = DemoStrategy(TRADE_SERVER, REAL_TIME_QUOTE_SERVER, _user_name, _pass_word,
                               description="my demo strategy")
err = simple_strategy.init()
if err.errorId != 0:
    print err
    exit(1)
(order_id, error) = simple_strategy.send_order("SHFE.rb1701", bs_flag=BSFlag_BUY,
                                               position_effect=PositionEffect_OPEN,
                                               price_type=PRTP_FIX,
                                               price=2089, volume=4)
```

send_order(证券代码,bs_flag = 下单方向,position_offect = 对持仓的影响,price_type = 价格的类型, price = 委托价, volume = 委托量)

综合范例

第一部分 定义策略类

```python
import logging #引用python日志系统
from sd.communication.server.protocol.function_constants import * #引用所有的变量
from sd.configuration.logconfig import configure_log #引用提供的日志
from sd.strategy.simplestragety import SimpleStrategy #引用提供的策略

_user_name = u"your-user-name"#定义你的账号
_pass_word = u"your-pass-word"#定义你的密码
_strategy_id = "your-strategy-name"#定义你的策略名

class DemoStrategy(SimpleStrategy):#定义你的策略,继承自引用策略

    '''定义成员变量
    分别含义
    trade_servers 交易服务端           quote_servers 数据服务端       user_name 账号
    pass_word 密码       strategy 策略      description 描述
    '''
    def __init__(self, trade_servers, quote_servers, user_name, pass_word, strategy=None, description=None):
        super(DemoStrategy, self).__init__(trade_servers, quote_servers, user_name, pass_word,
                                           strategy=strategy,description=description)
    '''
    下单通知的订单交易最新情况，接收后更新状态，字段参考class StrategyOrderDetail，  判断订单执行情况
    '''
    def on_notify_quote(self, data_type, data):
        logging.info("%s", data.to_json())

    def on_notify_order_detail(self, order_detail):
        logging.info("[on_notify_order_detail]:%s", order_detail.to_json())

    def on_notify_connection_error(self, reason=""):
        logging.warn("on disconnection:" + reason)

    def on_command_strategy_exit(self, reason=None):
        logging.warn("on command exit")
```

1 **引用所需*python*自带包 、*sdk*提供变量 、函数、 策略类**

2 **定义所需变量 ， 如账号、密码、策略名**

3 **定义你的策略类 、策略方法 、日志输出类型**

第二部分 方法调用

```python
if __name__ == "__main__":
    configure_log("DemoStrategy.log", logging.DEBUG)#日志打印地址
    simple_strategy = DemoStrategy(TRADE_SERVER, REAL_TIME_QUOTE_SERVER, _user_name, \
    _pass_word, _strategy_id,description="my demo strategy")#实例化你的策略

    '''验证是否连接成功'''
    err = simple_strategy.init()
    if err.errorId != 0:
        logging.error("Failed to login,exit:%s", err.to_json())
        exit(1)

    ''' 查询用户的持仓详情  '''
    (result, error) = simple_strategy.req_position()
    if error.errorId != 0:
        logging.error(error.to_json())
    else:
        for item in result:
            logging.info(item.to_json())

    ''' 获取半小时K线'''
    (data, error) = simple_strategy.get_bars(symbol="SHSE.600895", begin_time="2016-05-27 09:00:00",\
    end_time="2016-05-27 15:00:00", bar_type=60 * 30, time_out=20)
    if data is not None:
        for item in data:
            print item.to_dict()

    '''订阅数据'''
    error = simple_strategy.req_subscribe(TICK, "SHFE.rb1610")
    if error.errorId != 0:
        logging.error(error.to_json())

    '''买入下单'''
    (order_id, error) = simple_strategy.send_order("SHFE.rb1701", bs_flag=BSFlag_BUY,
                                                   position_effect=PositionEffect_OPEN,
                                                   price_type=PRTP_FIX,
                                                   price=2089, volume=4)
    simple_strategy.run() #策略运行
```

1 **设置打印日志**
2 **验证连接结果**
3 **数据获取订阅**
4 **订单买入卖出**
5 **策略运行**

Python api接口

常量定义

连接类型常量值

(import sd.communication.server.protocol.function_constants)

AgentType_STRATEGY_TRADE	交易策略
AgentType_STRATEGY_QUOTE	数据策略
REAL_TIME_QUOTE_SERVER	实时数据服务连接
HISTORY_QUOTE_SERVER	历史数据服务连接
TRADE_SERVER	交易服务连接

期货对冲常量值

(import sd.communication.server.protocol.function_constants)

HEDGE_FLAG_SPECULATION
HEDGE_FLAG_ARBITRAGE
HEDGE_FLAG_HEDGE

下单方向

(import sd.communication.server.protocol.function_constants)

BSFlag_BUY	卖出
BSFlag_SELL	买入

订单对持仓的影响

(import sd.communication.server.protocol.function_constants)

PositionEffect_OPEN
PositionEffect_CLOSE
PositionEffect_CLOSE_TODAY
PositionEffect_CLOSE_YESTERDAY

交易所

(import sd.communication.server.protocol.function_constants)

EXCHANGEID_SHFE	上期所
EXCHANGEID_DCE	大商所
EXCHANGEID_CFFEX	中金所
EXCHANGEID_CZCE	郑商所
EXCHANGEID_SHSE	上海股票交易所

EXCHANGEID_SZSE	深圳股票交易所

数据类型

(import sd.communication.server.protocol.function_constants)

TICK

BAR

DAILY_BAR

订单状态

(import sd.communication.server.protocol.function_constants)

ORDER_STATUS_CREATED	创建订单
ORDER_STATUS_UNREPORTED	订单待确认
ORDER_STATUS_REPORTED	订单已确认
ORDER_STATUS_NOT_TRADE	未成交
ORDER_STATUS_PART_SUCC	部分成交
ORDER_STATUS_SUCCEEDED	交易成交
ORDER_STATUS_UNREPORTED_CANCEL	取消订单待确认
ORDER_STATUS_REPORTED_CANCEL	取消订单已确认
ORDER_STATUS_INVALID	废单
ORDER_STATUS_CANCELED	取消成功

价格类型常量值

(import sd.communication.server.protocol.function_constants)

PRTP_FIX	限价单
PRTP_MARKET	市价单
PRTP_OPPOSIT	对手价格

策略初始化

基础策略类

(from sd.strategy.simplestragety import BaseStrategy)

BaseStrategy(servers, user_name, pass_word, strategy=None, description=None)

参数	类型	说明
servers	常量值	需要连接的服务端(引用常量)
user_name	string	账号
pass_word	string	密码
strategy	string	策略名称
description	string	策略描述

日志函数

(from sd.configuration.logconfig import configure_log)

configure_log(filename, level)

参数	类型	说明
filename	string	日志名
level	logging.{value}	日志级别

logging 模块日志级别

CRITICAL、**ERROR**、 **WARNING**、**INFO**、**DEBUG**、 **NOTSET**

简单策略类

(from sd.strategy.simplestragety import SimpleStrategy)

BaseStrategy(servers, user_name, pass_word, strategy, usage_type, description, mode=1, start_time, end_time,base_money, symbol commission_ratio,adjust_price_type ,history_data_server)

参数	类型	说明
servers	string	需要连接的服务端(一般引用常量)
user_name	string	账号
pass_word	string	密码
strategy	string	策略名称
description	string	策略描述
mode	int	策略模式
start_time	string	起始时间,如2016-05-27 09:00:00
end_time	string	结束时间,如2016-05-27 09:00:00
base_money	int	基础资金
symbol	string	证券代码
commission_ratio	float	佣金比例
adjust_price_type	int	调价类型
history_data_server	常量值	历史数据服务端

策略开始方法

run()

策略验证方法

init()

数据提取方法

历史bar数据方法

get_bars(symbol, begin_time, end_time,bar_type,time_out)

参数	类型	说明
symbol	string	证券代码
begin_time	string	起始时间,如2016-05-27 09:00:00
end_time	string	结束时间,如2016-05-27 09:00:00
bar_type	int	bar周期，以秒为单位，比如60即1分钟bar
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

最新bar数据获取方法

get_last_n_bars(symbol, bar_type, n, time_out)

参数	类型	说明
symbol	string	证券代码
n	int	提取的数据条数
bar_type	int	bar周期，以秒为单位，比如60即1分钟bar
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

最新一天bar数据获取方法

get_last_n_daily_bars(symbol, n, time_out)

参数	类型	说明
symbol	string	证券代码
n	int	提取的数据条数
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

指定一天bar数据获取方法

get_daily_bars(symbol, begin_time, end_time, time_out)

参数	类型	说明
symbol	string	证券代码
begin_time	string	起始时间,如2016-05-27 09:00:00
end_time	string	结束时间,如2016-05-27 09:00:00
bar_type	int	bar周期，以秒为单位，比如60即1分钟bar
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

最新tick数据获取方法

get_last_n_ticks(symbol, n, time_out)

参数	类型	说明
symbol	string	证券代码
n	int	提取的数据条数
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

历史tick数据获取方法

get_ticks(symbol, begin_time, end_time,time_out)

参数	类型	说明
symbol	string	证券代码
begin_time	string	起始时间,如2016-05-26
end_time	string	结束时间,如2016-05-27
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

最新tick数据获取方法

get_last_ticks(symbol, time_out)

参数	类型	说明
symbol	string	证券代码
time_out	int	错误退出时间

返回值*:*

查询数据,错误消息

用户操作方法

一日帐号详情获取方法

req_account_detail(trading_day)

参数	类型	说明
trading_day	string	需要获取的帐号详情日期,如:20160827

返回值*:*

查询数据

查询用户持仓详情

req_position(strategy_id)

参数	类型	说明
strategy_id	string	策略id

返回值*:*

查询数据

查询用户的委托的订单列表

req_order_list(strategy_id=None)

参数	类型	说明
strategy_id	string	策略id

返回值*:*

查询数据

订阅数据

req_subscribe(data_type, symbol)

参数	类型	说明
data_type	常量值	需要返回数据类型,使用定义常量,如:TICK,BAR
symbol	string	证券代码

返回值*:*

错误提示

用户下单

send_order(symbol, bs_flag, position_effect, price_type, price, volume, strategy,hedge_type,order_time)

参数	类型	说明
symbol	string	证券代码
bs_flag	常量值	下单方向
position_effect	常量值	订单对持仓的影响
price_type	常量值	价格类型
price	float	委托价
volume	float	委托量
strategy	string	策略名称
hedge_type	常量值	对冲类型
order_time	string	下单时间,只用于回测

返回值*:*

订单id,错误提示