目　录

- run - 运行策略
- stop - 停止策略

数据订阅

- subscribe - 行情订阅
- unsubscribe - 取消订阅

数据事件

- on_tick - tick数据推送事件
- on_bar - bar数据推送事件

数据查询函数

- current - 查询当前行情快照
- history - 查询历史行情
- history_n - 查询历史行情最新n条
- get_fundamentals - 查询基本面数据
- get_fundamentals_n - 查询基本面数据最新n条
- get_instruments - 查询最新交易标的信息
- get_history_instruments - 查询交易标的历史数据
- get_instrumentinfos - 查询交易标的基本信息
- get_history_constituents - 查询指数成份股的历史数据
- get_constituents - 查询指数最新成份股
- get_industry - 查询行业股票列表
- get_concept - 查询概念板块股票列表
- get_trading_dates - 查询交易日列表
- get_previous_trading_date - 返回指定日期的上一个交易日
- get_next_trading_date - 返回指定日期的下一个交易日
- get_dividend - 查询分红送配
- get_continuous_contracts - 获取连续合约

交易函数

- order_volume - 按指定量委托
- order_value - 按指定价值委托
- order_percent - 按总资产指定比例委托
- order_target_volume - 调仓到目标持仓量
- order_target_value - 调仓到目标持仓额
- order_target_percent - 调仓到目标持仓比例（总资产的比例）
- order_batch - 批量委托接口

快速创建我的策略

定时任务

以下的范例代码片段是一个非常简单的例子， 在每个交易日的14: 50: 00 市价购买200股浦发银行股票:

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals

from gm.api import *

def init(context):
    schedule(schedule_func=algo, date_rule='1d', time_rule='14:50:00')


def algo(context):
    # 购买200股浦发银行股票
    order_volume(symbol='SHSE.600000', volume=200, side=OrderSide_Buy, order_type=OrderType_Market, position_effect=PositionEffect_Open, price=0)

if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py', mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00', backtest_end_time='2017-08-21 15:00:00')
```

整个策略需要三步：

1. 设置初始化函数：`init` ，使用 `schedule` 函数进行定时任务配置
2. 配置任务，到点会执行该任务
3. 执行策略

数据事件驱动

策略订阅的每个代码的每一个bar，都会触发策略逻辑

以下的范例代码片段是一个非常简单的例子， 订阅浦发银行的日线bar，bar数据的更新会自动触发on_bar的调用：

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
from gm.api import *


def init(context):
    subscribe(symbols='SHSE.600000', frequency='1d')


def on_bar(context, bars):
    # 打印当前获取的bar信息
    bar = bars[0]
    # 执行策略逻辑操作
    print(bar)

if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py', mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00', backtest_end_time='2017-08-21 15:00:00')
```

整个策略需要三步：

1. 设置初始化函数：init，使用subscribe函数进行数据订阅
2. 实现一个函数：on_bar，来根据数据推送进行逻辑处理
3. 执行策略

时间序列数据事件驱动

策略订阅代码时指定数据窗口大小与周期，平台创建数据滑动窗口，加载初始数据，并在新的bar到

来时自动刷新数据。

on_bar事件触发时，策略可以取到订阅代码的准备好的时间序列数据。

以下的范例代码片段是一个非常简单的例子， 订阅浦发银行的日线bar，bar数据的更新会自动触发on_bar的调用，每次调用 `context.data` 来获取最新的50条日线bar信息：

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
from gm.api import *


def init(context):
    subscribe(symbols='SHSE.600000', frequency='1d', count=50)


def on_bar(context, bars):
    print(context.data(symbol='SHSE.600000', frequency='1d', count=50, fields='close,bob'))


if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py', mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00', backtest_end_time='2017-08-21 15:00:00')
```

整个策略需要三步：

1. 设置初始化函数: `init` ，使用 `subscribe` 函数进行数据订阅
2. 实现一个函数: `on_bar` ，来根据数据推送进行逻辑处理，通过 `context.data` 获取数据滑窗
3. 执行策略

多个代码数据事件驱动

策略订阅多个代码，并且要求同一频度的数据到齐后，再触发事件.

以下的范例代码片段是一个非常简单的例子， 订阅浦发银行和平安银行的日线bar，在浦发银行bar和平安银行bar到齐后会自动触发on_bar的调用：

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
```

```
from gm.api import *


def init(context):
    subscribe(symbols='SHSE.600000,SZSE.000001', frequency='1d', count=5, wait_group=True)


def on_bar(context, bars):
    for bar in bars:
        print(bar['symbol'], bar['eob'])


if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py', mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00', backtest_end_time='2017-08-21 15:00:00')
```

整个策略需要三步：

1. 设置初始化函数: `init` , 使用 `subscribe` 函数进行数据订阅多个代码， 设置 `wait_group=True`
2. 实现一个函数: `on_bar` , 来根据数据推送(多个代码行情)进行逻辑处理
3. 执行策略

默认账户交易

如果策略只关联一个账户，该账户是策略的默认账户。

进行交易时可以不指定交易账户， 以默认账户进行交易。

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
from gm.api import *


def init(context):
    subscribe(symbols='SHSE.600000,SZSE.000001', frequency='1d')

def on_bar(context, bars):
    for bar in bars:
        order_volume(symbol=bar['symbol'], volume=200, side=OrderSide_Buy,
```

```
                    order_type=OrderType_Market,
                    position_effect=PositionEffect_Open, price=0)

if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py',
mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00',
backtest_end_time='2017-08-21 15:00:00')
```

指定账户交易

如果策略在多个账户交易，需要显式指定交易账户。

以下的范例代码片段是一个非常简单的例子， 在下单时使用指定账户 `account='account_1'（参数为账户名称或账户ID）` 来进行交易:

```
# coding=utf-8
from __future__ import print_function, absolute_import,
unicode_literals
from gm.api import *


def init(context):
    subscribe(symbols='SHSE.600000,SZSE.000001', frequency='1m')

def on_bar(context, bars):
    for bar in bars:
      order_volume(symbol=bar['symbol'], volume=200,
side=OrderSide_Buy,
                 order_type=OrderType_Market,
                 position_effect=PositionEffect_Open, price=0,
account='account_1')


if __name__ == '__main__':
    run(strategy_id='strategy_1', filename='main.py',
mode=MODE_BACKTEST, token='token_id',
        backtest_start_time='2016-06-17 13:00:00',
backtest_end_time='2017-08-21 15:00:00')
```

回测模式与实时模式

掘金3策略只有两种模式， 回测模式(backtest)与实时模式(live)。在加载策略时指定mode参数。

回测模式

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals

run(strategy_id='strategy_1', filename='main.py', mode=MODE_BACKTEST, token='token_id',
    backtest_start_time='2016-06-17 13:00:00', backtest_end_time='2017-08-21 15:00:00')
```

`mode=MODE_BACKTEST` 表示回测模式。

实时模式

掘金3策略只有两种模式， 回测模式(backtest)与实时模式(live)。在加载策略时指定mode参数。

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
from gm.api import *

run(strategy_id='strategy_id', filename='main.py', mode=MODE_LIVE, token='token_id')
```

`mode=MODE_LIVE` 表示实时模式。

提取数据研究

如果只想提取数据，无需实时数据驱动策略， 无需交易下单可以直接通过数据查询函数来进行查询。

```
# coding=utf-8
from __future__ import print_function, absolute_import, unicode_literals
from gm.api import *

set_token('xxxx')
data = history(symbol='SHSE.600000', frequency='1d', start_time='2015-01-01', end_time='2015-12-31', fields='open,high,low,close')
```

整个过程只需要两步：

1. set_token 设置用户token， 如果token不正确，函数调用会抛出异常
2. 调用数据查询函数， 直接进行数据查询

回测模式下高速处理数据

本示例提供一种在init中预先取全集数据，规整后索引调用的高效数据处理方式，能够避免反复调用服务器接口导致的低效率问题，可根据该示例思路，应用到其他数据接口以提高效率.

```
# coding=utf-8
from __future__ import print_function, absolute_import
from gm.api import *

def init(context):
    # 在init中一次性拿到所有需要的instruments信息
    instruments = get_history_instruments(symbols='SZSE.000001,SZSE.000002', start_date=context.backtest_start_time,end_date=context.backtest_end_t
    # 将信息按symbol,date作为key存入字典
    context.ins_dict = {(i.symbol, i.trade_date): i for i in instruments}
    subscribe(symbols='SZSE.000001,SZSE.000002', frequency='1d')

def on_bar(context, bars):
    print(context.ins_dict[(bars[0].symbol, bars[0].eob.date())])

if __name__ == '__main__':
    run(strategy_id='*',
        mode=MODE_BACKTEST,
        token='*',
        backtest_start_time='2018-06-17 13:00:00',
        backtest_end_time='2018-08-21 15:00:00')
```

策略程序架构

策略程序架构

掘金策略程序初始化

通过init函数初始化策略,策略启动即会自动执行。在init函数中可以:

- 定义全局变量
  通过添加context包含的属性可以定义全局变量，如context.x,该属性可以在全文中进行传递。
- 定义调度任务
  可以通过schedule配置定时任务，程序在指定时间自动执行策略算法。
- 准备历史数据
  通过数据查询函数获取历史数据
- 订阅实时行情
  通过subscribe订阅行情，用以触发行情事件处理函数。

行情事件处理函数

- 处理盘口 `tick` 数据事件
  通过on_tick响应tick数据事件，可以在该函数中继续添加自己的策略逻辑,如进行数据计算、交易等
- 处理分时 `bar` 数据事件
  通过on_bar响应bar数据事件，可以在该函数中继续添加自己的策略逻辑，如进行数据计算、交易等

交易事件处理函数

- 处理回报 `execrpt` 数据事件
  当交易委托被执行后会触发on_execution_report，用于监测委托执行状态。
- 处理委托 `order` 状态变化数据事件
  当订单状态产生变化时会触发on_order_status,用于监测委托状态变更

策略入口

run函数用于启动策略，策略类交易类策略都需要run函数。在只需提取数据进行研究（即仅使用数据查询函数时）的情况下可以不调用run函数，在策略开始调用set_token即可

- 用户 `token` ID
  用户身份的唯一标识，token位置参见终端-系统设置界面-密钥管理（token）
- 策略ID `strategy_id`
  策略文件与终端连接的纽带，是策略的身份标识。每创建一个策略都会对应生成一个策略id，创建策略时即可看到。
- 策略工作模式
  策略支持两种运行模式， 实时模式和回测模式,实时模式用于仿真交易及实盘交易，回测模式用于策略研究，用户需要在运行策略时选择模式.

重要概念

symbol - 代码标识

掘金代码(**symbol**)是掘金平台用于唯一标识交易标的代码,

格式为: 交易所代码.交易标代码, 比如深圳平安的**symbol** 示例: `SZSE.000001`

交易所代码

目前掘金支持国内的7个交易所, 各交易所的代码缩写如下:

市场中文名	市场代码
上交所	SHSE
深交所	SZSE
中金所	CFFEX
上期所	SHFE
大商所	DCE
郑商所	CZCE
上海国际能源交易中心	INE

交易标的代码

交易表代码是指交易所给出的交易标的代码, 包括股票, 期货, 期权, 指数, 基金等代码。

具体的代码请参考交易所的给出的证券代码定义

mode - 模式选择

策略支持两种运行模式， 实时模式和回测模式,用户需要在运行策略时选择模式.

实时模式

订阅行情服务器推送的实时行情，也就是交易所的实时行情，只在交易时段提供。

回测模式

订阅指定时段、指定交易代码、指定数据类型的行情，行情服务器将按指定条件全速回放对应的行情数据。适用的场景是策略回测阶段，快速验证策略的绩效是否符合预期。

context - 上下文对象

context是策略运行上下文环境对象， 该对象将会在你的算法策略的任何方法之间做传递。

除了系统提供的， 用户也可以根据需求自己定义无限多种自己随后需要的属性

context.symbols - 订阅代码集合

函数原型:

```
context.symbols
```

返回值:

类型	说明
set(str)	订阅代码集合

示例:

```
context.symbols
```

返回:

```
symbol1     symbol2
--------    ----------
DCE.i1801   SHSE.600000
```

context.now - 当前时间

实时模式返回当前本地时间，回测模式返回当前回测时间

函数原型：

```
context.now
```

返回值：

类型	说明
datetime.datetime	当前时间(回测与实时模式不同)

示例：

```
context.now
```

返回：

```
now
--------------------
2017-07-04 09:00:00
```

context.data - 数据滑窗

获取订阅的bar或tick滑窗，数据为包含当前时刻推送bar或tick的前count条bar(tick)数据

原型：

```
context.data(symbol,frequency,count,fields)
```

参数：

参数名	类型	说明
symbol	str	标的代码
frequency	str	频率，所填频率应包含在subscribe订阅过频率中。
count	int	滑窗大小，正整数，此处count值应小于等于subscribe中指定的count值
fields	str	所需bar或tick的字段,如有多属性，中间用 , 隔开,具体字段见:股票字段 ,期货字段

返回值：

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

示例:

```
Subcribe_data = context.data(symbol='SHSE.600000', frequency='60s', count=2, fields='symbol,open,close,volume,eob')
```

输出:

```
  symbol        open          close         volume             eob
------------  ----------  -----------  ----------- -----------------
SHSE.600000    12.64000     12.65000       711900   2017-06-30 15:00:00
SHSE.600000    12.64000     12.62000       241000   2017-07-03 09:31:00
```

注意:

1.所得数据按eob时间正序排列。
2.不支持传入多个symbol或frequency,若输入多个,则返回空dataframe。
3.若fields查询字段包含无效字段，返回KeyError错误。

context.account - 账户信息

可通过此函数获取账户资金信息及持仓信息。

原型:

```
context.account(account_id=None)
```

参数:

参数名	类型	说明
account_id	str	账户信息，默认返回默认账户， 如多个账户需指定account_id

返回值:

返回类型为account - 账户对象。

示例-获取当前持仓:

```
Account_positions = context.account().positions()
```

返回值:

类型	说明
list[position]	持仓对象列表

context.parameters - 动态参数

函数原型:

```
context.parameters
```

返回值:

类型	说明
dict	key为动态参数的key, 值为动态参数对象, 参见动态参数设置

数据类

Tick - Tick对象

逐笔行情数据

参数名	类型	说明
symbol	str	标的代码
open	float	开盘价
high	float	最高价
low	float	最低价
price	float	最新价
cum_volume	long	成交总量/最新成交量,累计值
cum_amount	float	成交总金额/最新成交额,累计值
trade_type	int	交易类型 1: ‘双开’, 2: ‘双平’, 3: ‘多开’, 4: ‘空开’, 5: ‘空平’, 6: ‘多平’, 7: ‘多换’, 8: ‘空换’
last_volume	int	瞬时成交额
cum_position	int	合约持仓量(期),累计值（股票此值为0）
last_amount	float	瞬时成交额
created_at	datetime.datetime	创建时间
quotes	[] (list of dict)	股票提供买卖5档数据, list[0]~list[4]分别对应买卖一档到五档, 期货提供买卖1档数据, list[0]表示买卖一档; 注意: 可能会有买档或卖档报价缺失, 比如跌停时无买档报价（没有bid_p, bid_v）, 涨停时无卖档报价(没有ask_p, ask_v); 其中每档报价 `quote` 结构如下:

> 报价 ***`quote`*** ***- (dict类型)***

参数名	类型	说明
bid_p	float	买价
bid_v	int	买量
ask_p	float	卖价

ask_v	int	卖量

Bar - Bar对象

bar数据是指各种频率的行情数据

参数名	类型	说明
symbol	str	标的代码
frequency	str	频率，支持多种频率，具体见股票行情数据和期货行情数据
open	float	开盘价
close	float	收盘价
high	float	最高价
low	float	最低价
amount	float	成交额
volume	long	成交量
position	long	持仓量（仅期货）
bob	datetime.datetime	bar开始时间
eob	datetime.datetime	bar结束时间

交易类

Account - 账户对象

属性	类型	说明
id	str	账户id，实盘时用于指定交易账户
title	str	账户标题，实盘时用于指定账户名称
cash	dict	资金字典
positions(symbol='', side=None)	list	持仓情况 列表， 默认全部持仓， 可根据单一symbol（类型str）， side参数缩小查询范围
position(symbol, side)	dict	持仓情况，查询指定单一symbol（类型str）及持仓方向的持仓情况

Order - 委托对象

属性	类型	说明
strategy_id	str	策略ID
account_id	str	账号ID
account_name	str	账户登录名
cl_ord_id	str	委托客户端ID，下单生成，固定不变
order_id	str	委托柜台ID（系统字段）
ex_ord_id	str	委托交易所ID（系统字段）
symbol	str	标的代码
side	int	买卖方向 取值参考 OrderSide
position_effect	int	开平标志 取值参考 PositionEffect
position_side	int	持仓方向 取值参考 PositionSide
order_type	int	委托类型 取值参考 OrderType

order_duration	int	委托时间属性 取值参考 OrderDuration
order_qualifier	int	委托成交属性 取值参考 OrderQualifier
order_src	int	委托来源（系统字段）
status	int	委托状态 取值参考 OrderStatus
ord_rej_reason	int	委托拒绝原因 取值参考 OrderRejegectReason
ord_rej_reason_detail	str	委托拒绝原因描述
price	float	委托价格
stop_price	float	委托止损/止盈触发价格
order_style	int	委托风格 取值参考 OrderStyle
volume	long	委托量
value	int	委托额
percent	float	委托百分比
target_volume	long	委托目标量
target_value	int	委托目标额
target_percent	float	委托目标百分比
filled_volume	long	已成量
filled_vwap	float	已成均价
filled_amount	float	已成金额
created_at	datetime.datetime	委托创建时间
updated_at	datetime.datetime	委托更新时间

ExecRpt - 回报对象

属性	类型	说明
strategy_id	str	策略ID
account_id	str	账号ID
account_name	str	账户登录名
cl_ord_id	str	委托客户端ID
order_id	str	委托柜台ID
ex_ord_id	str	委托交易所ID
position_effect	int	开平标志 取值参考 PositionEffect
side	int	买卖方向 取值参考 OrderSide

ord_rej_reason	int	委托拒绝原因 取值参考 OrderRejectReason
ord_rej_reason_detail	str	委托拒绝原因描述
exec_type	int	执行回报类型 取值参考 ExecType
price	float	委托成交价格
volume	long	委托成交量
amount	float	委托成交金额
created_at	datetime.datetime	回报创建时间

Cash - 资金对象

属性	类型	说明
account_id	str	账号ID
account_name	str	账户登录名
currency	int	币种
nav	float	净值
pnl	float	净收益
fpnl	float	浮动盈亏
frozen	float	持仓占用资金
order_frozen	float	挂单冻结资金
available	float	可用资金
cum_inout	float	累计出入金
cum_trade	float	累计交易额
cum_pnl	float	累计平仓收益(没扣除手续费)
cum_commission	float	累计手续费
last_trade	float	上一次交易额
last_pnl	float	上一次收益
last_commission	float	上一次手续费
last_inout	float	上一次出入金
change_reason	int	资金变更原因 取值参考 CashPositionChangeReason
change_event_id	str	触发资金变更事件的ID
created_at	datetime.datetime	资金初始时间
updated_at	datetime.datetime	资金变更时间

Position - 持仓对象

属性	类型	说明
account_id	str	账号ID
account_name	str	账户登录名
symbol	str	标的代码
side	int	持仓方向 取值参考 PositionSide
volume	long	总持仓量; 昨持仓量 (volume - volu
volume_today	long	今日持仓量
vwap	float	持仓均价 new_vwap=((position.vwa (trade.volume*trade.price))/(pos
amount	float	持仓额 (volume*vwap*multiplier)
price	float	当前行情价格 (回测时值为0)
fpnl	float	持仓浮动盈亏 ((price - vwap) * v 率的考虑, 回测模式fpnl只有仓位变化时 次)
cost	float	持仓成本 (vwap * volume * multip
order_frozen	int	挂单冻结仓位
order_frozen_today	int	挂单冻结今仓仓位
available	long	可平总仓位 (volume - order_froze available_today)
available_today	long	可平今仓位 (volume_today - order
last_price	float	上一次成交价 (回测时值为0)
last_volume	long	上一次成交量 (回测时值为0)
last_inout	int	上一次出入持仓量 (回测时值为0)
change_reason	int	仓位变更原因, 取值参考 CashPosition
change_event_id	str	触发资金变更事件的ID
created_at	datetime.datetime	建仓时间
updated_at	datetime.datetime	仓位变更时间

Indicator - 绩效指标对象

属性	类型	说明
account_id	str	账号ID
pnl_ratio	float	累计收益率 (pnl/cum_inout)

pnl_ratio_annual	float	年化收益率 (pnl_ratio/自然天数 *365)
sharp_ratio	float	夏普比率 ([E(Rp)-Rf]/δp,E(Rp) = mean(pnl_ratio),Rf = 0,δp = std(pnl_ratio))
max_drawdown	float	最大回撤 max_drawdown=max(Di-Dj)/Di; D为某一天的净值(j>i)
risk_ratio	float	风险比率 (持仓市值/nav)
open_count	int	开仓次数
close_count	int	平仓次数
win_count	int	盈利次数
lose_count	int	亏损次数
win_ratio	float	胜率 (win_count / (win_count + lose_count))
created_at	datetime.datetime	指标创建时间
updated_at	datetime.datetime	指标变更时间

基本函数

init - 初始化策略

初始化策略， 策略启动时自动执行。可以在这里初始化策略配置参数。

函数原型：

```
init(context)
```

参数：

参数名	类型	说明
context	context	上下文，全局变量可存储在这里

示例：

```
def init(context):
    # 订阅bar
    subscribe(symbols='SHSE.600000,SHSE.600004', frequency='30s', count=5, wait_group=True, wait_group_timeout='5s')
    # 增加对象属性，如:设置一个股票资金占用百分比
    context.percentage_stock = 0.8
```

注意：
回测模式下init函数里不支持交易操作，仿真模式和实盘模式支持

schedule - 定时任务配置

在指定时间自动执行策略算法， 通常用于选股类型策略

函数原型：

```
schedule(schedule_func, date_rule, time_rule)
```

参数：

参数名	类型	说明
schedule_func	function	策略定时执行算法
date_rule	str	n + 时间单位， 可选’d/w/m’ 表示n天/n周/n月
time_rule	str	执行算法的具体时间 (%H:%M:%S 格式)

返回值:

`None`

示例:

```
def init(context):
    #每天的19:06:20执行策略algo_1
    schedule(schedule_func=algo_1, date_rule='1d',
time_rule='19:06:20')
    #每月的第一个交易日的09:40:00执行策略algo_2
    schedule(schedule_func=algo_2, date_rule='1m',
time_rule='9:40:00')

def algo_1(context):
    print(context.symbols)
def algo_2(context):
    order_volume(symbol='SHSE.600000', volume=200,
side=OrderSide_Buy, order_type=OrderType_Market,
position_effect=PositionEffect_Open)
```

注意:

1.time_rule的时,分,秒均可以只输入个位数，例: `'9:40:0'` 或 `'14:5:0'` ,但若对应位置为零，则0不可被省略，比如不能输入 `'14:5: '`

2.目前暂时支持 `1d` 、 `1w` 、 `1m` ，其中 `1w` 、 `1m` 仅用于回测

run - 运行策略

函数原型:

```
run(strategy_id='', filename='', mode=MODE_UNKNOWN, token='',
backtest_start_time='',
    backtest_end_time='', backtest_initial_cash=1000000,
    backtest_transaction_ratio=1, backtest_commission_ratio=0,
    backtest_slippage_ratio=0, backtest_adjust=ADJUST_NONE,
backtest_check_cache=1,
```

```
5.      serv_addr='')
```

参数:

参数名	类型	说明
strategy_id	str	策略id
filename	str	策略文件名称
mode	int	策略模式 MODE_LIVE(实时)=1 MODE_BACKTEST(回测) =2
token	str	用户标识
backtest_start_time	str	回测开始时间 (%Y-%m-%d %H:%M:%S格式)
backtest_end_time	str	回测结束时间 (%Y-%m-%d %H:%M:%S格式)
backtest_initial_cash	double	回测初始资金, 默认1000000
backtest_transaction_ratio	double	回测成交比例, 默认1.0, 即下单100%成交
backtest_commission_ratio	double	回测佣金比例, 默认0
backtest_slippage_ratio	double	回测滑点比例, 默认0
backtest_adjust	int	回测复权方式(默认不复权) ADJUST_NONE(不复权)=0 ADJUST_PREV(前复权)=1 ADJUST_POST(后复权)=2
backtest_check_cache	int	回测是否使用缓存: 1 - 使用, 0 - 不使用; 默认使用
serv_addr	str	终端服务地址, 默认本地地址, 可不填, 若需指定应输入ip+端口号, 如”127.0.0.1:8080”

返回值:

`None`

示例:

```
1. run(strategy_id='strategy_1', filename='main.py',
   mode=MODE_BACKTEST, token='token_id',
2.     backtest_start_time='2016-06-17 13:00:00',
   backtest_end_time='2017-08-21 15:00:00')
```

注意:

1.run函数中, `mode=1` 也可改为 `mode=MODE_LIVE` , 两者等

价， `backtest_adjust` 同理

2.backtest_start_time和backtest_end_time中月,日,时,分,秒均可以只输入个位数，例: `'2016-6-7 9:55:0'` 或 `'2017-8-1 14:6:0'` ,但若对应位置为零，则0不可被省略，比如不能输入 `"2017-8-1 14:6: "`

3. filename指运行的py文件名字，如该策略文件名为Strategy.py,则此处应填”Strategy.py”

stop - 停止策略

函数原型:

```
stop()
```

返回值:

`None`

示例:

```
#若订阅过的代码集合为空，停止策略
if not context.symbols:
    stop()
```

数据订阅

subscribe - 行情订阅

订阅行情，可以指定symbol，数据滑窗大小，以及是否需要等待全部代码的数据到齐再触发事件。

函数原型：

```
subscribe(symbols, frequency='1d', count=1, wait_group=False,
wait_group_timeout='10s', unsubscribe_previous=False)
```

参数：

参数名	类型	说明
symbols	str or list	订阅标的代码，支持字串格式，如有多个代码，中间用 `,` (英文逗号) 隔开，也支持 `['symbol1', 'symbol2']` 这种列表格式
frequency	str	频率，支持 ‘tick’, ‘1d’, ‘15s’, ‘30s’ 等，默认 ‘1d’, 详情见股票行情数据和期货行情数据
count	int	订阅数据滑窗大小，默认 `1` ,详情见数据滑窗
wait_group	bool	是否需要等待全部代码的数据到齐再触发事件，默认 `False` 不到齐。设置为 `True` 则等待订阅标的 `eob` 相同的 `bar` 全部到齐再被调用。该参数只对Bar数据有效。
wait_group_timeout	str	超时时间设定，支持 `s` 结尾表示单位 `秒` ，默认 `10s`
unsubscribe_previous	bool	是否取消过去订阅的symbols，默认 `False` 不取消，输入 `True` 则取消所有原来的订阅。

返回值：

`None`

示例：

```
subscribe(symbols='SHSE.600000,SHSE.600004', frequency='tick',
count=5, wait_group=True, wait_group_timeout='6s',
unsubscribe_previous=True)
```

注意：

subscribe支持多次调用，并可以重复订阅同一代码。

unsubscribe - 取消订阅

取消行情订阅，默认取消所有已订阅行情

函数原型：

```
1. unsubscribe(symbols='*', frequency='1d')
```

参数：

参数名	类型	说明
symbols	str or list	标的代码，支持字串格式，如果有多个代码，中间用 `,` (英文逗号) 隔开; `*` 表示所有，默认退订所有代码 也支持 `['symbol1', 'symbol2']` 这种列表格式的参数
frequency	str	频率，支持 ‘tick’, ‘1d’, ‘15s’, ‘30s’ 等，默认’1d’, 详情见股票行情数据和期货行情数据

返回值：

`None`

示例：

```
1. unsubscribe(symbols='SHSE.600000,SHSE.600004', frequency='tick')
```

注意：
如示例所示代码，取消 `SHSE.600000,SHSE.600004` 两只代码 `tick` 行情的订阅，
若 `SHSE.600000` 同时还订阅了 `"60s"` 频度的行情，该代码不会取消该标的此频度的订阅

数据事件

on_tick - tick数据推送事件

函数原型:

```
on_tick(context, tick)
```

参数:

参数名	类型	说明
context	context对象	上下文
tick	tick对象	当前被推送的tick

示例:

```
def on_tick(context, tick):
    print(tick)
```

输出:

```
  cum_volume    trade_type  created_at             cum_position
symbol          high    low    cum_amount    open    price    quotes
------------  ------------  -------------------  --------------  ---
---------  ------    -----  ------------  ------  -------  ---------
------------
       11608             8  2017-07-28 06:59:52           22023
CFFEX.IF1708    3712    3684   1.28903e+10    3687     3708
[{u'bid_p': 0.0, u'ask_v': 0, u'ask_p': 0.0, u'bid_v': 0}...]
```

on_bar - bar数据推送事件

函数原型:

```
on_bar(context, bars)
```

参数:

参数名	类型	说明
context	context对象	上下文对象
bars	list(bar)	当前被推送的bar列表

示例:

```
def on_bar(context, bars):
    for bar in bars:
        print(bar)
```

输出:

```
  volume  eob                   pre_close  symbol          high
amount  frequency        low    close  bob                      open
--------  -------------------  -----------  -----------  -------  --
---------  -----------  -------  -------  -------------------  -----
--
95965409  2010-07-28 16:00:00      2863.72  SHSE.000300  2888.6
9.22782e+10  1d           2852     2877.98  2010-07-28 16:00:00
2866.77
```

注意:

1.若在subscribe函数中订阅了多个标的的bar，但wait_group参数值为False,则多次触发On_bar,每次返回只包含单个标的list长度为1的bars;若参数值为True,则只会触发一次On_bar,返回包含多个标的的bars。

2.bar在本周期结束时间后才会推送，标的在这个周期内无交易则不推送bar。

数据查询函数

current - 查询当前行情快照

查询当前行情快照，返回tick数据。回测时，返回回测时间点的tick数据

函数原型：

```
current(symbols, fields='')
```

参数：

参数名	类型	说明
symbols	str or list	查询代码，如有多个代码，中间用 `,` (英文逗号) 隔开，也支持 `['symbol1', 'symbol2']` 这种列表格式
fields	str	查询字段，默认所有字段 具体字段见:Tick

返回值：
`list[dict]`

示例：

```
current_data = current(symbol='SZSE.000001')
```

输出：

```
    symbol          open                    high            cum_volumn ...... bid_p            bid_v            ask_p                ask_v     ......
---------- ------------------ ------------------- ------------ -------- ------------------ ----------- ------------------- ------- ---------
SZSE.000001  12.150               12.120                   27160202 ......     12.070              184400      12.090               88700     ......
```

注意：

1.若输入包含无效标的代码，则返回的列表只包含有效标的代码对应的 `dict`

2.若输入代码正确，但查询字段中包括错误字段，返回的列表仍包含对应数量的 `dict` ，但每个 `dict` 中除有效字段外，其他字段的值均为 `空字符串/0`

history - 查询历史行情

函数原型：

```
history(symbol, frequency, start_time, end_time, fields=None, skip_suspended=True,
        fill_missing=None, adjust=ADJUST_NONE, adjust_end_time='', df=False)
```

参数：

参数名	类型	说明
symbols	str	标的代码,若要获取多个标的的历史数据，可以采用中间用 `,` （英文逗号）隔开的方法，但不能是 `list` 类型的输入参数
frequency	str	频率，支持 ‘tick’, ‘1d’, ‘15s’, ‘30s’ 等，默认 ‘1d’, 详情见股票行情数据和期货行情数据
start_time	str or datetime.datetime	开始时间 (%Y-%m-%d %H:%M:%S 格式), 也支持datetime.datetime格式, 其中日线包含start_time数据, 非日线不包含start_time数据
end_time	str or datetime.datetime	结束时间 (%Y-%m-%d %H:%M:%S 格式), 也支持datetime.datetime格式
		指定返回对象字段, 如有多个字段, 中间

fields	str	用, 隔开, 默认所有, 具体字段见:股票字段 ,期货字段
skip_suspended	bool	是否跳过停牌, 默认跳过
fill_missing	str or None	填充方式, `None` - 不填充, `'NaN'` - 用空值填充, `'Last'` - 用上一个值填充, 默认None
adjust	int	`ADJUST_NONE or 0: 不复权`, `ADJUST_PREV or 1: 前复权`, `ADJUST_POST or 2: 后复权` 默认不复权
adjust_end_time	str	复权基点时间, 默认当前时间
df	bool	是否返回 dataframe格式, 默认 `False`, 返回list[dict]

返回值:

`df = True`

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

`df = False`

类型	说明
list	tick 列表 或者 bar 列表

示例:

```
history_data = history(symbol='SHSE.000300', frequency='1d',
start_time='2010-07-28',  end_time='2017-07-30', df=False)
```

输出:

```
  volume  eob                   pre_close  symbol          high
amount  frequency        low    close  bob                    open
--------  -------------------  -----------  -----------  -------  --
---------  -----------  -------  -------  -------------------  -----
--
95965409  2010-07-28 16:00:00      2863.72  SHSE.000300  2888.6
9.22782e+10  1d           2852     2877.98  2010-07-28 16:00:00
2866.77
62575591  2010-07-29 16:00:00      2877.98  SHSE.000300  2876.14
6.52245e+10  1d           2844.68  2868.85  2010-07-29 16:00:00
```

```
    2871.48
```

注意:

1.返回的 `list/DataFrame` 是以参数 `eob/bob` 的升序来排序的，若要获取多标的的数据，通常需进一步的数据处理来分别提取出每只标的的历史数据

2.start_time和end_time中月,日,时,分,秒均可以只输入个位数,例: `'2010-7-8 9:40:0'` 或 `'2017-7-30 12:3:0'` ,但若对应位置为零，则 `0` 不可被省略,如不可输入 `'2017-7-30 12:3: '`

3.若输入无效标的代码，返回 `空列表/空DataFrame`

4.若输入代码正确，但查询字段包含无效字段，返回的列表、DataFrame只包含 `eob、symbol` 和输入的其他有效字段

5. skip_suspended 和 fill_missing 参数暂不支持

history_n - 查询历史行情最新n条

函数原型:

```
1. history_n(symbol, frequency, count, end_time=None, fields=None,
   skip_suspended=True,
2.           fill_missing=None, adjust=ADJUST_NONE, adjust_end_time='',
   df=False)
```

参数:

参数名	类型	说明
symbol	str	标的代码(只允许单个标的的代码字符串)
frequency	str	频率, 支持 ‘tick’, ‘1d’, ‘15s’, ‘30s’ 等,详情见股票行情数据和期货行情数据
count	int	数量(正整数)
end_time	str or datetime.datetime	结束时间 (%Y-%m-%d %H:%M:%S 格式), 也支持datetime.datetime格式
fields	str	指定返回对象字段, 如有多个字段, 中间用 , 隔开, 默认所有, 具体字段见:股票字段 ,期货字段
skip_suspended	bool	是否跳过停牌, 默认跳过
fill_missing	str or None	填充方式, `None` - 不填充, `'NaN'` - 用空值填充, `'Last'` - 用上一个值填充, 默认 `None`

adjust	int	`ADJUST_NONE or 0: 不复权`, `ADJUST_PREV or 1: 前复权`, `ADJUST_POST or 2: 后复权` 默认不复权
adjust_end_time	str	复权基点时间, 默认当前时间
df	bool	是否返回dataframe 格式, 默认False, 返回list[dict]

返回值:

`df = True`

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

`df = False`

类型	说明
list	tick 列表 或者 bar 列表

示例:

```
history_n_data = history_n(symbol='CFFEX.IF1708', frequency='tick',
end_time='2017-07-30 20:00:20',
fields='cum_amount,open,symbol,price', count=20, df=False)
```

输出:

```
symbol         cum_amount    open    price
----------     ------------  ------  -------
CFFEX.IF1708    1.28903e+10    3687     3708
CFFEX.IF1708    1.28903e+10    3687     3708
```

注意:

1.返回的 `list/DataFrame` 是以参数 `eob/bob` 的升序来排序的

2.若输入无效标的代码, 返回 `空列表/空DataFrame`

3.若输入代码正确, 但查询字段包含无效字段, 返回的列表、DataFrame只包含 `eob、symbol` 和输入的其他有效字段

4.end_time中月,日,时,分,秒均可以只输入个位数,例: `'2017-7-30 20:0:20'` ,但若对应位置为零, 则 `0` 不可被省略,如不可输入 `'2017-7-30 20: :20'`

5. skip_suspended 和 fill_missing 参数暂不支持

get_fundamentals - 查询基本面数据

函数原型:

```
1. get_fundamentals(table, symbols, start_date, end_date, fields=None,
   filter=None, order_by=None, limit=1000, df=False)
```

参数:

参数名	类型	说明
table	str	表名, 只支持单表查询. 具体表名及fields字段名及filter可过滤的字段参考 财务数据文档
symbols	str or list	标的代码, 多个代码可用 , (英文逗号)分割, 也支持 ['symbol1', 'symbol2'] 这种列表格式
start_date	str	开始时间, (%Y-%m-%d 格式)
end_date	str	结束时间, (%Y-%m-%d 格式)
fields	str	查询字段 (必填)
filter	str	查询过滤,使用方法参考例3、例4
order_by	str or None	排序方式, 默认 None . TCLOSE 表示按 TCLOSE 升序排序. -TCLOSE 表示按 TCLOSE 降序排序. TCLOSE, -NEGOTIABLEMV 表示按 TCLOSE 升序, NEGOTIABLEMV 降序综合排序
limit	int	数量. 为保护服务器, 一次查询最多返回 40000 条记录
df	bool	是否返回dataframe格式, 默认False, 返回list[dict]

返回值:

key	**value类型**	说明
symbol	str	标的代码
pub_date	datetime.datetime	公司发布财报的日期.
end_date	datetime.datetime	财报统计的季度的最后一天.
fields	dict	相应指定查询 fields 字段的值. 字典key值请参考 财务数据文档

示例:

例1: 取股票代码 SHSE.600000, SZSE.000001 , 离 2017-01-01 最近一个交易日的股票交易财务衍生表 的

`TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFYNPAAEI,PETTMNPAAEI` 字段的值

```
get_fundamentals(table='trading_derivative_indicator',
symbols='SHSE.600000, SZSE.000001', start_date='2017-01-01',
end_date='2017-01-01',

fields='TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFY
                )
```

输出:

```
symbol       pub_date             end_date
NEGOTIABLEMV     PEMRQ     PELFYNPAAEI     PETTMNPAAEI     PELFY
TURNRATE     PETTM     TOTMKTCAP     TCLOSE
-----------  -------------------  -------------------  -------------
-  -------  -------------  -------------  -------  ----------  -----
--  -----------  --------
SHSE.600000  2016-12-30 00:00:00  2016-12-30 00:00:00     3.3261e+11
6.4605         7.0707         6.6184   6.925       0.0598   6.4746
3.50432e+11     16.21
SZSE.000001  2016-12-30 00:00:00  2016-12-30 00:00:00
1.33144e+11   6.2604         7.1341         6.2644   7.1462
0.2068   6.8399  1.56251e+11      9.1
```

例2: 取股票代码 `SHSE.600000, SZSE.000001` , 指定时间段 `2016-01-01 -- 2017-01-01` 股票交易财务衍生表 的 `TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFYNPAAEI,PETTMNPAAEI` 字段的值

```
get_fundamentals(table='trading_derivative_indicator',
symbols='SHSE.600000, SZSE.000001', start_date='2016-01-01',
end_date='2017-01-01',

fields='TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFY
                )
```

例3: 取指定股票 `SHSE.600000, SHSE.600001, SHSE.600002` 离 `2017-01-01` 最近一个交易日, 且满足条件 `PCTTM > 0 and PCTTM < 10` 股票交易财务衍生表 的 `TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFYNPAAEI,PETTMNPAAEI` 字段的值

```
get_fundamentals(table='trading_derivative_indicator', symbols='SHSE.600000, SHSE.600001, SHSE.600002', start_date='2017-01-01', end_date='2017-01-01', filter='PCTTM > 0 and PCTTM < 10',
fields='TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFY
                )
# 或者这样写
my_symbols = ['SHSE.600000', 'SHSE.600001', 'SHSE.600002']
get_fundamentals(table='trading_derivative_indicator', start_date='2017-01-01', end_date='2017-01-01', filter='PCTTM > 0 and PCTTM < 10',
                symbols=my_symbols,
fields='TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFY
                )
```

例4: 取指定股票 `SHSE.600000, SZSE.000001` 离 `2016-01-20` 最近一个财报, 同时满足条件 `CURFDS > 0 and TOTLIABSHAREQUI > 0` 的 资产负债 的数据

```
get_fundamentals(table='balance_sheet', start_date='2016-01-20', end_date='2016-01-20',
                fields='CURFDS, SETTRESEDEPO, PLAC, TRADFINASSET, ',
                symbols='SHSE.600000, SZSE.000001',
                filter='CURFDS > 0 and TOTLIABSHAREQUI > 0'
                )
# 或者这样写
my_symbols = ['SHSE.600000', 'SZSE.000001']
get_fundamentals(table='balance_sheet', end_date='2016-01-20',
                fields='CURFDS, SETTRESEDEPO, PLAC, TRADFINASSET, ',
                symbols=my_symbols,
                filter='CURFDS > 0 and TOTLIABSHAREQUI > 0'
                )
```

注意:

1.当 `start_date` == `end_date` 时, 取所举每个股票离 `end_date` 最近业务日期(交易日期或报告日期) 一条数据,当 `start_date` < `end_date` 时, 取指定时间段的数据,当 `start_date` > `end_date` 时, 返回 `空list/空DataFrame`

2.当不指定排序方式时,返回的 `list/DataFrame` 以参数 `pub_date/end_date` 来排序

3.start_date和end_date中月,日均可以只输入个位数,例: `'2010-7-8'` 或 `'2017-7-30'`

4.若输入包含无效标的代码，则返回的 `list/DataFrame` 只包含有效标的代码对应的数据

5.在该函数中，table参数只支持输入一个表名，若表名输入错误或以 `'table1,table2'` 方式输入多个表名，函数返回 `空list/空DataFrame`

6.若表名输入正确，但查询字段中出现非指定字符串，则程序直接报错

get_fundamentals_n - 查询基本面数据最新n条

取指定股票的最近 `end_date` 的 `count` 条记录

函数原型:

```
get_fundamentals_n(table, symbols, end_date, fields=None, filter=None, order_by=None, count=1, df=False)
```

参数:

参数名	类型	说明
table	str	表名. 具体表名及fields字段名及filter可过滤的字段参考 财务数据文档
symbols	str	标的代码, 多个代码可用 `,` (英文逗号)分割, 也支持 `['symbol1', 'symbol2']` 这种列表格式
end_date	str	结束时间, (%Y-%m-%d 格式)
fields	str	查询字段 (必填)
filter	str	查询过滤,,使用方法参考 `get_fundamentals` 的例3、例4
count	int	每个股票取最近的数量(正整数)
df	bool	是否返回dataframe格式, 默认False, 返回list[dict]

返回值:

key	**value类型**	说明
symbol	str	标的代码
pub_date	datetime.datetime	公司发布财报的日期.
end_date	datetime.datetime	财报统计的季度的最后一天.
fields	dict	相应指定查询 `fields` 字段的值. 字典key值请参考 财务数据文档

示例:

例1: 取股票代码 `SHSE.600000, SZSE.000001` , 离 `2017-01-01` 最近3条(每个股票都有3条) 股票交易财务衍生表 的

`TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFYNPAAEI,PETTMNPAAEI` 字段的值

```
get_fundamentals_n(table='trading_derivative_indicator',
symbols='SHSE.600000, SZSE.000001', end_date='2017-01-01', count=3,

fields='TCLOSE,NEGOTIABLEMV,TOTMKTCAP,TURNRATE,PELFY,PETTM,PEMRQ,PELFY
                    )
```

输出:

```
symbol       pub_date             end_date               TCLOSE
TOTMKTCAP    PETTM    TURNRATE    PETTMNPAAEI    PELFY
PELFYNPAAEI    NEGOTIABLEMV    PEMRQ
-----------  -------------------  -------------------  --------  ---
--------  -------  ----------  -------------  -------  -------------
--------------  -------
SZSE.000001  2016-12-30 00:00:00  2016-12-30 00:00:00       9.1
1.56251e+11   6.8399      0.2068         6.2644   7.1462
7.1341     1.33144e+11   6.2604
SZSE.000001  2016-12-29 00:00:00  2016-12-29 00:00:00      9.08
1.55907e+11   6.8249      0.2315         6.2506   7.1305
7.1184     1.32851e+11   6.2466
SZSE.000001  2016-12-28 00:00:00  2016-12-28 00:00:00      9.06
1.55564e+11   6.8098      0.2297         6.2369   7.1147
7.1027     1.32558e+11   6.2329
SHSE.600000  2016-12-30 00:00:00  2016-12-30 00:00:00     16.21
3.50432e+11   6.4746      0.0598         6.6184   6.925
7.0707     3.3261e+11    6.4605
SHSE.600000  2016-12-29 00:00:00  2016-12-29 00:00:00     16.07
3.47406e+11   6.4187      0.0578         6.5613   6.8652
7.0097     3.29737e+11   6.4047
SHSE.600000  2016-12-28 00:00:00  2016-12-28 00:00:00     16.09
3.47838e+11   6.4267      0.0704         6.5694   6.8737
7.0184     3.30148e+11   6.4126
```

注意:

1.对每个标的,返回的 `list/DataFrame` 以参数 `pub_date/end_date` 的倒序来排序

2.end_date中月,日均可以只输入个位数,例: `'2010-7-8'` 或 `'2017-7-30'`

3.若输入包含无效标的代码，则返回的 `list/DataFrame` 只包含有效标的代码对应的数据

4.在该函数中，table参数只支持输入一个表名，若表名输入错误或以 `'table1,table2'` 方式输入多个表名，函数返回 `空list/空DataFrame`

5.若表名输入正确，但查询字段中出现非指定字符串，则程序直接报错

get_instruments - 查询最新交易标的信息

查询最新交易标的信息,有基本数据及最新日频数据

函数原型:

```
get_instruments(symbols=None, exchanges=None, sec_types=None,
names=None, skip_suspended=True, skip_st=True, fields=None,
df=False)
```

参数:

参数名	类型	说明
symbols	str or list or None	标的代码 多个代码可用 `,` (英文逗号)分割, 也支持 `['symbol1', 'symbol2']` 这种列表格式,默认None表示所有
exchanges	str or list or None	见交易所代码, 多个交易所代码可用 `,` (英文逗号)分割, 也支持 `['exchange1', 'exchange2']` 这种列表格式,默认None表示所有
sec_types	list	指定类别, 默认所有, 其他类型见sec_type 类型
names	str or None	查询代码, 默认None 表示所有
skip_suspended	bool	是否跳过停牌, 默认True 跳过停牌
skip_st	bool	是否跳过ST, 默认True 跳过ST
fields	str or None	查询字段 默认None 表示所有
df	bool	是否返回dataframe格式, 默认False, 返回list[dict]

返回值:

key	类型	说明
symbol	str	标的代码
sec_type	int	1: 股票, 2: 基金, 3: 指数, 4: 期货, 5: 期权, 10: 虚拟合约
exchange	str	见交易所代码

sec_id	str	代码
sec_name	str	名称
sec_abbr	str	拼音简称
price_tick	float	最小变动单位
listed_date	datetime.date	上市日期
delisted_date	datetime.date	退市日期
trade_date	datetime.date	交易日期
sec_level	int	1-正常,2-ST 股票,3-*ST 股票,4-股份转让,5-处于退市整理期的证券,6-上市开放基金LOF,7-交易型开放式指数基金(ETF),8-非交易型开放式基金(暂不交易,仅揭示基金净值及开放申购赎回业务),9-仅提供净值揭示服务的开放式基金;,10-仅在协议交易平台挂牌交易的证券,11-仅在固定收益平台挂牌交易的证券,12-风险警示产品,13-退市整理产品,99-其它
is_suspended	int	是否停牌. 1: 是, 0: 否
multiplier	float	合约乘数
margin_ratio	float	保证金比率
settle_price	float	结算价
position	int	持仓量
pre_close	float	昨收价
upper_limit	float	涨停价
lower_limit	float	跌停价
adj_factor	float	复权因子.基金跟股票才有

示例:

```
1. get_instruments(exchanges='SZSE')
```

输出:

```
1.   adj_factor    is_st    upper_limit  sec_name      pre_close
   symbol         price_tick  delisted_date        exchange
   listed_date            sec_type    settle_price    lower_limit
   multiplier  sec_abbr      position  trade_date              sec_id
   is_suspended    margin_ratio
2. ------------  -------  -------------  ----------  -----------  -----
   ------  ------------  -------------------  ----------  -------------
   ------  ----------  --------------  -------------  ------------  ---
   -------  ----------  -------------------    --------  --------------
```

```
--------------
   115.338           0          12.38  平安银行              11.25
SZSE.000001          0.01  2038-01-01 00:00:00  SZSE        1991-04-
03 00:00:00          1              0          10.13             1
payx               0   2017-09-19 00:00:00    000001              0
1
   127.812           0          30.84  万科A                 28.04
SZSE.000002          0.01  2038-01-01 00:00:00  SZSE        1991-01-
29 00:00:00          1              0          25.24             1
wkA                0   2017-09-19  00:00:00   000002              0
1
   7.44538           0          27.24  国农科技              24.76
SZSE.000004          0.01  2038-01-01 00:00:00  SZSE        1991-01-
14 00:00:00          1              0          22.28             1
gnkj               0   2017-09-19 00:00:00    000004              0
1
   ......
```

注意:

1.停牌时且股票发生除权除息, 涨停价和跌停价可能有误差

2.预上市股票以1900-01-01为虚拟发布日期, 未退市股票以2038-01-01为虚拟退市日期。

3.对于检索所需标的信息的4种手段 `symbols,exchanges,sec_types,names` ,若输入参数之间出现任何矛盾(换句话说, 所有的参数限制出满足要求的交集为空),则返回 `空list/空DataFrame`
例如 `get_instruments(exchanges='SZSE',sec_types=[4])` 返回的是空值

4.若查询字段包含无效字段, 返回的 `列表/DataFrame` 只包含有效字段数据

get_history_instruments - 查询交易标的历史数据

返回指定symbols的标的日频历史数据

函数原型:

```
get_history_instruments(symbols, fields=None, start_date=None,
end_date=None, df=False)
```

参数:

参数名	类型	说明
	str	

symbols	or list	`['symbol1', 'symbol2']` 这种列表格式, 是必填参数
fields	str or None	查询字段. 默认 `None` 表示所有
start_date	str or None	开始时间. (%Y-%m-%d 格式) 默认 `None` 表示当前时间
end_date	str or None	结束时间. (%Y-%m-%d 格式) 默认 `None` 表示当前时间
df	bool	是否返回 dataframe 格式, 默认False, 返回 `list[dict]`, 列表每项的dict的key值为参数指定的 `fields`

返回值:

key	类型	说明
symbol	str	标的代码
trade_date	datetime.date	交易日期
sec_level	int	1-正常,2-ST 股票,3-*ST 股票,4-股份转让,5-处于退市整理期的证券,6-上市开放基金LOF,7-交易型开放式指数基金(ETF),8-非交易型开放式基金(暂不交易,仅揭示基金净值及开放申购赎回业务),9-仅提供净值揭示服务的开放式基金;,10-仅在协议交易平台挂牌交易的证券,11-仅在固定收益平台挂牌交易的证券,12-风险警示产品,13-退市整理产品,99-其它
is_suspended	int	是否停牌. 1: 是, 0: 否
multiplier	float	合约乘数
margin_ratio	float	保证金比率
settle_price	float	结算价
position	int	持仓量
pre_close	float	昨收价
upper_limit	float	涨停价
lower_limit	float	跌停价
adj_factor	float	复权因子.基金跟股票才有

示例:

```
1. get_history_instruments(symbols='SZSE.000001,SZSE.000002',
   start_date='2017-09-19', end_date='2017-09-19')
```

输出:

输出:

```
  adj_factor    is_st    settle_price    upper_limit  symbol
pre_close    lower_limit    is_suspended    multiplier    position
trade_date             margin_ratio
------------  -------  --------------  -------------  -----------  -
----------  -------------  --------------  ------------  ----------
-------------------  --------------
     115.338        0               0          12.38  SZSE.000001
11.25          10.13               0             1           0
2017-09-19 00:00:00               1
     127.812        0               0          30.84  SZSE.000002
28.04          25.24               0             1           0
2017-09-19 00:00:00               1
```

注意:

1.停牌时且股票发生除权除息, 涨停价和跌停价可能有误差

2.为保护服务器, 单次查询最多返回 **3300** 条记录

3.对每个标的, 数据根据参数 `trade_date` 的升序进行排序

4.start_date和end_date中月,日均可以只输入个位数,例: `'2010-7-8'` 或 `'2017-7-30'`

5.若查询字段中出现非指定字符串, 则程序直接报错

6.若输入包含无效标的代码, 则返回的 `list/DataFrame` 只包含有效标的代码对应的数据

get_instrumentinfos - 查询交易标的基本信息

获取到交易标的基本信息, 与时间无关.

函数原型:

```
get_instrumentinfos(symbols=None, exchanges=None, sec_types=None,
names=None, fields=None, df=False)
```

参数:

参数名	类型	说明
symbols	str or list or None	标的代码, 多个代码可用 `,` (英文逗号)分割,也支持 `['symbol1', 'symbol2']` 这种列表格式, 默认 `None` 表示所有

exchanges	str or list or None	多个交易所代码可用 `,`(英文逗号)分割, 也支持 `['exchange1', 'exchange2']` 这种列表格式, 默认 `None` 表示所有
sec_types	list	指定类别, 默认所有, 其他类型见sec_type 类型
names	str or None	查询代码, 默认 `None` 表示所有
fields	str or None	查询字段 默认 `None` 表示所有
df	bool	是否返回dataframe 格式, 默认False, 返回字典格式, 返回 `list[dict]` , 列表每项的dict的key值为参数指定的 `fields`

返回值:

key	类型	说明
symbol	str	标的代码
sec_type	int	1: 股票, 2: 基金, 3: 指数, 4: 期货, 5: 期权, 10: 虚拟合约
exchange	str	见交易市场代码.
sec_id	str	代码
sec_name	str	名称
sec_abbr	str	拼音简称
price_tick	float	最小变动单位
listed_date	datetime.date	上市日期
delisted_date	datetime.date	退市日期

示例:

```
get_instrumentinfos(symbols=['SHSE.000001','SHSE.000002'],df=True)
```

输出:

```
sec_name    symbol         price_tick  delisted_date
sec_type  sec_abbr      sec_id  listed_date          exchange
----------  -----------  ------------  -------------------  --------
--  ----------  --------  -------------------  ----------
上证指数      SHSE.000001             0  2038-01-01 00:00:00           3
szzs          000001    1991-07-15 00:00:00  SHSE
A股指数       SHSE.000002             0  2038-01-01 00:00:00
3  Agzs          000002    1992-02-21 00:00:00  SHSE
```

注意:

1.对于检索所需标的信息的4种手段 `symbols,exchanges,sec_types,names` ,若输入参数之间出现任何矛盾(换句话说,所有的参数限制出满足要求的交集为空),则返回 `空list/空DataFrame`
例如 `get_instrumentinfos(exchanges='SZSE',sec_types=[4])` 返回的是空值

2.若查询字段包含无效字段,返回的 `列表/DataFrame` 只包含有效字段数据

get_history_constituents - 查询指数成份股的历史数据

函数原型:

```
1. get_history_constituents(index, start_date=None, end_date=None)
```

参数:

参数名	类型	说明
index	str	指数代码
start_date	str or datetime.datetime or None	开始时间 (%Y-%m-%d 格式) 默认 `None` 表示当前日期
end_date	str or datetime.datetime or None	结束时间 (%Y-%m-%d 格式) 默认 `None` 表示当前日期

返回值:

`list[constituent]`
`constituent` 为 dict,包含key值 `constituents` 和 `trade_date` :

参数名	类型	说明
constituents	dict	股票代码作为key, 所占权重作为value的键值对
trade_date	datetime.datetime	交易日期

示例:

```
1. get_history_constituents(index='SHSE.000001', start_date='2017-07-
   10')
```

输出:

```
constituents
trade_date
------------------------------------------------------------------------
---------------  -------------------
{u'SHSE.600527': 0.009999999776482582, u'SHSE.600461':
0.019999999552965164,···}     2017-07-31 00:00:00
{u'SHSE.603966': 0.009999999776482582, u'SHSE.603960':
0.009999999776482582,···}     2017-08-31 00:00:00
···
```

注意:

1.函数返回的数据每月发布一次，故返回的数据是月频数据， `trade_date` 为各月最后一天

2.在该函数中，index参数只支持输入一个指数代码，若代码输入错误或以 `'index1,inedx2'` 方式输入多个代码，函数返回 `空list`

3.start_date和end_date中月,日均可以只输入个位数,例: `'2010-7-8'` 或 `'2017-7-30'` ,但若对应位置为零，则 `0` 不可被省略

get_constituents - 查询指数最新成份股

函数原型:

```
get_constituents(index, fields=None, df=False)
```

参数:

参数名	类型	说明
index	str	指数代码
fields	str or None	若要有返回权重字段, 可以设置为 ‘symbol, weight’
df	bool	是否返回dataframe 格式, 默认False, 返回list[dict]

返回值:

df = False

1. 参数 fields 为 None时, 返回 `list[str]` , 成分股列表
2. 参数 fields 指定为 `'symbol, weight'` , 返回 `list[dict]` , dict包含key值:

参数名	类型	说明

symbol	str	股票symbol
weight	float	权重

df = True

返回 `DataFrame`

示例**1:**

```
get_constituents(index='SHSE.000001', fields='symbol, weight')
```

输出:

```
symbol          weight
-----------  --------
SHSE.603966       0.01
SHSE.603960       0.01
...
```

示例**2:**
只输出 `symbol` 列表

```
get_constituents(index='SHSE.000001')
```

输出:

```
[u'SHSE.603966', u'SHSE.603960', u'SHSE.603218',···]
```

注意:

1.在该函数中，index参数只支持输入一个指数代码，若代码输入错误或以 `'index1,inedx2'` 方式输入多个代码，函数返回 `空list/空DataFrame`

get_industry - 查询行业股票列表

函数原型:

```
get_industry(code)
```

参数:

参数名	类型	说明

code	str	行业代码 不区分大小写

返回值:

`list` 返回指定行业的成份股symbol 列表

示例:

```
#返回所有以J6开头的行业代码对应的成分股(包括:J66,J67,J68,J69的成分股)
get_industry(code='j6')
```

输出:

```
[u'SHSE.600000', u'SHSE.600016', u'SHSE.600030',···]
```

注意:

1.在该函数中，code参数只支持输入一个行业代码，若代码输入错误或以 `'code1,code2'` 方式输入多个代码，函数返回 `空list`

get_concept - 查询概念板块股票列表

函数原型:

```
get_concept(code)
```

参数:

参数名	类型	说明
code	str	概念名称，该数据暂不支持

返回值:

`list` 返回指定概念的成份股symbol 列表

示例:

```
get_concept(code='新能源')
```

输出:

```
[u'SHSE.600000', u'SHSE.600016', u'SHSE.600030',···]
```

注意:

1.在该函数中，code参数只支持输入一个概念名称，若名称输入错误或以 `'code1,code2'` 方式输入多个代码，函数返回 `空list`

2. 数据优化，暂不支持使用

get_trading_dates - 查询交易日列表

函数原型:

```
1. get_trading_dates(exchange, start_date, end_date)
```

参数:

参数名	类型	说明
exchange	str	见交易市场代码
start_date	str	开始时间 (%Y-%m-%d 格式)
end_date	str	结束时间 (%Y-%m-%d 格式)

返回值:

交易日期字符串(%Y-%m-%d 格式)列表,

示例:

```
1. get_trading_dates(exchange='SZSE', start_date='2017-01-01',
   end_date='2017-01-30')
```

输出:

```
1. ['2017-01-03', '2017-01-04', '2017-01-05', '2017-01-06', ...]
```

注意:

1. `exchange` 参数仅支持输入单个交易所代码,若代码错误，返回 `空list`

2. `start_date` 和 `end_date` 中月,日均可以只输入个位数,
例: `'2010-7-8'` 或 `'2017-7-30'`

get_previous_trading_date - 返回指定日期的上一个交易日

函数原型:

```
1. get_previous_trading_date(exchange, date)
```

参数:

参数名	类型	说明
exchange	str	见交易市场代码
date	str	时间 (%Y-%m-%d 格式)

返回值:

`str` 返回指定日期的上一个交易日字符串(%Y-%m-%d 格式)

示例:

```
1. get_previous_trading_date(exchange='SZSE', date='2017-05-01')
```

输出:

```
1. '2017-04-28'
```

注意:

1. `exchange` 参数仅支持输入单个交易所代码,若代码错误，返回 `空list`

2. `date` 中月,日均可以只输入个位数,
例: `'2010-7-8'` 或 `'2017-7-30'`

get_next_trading_date - 返回指定日期的下一个交易日

函数原型:

```
1. get_next_trading_date(exchange, date)
```

参数:

参数名	类型	说明
exchange	str	见交易市场代码
date	str	时间 (%Y-%m-%d 格式)

返回值:

`str` 返回指定日期的下一个交易日字符串 (%Y-%m-%d 格式)

示例:

```
1. get_next_trading_date(exchange='SZSE', date='2017-05-01')
```

输出:

```
1. '2017-05-02'
```

注意:

1. `exchange` 参数仅支持输入单个交易所代码,若代码错误，返回 `空list`

2. `date` 中月,日均可以只输入个位数,
例: `'2010-7-8'` 或 `'2017-7-30'`

get_dividend - 查询分红送配

函数原型:

```
1. get_dividend(symbol, start_date, end_date=None)
```

参数:

参数名	类型	说明
symbol	str	标的代码, (必填)
start_date	str	开始时间 (%Y-%m-%d 格式)
end_date	str	结束时间 (%Y-%m-%d 格式)
df	bool	是否返回dataframe格式, 默认False, 返回返回 `list[dividend]`, dividend 为 dict

返回值:

keyv	value类型	说明
symbol	str	标的代码
cash_div	float	每股派现
allotment_ratio	float	每股配股比例
allotment_price	float	配股价

share_div_ratio	float	每股送股比例
share_trans_ratio	float	每股转增比例

示例:

```
get_dividend(symbol='SHSE.600000',start_date='2000-01-01',
end_date='2017-12-31')
```

输出:

```
  share_trans_ratio  symbol         cash_div    allotment_ratio
allotment_price    share_div_ratio
------------------  -----------  ----------  -----------------  ---
--------------  -----------------
                0    SHSE.600000        0.15                  0
0                  0
              0.5  SHSE.600000         0.2                  0
0                  0
                ...
```

注意:

1.在该函数中，symbol参数只支持输入一个标的代码，若代码输入错误或以 `'symbol1,symbol2'` 方式输入多个代码，函数返回 `空list/空DataFrame`

2. `start_date` 和 `end_date` 中月,日均可以只输入个位数,例: `'2010-7-8'` 或 `'2017-7-30'`

get_continuous_contracts - 获取连续合约

函数原型:

```
get_continuous_contracts(csymbol, start_date=None, end_date=None)
```

参数:

参数名	类型	说明
csymbol	str	查询代码 比如获取主力合约，输入SHFE.AG;获取连续合约，输入SHFE.AG01
start_date	str	开始时间 (%Y-%m-%d 格式)
end_date	str	结束时间 (%Y-%m-%d 格式)

返回值:

返回 `list[dict]` , dict包含key值:

参数名	类型	说明
symbol	str	真实合约symbol
trade_date	datetime.datetime	交易日期

示例:

```
get_continuous_contracts(csymbol='SHFE.AG', start_date='2017-07-01',
end_date='2017-08-01')
```

输出:

```
symbol        trade_date
------------  -------------------
SHFE.ag1712   2017-07-01 00:00:00
SHFE.ag1712   2017-07-02 00:00:00
```

注意:

1.在该函数中，csymbol参数只支持输入一个标的代码，若代码输入错误或以 `'csymbol1,csymbol2'` 方式输入多个代码，函数返回 `空list`

2. `start_date` 和 `end_date` 中月,日均可以只输入个位数,例: `'2017-7-1'` 或 `'2017-8-1'`

交易函数

order_volume - 按指定量委托

函数原型：

```
order_volume(symbol, volume, side, order_type,position_effect,
price=0,order_duration=OrderDuration_Unknown,
order_qualifier=OrderQualifier_Unknown,account='')
```

参数：

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
side	int	参见订单委托方向
order_type	int	参见订单委托类型
position_effect	int	参见开平仓类型
price	float	价格
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
account	account id or account name or None	帐户

示例：

```
data = order_volume(symbol='SHSE.600000', volume=10000,
side=OrderSide_Buy, order_type=OrderType_Limit,
position_effect=PositionEffect_Open, price=11)
```

返回:

```
 status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
-------- -------- ------------ ------------------ -------------
-- ----------- --------------- --------- ------- ------ -----
------------ --------------- --------------- --------------- ---
---------- ------------- ------- ------------- -------------- -
-----------
       3     10000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000              0.11       0.11   110000       1
1            10000           110000            10000              1
11       11  strategy_id            110000             1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2.若下单数量输入有误，终端会拒绝此单，并显示 `委托量不正确` 。股票买入最小单位为 `100` ，卖出最小单位为 `1` ,如存在不足100股的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5.输入无效参数报 `NameError` 错误，缺少参数报 `TypeError` 错误。

order_value - 按指定价值委托

函数原型:

```
order_value(symbol, value, side,order_type, position_effect,
```

```
    price=0, order_duration=OrderDuration_Unknown,
    order_qualifier=OrderQualifier_Unknown,account='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
value	int	股票价值
side	int	参见订单委托方向
order_type	int	参见 订单委托类型
position_effect	int	参见 开平仓类型
price	float	价格
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
account	account id or account name or None	帐户

示例:

下限价单，以11元每股的价格买入价值为100000的SHSE.600000,根据volume = value / price,计算并取整得到volume = 9000

```
order_value(symbol='SHSE.600000', value=100000, price=11,
side=OrderSide_Buy, order_type=OrderType_Limit,
position_effect=PositionEffect_Open)
```

返回:

```
 status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  ------------------  -------------
--  -----------  ----------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
        3      9000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000             0.099        0.1   100000       1
1             9000            99000             9000              2
```

```
    11        11  strategy_id              99000              1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2.根据指定价值计算购买标的数量，即 `value/price` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单,计算使用价格为最新一个tick中的最新价，price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5.输入无效参数报NameError错误，缺少参数报TypeError错误。

order_percent - 按总资产指定比例委托

函数原型:

```
1. order_percent(symbol, percent, side,order_type, position_effect,
   price=0, order_duration=OrderDuration_Unknown,
   order_qualifier=OrderQualifier_Unknown, account='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
percent	double	委托占总资产比例
side	int	参见 订单委托方向
order_type	int	参见 订单委托类型
position_effect	int	参见 开平仓类型
price	float	价格
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
account	account id or account name or None	帐户

示例:

当前总资产为1000000。下限价单，以11元每股的价格买入SHSE.600000,期望买入比例占总资产的

10%，根据volume = nav * precent / price 计算取整得出volume = 9000

```
order_percent(symbol='SHSE.600000', percent=0.1, side=OrderSide_Buy,
order_type=OrderType_Limit, position_effect=PositionEffect_Open,
price=11)
```

返回:

```
   status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  ------------------  -------------
--  -----------  ---------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
       3      9000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000             0.099        0.1   100000       1
1             9000            99000             9000              3
11       11  strategy_id             99000             1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2.根据指定比例计算购买标的数量,即 `(nav*precent)/price` ,股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端会拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` 。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，计算使用价格为最新一个tick中的最新价，price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5.输入无效参数报NameError错误，缺少参数报TypeError错误。

order_target_volume - 调仓到目标持仓量

函数原型:

```
order_target_volume(symbol, volume, position_side, order_type,
price=0, order_duration=OrderDuration_Unknown,
order_qualifier=OrderQualifier_Unknown, account='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	期望的最终数量
position_side	int	参见 持仓方向
order_type	int	参见 订单类型
price	float	价格
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
account	account id or account name or None	帐户

示例:

当前SHSE.600000多方向持仓量为0，期望持仓量为10000，下单量为期望持仓量 - 当前持仓量 = 10000

```
order_target_volume(symbol='SHSE.600000', volume=10000,
position_side=PositionSide_Long, order_type=OrderType_Limit,
price=13)
```

返回:

```
   status    volume  account_id    created_at
position_side  symbol       target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  ------------------  -------------
--  -----------  ----------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
       3      10000  strategy_id   2017-07-06 13:32:01
1  SHSE.600000             0.13      0.13   130000       1
1             10000           130000             10000             3
13       13  strategy_id             130000             1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，订单会被拒绝， `终端无显示，无回报` 。回测模式可参看order_reject_reason。

2.根据目标数量计算下单数量，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则订单会被拒绝， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` ,上期所昨仓不能平掉。应研究需要，股票也支持卖空操作。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，使用价格为最新一个tick中的最新价，price参数失效。若OrderTpye_Limit限价单价格错误，终端拒绝此单，显示委托价格错误。 `回测模式下对价格无限制` 。

5.输入无效参数报NameError错误，缺少参数报Typeerror错误。

order_target_value - 调仓到目标持仓额

函数原型:

```
order_target_value(symbol, value, position_side, order_type,
price=0, order_duration=OrderDuration_Unknown,
order_qualifier=OrderQualifier_Unknown, account='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
value	int	期望的股票最终价值
position_side	int	参见持仓方向
order_type	int	参见订单类型
price	float	价格
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
account	account id or account name or None	帐户

示例:

当前SHSE.600000多方向当前持仓量为0，目标持有价值为100000的该股票，根据value / price计算取整得出目标持仓量volume为9000，目标持仓量 - 当前持仓量 = 下单量为9000

```
order_target_value(symbol='SHSE.600000', value=100000,
position_side=PositionSide_Long, order_type=OrderType_Limit,
price=11)
```

返回:

```
   status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  ------------------  -------------
--  -----------  ---------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
       3      9000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000             0.099        0.1   100000       1
1             9000            99000             9000              3
11       11  strategy_id             99000             1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，订单会被拒绝， `终端无显示，无回报` 。回测模式可参看order_reject_reason。

2.根据目标价值计算下单数量，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则本地拒绝此单， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端拒绝此单，显示 `仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` ，目前不可修改。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，计算使用价格为最新一个tick中的最新价，price参数失效。若OrderTpye_Limit限价单价格错误，终端拒绝此单，显示委托价格错误。 `回测模式下对价格无限制` 。

5.输入无效参数报NameError错误，缺少参数报Typeerror错误。

order_target_percent - 调仓到目标持仓比例（总资产

的比例）

函数原型：

```
order_target_percent(symbol, percent, position_side, order_type,
price=0, order_duration=OrderDuration_Unknown,
order_qualifier=OrderQualifier_Unknown, account='')
```

参数：

参数名	类型	说明
symbol	str	标的代码
percent	double	期望的最终占总资产比例
position_side	int	参见持仓方向
order_type	int	参见订单类型
order_duration	int	参见 委托时间属性
order_qualifier	int	参见 委托成交属性
price	float	价格
account	account id or account name or None	帐户

示例：

当前总资产价值为1000000，目标为以11元每股的价格买入SHSE.600000的价值占总资产的10%，根据volume = nav * percent / price 计算取整得出应持有9000股。当前该股持仓量为零，因此买入量为9000

```
order_target_percent(symbol='SHSE.600000', percent=0.1,
position_side=PositionSide_Long, order_type=OrderType_Limit,
price=11)
```

返回：

```
   status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  -------------------  -------------
--  -----------  ----------------  ---------  -------  ------  -----
```

```
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
3.        3       9000   strategy_id    2017-07-06 07:00:01
1   SHSE.600000              0.099         0.1    100000        1
1              9000             99000              9000              3
11        11   strategy_id              99000              1
```

注意:

1.仅支持一个标的代码，若交易代码输入有误，订单会被拒绝， `终端无显示，无回报` 。回测模式可参看order_reject_reason。

2.根据目标比例计算下单数量，为占 `总资产(nav)` 比例，系统判断开平仓类型。若下单数量有误，终端拒绝此单，并显示 `委托量不正确` 。若实际需要买入数量为0，则本地拒绝此单， `终端无显示，无回报` 。股票买卖最小单位为 `100` ，不足100部分 `向下取整` ，如存在不足100的持仓一次性卖出;期货买卖最小单位为 `1` ， `向下取整` 。

3.若仓位不足，终端拒绝此单， `显示仓位不足` 。平仓时股票默认 `平昨仓` ，期货默认 `平今仓` ，目前不可修改。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，计算使用价格为最新一个tick中的最新价，price参数失效。若OrderTpye_Limit限价单价格错误，终端拒绝此单，显示委托价格错误。 `回测模式下对价格无限制` 。

6.输入无效参数报NameError错误，缺少参数报Typeerror错误。

order_batch - 批量委托接口

函数原型:

```
order_batch(orders, combine=False, account='')
```

参数:

参数名	类型	说明
orders	list[order]	委托对象列表，其中委托至少包含交易接口的必选参数，参见委托
combine	bool	是否是组合单，默认不是
account	account id or account name or None	帐户

示例:

```
    order_1 = {'symbol': 'SHSE.600000', 'volume': 100, 'price': 11, 'side': 1,
               'order_type': 2, 'position_effect':1}
    order_2 = {'symbol': 'SHSE.600004', 'volume': 100, 'price': 11, 'side': 1,
               'order_type': 2, 'position_effect':1}
    orders = [order_1, order_2]
    batch_orders = order_batch(orders, combine=True)
    for order in batch_orders:
        print(order)
```

返回:

```
   status    volume  account_id    created_at position_side  symbol         target_percent    percent    value side    position_effect    target_volume    filled_amount filled_volume    order_style    filled_vwap    price  strategy_id target_value    order_type
-------- -------- ------------ ------------------- --------------- ----------- ---------------- --------- ------- ------ ----- ------------ --------------- --------------- --------------- ------------- ------------- ------- ------------- -------------- ------------
       3      9000  strategy_id   2017-07-06 07:00:01 1  SHSE.600000             0.099         0.1    100000       1 1          9000           99000           9000              3 11        11  strategy_id            99000            1
```

注意:

1.每个order的symbol仅支持一个标的代码，若交易代码输入有误，终端会拒绝此单，并显示 `委托代码不正确` 。

2.若下单数量输入有误，终端会拒绝此单，并显示 `委托量不正确` 。 `下单数量严格按照指定数量下单` ，需注意股票买入最小单位为100。

3.若仓位不足，终端会拒绝此单， `显示仓位不足` 。应研究需要， `股票也支持卖空操作` 。

4.Order_type优先级高于price,若指定OrderTpye_Market下市价单，则price参数失效。若OrderTpye_Limit限价单，仿真模式价格错误，终端拒绝此单，显示委托价格错误， `回测模式下对价格无限制` 。

5.输入无效参数报NameError错误，缺少参数不报错，可能会出现下单被拒。

order_cancel - 撤销委托

函数原型:

```
order_cancel(wait_cancel_orders)
```

参数:

参数名	类型	说明
wait_cancel_orders	list[str]	委托对象列表 or 单独委托对象, 至少包含 cl_ord_id, 参见委托

示例:

```
order_1 = {'symbol': 'SHSE.600000', 'cl_ord_id': 'cl_ord_id_1', 'price': 11, 'side': 1, 'order_type':1 }
order_2 = {'symbol': 'SHSE.600004', 'cl_ord_id': 'cl_ord_id_2', 'price': 11, 'side': 1, 'order_type':1 }
orders = [order_1, order_2]
order_cancel(wait_cancel_orders=orders)
```

order_cancel_all - 撤销所有委托

函数原型:

```
order_cancel_all()
```

示例:

```
order_cancel_all()
```

order_close_all - 平当前所有可平持仓

函数原型:

```
order_close_all()
```

示例:

```
order_close_all()
```

get_unfinished_orders - 查询日内全部未结委托

函数原型:

```
get_unfinished_orders()
```

返回值:

类型	说明
list[order]	委托对象列表，参见委托

示例:

```
get_unfinished_orders()
```

返回:

```
   status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
-------- -------- ------------ ------------------- -------------
--  -----------  ---------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
       3      9000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000             0.099        0.1   100000       1
1             9000            99000             9000              3
11       11  strategy_id             99000             1
```

get_orders - 查询日内全部委托

函数原型:

```
get_orders()
```

返回值:

类型	说明
list[order]	委托对象列表，参见委托

示例：

```
get_orders()
```

返回：

```
   status    volume  account_id    created_at
position_side  symbol         target_percent    percent    value
side    position_effect    target_volume    filled_amount
filled_volume    order_style    filled_vwap    price  strategy_id
target_value    order_type
--------  --------  ------------  ------------------  -------------
--  -----------  ---------------  ---------  -------  ------  -----
------------  ---------------  ---------------  ---------------  ---
----------  -------------  -------  -------------  --------------  -
-----------
        3      9000  strategy_id   2017-07-06 07:00:01
1  SHSE.600000             0.099        0.1   100000       1
1             9000            99000             9000              3
11       11  strategy_id             99000             1
```

get_execution_reports - 查询日内全部执行回报

函数原型：

```
get_execution_reports()
```

返回值：

类型	说明
list[execrpt]	回报对象列表， 参见成交回报

示例：

```
get_execution_reports()
```

交易事件

on_order_status - 委托状态更新事件

响应委托状态更新事情，下单后及委托状态更新时被触发。

函数原型：

```
on_order_status(context, order)
```

参数：

参数名	类型	说明
context	context	上下文
order	order	委托

示例：

```
def on_order_status(context, order):
    print(order)
```

输出：

```
  status    ord_rej_reason  account_id      position_side    volume
symbol         target_percent    percent  updated_at
value    side    position_effect    target_volume    price
order_style  created_at           ord_rej_reason_detail
strategy_id      target_value    order_type
--------  ---------------  ------------  ---------------  --------
-----------  ----------------  ---------  -------------------  -----
--  ------  -----------------  --------------  -------  -----------
--  -------------------  ----------------------  -------------  ---
-----------  ------------
    8                 3       strategy_id             1     18229
SZSE.002528        -0.0999982        0.1  2017-07-27 07:00:01
100261       2                 2           -18229      5.5
3   2017-07-27 07:00:01    仓位不足 可用=16479       strategy_id
```

```
    -100260            2
```

on_execution_report - 委托执行回报事件

响应委托被执行事件，委托成交后被触发。

函数原型：

```
1. on_execution_report(context, execrpt)
```

参数：

参数名	类型	说明
context	context	上下文
execrpt	execrpt	回报

示例：

```
1. def on_execution_report(context, execrpt):
2.     print(execrpt)
```

on_account_status - 交易账户状态更新事件

函数原型：

```
1. on_account_status(context, account)
```

参数：

参数名	类型	说明
context	context	上下文
account	object, 包含account_id(账户id), account_name(账户名),ConnectionStatus(账户状态)	交易账户状态对象，仅响应 已连接，已登录，已断开 和 错误 事件。

动态参数

- add_parameter - 增加动态参数
- set_parameter - 修改已经添加过的动态参数
- on_parameter - 动态参数修改事件推送
- context.parameters - 获取所有动态参数

动态参数仅在实盘模式下生效， 可在终端设置和修改。

add_parameter - 增加动态参数

函数原型：

```
1. add_parameter(key, value, min=0, max=0, name='', intro='', group='', readonly=False)
```

参数：

参数名	类型	说明
key	str	参数的键
value	double	参数的值
min	double	最小值
max	double	最大值
name	str	参数名称
intro	str	参数说明
group	str	参数的组
readonly	bool	是否为只读参数

返回值：

`None`

示例：

```
1. add_parameter(key='signal', value=1)
```

set_parameter - 修改已经添加过的动态参数

函数原型：

```
set_parameter(key, value, min=0, max=0, name='', intro='', group='', readonly=False)
```

参数:

参数名	类型	说明
key	str	参数的键
value	double	参数的值
min	double	最小值
max	double	最大值
name	str	参数名称
intro	str	参数说明
group	str	参数的组
readonly	bool	是否为只读参数

返回值:

`None`

示例:

```
set_parameter(key='signal', value=0)
```

on_parameter - 动态参数修改事件推送

函数原型:

```
on_parameter(context, parameter)
```

参数:

参数名	类型	说明
context	context	上下文
parameter	dict	当前被推送的动态参数对象

示例:

```
def on_parameter(context, parameter):
    print(parameter)
```

输出:

```
group              max     value     min      readonly     intro                key
---------------  -----  -------  -------  ----------  ----------  ---
strategy_group       0        1        0     False       下单信号              signal
```

context.parameters - 获取所有动态参数

返回数据类型为字典, key为动态参数的key, 值为动态参数对象

其他函数

set_token - 设置token

用户有时只需要提取数据，set_token后就可以直接调用数据函数，无需编写策略结构。如果token不合法，访问需要身份验证的函数会抛出异常。

token位置参见终端-系统设置界面-密钥管理（token）

函数原型：

```
set_token(token)
```

参数：

参数名	类型	说明
token	str	身份标识

返回值：

`None`

示例：

```
set_token('xxxxx')
history_data = history(symbol='SHSE.000300', frequency='1d',
start_time='2010-07-28',  end_time='2017-07-30', df=True)
```

注意：
token输入错误不会报错，但获取数据时为空值。

log - 日志函数

函数原型：

```
log(level, msg, source)
```

参数:

参数名	类型	说明
level	str	日志级别 `debug` , `info` , `warning` , `error`
msg	str	信息
source	str	来源

返回值:

`None`

示例:

```
log(level='info', msg='平安银行信号触发', source='strategy')
```

注意:

1.log函数仅支持实时模式，输出到终端策略日志处。

2.level输入无效参数不会报错，终端日志无显示。

3.参数类型报NameError错误,缺少参数报TypeError错误。

get_strerror - 查询错误码的错误描述信息

函数原型:

```
get_strerror(error_code)
```

参数:

参数名	类型	说明
error_code	int	错误码

全部 错误码详细信息

返回值:

错误原因描述信息字符串

示例:

```
err = get_strerror(error_code=1010)
```

```
print(err)
```

注意：

error_code值输入错误无报错，返回值为空。

get_version - 查询api版本

函数原型：

```
get_version()
```

返回值：

字符串 当前API版本号

示例：

```
version = get_version()
```

其他事件

on_backtest_finished - 回测结束事件

函数原型:

```
on_backtest_finished(context, indicator)
```

参数:

参数名	类型	说明
context	context	上下文
indicator	indicator	绩效指标

示例:

```
def on_backtest_finished(context, indicator):
    print(indicator)
```

返回:

```
account_id    close_count    lose_count      max_drawdown
open_count    pnl_ratio_annual     pnl_ratio        risk_ratio
sharp_ratio    win_count     win_ratio
------------  ------------  --------------  ----------------  ------
-------  ------------------  ------------------ ----------------- ---
----------- -----------  --------------
strategy_id        29              19        0.0246530717172
28         0.123621715428      0.23315168108    0.0404759897319
2.58220641506      10       0.344827586207
```

on_error - 错误事件

函数原型：

```
on_error(context, code, info)
```

参数：

参数名	类型	说明
context	context	上下文
code	int	错误码
info	str	错误信息

示例：

```
def on_error(context, code, info):
    stop()
```

on_market_data_connected - 实时行情网络连接成功事件

函数原型：

```
on_market_data_connected(context)
```

参数：

参数名	类型	说明
context	context	上下文

示例：

```
def on_market_data_connected(context):
    print ('链接成功')
```

on_trade_data_connected - 交易通道网络连接成功事件

函数原型：

```
on_trade_data_connected(context)
```

参数:

参数名	类型	说明
context	context	上下文

示例:

```
def on_trade_data_connected(context):
    print ('链接成功')
```

on_market_data_disconnected - 实时行情网络连接断开事件

函数原型:

```
on_market_data_disconnected(context)
```

参数:

参数名	类型	说明
context	context	上下文

示例:

```
def on_market_data_disconnected(context):
    print ('链接失败')
```

on_trade_data_disconnected - 交易通道网络连接断开事件

函数原型:

```
on_trade_data_disconnected(context)
```

参数:

参数名	类型	说明
context	context	上下文

示例:

```
def on_trade_data_disconnected(context):
    print ('链接失败')
```

枚举常量

OrderStatus - 委托状态

```
OrderStatus_Unknown = 0
OrderStatus_New = 1                    # 已报
OrderStatus_PartiallyFilled = 2        # 部成
OrderStatus_Filled = 3                 # 已成
OrderStatus_Canceled = 5               # 已撤
OrderStatus_PendingCancel = 6          # 待撤
OrderStatus_Rejected = 8               # 已拒绝
OrderStatus_Suspended = 9              # 挂起
OrderStatus_PendingNew = 10            # 待报
OrderStatus_Expired = 12               # 已过期
```

OrderSide - 委托方向

```
OrderSide_Unknown = 0
OrderSide_Buy = 1              # 买入
OrderSide_Sell = 2             # 卖出
```

OrderType - 委托类型

```
OrderType_Unknown = 0
```

```
OrderType_Limit = 1             # 限价委托
OrderType_Market = 2            # 市价委托
OrderType_Stop = 3              # 止损止盈委托
```

OrderDuration - 委托时间属性

仅在实盘模式生效，具体执行模式请参考交易所给出的定义

```
OrderDuration_Unknown = 0
OrderDuration_FAK = 1           # 即时成交剩余撤销(fill and kill)
OrderDuration_FOK = 2           # 即时全额成交或撤销(fill or kill)
OrderDuration_GFD = 3           # 当日有效(good for day)
OrderDuration_GFS = 4           # 本节有效(good for section)
OrderDuration_GTD = 5           # 指定日期前有效(goodltilldate)
OrderDuration_GTC = 6           # 撤销前有效(goodtillcancel)
OrderDuration_GFA = 7           # 集合竞价前有效(good for auction)
```

OrderQualifier - 委托成交属性

仅在实盘模式生效，具体执行模式请参考交易所给出的定义

```
OrderQualifier_Unknown = 0
OrderQualifier_BOC     = 1          # 对方最优价格(best of
counterparty)
OrderQualifier_BOP     = 2          # 己方最优价格(best of party)
OrderQualifier_B5TC    = 3          # 最优五档剩余撤销(best 5 then
cancel)
OrderQualifier_B5TL    = 4          # 最优五档剩余转限价(best 5 then
limit)
```

ExecType - 执行回报类型

```
ExecType_Unknown = 0
ExecType_New = 1                    # 已报
ExecType_Canceled = 5               # 已撤销
ExecType_PendingCancel = 6          # 待撤销
ExecType_Rejected = 8               # 已拒绝
ExecType_Suspended = 9              # 挂起
ExecType_PendingNew = 10            # 待报
ExecType_Expired = 12               # 过期
```

```
ExecType_Trade = 15                    # 成交
ExecType_OrderStatus = 18              # 委托状态
ExecType_CancelRejected = 19           # 撤单被拒绝
```

PositionEffect - 开平仓类型

```
PositionEffect_Unknown = 0
PositionEffect_Open = 1                    # 开仓
PositionEffect_Close = 2                   # 平仓, 具体语义取决于对应的交易所
PositionEffect_CloseToday = 3              # 平今仓
PositionEffect_CloseYesterday = 4          # 平昨仓
```

PositionSide - 持仓方向

```
PositionSide_Unknown = 0
PositionSide_Long = 1              # 多方向
PositionSide_Short = 2             # 空方向
```

OrderRejectReason - 订单拒绝原因

```
OrderRejectReason_Unknown = 0                          # 未知原因
OrderRejectReason_RiskRuleCheckFailed = 1              # 不符合风控规则
OrderRejectReason_NoEnoughCash = 2                     # 资金不足
OrderRejectReason_NoEnoughPosition = 3                 # 仓位不足
OrderRejectReason_IllegalAccountId = 4                 # 非法账户ID
OrderRejectReason_IllegalStrategyId = 5                # 非法策略ID
OrderRejectReason_IllegalSymbol = 6                    # 非法交易标的
OrderRejectReason_IllegalVolume = 7                    # 非法委托量
OrderRejectReason_IllegalPrice = 8                     # 非法委托价
OrderRejectReason_AccountDisabled = 10                 # 交易账号被禁止交易
OrderRejectReason_AccountDisconnected = 11             # 交易账号未连接
OrderRejectReason_AccountLoggedout = 12                # 交易账号未登录
OrderRejectReason_NotInTradingSession = 13             # 非交易时段
OrderRejectReason_OrderTypeNotSupported = 14           # 委托类型不支持
OrderRejectReason_Throttle = 15                        # 流控限制
```

CancelOrderRejectReason - 取消订单拒绝原因

```
CancelOrderRejectReason_OrderFinalized = 101            # 委托已完成
CancelOrderRejectReason_UnknownOrder = 102              # 未知委托
CancelOrderRejectReason_BrokerOption = 103              # 柜台设置
CancelOrderRejectReason_AlreadyInPendingCancel = 104    # 委托撤销中
```

OrderStyle - 订单类型

```
OrderStyle_Unknown = 0
OrderStyle_Volume = 1                                   # 按指定量委托
OrderStyle_Value = 2                                    # 按指定价值委托
OrderStyle_Percent = 3                                  # 按指定比例委托
OrderStyle_TargetVolume = 4                             # 调仓到目标持仓量
OrderStyle_TargetValue = 5                              # 调仓到目标持仓额
OrderStyle_TargetPercent = 6                            # 调仓到目标持仓比例
```

CashPositionChangeReason - 仓位变更原因

```
CashPositionChangeReason_Unknown = 0
CashPositionChangeReason_Trade = 1              # 交易
CashPositionChangeReason_Inout = 2              # 出入金 / 出入持仓
```

SecType - 标的类别

```
SEC_TYPE_STOCK = 1                              # 股票
SEC_TYPE_FUND = 2                               # 基金
SEC_TYPE_INDEX = 3                              # 指数
SEC_TYPE_FUTURE = 4                             # 期货
SEC_TYPE_OPTION = 5                             # 期权
SEC_TYPE_CONFUTURE = 10                         # 虚拟合约
```

AccountStatus - 交易账户状态

```

State_UNKNOWN = 0;        //未知
```

```
State_CONNECTING = 1;     //连接中
State_CONNECTED = 2;      //已连接
State_LOGGEDIN = 3;       //已登录
State_DISCONNECTING = 4; //断开中
State_DISCONNECTED = 5;   //已断开
State_ERROR = 6;          //错误
```

错误码

错误码	描述
0	成功
1010	无法获取掘金服务器地址列表
1011	消息包解析错误
1012	网络消息包解析错误
1013	交易服务调用错误
1014	历史行情服务调用错误
1015	策略服务调用错误
1016	动态参数调用错误
1017	基本面数据服务调用错误
1018	回测服务调用错误
1019	交易网关服务调用错误
1020	无效的ACCOUNT_ID
1021	非法日期格式
1100	交易消息服务连接失败
1101	交易消息服务断开
1200	实时行情服务连接失败
1201	实时行情服务连接断开
1300	初始化回测失败，可能是终端未启动或无法连接到终端
1301	回测时间区间错误
1302	回测读取缓存数据错误
1303	回测写入缓存数据错误