

python api 文档

快速开始	3
安装Python	3
Windows	3
mac	6
下载sdk	9
安装配置	9
我的第一个策略	10
方式1 直接使用例子运行	10
方式2 创建自己的策略	10
Python API 范例	11
数据接口范例	11
历史数据提取	11
行情订阅	12
策略接口范例	13
获取用户持仓	13
查询用户的委托的订单列表	13
查询用户在策略上的持仓详情和 查询用户持仓不同的策略名称	14
策略下单	14
Python api接口	15
常量定义	15
期权相并常量值	15
连接类型常量值	17
期货对冲常量值	17
下单方向	17
订单对持仓的影响	17
交易所	17
数据类型	18
订单状态	18
价格类型常量值	18

策略初始化	19
基础策略类	19
日志函数	19
简单策略类	20
策略开始方法	20
策略验证方法	20
数据提取方法	21
历史bar数据方法	21
最新bar数据获取方法	21
最新一天bar数据获取方法	22
指定一天bar数据获取方法	22
最新tick数据获取方法	23
历史tick数据获取方法	23
最新tick数据获取方法	24
用户操作方法	25
一日帐号详情获取方法	25
查询用户持仓详情	25
查询用户的委托的订单列表	25
订阅数据	26
用户下单	26

快速开始

安装Python

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

Windows

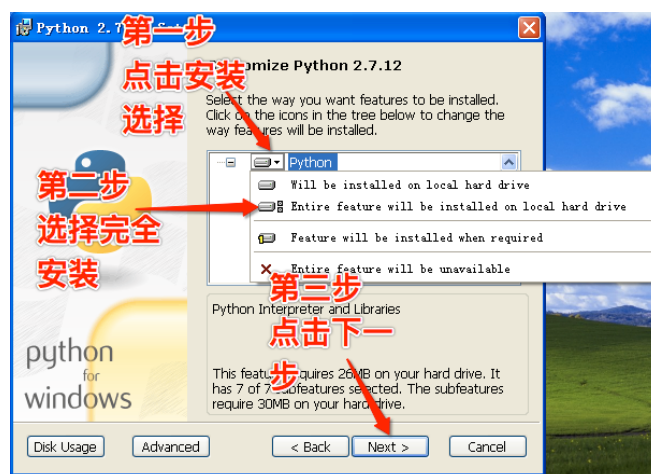
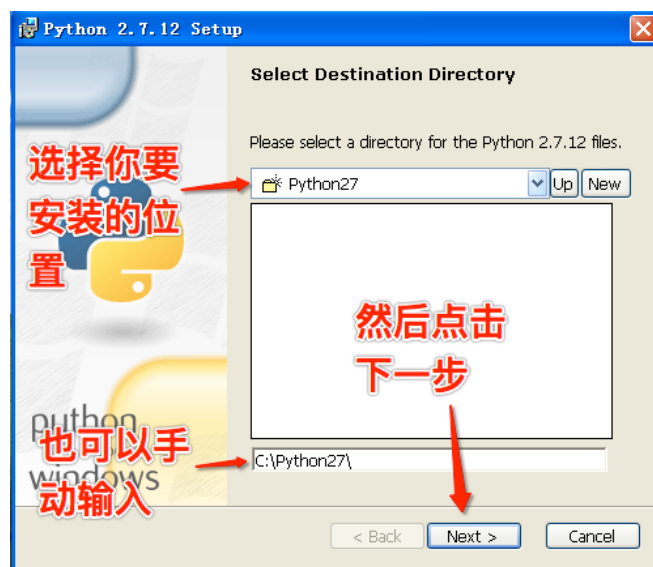
1.使用浏览器登录<https://www.python.org/downloads/release/python-2712/>前往下载Windows对应安装软件。

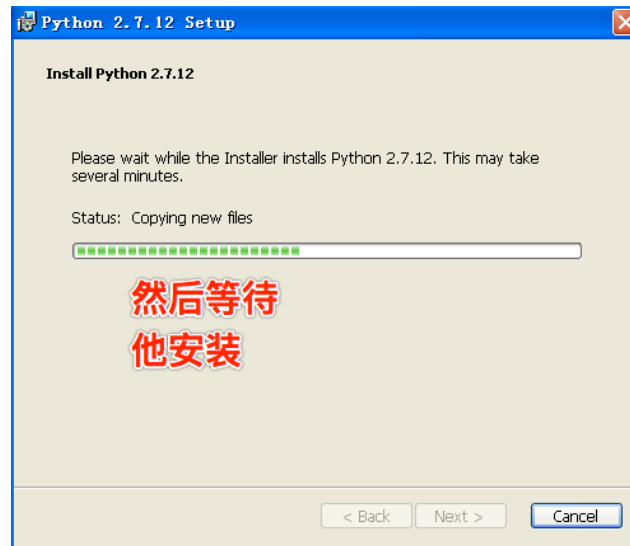
Version	Operating System	Description	MD5 Sum
Gzipped source tarball	Source release		88d61f82e3616
XZ compressed source tarball	Source release		57dffcee9cee8
Mac OS X 32-bit i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later	3adbedcc935a
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	86bedde2becc
Windows debug information files			1751598e1643
Windows debug information files for 64-bit binaries	Windows		c5433a7fca9ec
Windows help file	Windows		7bc4e15ecae8
Windows x86-64 MSI installer	Windows	Windows 64, not Itanium	8fa13925db870
Windows x86 MSI installer	Windows		fe0ef5b8fd0270



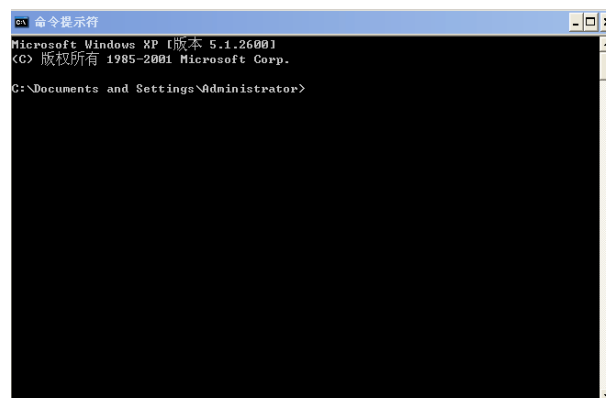
2.点击打开下载好的安装包

3进行如下步骤：





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

```
C:\Documents and Settings\Administrator>python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

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。

Files		
Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
Mac OS X 32-bit/i386/PPC installer	Mac OS X	for Mac OS X 10.5 and later
Mac OS X 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later
Windows debug information files	Windows	
Windows debug information files for 64-bit binaries	Windows	
Windows help file	Windows	
Windows x86-64 MSI installer	Windows	for AMD64/EM64T/x64, not Itanium processors
Windows x86 MSI installer	Windows	

2.点击下载后软件







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

```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 12:54:16)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

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

```
#coding:utf-8

from sd.communication.server.protocol.function_constants import AgentType_STRATEGY_QUOTE, REAL_TIME_QUOTE_SERVER
from sd.strategy.basestrategy import BaseStrategy

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

class DataExample(BaseStrategy):
    def __init__(self, servers, user_name, pass_word, strategy=None, description=None):
        super(DataExample, self).__init__(servers, user_name, pass_word, strategy,
                                          AgentType_STRATEGY_QUOTE, description)

if __name__ == "__main__":
    reqDataTerminal = BaseStrategy(REAL_TIME_QUOTE_SERVER, _user_name, _pass_word)
    (data,error) = reqDataTerminal.get_bars(symbol="SHFE.rb1610", begin_time="2016-05-27 09:00:00",
                                           end_time="2016-05-27 15:00:00", bar_type=60, time_out=20) # 获取60秒K线
    for item in data:
        print item.to_json()
```

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

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.basestrategy 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()
```

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.basestrategy 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()
```

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.basestrategy 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()
```

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.basestrategy 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()
```

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()
```

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()
```

查询用户在策略上的持仓详情和 查询用户持仓不同的策略名称

```
from sd.communication.server.protocol.function_constants import *
from sd.configuration.logconfig import configure_log
from sd.strategy.simplestrategy 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(simple_strategy.quoteStrategy.connectionInfo.strategyName)

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

策略下单

[illegible]

Python api接口

常量定义

期权相并常量值

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

ENTRUST_PROP_GFD	限价GFD
ENTRUST_PROP_OPA	限价即时全部成交否则撤单
ENTRUST_PROP_OPB	市价即时成交剩余撤单
ENTRUST_PROP_OPC	市价即时全部成交否则撤单
ENTRUST_PROP_OPD	市价剩余转限价
ENTRUST_PROP_OTE	期权行权
ENTRUST_PROP_OTU	备兑证券划转

期权类型

OPTION_TYPE_C	认购
OPTION_TYPE_P	认沽

开平仓方向

ENTRUST_OC_O	开仓
ENTRUST_OC_C	平仓
ENTRUST_OC_X	行权
ENTRUST_OC_A	自动行权

备兑标志

COVERED_FLAG_1	备兑
----------------	----

期权持仓类别

OPTHOLD_TYPE_0	权利方
OPTHOLD_TYPE_1	义务方
OPTHOLD_TYPE_2	备兑方

期权开仓限制

OPT_OPEN_STATUS_0	否
OPT_OPEN_STATUS_1	是

期权合约挂牌标志

OPTION_FLAG_A	当日新挂牌合约
OPTION_FLAG_E	存续合约
OPTION_FLAG_D	当日摘牌合约

期权开仓限制

OPT_FINAL_STATUS_0	否
OPT_FINAL_STATUS_1	是

金融品种

FINANCE_TYPE_8	期权
----------------	----

成交状态

BUSINESS_STATUS_0	成交
BUSINESS_STATUS_2	废单
BUSINESS_STATUS_4	确认

自动行权策略类别

OPTEXE_STG_KIND_0	实值X元即行权
OPTEXE_STG_KIND_1	盈利X元即行权
OPTEXE_STG_KIND_2	盈利百分比即行权
OPTEXE_STG_KIND_3	亏损百分比即行权
OPTEXE_STG_KIND_4	实值百分比即行权

委托来源

ENTRUST_SRC_0	个人投资者发起
ENTRUST_SRC_1	交易所发起
ENTRUST_SRC_2	会员发起
ENTRUST_SRC_3	机构投资者发起
ENTRUST_SRC_4	自营交易发起
ENTRUST_SRC_5	流动性服务提供商发起

风险类别

OPTRIKE_TYPE_1	关注
OPTRIKE_TYPE_2	警告
OPTRIKE_TYPE_3	强平

负债类型

DEBT_TYPE_1	欠资
DEBT_TYPE_2	欠券

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

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_dailyBars(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_dailyBars(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