目　录

- 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
- order_cancel - 撤销委托
- order_cancel_all - 撤销所有委托
- order_close_all - 平当前所有可平持仓
- get_unfinished_orders - 查询日内全部未结委托
- get_orders - 查询日内全部委托
- get_execution_reports - 查询日内全部执行回报

交易查询函数

- context.account().positions() - 查询当前账户全部持仓
- context.account().position(symbol, side) - 查询当前账户指定持仓
- context.account().cash - 查询当前账户资金

两融交易函数

算法交易函数

新股交易函数

基金交易函数

债券交易函数

交易事件

- on_order_status - 委托状态更新事件
- on_execution_report - 委托执行回报事件
- on_account_status - 交易账户状态更新事件

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

其他函数

- set_token - 设置token
- log - 日志函数
- get_strerror - 查询错误码的错误描述信息
- get_version - 查询api版本

其他事件

- on_backtest_finished - 回测结束事件
- on_error - 错误事件
- on_market_data_connected - 实时行情网络连接成功事件
- on_trade_data_connected - 交易通道网络连接成功事件
- on_market_data_disconnected - 实时行情网络连接断开事件
- on_trade_data_disconnected - 交易通道网络连接断开事件

枚举常量

- OrderStatus - 委托状态
- OrderSide - 委托方向
- OrderType - 委托类型
- OrderDuration - 委托时间属性
- OrderQualifier - 委托成交属性
- OrderBusiness - 委托业务类型
- ExecType - 执行回报类型
- PositionEffect - 开平仓类型
- PositionSide - 持仓方向
- OrderRejectReason - 订单拒绝原因
- CancelOrderRejectReason - 取消订单拒绝原因
- OrderStyle - 委托风格
- CashPositionChangeReason - 仓位变更原因
- SecType - 标的类别
- AccountStatus - 交易账户状态
- PositionSrc - 头寸来源(仅适用融券融券)
- AlgoOrderStatus 算法单状态,暂停/恢复算法单时有效

错误码

快速开始

常见的策略结构主要包括3类，如下图所示。

定时任务

```
# 第一步：初始化，配置定时任务
def init(context):
    schedule(schedule_func,
             date_rule, time_rule)

# 第二步：定时任务函数
def algo(context):
    # 在这里输入需要定时执行的内容

# 第三步：执行策略
if __name__ == '__main__':
    run(strategy_id='',
        filename='',
        mode=MODE_UNKNOWN,
        token='')
```

示例策略：
小市值、alpha对冲、多因子选股、行业轮动

事件驱动任务

```
# 第一步：初始化，数据订阅
def init(context):
    subscribe
(symbols,frequency,count)

# 第二步：事件驱动函数下单
# 可以用on_xxx()类函数
def on_bar(context,bar):
    # 在这里定义下单逻辑

# 第三步：执行策略
if __name__ == '__main__':
    run(strategy_id='',
       filename='',
       mode=MODE_UNKNOWN,
       token='')
```

示例策略：
双均线、布林带均值回归、网格交易
指数增强、跨品种套利、跨期套利、做市商交易、日内回转交易、海龟交易法、机器学习

定时+事件驱动任务

```
# 第一步：初始化函数，订阅数据，配置定时任务
def init(context):
    # 数据订阅
    subscribe(symbols,frequency,count)
    # 定时任务配置
    schedule(schedule_func,date_rule,
time_rule)

# 第二步：定时任务函数
def algo(context):
    # 在这里输入需要定时执行的内容

# 第三步：事件驱动函数下单
def on_bar(context,bars):
    # 在这里定义事件驱动内容

# 第四步：执行策略
if __name__ == '__main__':
    run(strategy_id='',
        filename='',
        mode=MODE_UNKNOWN,
        token='')
```

示例策略：
Dual Thrust、R-Breaker、菲阿里四价

用户可以根据策略需求选择相应的策略结构，具体可以参考经典策略。

定时任务示例

以下代码的内容是：在每个交易日的14：50：00 市价买入200股浦发银行股票：

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

```

```
def init(context):
    # 每天14:50 定时执行algo任务,
    # algo执行定时任务函数，只能传context参数
    # date_rule执行频率，目前暂时支持1d、1w、1m，其中1w、1m仅用于回测，实时模式1d以上的频率，需要在algo判断日期
    # time_rule执行时间， 注意多个定时任务设置同一个时间点，前面的定时任务会被后面的覆盖
    schedule(schedule_func=algo, date_rule='1d', time_rule='14:50:00')


def algo(context):
    # 以市价购买200股浦发银行股票， price在市价类型不生效
    order_volume(symbol='SHSE.600000', volume=200, side=OrderSide_Buy,
                 order_type=OrderType_Market, position_effect=PositionEffect_Open, price=0)


# 查看最终的回测结果
def on_backtest_finished(context, indicator):
    print(indicator)


if __name__ == '__main__':
    '''
        strategy_id策略ID, 由系统生成
        filename文件名, 请与本文件名保持一致
        mode运行模式, 实时模式:MODE_LIVE回测模式:MODE_BACKTEST
        token绑定计算机的ID, 可在系统设置-密钥管理中生成
        backtest_start_time回测开始时间
        backtest_end_time回测结束时间
        backtest_adjust股票复权方式, 不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
        backtest_initial_cash回测初始资金
        backtest_commission_ratio回测佣金比例
        backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_BACKTEST,
        token='token_id',
        backtest_start_time='2020-11-01 08:00:00',
        backtest_end_time='2020-11-10 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

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

数据事件驱动示例

在用subscribe()接口订阅标的后，后台会返回tick数据或bar数据。每产生一个或一组数据，就会自动触发on_tick()或on_bar()里面的内容执行。比如以下范例代码片段，订阅浦发银行频率为1天和60s的bar数据，每产生一次bar，就会自动触发on_bar()调用，打印获取的bar信息：

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


def init(context):
    # 订阅浦发银行, bar频率为一天和一分钟
    # 订阅订阅多个频率的数据, 可多次调用subscribe
    subscribe(symbols='SHSE.600000', frequency='1d')
    subscribe(symbols='SHSE.600000', frequency='60s')


def on_bar(context, bars):

    # 打印bar数据
    print(bars)


if __name__ == '__main__':
    '''
        strategy_id策略ID, 由系统生成
        filename文件名, 请与本文件名保持一致
        mode运行模式, 实时模式:MODE_LIVE回测模式:MODE_BACKTEST
        token绑定计算机的ID, 可在系统设置-密钥管理中生成
        backtest_start_time回测开始时间
        backtest_end_time回测结束时间
        backtest_adjust股票复权方式, 不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
        backtest_initial_cash回测初始资金
        backtest_commission_ratio回测佣金比例
        backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_BACKTEST,
        token='token_id',
        backtest_start_time='2020-11-01 08:00:00',
        backtest_end_time='2020-11-10 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

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示例

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和分钟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条分钟bar信息：

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


def init(context):
    # 订阅浦发银行，bar频率为一天和一分钟
    # 指定数据窗口大小为50
    # 订阅订阅多个频率的数据，可多次调用subscribe
    subscribe(symbols='SHSE.600000', frequency='1d', count=50)
    subscribe(symbols='SHSE.600000', frequency='60s', count=50)


def on_bar(context, bars):
    # context.data提取缓存的数据滑窗，可用于计算指标
    # 注意：context.data里的count要小于或者等于subscribe里的count
    data = context.data(symbol=bars[0]['symbol'], frequency='60s', count=50,
fields='close,bob')

    # 打印最后5条bar数据（最后一条是最新的bar）
    print(data.tail())


if __name__ == '__main__':
    '''
        strategy_id策略ID，由系统生成
        filename文件名，请与本文件名保持一致
        mode运行模式，实时模式:MODE_LIVE回测模式:MODE_BACKTEST
        token绑定计算机的ID，可在系统设置-密钥管理中生成
        backtest_start_time回测开始时间
        backtest_end_time回测结束时间
        backtest_adjust股票复权方式，不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
        backtest_initial_cash回测初始资金
        backtest_commission_ratio回测佣金比例
        backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_BACKTEST,
        token='token_id',
        backtest_start_time='2020-11-01 08:00:00',
        backtest_end_time='2020-11-10 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

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

选择回测模式/实时模式运行示例

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
from gm.api import *


def init(context):
    # 订阅浦发银行的tick
    subscribe(symbols='SHSE.600000', frequency='60s')


def on_bar(context, bars):
    # 打印当前获取的bar信息
    print(bars)

if __name__ == '__main__':
    # 在终端仿真交易和实盘交易的启动策略按钮默认是实时模式，运行回测默认是回测模式，在外部IDE里运行策略需要修改成对应的运行模式
    # mode=MODE_LIVE 实时模式，回测模式的相关参数不生效
    # mode=MODE_BACKTEST  回测模式

    '''
        strategy_id策略ID，由系统生成
        filename文件名，请与本文件名保持一致
        mode运行模式，实时模式:MODE_LIVE回测模式:MODE_BACKTEST
        token绑定计算机的ID，可在系统设置-密钥管理中生成
        backtest_start_time回测开始时间
        backtest_end_time回测结束时间
        backtest_adjust股票复权方式，不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
        backtest_initial_cash回测初始资金
        backtest_commission_ratio回测佣金比例
        backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_LIVE,
        token='token_id',
        backtest_start_time='2020-11-01 08:00:00',
        backtest_end_time='2020-11-10 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

整个策略需要三步：

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

提取数据研究示例

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
from gm.api import *


# 可以直接提取数据，掘金终端需要打开，接口取数是通过网络请求的方式，效率一般，行情数据可通过subscribe订阅方式
# 设置token， 查看已有token ID,在用户-密钥管理里获取
set_token('your token_id')

# 查询历史行情，采用定点复权的方式， adjust指定前复权，adjust_end_time指定复权时间点
data = history(symbol='SHSE.600000', frequency='1d', start_time='2020-01-01 09:00:00', end_time='2020-12-31 16:00:00',
               fields='open,high,low,close', adjust=ADJUST_PREV, adjust_end_time='2020-12-31', df=True)
print(data)
```

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示例

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_time)
    # 将信息按symbol,date作为key存入字典
    context.ins_dict = {(i.symbol, i.trade_date.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())])

```

```
16.
17. if __name__ == '__main__':
18.     '''
19.         strategy_id策略ID, 由系统生成
20.         filename文件名, 请与本文件名保持一致
21.         mode运行模式, 实时模式:MODE_LIVE回测模式:MODE_BACKTEST
22.         token绑定计算机的ID, 可在系统设置-密钥管理中生成
23.         backtest_start_time回测开始时间
24.         backtest_end_time回测结束时间
25.         backtest_adjust股票复权方式, 不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
26.         backtest_initial_cash回测初始资金
27.         backtest_commission_ratio回测佣金比例
28.         backtest_slippage_ratio回测滑点比例
29.     '''
30.     run(strategy_id='strategy_id',
31.         filename='main.py',
32.         mode=MODE_BACKTEST,
33.         token='token_id',
34.         backtest_start_time='2020-11-01 08:00:00',
35.         backtest_end_time='2020-11-10 16:00:00',
36.         backtest_adjust=ADJUST_PREV,
37.         backtest_initial_cash=10000000,
38.         backtest_commission_ratio=0.0001,
39.         backtest_slippage_ratio=0.0001)
```

整个策略需要三步：

1. 设置初始化函数: `init` , 一次性拿到所有需要的instruments信息， 将信息按symbol,date作为key存入字典， 使用 `subscribe` 函数进行数据订阅代码
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

实时模式下动态参数示例

本示例提供一种通过策略设置动态参数，可在终端界面显示和修改，在不停止策略的情况下手动修改参数传入策略方法.

```
1. # coding=utf-8
2. from __future__ import print_function, absolute_import, unicode_literals
3. from gm.api import *
4. import numpy as np
5. import pandas as pd
6.
7.
8. '''动态参数, 是指在不终止策略的情况下, 掘金终端UI界面和策略变量做交互,
9.     通过add_parameter在策略代码里设置动态参数, 终端UI界面会显示对应参数
10. '''
11.
12.
13. def init(context):
14.     # log日志函数, 只支持实时模式, 在仿真交易和实盘交易界面查看, 重启终端log日志会被清除, 需要记录到本地可以使用logging库
15.     log(level='info', msg='平安银行信号触发', source='strategy')
16.     # 设置k值阀值作为动态参数
```

```
    context.k_value = 23
    # add_parameter设置动态参数函数，只支持实时模式，在仿真交易和实盘交易界面查看，重启终端动态参数会被清除，重新运行策略会重新设置
    add_parameter(key='k_value', value=context.k_value, min=0, max=100, name='k值阀值', intro='设置k值阀值',
                  group='1', readonly=False)

    # 设置d值阀值作为动态参数
    context.d_value = 20
    add_parameter(key='d_value', value=context.d_value, min=0, max=100, name='d值阀值', intro='设置d值阀值',
                  group='2', readonly=False)

    print('当前的动态参数有', context.parameters)
    # 订阅行情
    subscribe(symbols='SZSE.002400', frequency='60s', count=120)


def on_bar(context, bars):

    data = context.data(symbol=bars[0]['symbol'], frequency='60s', count=100)

    kdj = KDJ(data, 9, 3, 3)
    k_value = kdj['kdj_k'].values
    d_value = kdj['kdj_d'].values

    if k_value[-1] > context.k_value and d_value[-1] < context.d_value:
        order_percent(symbol=bars[0]['symbol'], percent=0.01, side=OrderSide_Buy, order_type=OrderType_Market, position_effect=PositionEffect_Open)
        print('{}下单买入，  k值为{}'.format(bars[0]['symbol'], context.k_value))


# 计算KDJ
def KDJ(data, N, M1, M2):
    lowList= data['low'].rolling(N).min()
    lowList.fillna(value=data['low'].expanding().min(), inplace=True)
    highList = data['high'].rolling(N).max()
    highList.fillna(value=data['high'].expanding().max(), inplace=True)
    rsv = (data['close'] - lowList) / (highList - lowList) * 100
    data['kdj_k'] = rsv.ewm(alpha=1/M1).mean()
    data['kdj_d'] = data['kdj_k'].ewm(alpha=1/M2).mean()
    data['kdj_j'] = 3.0 * data['kdj_k'] - 2.0 * data['kdj_d']
    return data


# 动态参数变更事件
def on_parameter(context, parameter):
    # print(parameter)
    if parameter['name'] == 'k值阀值':
        # 通过全局变量把动态参数值传入别的事件里
        context.k_value = parameter['value']
        print('{}已经修改为{}'.format(parameter['name'], context.k_value))

    if parameter['name'] == 'd值阀值':
        context.d_value = parameter['value']
```

```
        print('{}已经修改为{}'.format(parameter['name'], context.d_value))


def on_account_status(context, account):
    print(account)


if __name__ == '__main__':
    '''
    strategy_id策略ID,由系统生成
    filename文件名,请与本文件名保持一致
    mode实时模式:MODE_LIVE回测模式:MODE_BACKTEST
    token绑定计算机的ID,可在系统设置-密钥管理中生成
    backtest_start_time回测开始时间
    backtest_end_time回测结束时间
    backtest_adjust股票复权方式不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
    backtest_initial_cash回测初始资金
    backtest_commission_ratio回测佣金比例
    backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='07c08563-a4a8-11ea-a682-7085c223669d',
        filename='main.py',
        mode=MODE_LIVE,
        token='2c4e3c59cde776ebc268bf6d7b4c457f204482b3',
        backtest_start_time='2020-09-01 08:00:00',
        backtest_end_time='2020-10-01 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=500000,
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

level2数据驱动事件示例

本示例提供level2行情的订阅， 包括逐笔成交、逐笔委托、委托队列
仅券商托管版本支持

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


def init(context):
    # 查询历史L2 Tick行情
    history_l2tick=get_history_l2ticks('SHSE.600519', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None,
                          skip_suspended=True, fill_missing=None,
                          adjust=ADJUST_NONE, adjust_end_time='', df=False)
    print(history_l2tick[0])

    # 查询历史L2 Bar行情
    history_l2bar=get_history_l2bars('SHSE.600000', '60s', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None,
                         skip_suspended=True, fill_missing=None,
```

```
                        adjust=ADJUST_NONE, adjust_end_time='', df=False)
    print(history_l2bar[0])

    # 查询历史L2 逐笔成交
    history_transactions = get_history_l2transactions('SHSE.600000', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None, df=False)
    print(history_transactions[0])

    # 查询历史L2 逐笔委托
    history_order=get_history_l2orders('SZSE.000001', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None, df=False)
    print(history_order[0])

    # 查询历史L2 委托队列
    history_order_queue = get_history_l2orders_queue('SZSE.000001', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None, df=False)
    print(history_order_queue[0])
    # 订阅浦发银行的逐笔成交数据
    subscribe(symbols='SHSE.600000', frequency='l2transaction')
    # 订阅平安银行的逐笔委托数据（仅支持深市标的）
    subscribe(symbols='SZSE.000001', frequency='l2order')


def on_l2order(context, order):
    # 打印逐笔成交数据
    print(order)


def on_l2transaction(context, transition):
    # 打印逐笔委托数据
    print(transition)



if __name__ == '__main__':
    '''
        strategy_id策略ID, 由系统生成
        filename文件名, 请与本文件名保持一致
        mode运行模式, 实时模式:MODE_LIVE回测模式:MODE_BACKTEST
        token绑定计算机的ID, 可在系统设置-密钥管理中生成
        backtest_start_time回测开始时间
        backtest_end_time回测结束时间
        backtest_adjust股票复权方式, 不复权:ADJUST_NONE前复权:ADJUST_PREV后复权:ADJUST_POST
        backtest_initial_cash回测初始资金
        backtest_commission_ratio回测佣金比例
        backtest_slippage_ratio回测滑点比例
    '''
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_BACKTEST,
        token='token_id',
        backtest_start_time='2020-11-01 08:00:00',
        backtest_end_time='2020-11-10 16:00:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
```

```
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001)
```

可转债数据获取、交易示例

本示例提供可转债数据获取、可转债交易

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


def init(context):
    # 订阅可转债行情。与股票无异
    subscribe(symbols='SHSE.113038', frequency='tick', count=2)

    # 获取可转债基本信息，输入可转债代码即可
    infos = get_instrumentinfos(symbols='SHSE.113038', df=True)

    # 输入可转债标的代码，可以获取到历史行情，但是只能是分钟线，不能获取日线。
    history_data = history(symbol='SHSE.113038', frequency='60s', start_time='2021-02-24 14:50:00',
                           end_time='2021-02-24 15:30:30', adjust=ADJUST_PREV, df=True)


    # 可转债回售、转股、转股撤销，需要券商实盘环境，仿真回测不可用。
    bond_convertible_call('SHSE.110051', 100, 0)
    bond_convertible_put('SHSE.183350', 100, 0)
    bond_convertible_put_cancel('SHSE.183350', 100)

    # 可转债下单，仅将symbol替换为可转债标的代码即可
    order_volume(symbol='SZSE.128041', volume=100, side=OrderSide_Buy, order_type=OrderType_Limit, position_effect=PositionEffect_Open, price=340)

    # 直接获取委托，可以看到相应的可转债委托，普通买卖通过标的体现可转债交易，转股、回售、回售撤销通过order_business字段的枚举值不同来体现。
    A = get_orders()


def on_tick(context, tick):
    # 打印频率为tick的浦发银行的50条最新tick
    print(tick)


if __name__ == '__main__':
    run(strategy_id='strategy_id',
        filename='main.py',
        mode=MODE_LIVE,
        token='token_id',
        backtest_start_time='2020-12-16 09:00:00',
        backtest_end_time='2020-12-16 09:15:00',
        backtest_adjust=ADJUST_PREV,
        backtest_initial_cash=10000000,
```

```
        backtest_commission_ratio=0.0001,
        backtest_slippage_ratio=0.0001
        )
```

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` 委托状态变化数据事件
  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.
- 处理账户 `account` 交易账户状态变化数据事件
  当交易账户状态产生变化时会触发on_account_status，用于监测 `交易账户委托状态` 变更.

其他事件处理函数

- 处理 `error` 错误事件
  当发生异常情况时触发错误事件，并返回错误码和错误信息
- 处理动态参数 `parameter` 动态参数修改事件
  当动态参数产生变化时会触发on_parameter，用于监测动态参数修改.

- 处理绩效指标对象 `Indicator` 回测结束事件
  在回测模式下，回测结束后会触发on_backtest_finished，并返回回测得到的绩效指标对象.

- 处理实时行情网络连接成功事件
  当实时行情网络连接成功时触发实时行情网络连接成功事件.

- 处理实时行情网络连接断开事件
  当实时行情网络连接断开时触发实时行情网络连接断开事件.

- 处理交易通道网络连接成功事件
  当交易通道网络连接成功时触发交易通道网络连接成功事件.

- 处理交易通道网络连接断开事件
  当交易通道网络连接断开时触发交易通道网络连接断开事件.

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。

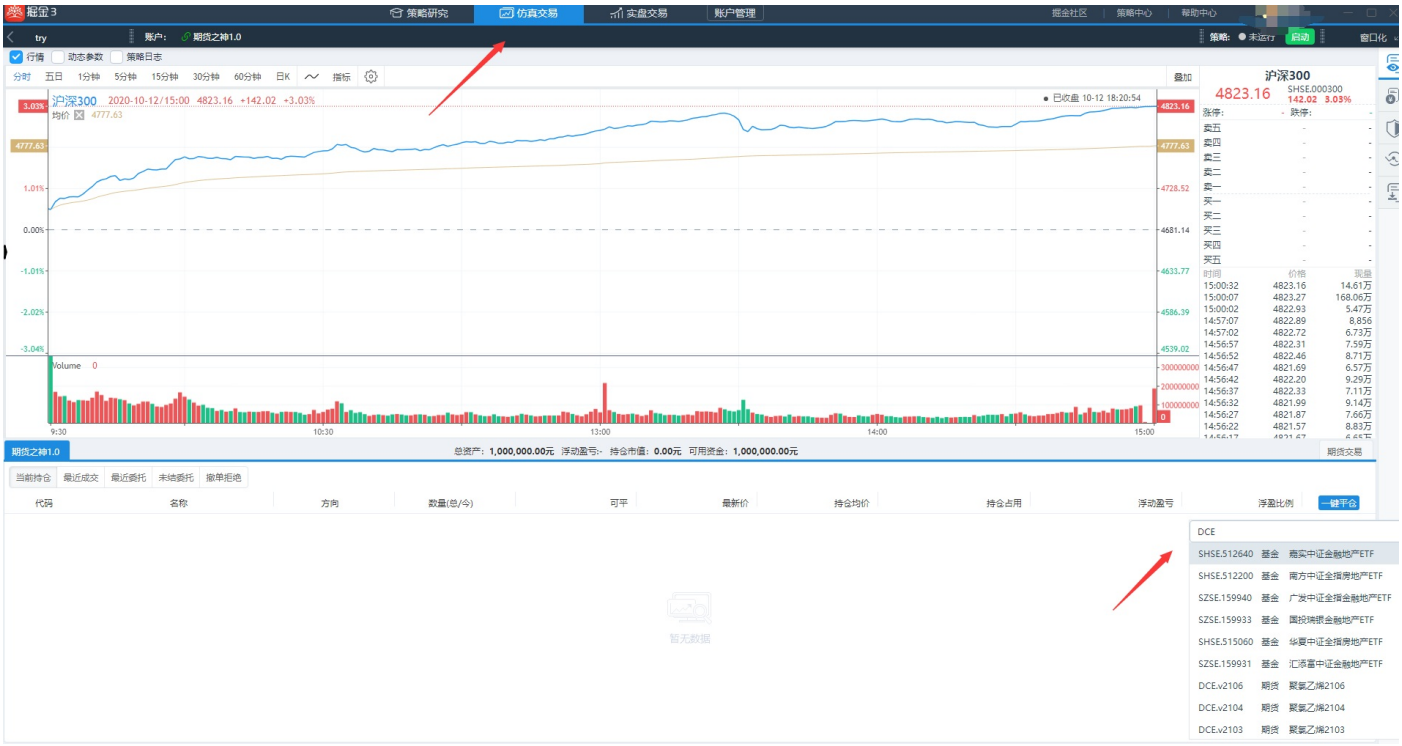
- 策略工作模式
  策略支持两种运行模式， 实时模式和回测模式,实时模式用于仿真交易及实盘交易，回测模式用于策略研究，用户需要在运行策略时选择模式.

变量约定

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` (注意区分大小写)。代码表示可以在掘金终端的仿真交易或交易工具中进行查询。



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（如600000）， 期货（如rb2011）， 期权(如10002498）， 指数（如000001）， 基金（如510300）等代码。

具体的代码请参考交易所的给出的证券代码定义。

symbol示例

市场中文名	市场代码	示例代码	证券简称
上交所	SHSE	SHSE.600000	浦发银行
深交所	SZSE	SZSE.000001	平安银行
中金所	CFFEX	CFFEX.IC2011	中证500指数2020年11月期货合约
上期所	SHFE	SHFE.rb2011	螺纹钢2020年11月期货合约
大商所	DCE	DCE.m2011	豆粕2020年11月期货合约
郑商所	CZCE	CZCE.FG101	玻璃2021年1月期货合约
上海国际能源交易中心	INE	INE.sc2011	原油2020年11月期货合约

期货主力连续合约

仅回测模式下使用， `期货主力连续合约` 为量价数据的简单拼接，未做平滑处理， 如SHFE.RB螺纹钢主力连续合约，其他主力合约请查看期货主力连续合约

mode - 模式选择

策略支持两种运行模式,需要在 `run()` 里面指定，分别为实时模式和回测模式。

实时模式

实时模式需指定 **mode = MODE_LIVE**

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，常用于仿真和实盘。

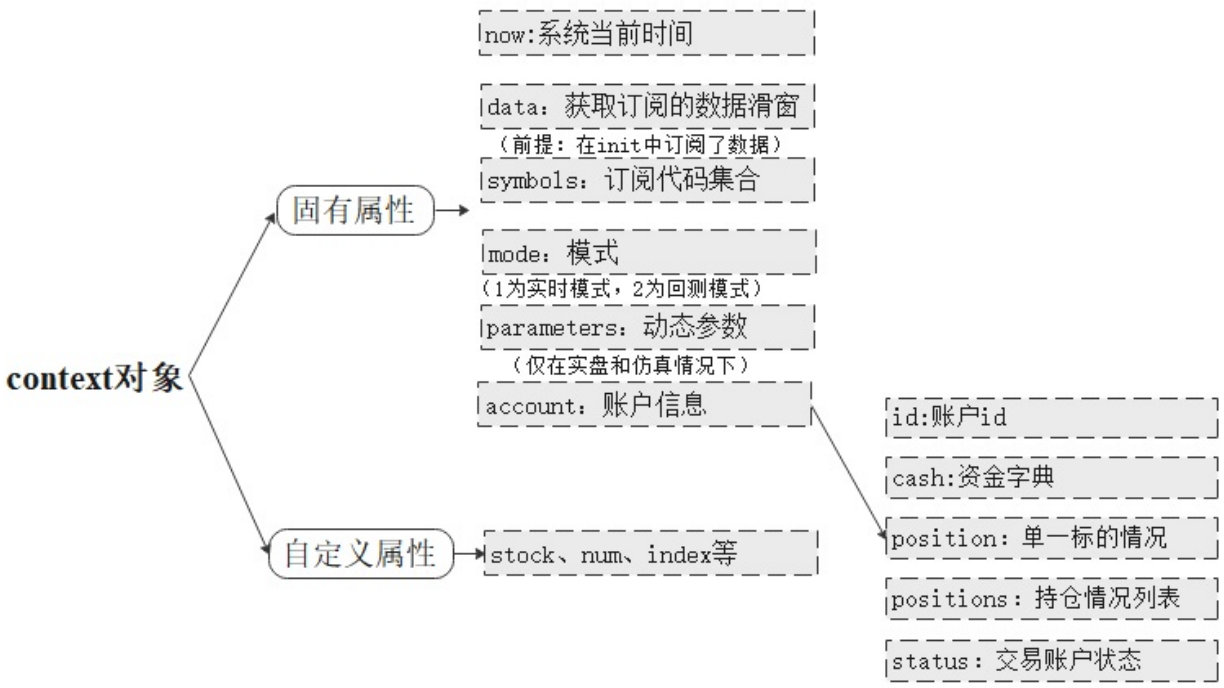
回测模式

回测模式需指定 **mode = MODE_BACKTEST**

订阅指定时段、指定交易代码、指定数据类型的历史行情，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。用户可以通过context定义多种自己需要的属性，也可以查看context固有属性，context结构如下图：



context.symbols - 订阅代码集合

通过subscribe行情订阅函数， 订阅代码会生成一个代码集合

函数原型：

```
context.symbols
```

返回值：

类型	说明
set(str)	订阅代码集合

示例：

```
subscribe(symbols=['SHSE.600519', 'SHSE.600419'], frequency='60s')
context.symbols
```

返回：

```
{'SHSE.600519', 'SHSE.600419'}
```

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(回测模式下是策略回测的当前历史时间， 实时模式下是用户的系统本地时间)

示例:

```
context.now
```

返回:

```
2020-09-01 09:40:00+08:00
```

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` 或者 `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(只允许单个标的的代码字符串)
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

订阅tick时
示例:

```
Subcribe_data = context.data(symbol='SHSE.600000', frequency='tick', count=2)
```

输出:

```
[{'symbol': 'SHSE.600000', 'open': 9.680000305175781, 'high': 9.720000267028809, 'low':
9.619999885559082, 'price': 9.630000114440918, 'quotes': [{'bid_p': 9.630000114440918,
```

```
'bid_v': 360197, 'ask_p': 9.640000343322754, 'ask_v': 124200}, {'bid_p':
9.619999885559082, 'bid_v': 1265300, 'ask_p': 9.649999618530273, 'ask_v': 172859},
{'bid_p': 9.609999656677246, 'bid_v': 1030400, 'ask_p': 9.65999984741211, 'ask_v':
233400}, {'bid_p': 9.600000381469727, 'bid_v': 1200000, 'ask_p': 9.670000076293945,
'ask_v': 150700}, {'bid_p': 9.59000015258789, 'bid_v': 208000, 'ask_p':
9.680000305175781, 'ask_v': 199543}], 'cum_volume': 29079145, 'cum_amount':
280888066.0, 'last_amount': 963.0, 'last_volume': 100, 'created_at':
datetime.datetime(2020, 11, 20, 11, 30, 1, 400000, tzinfo=tzfile('PRC')),
'cum_position': 0, 'trade_type': 0}, {'quotes': [{'bid_p': 9.630000114440918, 'bid_v':
315497, 'ask_p': 9.640000343322754, 'ask_v': 125900}, {'bid_p': 9.619999885559082,
'bid_v': 1291300, 'ask_p': 9.649999618530273, 'ask_v': 177959}, {'bid_p':
9.609999656677246, 'bid_v': 1035000, 'ask_p': 9.65999984741211, 'ask_v': 233400},
{'bid_p': 9.600000381469727, 'bid_v': 1213300, 'ask_p': 9.670000076293945, 'ask_v':
150700}, {'bid_p': 9.59000015258789, 'bid_v': 212100, 'ask_p': 9.680000305175781,
'ask_v': 173943}, {'bid_p': 0, 'bid_v': 0, 'ask_p': 0, 'ask_v': 0}, {'bid_p': 0,
'bid_v': 0, 'ask_p': 0, 'ask_v': 0}, {'bid_p': 0, 'bid_v': 0, 'ask_p': 0, 'ask_v': 0},
{'bid_p': 0, 'bid_v': 0, 'ask_p': 0, 'ask_v': 0}, {'bid_p': 0, 'bid_v': 0, 'ask_p': 0,
'ask_v': 0}], 'symbol': 'SHSE.600000', 'created_at': datetime.datetime(2020, 11, 20,
13, 0, 2, 430000, tzinfo=tzfile('PRC')), 'price': 9.630000114440918, 'open':
9.680000305175781, 'high': 9.720000267028809, 'low': 9.619999885559082, 'cum_volume':
29171845, 'cum_amount': 281780897.0, 'cum_position': 0, 'last_amount': 892831.0,
'last_volume': 92700, 'trade_type': 0, 'receive_local_time': 1605863292.163}]
```


订阅bar时
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。

Tips: context.data()与**bar**一起使用时的区别和联系

以订阅‘SHSE.600519’股票日频数据为例，在on_bar()中同时输出bar和context.data()。

- 当订阅的滑窗大小（count）为1时，bar返回的数据和context.data返回的数据是相同的
- 当订阅的滑窗大小（count）大于1时，bar返回的数据为最新的一条；而context.data()返回的数据是count条，其中最后一条和bar返回的数据相同
  也就是说，无论订阅滑窗大小如何设置，bar每次只返回一条最新数据，而context.data()返回数据条数等于count，并且最后一条为最新数据。

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：

```
# 所有持仓
Account_positions = context.account().positions()
# 指定持仓
Account_position = context.account().position(symbol='SHSE.600519',side =
PositionSide_Long)
```

返回值：

类型	说明
list[position]	持仓对象列表

注意：
没有持仓的情况下，用context.account().positions()查总持仓，返回空列表，用context.account().position()查单个持仓，返回None

输出

```
# 所有持仓输出
[{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'symbol': 'SHSE.600419',
'side': 1, 'volume': 2200, 'volume_today': 100, 'vwap': 16.43391600830338, 'amount':
36154.61521826744, 'fpnl': -2362.6138754940007, 'cost': 36154.61521826744, 'available':
2200, 'available_today': 100, 'created_at': datetime.datetime(2020, 9, 1, 9, 40,
tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 30, 9, 40,
tzinfo=tzfile('PRC')), 'account_name': '', 'vwap_diluted': 0.0, 'price': 0.0,
'order_frozen': 0, 'order_frozen_today': 0, 'available_now': 0, 'market_value': 0.0,
'last_price': 0.0, 'last_volume': 0, 'last_inout': 0, 'change_reason': 0,
'change_event_id': '', 'has_dividend': 0}, {'account_id': 'd7443a53-f65b-11ea-bb9d-
484d7eaefe55', 'symbol': 'SHSE.600519', 'side': 1, 'volume': 1100, 'vwap':
1752.575242219682, 'amount': 1927832.7664416502, 'fpnl': -110302.84700805641, 'cost':
1927832.7664416502, 'available': 1100, 'created_at': datetime.datetime(2020, 9, 1, 9,
40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 15, 9, 40,
tzinfo=tzfile('PRC')), 'account_name': '', 'volume_today': 0, 'vwap_diluted': 0.0,
'price': 0.0, 'order_frozen': 0, 'order_frozen_today': 0, 'available_today': 0,
'available_now': 0, 'market_value': 0.0, 'last_price': 0.0, 'last_volume': 0,
```

```
'last_inout': 0, 'change_reason': 0, 'change_event_id': '', 'has_dividend': 0}]
# 指定持仓输出
{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'symbol': 'SHSE.600519', 'side':
1, 'volume': 1100, 'vwap': 1752.575242219682, 'amount': 1927832.7664416502, 'fpnl':
-110302.84700805641, 'cost': 1927832.7664416502, 'available': 1100, 'created_at':
datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 15, 9, 40, tzinfo=tzfile('PRC')), 'account_name': '',
'volume_today': 0, 'vwap_diluted': 0.0, 'price': 0.0, 'order_frozen': 0,
'order_frozen_today': 0, 'available_today': 0, 'available_now': 0, 'market_value': 0.0,
'last_price': 0.0, 'last_volume': 0, 'last_inout': 0, 'change_reason': 0,
'change_event_id': '', 'has_dividend': 0}
```

示例-获取当前账户资金:

```
context.account().cash
```

返回值:

类型	说明
dict[cash]	资金对象字典

输出

```
{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'nav': 1905248.2789094353,
'pnl': -94751.72109056474, 'fpnl': -94555.35135529494, 'frozen': 1963697.3526980684,
'available': 36106.277566661825, 'cum_inout': 2000000.0, 'cum_trade':
1963697.3526980684, 'cum_commission': 196.3697352698069, 'last_trade':
1536.1536610412597, 'last_commission': 0.153615366104126, 'created_at':
datetime.datetime(2020, 9, 1, 8, 0, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 30, 9, 40, tzinfo=tzfile('PRC')), 'account_name': '',
'currency': 0, 'order_frozen': 0.0, 'balance': 0.0, 'market_value': 0.0, 'cum_pnl':
0.0, 'last_pnl': 0.0, 'last_inout': 0.0, 'change_reason': 0, 'change_event_id': ''}
```

示例-获取账户连接状态:

```
context.account().status
```

输出

```
state: 3
```

context.parameters - 动态参数

获取所有动态参数

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

示例-添加动态参数和查询所有设置的动态参数

```
add_parameter(key='k_value', value=context.k_value, min=0, max=100, name='k值阀值', intro='k值阀值',group='1', readonly=False)

context.parameters
```

输出

```
{'k_value': {'key': 'k_value', 'value': 80.0, 'max': 100.0, 'name': 'k值阀值', 'intro': 'k值阀值', 'group': '1', 'min': 0.0, 'readonly': False}}
```

context.xxxxx - 自定义属性

通过自定义属性设置参数, 随context全局变量传入策略各个事件里

```
context.my_value = 100000000
```

返回值:

类型	说明
any type	自定义属性

示例-输出自定义属性

```
print(context.my_value)
```

输出

```
100000000
```

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	日线开盘价
high	float	日线最高价
low	float	日线最低价
price	float	最新价
cum_volume	long	成交总量/最新成交量,累计值（日线成交量）
cum_amount	float	成交总金额/最新成交额,累计值 （日线成交金额）
cum_position	int	合约持仓量(只适用于期货),累计值（股票此值为0）
trade_type	int	交易类型（只适用于期货） 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	瞬时成交量
last_amount	float	瞬时成交额（郑商所last_amount为0）
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; , level2行情对应的是list[0]~list[9]买卖一档到十档, 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（bid_p和bid_v为0）, 涨停时无卖档报价(ask_p和ask_v为0); 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
bid_q	dict	委买队列 包含（total_orders （int）委托总个数, queue_volumes (list) 委托量队列）, 仅level2行情支持
ask_q	dict	委卖队列 包含（total_orders （int）委托总个数, queue_volumes (list) 委托量队列）, 仅level2行情支持

注意： 1、tick是分笔成交数据，股票频率为3s， 期货为0.5s， 指数5s， 包含集合竞价数据，股票早盘集合竞价数为09:15:00-09:25:00的tick数据
2、涨停时， 没有卖价和卖量， ask_p和ask_v用0填充，跌停时，没有买价和买量，bid_p和bid_v用0填充
3、queue_volumes 委托量队列，只能获取到最优第一档的前50个委托量（不活跃标的可能会不足50个）

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

注意：
不活跃标的，没有成交量是不生成bar

L2Order - Level2 逐笔委托

参数名	类型	说明
symbol	str	标的代码
side	str	委托方向 深市：’1’买， ‘2’卖， ‘F’借入， ‘G’出借， 沪市：’B’买，’S’卖
price	float	委托价
volume	int	委托量
order_type	str	委托类型 深市：’1’市价， ‘2’限价， ‘U’本方最优，沪市：’A’新增委托订单，’D’删除委托订单
order_index	int	委托编号
created_at	datetime.datetime	创建时间

L2Transaction - Level2 逐笔成交

参数名	类型	说明
symbol	str	标的代码
side	str	委托方向 沪市：B – 外盘,主动买， S – 内盘,主动卖， N – 集合竞价，深市无此字段

price	float	成交价
volume	int	成交量
exec_type	str	成交类型 深市：'4'撤单，'F'成交， 沪市无此字段
exec_index	int	成交编号
ask_order_index	int	叫卖委托编号
bid_order_index	int	叫买委托编号
created_at	datetime.datetime	创建时间

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
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 参数可缩小查询范围
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
status	dict	交易账户状态 查询交易账户连接状态

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（掘金维护，下单唯一标识）
order_id	str	委托柜台ID（系统字段，下单不会立刻生成，委托报到柜台才会生成）
ex_ord_id	str	委托交易所ID（系统字段，下单不会立刻生成，委托报到柜台才会生成）
algo_order_id	str	算法单ID
symbol	str	标的代码
status	int	委托状态 取值参考 OrderStatus
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_business	int	委托业务属性 取值参考 OrderBusiness
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
position_src	int	头寸来源（系统字段）
volume	int	委托量
price	float	委托价格
value	int	委托额
percent	float	委托百分比
target_volume	int	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	int	已成量 （一笔委托对应多笔成交为累计值）
filled_vwap	float	已成均价，公式为 `(price*(1+backtest_slippage_ratio))` （仅股票实盘支持，期货实盘不支持）
filled_amount	float	已成金额，公式为 `(filled_volume*filled_vwap)` （仅股票实盘支持，期货实盘不支持）
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	柜台委托ID
exec_id	str	交易所成交ID
symbol	str	委托标的
side	int	买卖方向 取值参考 OrderSide
position_effect	int	开平标志 取值参考 PositionEffect
order_business	int	委托业务属性 OrderBusiness
order_style	int	委托风格 OrderStyle
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	成交价格
volume	int	成交量
amount	float	成交金额
cost	float	成交成本金额（仅期货实盘支持，股票实盘不支持）
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	总资金
fpnl	float	浮动盈亏
frozen	float	持仓占用资金 （仅期货实盘支持，股票实盘不支持）
order_frozen	float	冻结资金
available	float	可用资金
market_value	float	市值 （仅股票实盘支持，期货实盘不支持）
balance	float	资金余额
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	int	总持仓量; 如果要得到昨持仓量，公式为 (vol
volume_today	int	今日买入量
market_value	float	持仓市值
vwap	float	持仓均价 new_vwap=((position.vwap * p (trade.volume*trade.price))/(position （实盘时，期货跨天持仓，会自动变成昨结价，仿
vwap_open	float	开仓均价（期货适用，实盘适用）
vwap_diluted	float	摊薄成本（股票适用，实盘适用）
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) * volume 率的考虑，回测模式fpnl只有仓位变化时或者一天 次， 回测的price为当天的收盘价） （根据持仓
fpnl_open	float	浮动盈亏（期货适用， 根据开仓均价计算）
cost	float	持仓成本 (vwap * volume * multiplier
order_frozen	int	挂单冻结仓位
order_frozen_today	int	挂单冻结今仓仓位(仅上期所和上海能源交易所标的

available	int	非挂单冻结仓位 ，公式为 (volume - order_ 昨仓位，公式为 (available - available_
available_today	int	非挂单冻结今仓位，公式为 (volume_today order_frozen_today) (仅上期所和上海能源
available_now	int	当前可用仓位
credit_position_sellable_volume	int	可卖担保品数
created_at	datetime.datetime	建仓时间 （实盘不支持）
updated_at	datetime.datetime	仓位变更时间 （实盘不支持）

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*sqrt(250),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）
calmar_ratio	float	卡玛比率 年化收益率/最大回撤
open_count	int	开仓次数
close_count	int	平仓次数
win_count	int	盈利次数（平仓价格大于持仓均价vwap的次数）
lose_count	int	亏损次数 （平仓价格小于或者等于持仓均价vwap的次数）
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

API介绍

        - order_cancel_all - 撤销所有委托
        - order_close_all - 平当前所有可平持仓
        - get_unfinished_orders - 查询日内全部未结委托
        - get_orders - 查询日内全部委托
        - get_execution_reports - 查询日内全部执行回报
        - ipo_buy - 新股申购
        - fund_etf_buy - ETF申购
        - fund_etf_redemption - ETF赎回
        - fund_subscribing - 基金认购
        - fund_buy - 基金申购
        - fund_redemption - 基金赎回
        - bond_reverse_repurchase_agreement - 国债逆回购
    - 交易查询函数
        - context.account().positions() - 查询当前账户全部持仓
        - context.account().position(symbol, side) - 查询当前账户指定持仓
        - context.account().cash - 查询当前账户资金
    - 两融交易函数
        - credit_buying_on_margin - 融资买入
        - credit_short_selling - 融券卖出
        - credit_repay_cash_directly - 直接还款
        - credit_repay_share_directly - 直接还券
        - credit_get_collateral_instruments - 查询担保证券
        - credit_get_borrowable_instruments - 查询标的证券
        - credit_get_borrowable_instruments_positions - 查询券商融券账户头寸
        - credit_get_contracts - 查询融资融券合约
        - credit_get_cash - 查询融资融券资金
        - credit_repay_share_by_buying_share - 买券还券
        - credit_repay_cash_by_selling_share - 卖券还款
        - credit_buying_on_collateral - 担保品买入
        - credit_selling_on_collateral - 担保品卖出
        - credit_collateral_in - 担保品转入
        - credit_collateral_out - 担保品转出
    - 算法交易函数
        - algo_order 算法交易委托
        - algo_order_cancel 撤销算法委托
        - get_algo_orders 查询算法委托
        - algo_order_pause 暂停或重启或者撤销算法委托
        - get_algo_child_orders 查询算法委托的所有子单
        - on_algo_order_status 算法单状态事件
    - 交易事件
        - on_order_status - 委托状态更新事件
        - on_execution_report - 委托执行回报事件
        - on_account_status - 交易账户状态更新事件
    - 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
    - 其他函数
        - set_token - 设置token
        - log - 日志函数
        - get_strerror - 查询错误码的错误描述信息
        - get_version - 查询api版本

        - set_mfp - 设置留痕信息
    - 其他事件
        - on_backtest_finished - 回测结束事件
        - on_error - 错误事件
        - on_market_data_connected - 实时行情网络连接成功事件
        - on_trade_data_connected - 交易通道网络连接成功事件
        - on_market_data_disconnected - 实时行情网络连接断开事件
        - on_trade_data_disconnected - 交易通道网络连接断开事件

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,秒均不可以只输入个位数, 例: `'9:40:0'` 或 `'14:5:0'`

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,
    serv_addr='', backtest_match_mode=0)
```

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”127.0.0.1:7001”
backtest_match_mode	int	回测市价撮合模式： 1-实时撮合：在当前bar的收盘价/当前tick的price撮合，0-延时撮合：在下个bar的开盘价/下个tick的price撮合，默认是模式0

返回值：

`None`

示例：

```
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')
```

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

停止策略，退出策略进程

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, 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’，‘60s’，‘300s’，‘900s’ 等，默认‘1d’，详情见股票行情数据和期货行情数据，实时行情支持的频率
count	int	订阅数据滑窗大小，默认 `1` ,详情见数据滑窗
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='60s', count=5,
unsubscribe_previous=True)
```

注意：

1. subscribe支持多次调用，并可以重复订阅同一代码。订阅后的数据储存在本地，需要通过context.data接口调用或是直接在on_tick或on_bar中获取。
2. 在实时模式下，最新返回的数据是不复权的。

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：

```
unsubscribe(symbols='*', frequency='60s')
```

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’, ‘60s’, ‘300s’, ‘900s’ 等, 默认‘1d’, 详情见股票行情数据和期货行情数据, 实时行情支持的频率

返回值:

`None`

示例:

```
1. unsubscribe(symbols='SHSE.600000,SHSE.600004', frequency='60s')
```

注意:
如示例所示代码, 取消 `SHSE.600000,SHSE.600004` 两只代码 `60s` 行情的订阅, 若 `SHSE.600000` 同时还订阅了 `"300s"` 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

数据事件是阻塞回调事件函数，通过**subscribe**函数订阅， 主动推送

on_tick - tick数据推送事件

接收tick分笔数据， 通过subscribe订阅tick行情，行情服务主动推送tick数据

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：

```
{'symbol': 'SHSE.600519', 'created_at': datetime.datetime(2020, 9, 2, 14, 7, 23,
620000, tzinfo=tzfile('PRC')), 'price': 1798.8800048828125, 'open': 1825.0, 'high':
1828.0, 'low': 1770.0, 'cum_volume': 2651191, 'cum_amount': 4760586491.0,
'cum_position': 0, 'last_amount': 179888.0, 'last_volume': 100, 'trade_type': 0,
'receive_local_time': 1602751345.262745}
```

on_bar - bar数据推送事件

接收固定周期bar数据， 通过subscribe订阅bar行情，行情服务主动推送bar数据

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:

```
{'symbol': 'SHSE.600519', 'eob': datetime.datetime(2020, 9, 30, 15, 15, 1,
tzinfo=tzfile('PRC')), 'bob': datetime.datetime(2020, 9, 30, 0, 0,
tzinfo=tzfile('PRC')), 'open': 1660.0, 'close': 1668.5, 'high': 1691.9000244140625,
'low': 1660.0, 'volume': 2708781, 'amount': 4536012540.0, 'pre_close':
1652.2999267578125, 'position': 0, 'frequency': '1d', 'receive_local_time':
1602751647.923199}
```

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。

on_l2transaction - 逐笔成交事件

接收逐笔成交数据（L2行情时有效）
仅特定券商版本可用
函数原型:

```
on_l2transaction(context, transaction)
```

参数:

参数名	类型	说明
context	context对象	上下文对象
transaction	L2Transaction对象	收到的逐笔成交行情

示例:

```
def on_l2transaction(context, transaction):
    print(transaction)
```

输出:

```
{'symbol': 'SZSE.300003', 'volume': 300, 'created_at': datetime.datetime(2020, 11, 24,
```

```
    9, 38, 16, 50, tzinfo=tzfile('PRC')), 'exec_type': '4', 'side': '', 'price': 0.0}
```

on_l2order - 逐笔委托事件

接收逐笔委托数据（深交所L2行情时有效）
仅特定券商版本可用
函数原型：

```
on_l2order(context, l2order)
```

参数：

参数名	类型	说明
context	context对象	上下文对象
l2order	L2Order对象	收到的逐笔委托行情

示例：

```
def on_l2order(context, l2order):
    print(l2order)
```

输出：

```
{'symbol': 'SZSE.300003', 'side': '1', 'price': 29.350000381469727, 'volume': 2400,
'created_at': datetime.datetime(2020, 11, 24, 9, 38, 16, 80, tzinfo=tzfile('PRC')),
'order_type': '2'}
```

on_l2order_queue - 委托队列事件

接收委托队列，L2行情时有效，最优价最大50笔委托量
仅特定券商版本可用

函数原型：

```
on_l2order_queue(context, l2order_queue)
```

参数：

参数名	类型	说明
context	context对象	上下文对象
l2order_queue	L2OrderQueue对象	收到的委托队列行情

```
def on_l2order_queue(context, l2order_queue):
    print(l2order_queue)
```

输出：

```
{'symbol': 'SHSE.600000', 'price': 9.90999984741211, 'total_orders': 155,
'queue_orders': 50, 'queue_volumes': [52452, 600, 1200, 3200, 10000, 1800, 1000, 300,
10000, 2000, 500, 500, 2000, 1000, 2000, 300, 1200, 1400, 1000, 200, 1000, 100, 500,
1000, 500, 2380
0, 25400, 1000, 2000, 200, 500, 1200, 5000, 2000, 17600, 5000, 1000, 1300, 1000, 1200,
1000, 3000, 1000, 1000, 15000, 400, 15000, 5000, 2000, 10000], 'created_at':
datetime.datetime(2020, 11, 23, 14, 0, 1, tzinfo=tzfile('PRC')), 'side': '', 'volume':
0}
```

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。实时模式，返回当前最新tick数据，回测模式，返回回测当前时间点的最近一分钟的收盘价

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 ，使用参考symbol
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(symbols='SZSE.000001')
```

输出:

```
[{'symbol': 'SZSE.000001', 'open': 16.200000762939453, 'high': 16.920000076293945,
'low': 16.149999618530273, 'price': 16.559999465942383, 'quotes': [{'bid_p':
16.549999237060547, 'bid_v': 209200, 'ask_p': 16.559999465942383, 'ask_v': 296455},
{'bid_p': 16.540000915527344, 'bid_v': 188900, 'ask_p': 16.56999969482422, 'ask_v':
374405}, {'bid_p': 16.530000686645508, 'bid_v': 44900, 'ask_p': 16.579999923706055,
'ask_v': 187220}, {'bid_p': 16.520000457763672, 'bid_v': 20800, 'ask_p':
16.59000015258789, 'ask_v': 102622}, {'bid_p': 16.510000228881836, 'bid_v': 37700,
'ask_p': 16.600000381469727, 'ask_v': 337002}], 'cum_volume': 160006232, 'cum_amount':
2654379585.66, 'last_amount': 14153832.0, 'last_volume': 854700, 'trade_type': 7,
'created_at': datetime.datetime(2020, 10, 15, 15, 0, 3, tzinfo=tzfile('PRC'))}]
```

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`

3. 回测只返回symbol、price和created_at字段，实时模式返回全部字段

4. 实时模式无法获取集合竞价的数据，可使用history_n

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=True)
```

参数:

参数名	类型	说明
symbol	str or list	标的代码，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 ，使用参考symbol
frequency	str	频率，支持 ‘tick’, ‘1d’, ‘60s’ 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
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 ，目前只支持股票
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:参考**Tick对象**或者**Bar对象**。

当df = True时， 返回

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

示例:

```
history_data = history(symbol='SHSE.000300', frequency='1d', start_time='2010-07-28',
end_time='2017-07-30', fields='open, close, low, high, eob', adjust=ADJUST_PREV, df=
True)
```

输出:

```
          open      close        low       high                       eob
0     2796.4829  2863.7241  2784.1550  2866.4041 2010-07-28 00:00:00+08:00
1     2866.7720  2877.9761  2851.9961  2888.5991 2010-07-29 00:00:00+08:00
2     2871.4810  2868.8459  2844.6819  2876.1360 2010-07-30 00:00:00+08:00
3     2868.2791  2917.2749  2867.4500  2922.6121 2010-08-02 00:00:00+08:00
4     2925.2539  2865.9709  2865.7610  2929.6140 2010-08-03 00:00:00+08:00
```

当df = False时， 返回

类型	说明
list	tick 列表 或者 bar 列表

注意:

```
history_data = history(symbol='SHSE.000300', frequency='1d', start_time='2017-07-30',
end_time='2017-07-31', fields='open, close, low, high, eob', adjust=ADJUST_PREV,
df=False)
```

输出:

```
[{'open': 3722.42822265625, 'close': 3737.873291015625, 'low': 3713.655029296875,
'high': 3746.520751953125, 'eob': datetime.datetime(2017, 7, 31, 0, 0,
tzinfo=tzfile('PRC'))}]
```

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: '`
获取数据目前采用前后闭区间的方式，即会获取前后两个时间节点的数据，使用时务必注意这点

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

6. 单次返回数据量最大返回33000，超出部分不返回

7. start_time和end_time输入不存在日期时，会报错details = “failed to parse datetime”

history_n - 查询历史行情最新n条

函数原型:

```
history_n(symbol, frequency, count, end_time=None, 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
symbol	str	标的代码(只允许单个标的的代码字符串)，使用时参考symbol
frequency	str	频率，支持 ‘tick’, ‘1d’, ‘60s’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，默认None时，用了实际当前时间，非回测当前时间
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 , 目前只支持股票
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:参考**Tick对象**或者**Bar对象**。

当df = True时，返回

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

示例:

```
history_n_data = history_n(symbol='SHSE.600519', frequency='1d', count=100,
end_time='2020-10-20 15:30:00', fields='symbol, open, close, low, high, eob',
adjust=ADJUST_PREV, df=True)
```

输出:

```
 symbol       open  ...       high                       eob
0   SHSE.600519  1350.2278  ...  1350.3265 2020-05-22 00:00:00+08:00
1   SHSE.600519  1314.6434  ...  1350.8010 2020-05-25 00:00:00+08:00
2   SHSE.600519  1354.0629  ...  1354.1321 2020-05-26 00:00:00+08:00
3   SHSE.600519  1343.3086  ...  1344.2970 2020-05-27 00:00:00+08:00
4   SHSE.600519  1322.5214  ...  1331.3878 2020-05-28 00:00:00+08:00
```

当df = False时， 返回

类型	说明
list	tick 列表 或者 bar 列表

示例:

```
history_n_data = history_n(symbol='SHSE.600519', frequency='1d', count=2,
end_time='2020-10-20 15:30:00', fields='symbol, open, close, low, high, eob',
adjust=ADJUST_PREV, df=False)
```

输出:

```
[{'symbol': 'SHSE.600519', 'open': 1725.0, 'close': 1699.0, 'low': 1691.9000244140625,
'high': 1733.97998046875, 'eob': datetime.datetime(2020, 10, 19, 0, 0,
tzinfo=tzfile('PRC'))}, {'symbol': 'SHSE.600519', 'open': 1699.989990234375, 'close':
1734.0, 'low': 1695.0, 'high': 1734.969970703125, 'eob': datetime.datetime(2020, 10,
20, 0, 0, tzinfo=tzfile('PRC'))}]
```

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

6. 单次返回数据量最大返回33000, 超出部分不返回

7. end_time输入不存在日期时，会报错details = “Can’t parse string as time: 2020-10-40 15:30:00”

context.data - 查询订阅数据

函数原型:

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

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)，使用时参考symbol
frequency	str	频率，支持 ‘tick’，‘1d’，‘60s’等,需和subscribe函数中指定的频率保持一致。详情见股票行情数据和期货行情数据
count	int	滑窗大小(正整数)，需小于等于subscribe函数中count值
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

返回值:参考**Tick对象**或者**Bar对象**。

类型	说明
dataframe	tick的dataframe 或者 bar的dataframe

示例:

```
def init(context):
    subscribe(symbols='SHSE.600519', frequency='60s', count=2)

def on_bar(context,bars):
    data = context.data(symbol='SHSE.600519', frequency='60s', count=1)
```

输出:

```
        symbol             eob                             bob             open
close     high         low          amount    pre_close    position    frequency
volume
0    SHSE.600519    2020-12-21 09:31:00+08:00    2020-12-21 09:30:00+08:00    1840
1845.5    1845.5    1838.199951    210503484      0            0          60s
114365
```

注意:

1. 只有在订阅后，此接口才能取到数据，如未订阅数据，则返回值为空。
2. symbols参数只支持输入一个标的。
3. count参数必须小于或等于订阅函数里面的count值

get_history_l2ticks - 查询历史L2 Tick行情

仅特定券商版本可用

函数原型:

```
 get_history_l2ticks(symbols, 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，使用时参考symbol
start_time	str	开始时间 (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间 (%Y-%m-%d %H:%M:%S 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
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`

返回值:参考**Tick对象**

当df = True时， 返回 `dataframe`

当df = Falst， 返回 `list`
示例:

```
history_l2tick=get_history_l2ticks('SHSE.600519', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None,
                        skip_suspended=True, fill_missing=None,
                        adjust=ADJUST_NONE, adjust_end_time='', df=False)
print(history_l2tick[0])
```

输出:

```
{'symbol': 'SHSE.600519', 'open': 1771.010009765625, 'high': 1809.9000244140625, 'low': 1771.010009765625, 'price': 1791.0999755859375, 'quotes': [{'bid_p': 1790.8800048828125, 'bid_v': 100, 'ask_p': 1794.760009765625, 'ask_v': 200}, {'bid_p': 1790.80004882812
5, 'bid_v': 123, 'ask_p': 1794.800048828125, 'ask_v': 100}, {'bid_p': 1790.699951171875, 'bid_v': 100, 'ask_p': 1794.8800048828125, 'ask_v': 416}, {'bid_p': 1790.68994140625, 'bid_v': 200, 'ask_p': 1794.8900146484375, 'ask_v': 300}, {'bid_p': 1790.630004882812
5, 'bid_v': 300, 'ask_p': 1794.9000244140625, 'ask_v': 1000}, {'bid_p': 1790.6199951171875, 'bid_v': 500, 'ask_p': 1794.949951171875, 'ask_v': 300}, {'bid_p': 1790.6099853515625, 'bid_v': 300, 'ask_p': 1794.9599609375, 'ask_v': 300}, {'bid_p': 1790.59997558593
75, 'bid_v': 200, 'ask_p': 1794.97998046875, 'ask_v': 100}, {'bid_p': 1790.510009765625, 'bid_v': 314, 'ask_p': 1794.989990234375, 'ask_v': 200}, {'bid_p': 1790.5, 'bid_v': 4200, 'ask_p': 1795.0, 'ask_v': 9700}], 'cum_volume': 5866796, 'cum_amount': 1049949547
1.0, 'last_amount': 1973854.0, 'last_volume': 1100, 'created_at': datetime.datetime(2020, 11, 23, 14, 0, 2, tzinfo=tzfile('PRC')), 'cum_position': 0, 'trade_type': 0}
```

注意:
1、 `get_history_l2ticks` 接口每次只能提取一天的数据， 如果取数时间超过一天，则返回按照结束时间的最近有一个交易日数据， 如果取数时间段超过1 个自然月（31）天，则获取不到数据

get_history_l2bars - 查询历史L2 Bar行情

仅特定券商版本可用

函数原型:

```
get_history_l2bars(symbols, 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，使用时参考symbol
frequency	str	频率， 支持 ‘1d’, ‘60s’等
start_time	str	开始时间 (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间 (%Y-%m-%d %H:%M:%S 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
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`

返回值:参考**Bar对象**。

当df = True时， 返回 `dataframe`

当df = Falst， 返回 `list`

示例:

```
history_l2bar=get_history_l2bars('SHSE.600000', '60s', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None,
                                 skip_suspended=True, fill_missing=None,
                                 adjust=ADJUST_NONE, adjust_end_time='', df=False)
print(history_l2bar[0])
```

输出:

```
{'symbol': 'SHSE.600000', 'frequency': '60s', 'open': 9.90999984741211, 'high':
```

```
    9.90999984741211, 'low': 9.890000343322754, 'close': 9.899999618530273, 'volume':
    1270526, 'amount': 12574276.0, 'bob': datetime.datetime(2020, 11, 23, 14, 0,
    tzinfo=tzfile('PRC'))
, 'eob': datetime.datetime(2020, 11, 23, 14, 1, tzinfo=tzfile('PRC')), 'position': 0,
'pre_close': 0.0}
```

注意:
1、 `get_history_l2bars` 接口每次最多可提取 1 个自然月（31）天的数据如: 2015.1.1-2015.1.31
错误设置: （2015.1.1-2015.2.1）超出 31 天则获取不到任何数据

get_history_l2transactions - 查询历史L2 逐笔成交

仅特定券商版本可用

函数原型:

```
get_history_l2transactions(symbols, start_time, end_time, fields=None, df=False)
```

参数:

参数名	类型	说明
symbols	str	标的代码，使用时参考symbol
start_time	str	开始时间 (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间 (%Y-%m-%d %H:%M:%S 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
df	bool	是否返回 dataframe格式, 默认 `False`

返回值:参考**level2逐笔成交数据**

当df = True时, 返回 `dataframe`

当df = Falst, 返回 `list`

示例:

```
history_transactions=get_history_l2transactions('SHSE.600000', '2020-11-23 14:00:00',
'2020-11-23 15:00:00', fields=None, df=False)
print(history_transactions[0])
```

输出:

```
{'symbol': 'SHSE.600000', 'side': 'B', 'price': 9.90999984741211, 'volume': 100,
'created_at': datetime.datetime(2020, 11, 23, 14, 0, 0, 820000, tzinfo=tzfile('PRC')),
'exec_type': ''}
```

注意:
1、 `get_history_l2transactions` 接口每次只能提取一天的数据, 如果取数时间超过一天，则返回按照开始时间的最近有一个交易日数据

get_history_l2orders -查询历史L2 逐笔委托

仅特定券商版本可用
注意：仅深市标的可用

函数原型：

```
get_history_l2orders(symbols, start_time, end_time, fields=None, df=False)
```

参数：

参数名	类型	说明
symbols	str	标的代码，使用时参考symbol
start_time	str	开始时间 (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间 (%Y-%m-%d %H:%M:%S 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
df	bool	是否返回 dataframe格式， 默认 `False`

返回值：参考**level2逐笔委托数据**

当df = True时， 返回 `dataframe`

当df = Falst， 返回 `list`

示例：

```
history_order=get_history_l2orders('SZSE.000001', '2020-11-23 14:00:00', '2020-11-23 15:00:00', fields=None, df=False)
print(history_order[0])
```

输出：

```
{'symbol': 'SZSE.000001', 'side': '1', 'price': 19.520000457763672, 'volume': 200, 'created_at': datetime.datetime(2020, 11, 23, 14, 0, 0, 110000, tzinfo=tzfile('PRC')), 'order_type': '2'}
```

注意：
1、 `get_history_l2orders` 接口每次只能提取一天的数据， 如果取数时间超过一天，则返回按照开始时间的最近有一个交易日数据

get_history_l2orders_queue -查询历史L2 委托队列

仅特定券商版本可用

函数原型：

```
get_history_l2orders_queue(symbols, start_time, end_time, fields=None, df=False)
```

参数:

参数名	类型	说明
symbols	str	标的代码，使用时参考symbol
start_time	str	开始时间 (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间 (%Y-%m-%d %H:%M:%S 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
df	bool	是否返回 dataframe格式, 默认 `False`

返回值:参考**level2委托队列据**

当df = True时， 返回 `dataframe`

当df = Falst， 返回 `list`

示例:

```
history_order_queue=get_history_l2orders_queue('SHSE.600000', '2020-11-23 14:00:00',
'2020-11-23 15:00:00', fields=None, df=False)
print(history_order_queue[0])
```

输出:

```
{'symbol': 'SHSE.600000', 'price': 9.90999984741211, 'total_orders': 155,
'queue_orders': 50, 'queue_volumes': [52452, 600, 1200, 3200, 10000, 1800, 1000, 300,
10000, 2000, 500, 500, 2000, 1000, 2000, 300, 1200, 1400, 1000, 200, 1000, 100, 500,
1000, 500, 2380
0, 25400, 1000, 2000, 200, 500, 1200, 5000, 2000, 17600, 5000, 1000, 1300, 1000, 1200,
1000, 3000, 1000, 1000, 15000, 400, 15000, 5000, 2000, 10000], 'created_at':
datetime.datetime(2020, 11, 23, 14, 0, 1, tzinfo=tzfile('PRC')), 'side': '', 'volume':
0}
```

注意:
1、 `get_history_l2orders_queue` 接口每次只能提取一天的数据， 如果取数时间超过一天，则返回按照开始时间的最近有一个交易日数据

get_fundamentals - 查询基本面数据

函数原型:

```
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, 使用时参考symbol , 免费版本只支持单个标的, 多标的只返回第一个
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. 默认是 `1000` , 为保护服务器, 单次查询最多返回 `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,PELFYNPAAEI,PETTMNPAAEI'
 df=True)
```

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,PELFYNPAAEI,PETTMNPAAEI'
```

```
    df=True)
```

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,PELFYNPAAEI,PETTMNPAAEI'
 df=True)
# 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,PELFYNPAAEI,PETTMNPAAEI'
 df=True)
```

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

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, 使用时参考symbol, 免费版本只支持单个标的, 多标的只返回第一个
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,PELFYNPAAEI,PETTMNPAAEI'
```

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
4.  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
5.  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
6.  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
7.  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:

```
1.  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，使用时参考symbol
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,参考返回值。
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, 8: 可转债, 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.datetime	上市日期
delisted_date	datetime.datetime	退市日期
trade_date	datetime.datetime	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 （可转债没有涨停价）
lower_limit	float	跌停价 （可转债没有跌停价）
adj_factor	float	复权因子.基金跟股票才有
conversion_price	float	可转债转股价
conversion_start_date	datetime.datetime	可转债开始转股时间
underlying_symbol	str	可转债正股标的

示例:

```
get_instruments(exchanges='SZSE', df=True)
```

输出:

```
  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
   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
3.    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
4.    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
5.    ......
```

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.获取全A股票代码示例 `get_instruments(exchanges='SZSE,SHSE', sec_types=1, fields='symbol',df=1)['symbol'].tolist()`
5.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
6.关于可转债的到期日(退市日期)为 `delisted_date` , 转股价值为 `转股价值 = 转股数*股价=(100/可转债转股价)*股价`

get_history_instruments - 查询交易标的历史信息数据

返回指定symbols的标的日频历史数据

函数原型:

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

参数:

参数名	类型	说明
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, 是必填参数, 使用时参考symbol
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

df	bool	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.datetime	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
pre_settle	float	昨结价
position	int	持仓量
pre_close	float	昨收价
upper_limit	float	涨停价 （可转债没有涨停价）
lower_limit	float	跌停价 （可转债没有跌停价）
adj_factor	float	复权因子.基金跟股票才有

示例:

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

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 **33000** 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，使用时参考symbol
exchanges	str or list or None	见交易市场代码, 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
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，8: 可转债， 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.datetime	上市日期
delisted_date	datetime.datetime	退市日期
conversion_price	float	可转债转股价
underlying_symbol	str	可转债正股标的

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_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:

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

当df = True时，返回

类型	说明
dataframe	dataframe

示例1:

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

输出:

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

当df = False时，返回

类型	说明
list[dict()]	列表镶嵌字典

示例2:

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

输出:

```
[{'symbol': 'SHSE.603681', 'weight': 0.009999999776482582}, {'symbol': 'SHSE.601518',
'weight': 0.009999999776482582}, {'symbol': 'SHSE.600010', 'weight':
0.12999999523162842}···]
```

示例3:
只输出 `symbol` 列表

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

输出:

```
['SHSE.603966', 'SHSE.603960', '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_history_constituents - 查询指数成份股的历史数据

函数原型:

```
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:

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

输出:

```
                              constituents
trade_date
{'SHSE.600527': 0.009999999776482582, 'SHSE.600461': 0.019999999552965164,···}
2017-07-31 00:00:00
{'SHSE.603966': 0.009999999776482582, '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_continuous_contracts - 获取主力合约

函数原型:
获取主力合约和次主力合约

```
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.AG00
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:

key	value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': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 1, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 2, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 3, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 4, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 5, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 6, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 7, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 8, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 9, 0, 0, tzinfo=tzfile('PRC'))}, {'symbol': 'SHFE.ag2012', 'trade_date': datetime.datetime(2020, 7, 10, 0, 0, tzinfo=tzfile('PRC'))}]
```

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'`

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:

```
['SHSE.600000', 'SHSE.600016', '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_dividend - 查询分红送配

函数原型:

```
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)，使用时参考symbol
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:

key	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
created_at	datetime.datetime	创建时间

示例：

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

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_trading_dates - 查询交易日列表

函数原型：

```
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：

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

输出：

```
['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:

```
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:

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

输出:

```
'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:

```
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'`

股票与指数数据函数

股票与指数数据函数(beta版)

python股票与指数数据API包含在gm3.0.145版本及以上版本，不需要引入新库

stk_get_index_constituents - 查询指数最新成分股

查询指定指数在最新交易日的成分股和权重

函数原型:

```
stk_get_index_constituents(index)
```

参数:

参数名	类型	中文名称	必填	默认值	参数用法说明
index	str	指数代码	Y	无	必填，只能输入一个指数，如: `'SHSE.000905'`

返回值: `dataframe`

字段名	类型	中文名称	说明
index	str	指数代码	查询成分股的指数代码
symbol	str	成分股代码	exchange.sec_id
weight	float	成分股权重	成分股symbol对应的指数权重
trade_date	datetime.datetime	交易日期	最新交易日，%Y-%m-%d 格式

示例:

```
stk_get_index_constituents(index='SHSE.000905')
```

输出:

```
           index       symbol  weight        date
0    SHSE.000905  SZSE.000009  0.0060  2022-09-07
1    SHSE.000905  SZSE.000012  0.0015  2022-09-07
2    SHSE.000905  SZSE.000021  0.0018  2022-09-07
3    SHSE.000905  SZSE.000027  0.0020  2022-09-07
4    SHSE.000905  SZSE.000028  0.0007  2022-09-07
..           ...         ...     ...         ...
495  SHSE.000905  SHSE.688390  0.0032  2022-09-07
496  SHSE.000905  SHSE.688521  0.0020  2022-09-07
497  SHSE.000905  SHSE.688777  0.0035  2022-09-07
```

```
498  SHSE.000905  SHSE.688819  0.0008  2022-09-07
499  SHSE.000905  SHSE.689009  0.0025  2022-09-07
[500 rows x 4 columns]
```

注意：

1. 数据日频更新，当日更新前返回前一交易日的成分数据，约在交易日20点左右更新当日数据。

期货数据函数

期货数据函数(beta版)

python期货数据API包含在gm3.0.145版本及以上版本，不需要引入新库

fut_get_continuous_contracts - 查询连续合约对应的真实合约

查询指定时间段连续合约在每个交易日上对应的真实合约

函数原型：

```
fut_get_continuous_contracts(csymbol, start_date="", end_date="")
```

参数：

参数名	类型	中文名称	必填	默认值	参数用法说明
csymbol	str	连续合约代码	Y	无	必填，使用时参考查询代码， 支持主力合约、次主力、前5个月份连续合约代码，如： 1000股指期货主力连续合约：CFFEX.IM， 1000股指期货次主力连续合约：CFFEX.IM22， 1000股指期货当月连续合约：CFFEX.IM00， 1000股指期货下月连续合约：CFFEX.IM01， 1000股指期货下季连续合约：CFFEX.IM02， 1000股指期货隔季连续合约：CFFEX.IM03
start_date	str	开始时间	N	“”	开始时间日期，%Y-%m-%d 格式，默认 "" 表示最新时间
end_date	str	结束时间	N	“”	结束时间日期，%Y-%m-%d 格式，默认 "" 表示最新时间

返回值：`list[dict]`

字段名	类型	中文名称	说明
symbol	str	标的代码	exchange.sec_id
trade_date	datetime.datetime	交易日期	具体合约对应的交易日期

示例：

```
fut_get_continuous_contracts(csymbol='SHFE.NI', start_date="2022-09-01",
end_date="2022-09-15")
```

输出：

```
[{'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-01'}, {'symbol': 'SHFE.ni2210',
```

```
    'trade_date': '2022-09-02'}, {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-05'},
    {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-06'}, {'symbol': 'SHFE.ni2210',
    'trade_date': '2022-09-07'}, {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-08'},
    {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-09'}, {'symbol': 'SHFE.ni2210',
    'trade_date': '2022-09-13'}, {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-14'},
    {'symbol': 'SHFE.ni2210', 'trade_date': '2022-09-15'}]
```

注意：

1. 具体合约（真实合约）：`交易所.品种名到期月份` 对应期货具体合约symbol，如CFFEX.IF2206
2. 主力连续合约（虚拟合约，由真实合约拼接）：`交易所.品种名` 对应主力连续合约symbol，如CFFEX.IF，CFFEX.IC

- 主力连续合约切换规则
  1、每个品种只选出唯一一个主力合约。
  2、日成交量和持仓量都为最大的合约，确定为新的主力合约，每日收盘结算后判断，于下一交易日进行指向切换，日内不会进行主力合约的切换。
  3、按照第二条规定产生新的主力合约之前，维持原来的主力合约不变。
  4、若出现当前主力合约的成交量和持仓量都不是最大的情况，当前指向合约在下一个交易日必须让出主力合约身份，金融期货新主力指向成交量最大的合约（中金所），商品期货新主力指向持仓量最大的合约（上期所、大商所、郑商所、上期能源）。

3. 次主力连续合约（虚拟合约，由真实合约拼接）：`交易所.品种名 22` 对应次主力连续合约symbol，如CFFEX.IF22，CFFEX.IC22

- 次主力连续合约切换规则
  1、每个品种只选出唯一一个次主力合约。
  2、金融期货日成交量第二大、或商品期货日持仓量第二大的合约，确定为新的次主力合约，每日收盘结算后判断，于下一交易日进行指向切换，日内不会进行次主力合约的切换。
  3、按照第二条规定产生新的次主力合约之前，维持原来的次主力合约不变。
  4、若金融期货出现当前次主力合约的成交量、或商品期货出现当前次主力合约持仓量不是第二大的情况，当前指向合约在下一个交易日必须让出次主力合约身份，金融期货新主力指向成交量第二大的合约（中金所），商品期货新主力指向持仓量第二大的合约（上期所、大商所、郑商所、上期能源）。

4. 月份连续合约（虚拟合约，由真实合约拼接）：`交易所.品种名 月份排序` 对应月份连续合约symbol，如SHFE.RB00，SHFE.RB01，…，SHFE.RB04（同一品种最多有最近5个月的月份连续合约）

- 月份连续合约的切换规则
  1、该品种上市合约按交割月份排序
  2、00对应最近月份合约，01对应其后一个合约，02对应再后一个合约，依次类推
  3、合约最后交易日盘后切换。

5. 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`

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(指股数)
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（限价下单和科创板市价保护价是必填字段）
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

返回值：

类型	说明
list[order]	委托对象列表，参见委托

示例：

```
order_volume(symbol='SHSE.600000', volume=10000, side=OrderSide_Buy,
```

```
    order_type=OrderType_Limit, position_effect=PositionEffect_Open, price=11)
```

返回:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1, 'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0, 'order_style': 1, 'volume': 10000, 'value': 110000.0, 'percent': 5.5e-05, 'target_volume': 10000, 'target_value': 110000.0, 'target_percent': 5.5e-05, 'filled_volume': 10000, 'filled_vwap': 11.0011, 'filled_amount': 110010.99999999999, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 11.0011, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0}]
```

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OrderType_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。

6.关于 `side` 与 `position_effect` 字段的使用说明
做多（买开）： `side=OrderSide_Buy , position_effect=PositionEffect_Open`
平仓（卖平）： `side=OrderSide_Sell , position_effect=PositionEffect_Close`

做空（卖开）： `side=OrderSide_Sell , position_effect=PositionEffect_Open`
平仓（买平）： `side= OrderSide_Buy , position_effect=PositionEffect_Close`

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（限价下单和科创板市价保护价是必填字段）
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

返回值:

类型	说明
list[order]	委托对象列表，参见委托

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:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-
11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1,
'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0,
'order_style': 1, 'volume': 9000, 'value': 100000.0, 'percent': 5e-05, 'target_volume':
9000, 'target_value': 99000.0, 'target_percent': 4.95e-05, 'filled_volume': 9000,
'filled_vwap': 11.0011, 'filled_amount': 99009.9, 'created_at': datetime.datetime(2020,
9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40,
tzinfo=tzfile('PRC')), 'filled_commission': 9.90099, 'account_name': '', 'order_id':
'', 'ex_ord_id': '', 'algo_order_id': '', 'order_business': 0, 'order_duration': 0,
'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0,
'ord_rej_reason_detail': '', 'stop_price': 0.0}]
```

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OrderType_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:

```
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（限价下单和科创板市价保护价是必填字段）
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

返回值:

类型	说明
list[order]	委托对象列表，参见委托

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:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-
11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1,
'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0,
'order_style': 1, 'volume': 18181800, 'value': 200000000.0, 'percent': 0.1,
'target_volume': 18181800, 'target_value': 199999800.0, 'target_percent': 0.0999999,
```

```
'filled_volume': 18181800, 'filled_vwap': 11.0011, 'filled_amount': 200019799.98,
'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission':
20001.979998, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '',
'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0,
'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price':
0.0}]
```

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OrderType_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。

6.期货实盘时，percent是以合约上市的初始保证金比例计算得到的，并非实时保证金比例。

order_target_volume - 调仓到目标持仓量

函数原型:

```
1. 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	表示将多仓还是空仓调到目标持仓量，参见 持仓方向
order_type	int	参见 订单类型
price	float	价格（限价下单和科创板市价保护价是必填字段）
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

返回值:

类型	说明
list[order]	委托对象列表，参见委托

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：

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-
11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1,
'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 13.0,
'order_style': 1, 'volume': 10000, 'value': 130000.0, 'percent': 6.5e-05,
'target_volume': 10000, 'target_value': 130000.0, 'target_percent': 6.5e-05,
'filled_volume': 10000, 'filled_vwap': 13.0013, 'filled_amount': 130013.0,
'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission':
13.0013, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '',
'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0,
'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price':
0.0}]
```

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	表示将多仓还是空仓调到目标持仓额，参见持仓方向
order_type	int	参见订单类型
price	float	价格（限价下单和科创板市价保护价是必填字段）
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

返回值：

类型	说明
list[order]	委托对象列表，参见委托

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：

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1, 'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0, 'order_style': 1, 'volume': 9000, 'value': 100000.0, 'percent': 5e-05, 'target_volume': 9000, 'target_value': 100000.0, 'target_percent': 5e-05, 'filled_volume': 9000, 'filled_vwap': 11.0011, 'filled_amount': 99009.9, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 9.90099, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0}]
```

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OrderType_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	表示将多仓还是空仓调到目标持仓比例，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（限价下单和科创板市价保护价是必填字段）
account	account id or account name or None	帐户

返回值:

类型	说明
list[order]	委托对象列表，参见委托

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:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-
11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1,
'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0,
```

```
    'order_style': 1, 'volume': 18181800, 'value': 200000000.0, 'percent': 0.1,
    'target_volume': 18181800, 'target_value': 199999800.0, 'target_percent': 0.1,
    'filled_volume': 18181800, 'filled_vwap': 11.0011, 'filled_amount': 200019799.98,
    'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at':
    datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission':
    20001.979998, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '',
    'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0,
    'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price':
    0.0}]
```

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OrderType_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。

6.期货实盘时，percent是以合约上市的初始保证金比例计算得到的，并非实时保证金比例。

order_batch - 批量委托接口

函数原型:

```
1. 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（预留字段，目前无效）
account	account id or account name or None	帐户

返回值:

类型	说明
list[order]	委托对象列表，参见委托

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:

```
{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1, 'position_effect': 1, 'order_type': 2, 'status': 3, 'price': 10.280000686645508, 'order_style': 1, 'volume': 100, 'filled_volume': 100, 'filled_vwap': 10.281028686714173, 'filled_amount': 1028.1028686714174, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 0.10281028686714173, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'position_side': 0, 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0, 'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0}
{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000001', 'symbol': 'SHSE.600004', 'side': 1, 'position_effect': 1, 'order_type': 2, 'status': 3, 'price': 15.050000190734863, 'order_style': 1, 'volume': 100, 'filled_volume': 100, 'filled_vwap': 15.051505190753936, 'filled_amount': 1505.1505190753935, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 0.15051505190753936, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'position_side': 0, 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0, 'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0}
{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000002', 'symbol': 'SHSE.600000', 'side': 1, 'position_effect': 1, 'order_type': 2, 'status': 3, 'price': 10.180000305175781, 'order_style': 1, 'volume': 100, 'filled_volume': 100, 'filled_vwap': 10.1810183052063, 'filled_amount': 1018.10183052063, 'created_at': datetime.datetime(2020, 9, 2, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 2, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 0.101810183052063, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'position_side': 0, 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0, 'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0}
{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000003', 'symbol': 'SHSE.600004', 'side': 1, 'position_effect': 1, 'order_type': 2, 'status': 3, 'price': 14.819999694824219, 'order_style': 1, 'volume': 100, 'filled_volume': 100, 'filled_vwap': 14.8214816947937, 'filled_amount': 1482.14816947937, 'created_at': datetime.datetime(2020, 9, 2, 9, 40,
```

```
tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 2, 9, 40,
tzinfo=tzfile('PRC')), 'filled_commission': 0.148214816947937, 'account_name': '',
'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'position_side': 0,
'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0,
'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0,
'value': 0.0, 'percent': 0.0, 'target_volume': 0, 'target_value': 0.0,
'target_percent': 0.0}
```

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OrderType_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[dict]	传入单个字典. 或者list字典. 每个字典包含key: cl_ord_id, account_id, 参见委托

示例:

```
order_1 = {'cl_ord_id': order1['cl_ord_id'], 'account_id': order1['account_id']}
order_2 = {'cl_ord_id': order2['cl_ord_id'], 'account_id': order2['account_id']}
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

注意:不支持市价委托类型的委托，会被柜台拒绝

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:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600519', 'side': 1, 'position_effect': 1, 'position_side': 1, 'order_type': 2, 'status': 3, 'price': 1792.0, 'order_style': 1, 'volume': 100, 'value': 179200.0, 'percent': 8.96e-05, 'target_volume': 100, 'target_value': 179200.0, 'target_percent': 8.96e-05, 'filled_volume': 100, 'filled_vwap': 1792.1792, 'filled_amount': 179217.92, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 17.921792000000003, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0}]
```

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:

```
[{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600519', 'side': 1, 'position_effect': 1, 'position_side': 1, 'order_type': 2, 'status': 3, 'price': 1792.0, 'order_style': 1, 'volume': 100, 'value': 179200.0, 'percent': 8.96e-05, 'target_volume': 100, 'target_value': 179200.0, 'target_percent': 8.96e-05, 'filled_volume': 100, 'filled_vwap': 1792.1792, 'filled_amount': 179217.92, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission': 17.921792000000003, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '', 'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0}]
```

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()
```

返回:

```
[{'strategy_id': '004beb61-1282-11eb-9313-00ff5a669ee2', 'account_id': '3acc8b6e-af54-11e9-b2de-00163e0a4100', 'account_name': '3acc8b6e-af54-11e9-b2de-00163e0a4100', 'cl_ord_id': '49764a82-14fb-11eb-89df-00ff5a669ee2', 'order_id': '4a06f925-14fb-11eb-9e8a-00163e0a4100', 'exec_id': '573b108b-14fb-11eb-9e8a-00163e0a4100', 'symbol':
```

```
    'SHSE.600714', 'position_effect': 1, 'side': 1, 'exec_type': 15, 'price':
    5.579999923706055, 'volume': 900, 'amount': 5021.999931335449, 'created_at':
    datetime.datetime(2020, 10, 23, 14, 45, 29, 776756, tzinfo=tzfile('PRC')),
    'commission': 5.0, 'cost': 5021.999931335449, 'ord_rej_reason': 0,
    'ord_rej_reason_detail': ''}]
```

交易查询函数

context.account().positions() - 查询当前账户全部持仓

原型:

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

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:

类型	说明
list[position]	持仓对象列表

注意：没有持仓时，返回空列表

示例-获取当前持仓:

```
# 所有持仓
Account_positions = context.account().positions()
```

输出

```
% 所有持仓输出
[{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'symbol': 'SHSE.600419',
'side': 1, 'volume': 2200, 'volume_today': 100, 'vwap': 16.43391600830338, 'amount':
36154.61521826744, 'fpnl': -2362.6138754940007, 'cost': 36154.61521826744, 'available':
2200, 'available_today': 100, 'created_at': datetime.datetime(2020, 9, 1, 9, 40,
tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 30, 9, 40,
tzinfo=tzfile('PRC')), 'account_name': '', 'vwap_diluted': 0.0, 'price': 0.0,
'order_frozen': 0, 'order_frozen_today': 0, 'available_now': 0, 'market_value': 0.0,
'last_price': 0.0, 'last_volume': 0, 'last_inout': 0, 'change_reason': 0,
'change_event_id': '', 'has_dividend': 0}, {'account_id': 'd7443a53-f65b-11ea-bb9d-
484d7eaefe55', 'symbol': 'SHSE.600519', 'side': 1, 'volume': 1100, 'vwap':
1752.575242219682, 'amount': 1927832.7664416502, 'fpnl': -110302.84700805641, 'cost':
1927832.7664416502, 'available': 1100, 'created_at': datetime.datetime(2020, 9, 1, 9,
40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 15, 9, 40,
tzinfo=tzfile('PRC')), 'account_name': '', 'volume_today': 0, 'vwap_diluted': 0.0,
'price': 0.0, 'order_frozen': 0, 'order_frozen_today': 0, 'available_today': 0,
'available_now': 0, 'market_value': 0.0, 'last_price': 0.0, 'last_volume': 0,
'last_inout': 0, 'change_reason': 0, 'change_event_id': '', 'has_dividend': 0}]
```

context.account().position(symbol, side) - 查询当前账户指定持仓

参数:

参数名	类型	说明
symbol	str	标的代码
side	int	持仓方向，取值参考PositionSide

返回值:

类型	说明
dict[position]	持仓对象列表

注意： 当指定标的没有持仓时，返回None
示例-获取当前持仓:

```
# 指定持仓
Account_position = context.account().position(symbol='SHSE.600519',side = PositionSide_Long)
```

输出

```
# 指定持仓输出
{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'symbol': 'SHSE.600519', 'side': 1, 'volume': 1100, 'vwap': 1752.575242219682, 'amount': 1927832.7664416502, 'fpnl': -110302.84700805641, 'cost': 1927832.7664416502, 'available': 1100, 'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 9, 15, 9, 40, tzinfo=tzfile('PRC')), 'account_name': '', 'volume_today': 0, 'vwap_diluted': 0.0, 'price': 0.0, 'order_frozen': 0, 'order_frozen_today': 0, 'available_today': 0, 'available_now': 0, 'market_value': 0.0, 'last_price': 0.0, 'last_volume': 0, 'last_inout': 0, 'change_reason': 0, 'change_event_id': '', 'has_dividend': 0}
```

context.account().cash - 查询当前账户资金

原型:

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

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:

类型	说明
dict[cash]	资金对象列表

示例-获取当前账户资金:

```
context.account().cash
```

输出

```
{'account_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'nav': 1905248.2789094353,
'pnl': -94751.72109056474, 'fpnl': -94555.35135529494, 'frozen': 1963697.3526980684,
'available': 36106.277566661825, 'cum_inout': 2000000.0, 'cum_trade':
1963697.3526980684, 'cum_commission': 196.3697352698069, 'last_trade':
1536.1536610412597, 'last_commission': 0.153615366104126, 'created_at':
datetime.datetime(2020, 9, 1, 8, 0, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 30, 9, 40, tzinfo=tzfile('PRC')), 'account_name': '',
'currency': 0, 'order_frozen': 0.0, 'balance': 0.0, 'market_value': 0.0, 'cum_pnl':
0.0, 'last_pnl': 0.0, 'last_inout': 0.0, 'change_reason': 0, 'change_event_id': ''}
```

两融交易函数

两融 SDK

python两融SDK包含在gm3.0.126版本及以上版本，不需要引入新库
融资融券暂时仅支持实盘委托，不支持仿真交易

credit_buying_on_margin - 融资买入

函数原型:

```
credit_buying_on_margin(position_src, symbol, volume, price,
order_type=OrderType_Limit, order_duration=OrderDuration_Unknown,
                        order_qualifier=OrderQualifier_Unknown, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
price	float	价格
position_src	int	头寸来源 取值参考 PositionSrc
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
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值:

请参考 order 返回值字段说明

示例代码

```
credit_buying_on_margin(position_src=PositionSrc_L1, symbol='SHSE.600000', volume=100,
price=10.67)
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ ------------
----------------------- ----------- ---------- ------ ----- ------ --------------------
--------------------------------------------------- ------------- ------------ -------
- --------- ------------ ---- -------------- ------------ ------------ ----------
---- --------- ------------- -------------------- ---------- ----------- ----- ------
- ------------ ----------- ------------- ------------ ---------- ------------ ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b853062-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10      10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 863549, tzinfo=tzfile('PRC')) 200
0    0               0              0             0               0         0
0          0           0.0    0.0     0             0.0          0.0            0
0.0         0.0           0.0                None
```

credit_short_selling - 融券卖出

函数原型

```
credit_short_selling(position_src, symbol, volume, price, order_type=OrderType_Limit,
order_duration=OrderDuration_Unknown,
                         order_qualifier=OrderQualifier_Unknown, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
price	float	价格
position_src	int	头寸来源 取值参考 PositionSrc
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

account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值：

请参考 order 返回值字段说明

示例代码

```
credit_short_selling(position_src=PositionSrc_L1, symbol='SHSE.600000', volume=100,
price=10.67, order_type=OrderType_Limit,
                                order_duration=OrderDuration_Unknown,
                                order_qualifier=OrderQualifier_Unknown, account_id='')
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ -------------
---------------------- ----------- ---------- ------ ----- ------ --------------------
-------------------------------------------------- -------------- ------------ -------
- --------- ------------- ---- --------------- ------------ -------------- -----------
---- --------- -------------- -------------------- ---------- ----------- ----- ------
- ------------ ------------ -------------- ------------ ----------- ------------- ---
-------------- -----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b853062-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1           10     10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 863549, tzinfo=tzfile('PRC')) 201
0    0               0              0              0                0         0
0.0        0           0.0   0.0     0             0.0          0.0            0
0.0         0.0           0.0               None
```

credit_repay_cash_directly - 直接还款

函数原型：

```
credit_repay_cash_directly(amount, account_id='')
```

参数：

参数名	类型	说明
amount	float	还款金额
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值： `dict`

字段	类型	说明

actual_repay_amount	float	实际还款金额
account_id	str	账号id
account_name	str	账户名称

示例代码:

```
credit_repay_cash_directly(amount=10000.00, account_id='')
```

示例返回值:

```
actual_repay_amount account_id                           account_name
------------------- ------------------------------------ ------------
10000.0             8f30e83f-a7c5-11ea-b510-309c231d28bd 001515018318
```

credit_repay_share_directly - 直接还券

函数原型

```
credit_repay_share_directly(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `[dict]`

请参考 `order` 返回值字段说明

示例代码

```
credit_repay_share_directly(symbol='SHSE.600000', volume=100, account_id='')
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail price stop_price order_style value percent target_volume
target_value target_percent filled_volume filled_vwap filled_amount filled_commission
updated_at
------------------------------------ ------------------------------------ ------------
---------------------- ---------- ---------- ------ ------ --------------------------
------------------------------------------- -------------- ------------ -------- ----
```

```
----- ------------ ---- --------------- ------------ -------------- --------------- -
-------- -------------- -------------------- ----- ---------- ----------- ----- ------
- ------------ ------------ -------------- ------------ ----------- ------------ ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b86685e-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10     100
datetime.datetime(2020, 6, 6, 15, 41, 44, 871536, tzinfo=tzfile('PRC')) 204
0    0               0             0              0               0         0
0.0   0.0        0           0.0   0.0     0             0.0          0.0            0
0.0         0.0           0.0               None
........ ......
```

credit_get_collateral_instruments - 查询担保证券

查询担保证券，可做担保品股票列表

函数原型：

```
credit_get_collateral_instruments(account_id='', df=False)
```

参数：

参数名	类型	说明
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号
df	bool	是否返回 dataframe 格式数据。默认 False，返回list[dict]

返回值：

字段	类型	说明
symbol	str	标的
pledge_rate	float	折算率

示例代码

```
credit_get_collateral_instruments(account_id='', df=False)
```

示例返回值

```
symbol      pledge_rate
----------- -----------
SHSE.010107  0.9
SHSE.010303  0.9
..........   ...
```

credit_get_borrowable_instruments - 查询可融标的证券

查询标的证券，可做融券标的股票列表

函数原型：

```
credit_get_borrowable_instruments(position_src, account_id='', df=False)
```

参数：

参数名	类型	说明
position_src	int	头寸来源 取值参考 PositionSrc
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号
df	bool	是否返回 `dataframe` 格式数据。默认 False，返回list[dict]

返回值： `list[dict]`

字段	类型	说明
symbol	str	标的
margin_rate_for_cash	float	融资保证金比率
margin_rate_for_security	float	融券保证金比率

示例代码

```
credit_get_borrowable_instruments(position_src=PositionSrc_L1, account_id='', df=False)
```

示例返回值

```
symbol      margin_rate_for_cash margin_rate_for_security
----------- -------------------- -------------------------
SHSE.510050  1.0                  0.6
SHSE.510160  1.0                  0.6
...........  ...                  ...
```

credit_get_borrowable_instruments_positions - 查询券商融券账户头寸

查询券商融券账户头寸，可用融券的数量

函数原型：

```
credit_get_borrowable_instruments_positions(position_src, account_id='', df=False)
```

参数：

参数名	类型	说明
position_src	int	头寸来源 取值参考 PositionSrc
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号
df	bool	是否返回 `dataframe` 格式数据。默认 False，返回list[dict]

返回值:
当df = True时, 返回 `dataframe`
当df = False时, 返回 `list[dict]`

字段	类型	说明
symbol	str	标的
balance	float	证券余额
available	float	证券可用金额

示例代码

```
credit_get_borrowable_instruments_positions(position_src=PositionSrc_L1, account_id='', df=False)
```

示例返回值

```
symbol       balance available
------------ ------- -----------
SHSE.600166  700.0   700.0
SHSE.688002  2000.0  2000.0
..........   ......  ......
```

credit_get_contracts - 查询融资融券合约

函数原型:

```
credit_get_contracts(position_src, account_id='', df=False)
```

参数:

参数名	类型	说明
position_src	int	头寸来源 取值参考 PositionSrc
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号
df	bool	是否返回 `dataframe` 格式数据。默认 False, 返回list[dict]

返回值:
当df = True时, 返回 `dataframe`
当df = False时, 返回 `list[dict]`

字段	类型	说明
symbol	str	标的
ordersno	str	委 托 号
creditdirect	str	融资融券方向, '0'表示融资, '1'表示融券
orderqty	float	委托数量
matchqty	float	成交数量

orderamt	float	委托金额
orderfrzamt	float	委托冻结金额
matchamt	float	成交金额
clearamt	float	清算金额
lifestatus	str	合约状态
creditrepay	float	T日之前归还金额
creditrepayunfrz	float	T日归还金额
fundremain	float	应还金额
stkrepay	float	T日之前归还数量
stkrepayunfrz	float	T日归还数量
stkremain	float	应还证券数量
stkremainvalue	float	应还证券市值
fee	float	融资融券息、费
overduefee	float	逾期未偿还息、费
fee_repay	float	已偿还息、费
punifee	float	利息产生的罚息
punifee_repay	float	已偿还罚息
rights	float	未偿还权益金额
overduerights	float	逾期未偿还权益
rights_repay	float	已偿还权益
lastprice	float	最新价
profitcost	float	浮动盈亏
sno	str	合约编号
punidebts	float	逾期本金罚息
punidebts_repay	float	本金罚息偿还
punidebtsunfrz	float	逾期本金罚息
punifeeunfrz	float	逾期息费罚息
punirights	float	逾期权益罚息
punirights_repay	float	权益罚息偿还
punirightsunfrz	float	逾期权益罚息
feeunfrz	float	实时偿还利息
overduefeeunfrz	float	实时偿还逾期利息
rightsqty	float	未偿还权益数量
overduerightsqty	float	逾期未偿还权益数量
orderdate	int	委托日期，时间戳类型
lastdate	int	最后一次计算息费日期，时间戳类型
closedate	int	合约全部偿还日期，时间戳类型
sysdate	int	系统日期，时间戳类型

enddate	int	负债截止日期，时间戳类型
oldenddate	int	原始的负债截止日期，时间戳类型

示例代码

```
credit_get_contracts(position_src=PositionSrc_L1, account_id='', df=False)
```

示例返回值

```
symbol      ordersno creditdirect orderqty matchamt clearamt lifestatus enddate
fundremain fee   rights    profitcost sno      rightsqty orderdate matchqty orderamt
orderfrzamt oldenddate creditrepay creditrepayunfrz stkrepay stkrepayunfrz stkremain
stkremainvalue overduefee fee_repay punifee punifee_repay overduerights rights_repay
lastprice sysdate lastdate closedate punidebts punidebts_repay punidebtsunfrz
punifeeunfrz punirights punirights_repay punirightsunfrz feeunfrz overduefeeunfrz
overduerightsqty
---------- -------- ------------ -------- -------- -------- ---------- -------- ------
---- ----- --------- ---------- -------- --------- --------- -------- -------- --------
--- ---------- ----------- ---------------- -------- ------------ --------- ----------
---- ---------- --------- ------- ------------ ------------ ------------ --------- --
----- -------- --------- --------- --------------- -------------- ------------ --------
-- ---------------- --------------- -------- --------------- -----------------
SZSE.159937 115906   0            59600.0  217957.2 217957.2 0          20200823
220666.32  32.69 220666.32 11112.52   15211131 59600.0   0         0.0      0.0
0.0         0          0.0         0.0              0.0      0.0           0.0
0.0            0.0        0.0       0.0     0.0           0.0           0.0
0.0       0.0     0.0      0.0       0.0       0.0             0.0           0.0
0.0        0.0              0.0             0.0      0.0             0.0
.......... ......
```

credit_get_cash - 查询融资融券资金

函数原型:

```
credit_get_cash(account_id='')
```

参数:

参数名	类型	说明
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `dict`

字段	类型	名称
fundintrrate	float	融资利率
stkintrrate	float	融券利率
punishintrrate	float	罚息利率
creditstatus	str	信用状态

marginrates	float	维持担保比例
realrate	float	实时担保比例
asset	float	总资产
liability	float	总负债
marginavl	float	保证金可用数
fundbal	float	资金余额
fundavl	float	资金可用数
dsaleamtbal	float	融券卖出所得资金
guaranteeout	float	可转出担保资产
gagemktavl	float	担保证券市值
fdealavl	float	融资本金
ffee	float	融资息费
ftotaldebts	float	融资负债合计
dealfmktavl	float	应付融券市值
dfee	float	融券息费
dtotaldebts	float	融券负债合计
fcreditbal	float	融资授信额度
fcreditavl	float	融资可用额度
fcreditfrz	float	融资额度冻结
dcreditbal	float	融券授信额度
dcreditavl	float	融券授信额度
dcreditfrz	float	融券额度冻结
rights	float	红利权益
serviceuncomerqrights	float	红利权益(在途)
rightsqty	float	红股权益
serviceuncomerqrightsqty	float	红股权益(在途)
acreditbal	float	总额度
acreditavl	float	总可用额度
acashcapital	float	所有现金资产（所有资产、包括融券卖出）
astkmktvalue	float	所有证券市值（包含融资买入、非担保品）
withdrawable	float	可取资金
netcapital	float	净资产
fcreditpnl	float	融资盈亏
dcreditpnl	float	融券盈亏
fcreditmarginoccupied	float	融资占用保证金
dcreditmarginoccupied	float	融券占用保证金
collateralbuyableamt	float	可买担保品资金
repayableamt	float	可还款金额

dcreditcashavl	float	融券可用资金

示例代码

```
credit_get_cash(account_id='')
```

示例返回值

```
marginrates realrate asset         liability  marginavl         fundbal       fundavl
guaranteeout gagemktavl  fdealavl   ffee   ftotaldebts dfee fcreditbal fcreditavl
dcreditbal dcreditavl acreditbal acreditavl  account_id
account_name     rid                                  fundintrrate  stkintrrate
punishintrrate creditstatus dsaleamtbal dealfmktavl dtotaldebts fcreditfrz dcreditfrz
rights    serviceuncomerqrights rightsqty serviceuncomerqrightsqty acashcapital
astkmktvalue withdrawable netcapital fcreditpnl dcreditpnl fcreditmarginoccupied
dcreditmarginoccupied collateralbuyableamt repayableamt dcreditcashavl
---------- -------- ------------ ---------- ---------------- ------------ ---------
---- ----------- ----------- ---------- ------ ----------- ---- ---------- -----------
------- ---------- ---------- ---------- ----------- --------------------------------
-- ---------------- ------------------------------------ ------------ ----------- ----
---------- ------------ ----------- ----------- ----------- ---------- ------------ ---
------ --------------------- --------- ---------------------- ------------ ----------
-- ------------ ---------- ---------- ---------- -------------------- ----------------
----- -------------------- ------------ --------------
229.5157    229.5157 1001926287.62 4356531.11 1994995956.360447 1002066280.62
2002149925.25 988838893.26 -4483562.29 4307241.23 668.89 4307910.12  1.28 8000000.0
3683189.8499999996 8000000.0  8000001.0  16000000.0 11683190.85 8f30e83f-a7c5-11ea-
b510-309c231d28bd 001515018318     3978e606-0fc0-43ec-87b0-01887fa280c5 0.0
0.0         0.0                         0.0         0.0         0.0         0.0
0.0          0.0       0.0                   0.0       0.0                     0.0
0.0          0.0          0.0        0.0        0.0        0.0                  0.0
0.0                  0.0          0.0
```

credit_repay_share_by_buying_share - 买券还券

函数原型

```
credit_repay_share_by_buying_share(symbol, volume, price, order_type=OrderType_Limit,
                                   order_duration=OrderDuration_Unknown,
                                   order_qualifier=OrderQualifier_Unknown,
account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
price	float	价格
order_type	int	参见订单委托类型

order_duration	int	参见 委托时间属性
order_qualifier	str	参见 委托成交属性
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值：

请参考 `order` 返回值字段说明

示例代码

```
credit_repay_share_by_buying_share(symbol='SHSE.600000', volume=100, price=10.67,
order_type=OrderType_Limit,
                                   order_duration=OrderDuration_Unknown,
                                   order_qualifier=OrderQualifier_Unknown,
                                   account_id='')
```

示例返回值

```
strategy_id                          account_id                          cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ -------------
---------------------- ----------- ---------- ------ ----- ------ --------------------
------------------------------------------------- -------------- ------------ -------
- --------- ------------- ---- --------------- ------------- -------------- -----------
---- --------- -------------- --------------------- ---------- ----------- ----- ------
- ------------ ------------ -------------- ------------- ----------- ------------- ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b857e02-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1           10     10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 865536, tzinfo=tzfile('PRC')) 202
0    0               0             0              0               0         0
0.0        0           0.0   0.0     0             0.0          0.0            0
0.0         0.0           0.0               None
```

credit_repay_cash_by_selling_share - 卖券还款

函数原型

```
credit_repay_cash_by_selling_share(symbol, volume, price, order_type=OrderType_Limit,
                                   order_duration=OrderDuration_Unknown,
                                   order_qualifier=OrderQualifier_Unknown,
account_id='')
```

参数：

参数名	类型	说明

symbol	str	标的代码
volume	int	数量
price	float	价格
order_type	int	参见订单委托类型
order_duration	int	参见 委托时间属性
order_qualifier	str	参见 委托成交属性
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值:

请参考 `order` 返回值字段说明

示例代码

```
credit_repay_cash_by_selling_share(symbol='SHSE.600000', volume=100, price=10.67,
order_type=OrderType_Limit,
                                   order_duration=OrderDuration_Unknown,
                                   order_qualifier=OrderQualifier_Unknown,
                                   account_id='')
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ -------------
---------------------- ----------- ---------- ------ ----- ------ --------------------
------------------------------------------------- -------------- ------------ -------
- --------- ------------- ---- --------------- ------------- -------------- -----------
---- --------- -------------- --------------------- ---------- ----------- ----- ------
- ------------ ------------ -------------- ------------ ----------- ------------- ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b85a50a-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10     10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 866535, tzinfo=tzfile('PRC')) 203
0    0               0             0              0               0         0
0.0        0           0.0   0.0     0             0.0          0.0            0
0.0         0.0           0.0               None
```

credit_buying_on_collateral - 担保品买入

函数原型

```
credit_buying_on_collateral(symbol, volume, price,
                            order_type=OrderType_Limit,
```

```
                                    order_duration=OrderDuration_Unknown,
                                    order_qualifier=OrderQualifier_Unknown, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
price	float	价格
order_type	int	参见订单委托类型
order_duration	int	参见 委托时间属性
order_qualifier	str	参见 委托成交属性
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `list[dict]`

请参考 `order` 返回值字段说明

示例代码

```
credit_buying_on_collateral(symbol='SHSE.600000', volume=100, price=10.67,
order_type=OrderType_Limit,
                                                order_duration=OrderDuration_Unknown,
                                                order_qualifier=OrderQualifier_Unknown,
                                                account_id='')
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ ------------
---------------------- ---------- ---------- ------ ----- ------ --------------------
------------------------------------------------- -------------- ------------ -------
- --------- ------------ ---- -------------- ------------ ------------- -----------
---- --------- -------------- --------------------- ---------- ---------- ----- ------
- ------------ ------------ -------------- ------------ ----------- ------------- ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b861a31-
a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10     10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 869534, tzinfo=tzfile('PRC')) 207
0    0               0             0              0               0         0
0.0        0           0.0   0.0     0             0.0          0.0           0
0.0         0.0           0.0              None
```

credit_selling_on_collateral - 担保品卖出

函数原型

```
credit_selling_on_collateral(symbol, volume, price,
                             order_type=OrderType_Limit,
                             order_duration=OrderDuration_Unknown,
                             order_qualifier=OrderQualifier_Unknown, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
price	float	价格
order_type	int	参见订单委托类型
order_duration	int	参见 委托时间属性
order_qualifier	str	参见 委托成交属性
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `list[dict]`

请参考order 返回值字段说明

示例代码

```
credit_selling_on_collateral(symbol='SHSE.600000', volume=100, price=10.67,
order_type=OrderType_Limit,
                             order_duration=OrderDuration_Unknown,
                             order_qualifier=OrderQualifier_Unknown,
                             account_id='')
```

示例返回值

```
strategy_id                          account_id                           cl_ord_id
symbol      order_type status price volume created_at
order_business account_name order_id ex_ord_id algo_order_id side position_effect
position_side order_duration order_qualifier order_src ord_rej_reason
ord_rej_reason_detail stop_price order_style value percent target_volume target_value
target_percent filled_volume filled_vwap filled_amount filled_commission updated_at
------------------------------------ ------------------------------------ -------------
---------------------- ---------- ---------- ------ ----- ------ --------------------
------------------------------------------------- -------------- ------------ -------
- --------- ------------- ---- --------------- ------------ -------------- -----------
---- --------- -------------- ---------------------- ---------- ---------- ----- ------
- ------------ ------------ -------------- ------------ ----------- ------------- ---
-------------- ----------
3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b861a31-
```

```
a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10     10.67 100
datetime.datetime(2020, 6, 6, 15, 41, 44, 869534, tzinfo=tzfile('PRC')) 208
0    0               0             0              0               0         0
0.0        0           0.0   0.0     0             0.0          0.0           0
0.0         0.0           0.0               None
```

credit_collateral_in - 担保品转入

函数原型

```
1. credit_collateral_in(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `list[dict]`

请参考 order 返回值字段说明

示例代码

```
1. credit_collateral_in(symbol='SHSE.600000', volume=100, account_id='')
```

示例返回值

```
1. strategy_id                          account_id                          cl_ord_id
   symbol      order_type status volume created_at
   order_business account_name order_id ex_ord_id algo_order_id side position_effect
   position_side order_duration order_qualifier order_src ord_rej_reason
   ord_rej_reason_detail price stop_price order_style value percent target_volume
   target_value target_percent filled_volume filled_vwap filled_amount filled_commission
   updated_at
2. ------------------------------------ ------------------------------------ ------------
   ---------------------- ----------- ---------- ------ ------ --------------------------
   ------------------------------------------------ -------------- ------------ -------- ----
   ----- ------------ ---- --------------- ------------ -------------- --------------- -
   -------- -------------- --------------------- ----- ---------- ---------- ----- ------
   - ------------ ------------ -------------- ------------ ----------- ------------ ---
   -------------- ----------
3. 3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b868f72-
   a7c9-11ea-b510-309c231d28bd SHSE.600000 1          10     100
   datetime.datetime(2020, 6, 6, 15, 41, 44, 872536, tzinfo=tzfile('PRC')) 209
   0    0               0             0              0               0         0
   0.0   0.0        0           0.0   0.0     0             0.0          0.0           0
   0.0         0.0           0.0               None
```

```
4. ...... ......
```

credit_collateral_out - 担保品转出

函数原型

```
1. credit_collateral_out(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	数量
account_id	str	账号id，不填或留空，表示使用默认账号，否则使用指定账号

返回值: `list[dict]`

请参考 order 返回值字段说明

示例代码

```
1. credit_collateral_out(symbol='SHSE.600000', volume=100, account_id='')
```

示例返回值

```
1. strategy_id                          account_id                          cl_ord_id
   symbol      order_type status volume created_at
   order_business account_name order_id ex_ord_id algo_order_id side position_effect
   position_side order_duration order_qualifier order_src ord_rej_reason
   ord_rej_reason_detail price stop_price order_style value percent target_volume
   target_value target_percent filled_volume filled_vwap filled_amount filled_commission
   updated_at
2. ------------------------------------ ------------------------------------ ------------
   ---------------------- ----------- ---------- ------ ------ --------------------------
   ------------------------------------------- -------------- ------------ -------- ----
   ----- ------------ ---- --------------- ------------ -------------- --------------- -
   -------- -------------- --------------------- ----- ---------- ---------- ----- ------
   - ------------ ------------ -------------- ------------ ----------- ------------- ---
   -------------- ----------
3. 3af55cb8-a7c5-11ea-b510-309c231d28bd 8f30e83f-a7c5-11ea-b510-309c231d28bd 2b868f72-
   a7c9-11ea-b510-309c231d28bd SHSE.600000 1           10     100
   datetime.datetime(2020, 6, 6, 15, 41, 44, 872536, tzinfo=tzfile('PRC')) 210
   0    0               0             0              0               0         0
   0.0   0.0        0           0.0   0.0     0             0.0          0.0           0
   0.0         0.0           0.0               None
4. ...... ......
```

算法交易函数

算法 SDK

python算法SDK包含在gm3.0.126版本及以上版本，不需要引入新库
仅支持实时模式，部分券商版本可用

algo_order 算法交易委托

委托算法母单

函数原型：

```
algo_order(symbol, volume, side, order_type,position_effect, price, algo_name, algo_param)
```

参数：

参数	类型	说明
symbol	str	标的代码
volume	int	数量
side	int	OrderSide_Buy = 1 买入OrderSide_Sell = 2 卖出
order_type	int	OrderType_Limit = 1 限价委托，OrderType_Market = 2 市价委托
position_effect	int	PositionEffect_Open = 1 开仓PositionEffect_Close = 2 平仓，
price	int	基准价格（ATS-SMART算法不生效）
algo_name	str	算法名称，ATS-SMART、ZC-POV
algo_param	dict	算法参数

返回值：

类型	说明
list[order]	委托对象列表，参见委托

当**algo_name = 'ATS-SMART'**时
algo_param的参数为

参数	类型	说明
start_time	str	开始时间

end_time_referred	str	结束参考时间(不能超过14:55:00)
time_end	str	结束时间(不能超过14:55:00)
end_time_valid	int	结束时间是否有效,如设为无效，则以收盘时间为结束时间， 1为有效， 0为无效
stop_sell_when_DL	int	涨停时是否停止卖出， 1为是，0为否
cancel_when_PL	int	跌停时是否撤单， 1为是， 0为否
min_trade_amount	int	最小交易金额

示例:

```
# 下算法母单，设定母单的执行参数
algo_param = {'start_time': '09:00:00', 'end_time_referred':'14:55:00', 'end_time':
'14:55:00', 'end_time_valid': 1, 'stop_sell_when_dl': 1,
              'cancel_when_pl': 0, 'min_trade_amount': 100000}
aorders = algo_order(symbol='SHSE.600000', volume=20000, side=OrderSide_Buy,
order_type=OrderType_Limit,
                     position_effect=PositionEffect_Open, price=5, algo_name='ATS-SMART',
algo_param=algo_param)
print(aorders)
```

输出:

```
[{'strategy_id': '6f534238-2883-11eb-a8fe-fa163ef85f63', 'account_id': '927f9095-27e5-
11eb-bb81-fa163ef85f63', 'account_name': '1001000002', 'cl_ord_id': '03f13690-2d64-
11eb-9e36-fa163ef85f63', 'symbol': 'SHSE.600000', 'side': 1, 'position_effect': 1,
'order_type': 1, 'status': 10, 'price': 5.0, 'order_style': 1, 'volume': 20000,
'created_at': datetime.datetime(2020, 11, 23, 16, 15, 15, 105141,
tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2020, 11, 23, 16, 15, 15,
105141,tzinfo=tzfile('PRC')), 'algo_name': 'ATS-SMART', 'algo_param':
'start_time&&1606093200||end_time_referred&&1606114500||end_time&&1606114500||end_time_va
 'order_id': '', 'ex_ord_id': '', 'position_side': 0, 'order_business': 0,
'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0,
'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0,
'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0,
'filled_volume': 0, 'filled_vwap': 0.0, 'filled_amount': 0.0, 'filled_commission': 0.0,
'algo_status': 0, 'algo_comment': ''}]
```

当algo_name = ‘ZC-POV’时
algo_param的参数为

参数	类型	说明
part_rate	float	市场参与率（0~45），单位%，默认30，即30%

示例:

```
# 下算法母单，设定母单的执行参数
algo_param = {"participation_rate" : 15}
aorder = algo_order(symbol=symbol, volume=1000, side=OrderSide_Buy,
order_type=OrderSide_Buy,
```

```
                position_effect=PositionEffect_Open, price=price, algo_name=algo_name,
algo_param=algo_param)
print(aorder)
```

输出：

```
[{'strategy_id': '6f534238-2883-11eb-a8fe-fa163ef85f63', 'account_id': '15b7afb1-e91d-
11eb-953b-025041000001', 'account_name': 'b6b2819b-e864-11eb-b146-00163e0a4100',
'cl_ord_id': 'e0f4ca3f-f97a-11eb-acee-165afc004509', 'symbol': 'SHSE.600007', 'side':
1, 'position_effect': 1, 'order_type': 1, 'status': 10, 'price': 15.470000267028809,
'order_style': 1, 'volume': 1000, 'created_at': datetime.datetime(2021, 8, 10, 9, 32,
52, 39737, tzinfo=tzfile('PRC')), 'updated_at': datetime.datetime(2021, 8, 10, 9, 32,
52, 42738, tzinfo=tzfile('PRC')), 'algo_name': 'ZC-POV', 'algo_param':
'TimeStart&&1628559000||TimeEnd&&1628578800||PartRate&&0.150000||MinAmount&&1000',
'order_id': '', 'ex_ord_id': '', 'position_side': 0, 'order_business': 0,
'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0,
'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0,
'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0,
'filled_volume': 0, 'filled_vwap': 0.0, 'filled_amount': 0.0, 'filled_commission': 0.0,
'algo_status': 0, 'algo_comment': '', 'properties': {}]
```

注意：回测模式不支持算法单

algo_order_cancel 撤销算法委托

撤销母单委托

函数原型：

```
algo_order_cancel(wait_cancel_orders)
```

参数：

参数	类型	说明
wait_cancel_orders	str	撤单算法委托. 传入单个字典. 或者list字典. 每个字典包含key:cl_ord_id key:account_id

cl_ord_id为委托id， account_id为账户id

返回值：

类型	说明
list[order]	委托对象列表，参见委托

示例：

```
aorders = get_algo_orders(account='')
wait_cancel_orders = [{'cl_ord_id': aorders[0]['cl_ord_id'], 'account_id': aorders[0]
['account_id']}]
algo_order_cancel(wait_cancel_orders)
```

get_algo_orders 查询算法委托

查询母单委托

函数原型：

```
algo_order_cancel(account)
```

参数：

参数	类型	说明
account	str	account_id 默认帐号时为 ‘’

返回值：

类型	说明
list[order]	委托对象列表，参见委托

示例：

```
get_algo_orders(account='')
```

输出：

```
[{'strategy_id': '6f534238-2883-11eb-a8fe-fa163ef85f63', 'account_id': '927f9095-27e5-
11eb-bb81-fa163ef85f63', 'account_name': '1001000002', 'cl_ord_id': 'fe0ec2d3-2d50-
11eb-9e36-fa163ef85f63', 'symbol': 'SHSE.510300', 'side': 1, 'position_effect': 1,
'order_type': 1, 'status': 10, 'price': 5.0, 'order_style': 1, 'volume': 20000,
'created_at': datetime.datetime(2020, 11, 23, 13, 59, 4,794594, tzinfo=tzfile('PRC')),
'updated_at': datetime.datetime(2020, 11, 23, 13, 59, 4, 795571, tzinfo=tzfile('PRC')),
'algo_name': 'ATS-SMART', 'algo_param':
'start_time&&1606093200||end_time_referred&&1606114500||end_time&&1606114500||end_time_va
 'order_id': '', 'ex_ord_id': '', 'position_side': 0, 'order_business': 0,
'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'position_src': 0,
'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0,
'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0,
'filled_volume': 0, 'filled_vwap': 0.0, 'filled_amount': 0.0, 'filled_commission': 0.0,
'algo_status': 0, 'algo_comment': ''}]
```

algo_order_pause 暂停或重启或者撤销算法委托

函数原型：

```
algo_order_pause(alorders)
```

参数：

	类	

	型	
alorders	str	传入单个字典. 或者list字典. 每个字典包含key:cl_ord_id, key:account_id key:algo_status

cl_ord_id为委托id， account_id为账户id，algo_status为算法单状态（1 - 重启 2 - 暂停 3 -暂停并撤子单）

返回值:

类型	说明
list[order]	委托对象列表，参见委托

```
aorders = get_algo_orders(account='')
# 暂停订单，修改订单结构的母单状态字段
alorders01 = [{'cl_ord_id': aorders[0]['cl_ord_id'], 'account_id': aorders[0]['account_id'], 'algo_status': 3}]
algo_order_pause(alorders01)
```

注意： ATS-SMART算法暂不支持此接口

get_algo_child_orders 查询算法委托的所有子单

函数原型:

```
get_algo_child_orders(cl_ord_id, account='')
```

参数:

参数	类型	说明
cl_ord_id	str	传入单个字典. 或者list字典. 每个字典包含key:cl_ord_id
account	str	account_id 默认帐号时为 ‘’

返回值:

类型	说明
list[order]	委托对象列表，参见委托

示例:

```
aorders = get_algo_orders(account='')
child_order= get_algo_child_orders(aorders[0]['cl_ord_id'], account='')
print(child_order[0])
```

输出:

```
[{'account_id': '17ceec74-2efb-11eb-b437-00ff5a669ee2', 'account_name': '0000001', 'cl_ord_id': '1606294231_9', 'order_id': '1606294231_9', 'symbol': 'SZSE.000001', 'side': 1, 'position_effect': 1, 'order_type': 1, 'status': 3, 'price': 19.06, 'volume': 100, 'filled_volume': 100, 'filled_vwap': 19.06, 'filled_amount': 1905.9999999999998, 'algo_order_id': '453b3064-2efb-11eb-b437-00ff5a669ee2',
```

```
1905.9999999999998, 'algo_order_id': '453b3064-2efb-11eb-b437-00ff5a669ee2',
'strategy_id': '', 'ex_ord_id': '', 'position_side': 0, 'order_business': 0,
'order_duration': 0, 'order_qualifier': 0, 'order_src': 0, 'ord_rej_reason': 0,
'ord_rej_reason_detail': '', 'stop_price': 0.0, 'order_style': 0, 'value': 0.0,
'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0,
'filled_commission': 0.0, 'created_at': None, 'updated_at': None}]
```

on_algo_order_status 算法单状态事件

响应算法单状态更新事情，下算法单后状态更新时被触发

函数原型：

```
1. on_algo_order_status(context, algo_order)
```

参数：

参数名	类型	说明
context	context	上下文
algo_order	order	委托

示例：

```
1. def on_algo_order_status(context, algo_order):
2.     print(algo_order)
```

输出：

```
1. {'strategy_id': '6f534238-2883-11eb-a8fe-fa163ef85f63', 'account_id': '927f9095-27e5-
   11eb-bb81-fa163ef85f63', 'account_name': '1001000002', 'cl_ord_id': '09baa735-2e01-
   11eb-ab6f-fa163ef85f63','symbol': 'SHSE.600000', 'side': 1, 'position_effect':
   1,'order_type': 1, 'status':1, 'price': 5.0, 'order_style': 1, 'volume': 20000,
   'created_at':datetime.datetime(2020,11, 24, 10, 59, 15, 800453, tzinfo=tzfile('PRC')),
   'updated_at': datetime.datetime(2020, 11, 24, 10, 59, 17, 922523,
   tzinfo=tzfile('PRC')), 'algo_name': 'ATS-SMART', 'algo_param':
   'start_time&&1606179600||end_time_referred&&1606200900||end_time&&1606200900||end_time_va
    'order_id': '', 'ex_ord_id':'', 'position_side': 0, 'order_business': 0,
   'order_duration': 0, 'order_qualifier':0, 'order_src': 0, 'position_src': 0,
   'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0, 'value': 0.0,
   'percent': 0.0, 'target_volume': 0, 'target_value': 0.0, 'target_percent': 0.0,
   'filled_volume': 0, 'filled_vwap': 0.0, 'filled_amount': 0.0, 'filled_commission': 0.0,
   'algo_status':0, 'algo_comment': ''}
```

新股交易函数

新股 SDK

ipo_buy - 新股申购

仅在实盘中可以使用

函数原型:

```
ipo_buy(symbol, volume, price, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	申购数量
price	float	新股发行价
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

ipo_get_quota - 查询新股申购额度

仅在实盘中可以使用

函数原型:

```
ipo_get_quota(account_id='')
```

参数:

参数名	类型	说明
account_id	str	账户ID，不指定则使用默认账户

返回值:

`List[Dict[str, Any]]`

key	value类型	说明

quota	float	可申购数量

ipo_get_instruments - 查询当日新股清单

仅在实盘中可以使用

函数原型:

```
1. ipo_get_instruments(sec_type, account_id='', df=False)
```

参数:

参数名	类型	说明
sec_type	int	标的类型，用以区别获取新股还是新债
account_id	str	账户ID，不指定则使用默认账户
df	bool	是否返回 DataFrame

返回值:

`List[Dict]`

key	value类型	说明
symbol	str	标的代码
price	float	申购价格
min_vol	int	申购最小数量
max_vol	int	申购最大数量

ipo_get_match_number - 查询配号

仅在实盘中可以使用

函数原型:

```
1. ipo_get_match_number(start_time, end_time, account_id='', df=False)
```

参数:

参数名	类型	说明
start_time	str	开始时间， (%Y-%m-%d %H:%M:%S 格式)
end_time	str	结束时间， (%Y-%m-%d %H:%M:%S 格式)
account_id	str	账户ID，不指定则使用默认账户
df	bool	是否返回 DataFrame

返回值:

`List[Dict]`

key	value类型	说明
symbol	str	标的代码
order_id	str	委托号
volume	int	成交数量
match_number	str	申购配号
order_at	datetime.datetime	委托日期
match_at	datetime.datetime	配号日期

ipo_get_lot_info - 中签查询

仅在实盘中可以使用

函数原型:

```
ipo_get_lot_info(account_id='', df=False)
```

参数:

参数名	类型	说明
account_id	str	账户ID，不指定则使用默认账户
df	bool	是否返回 DataFrame

返回值:

`List[Dict]`

key	value类型	说明
symbol	str	标的代码
order_at	datetime.datetime	委托日期
lot_at	datetime.datetime	中签日期
lot_volume	int	中签数量
give_up_volume	int	放弃数量
price	float	中签价格
amount	float	中签金额
pay_volume	float	已缴款数量
pay_amount	float	已缴款金额

基金交易函数

基金业务 SDK

fund_etf_buy - ETF申购

仅在实盘中可以使用

```
fund_etf_buy(symbol, volume, price, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	申购数量
price	float	申购价格
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

fund_etf_redemption - ETF赎回

仅在实盘中可以使用

```
fund_etf_redemption(symbol, volume, price, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	赎回数量
price	float	赎回价格
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

fund_subscribing - 基金认购

仅在实盘中可以使用

```
1. fund_subscribing(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

fund_buy - 基金申购

仅在实盘中可以使用

```
1. fund_buy(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	申购数量
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

fund_redemption - 基金赎回

仅在实盘中可以使用

```
1. fund_redemption(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

债券交易函数

债券业务 SDK

bond_reverse_repurchase_agreement - 国债逆回购

仅在实盘中可以使用

```
bond_reverse_repurchase_agreement(symbol, volume, price, order_type=OrderType_Limit,
order_duration=OrderQualifier_Unknown, order_qualifier=OrderQualifier_Unknown,
account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量
price	float	价格
order_type	int	委托类型
order_duration	int	委托时间属性
order_qualifier	int	委托成交属性
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict]`

注意：逆回购1张为100元。上海市场最少交易1000张，深圳最少交易10张。且上海数量必须是1000张的整数倍，深圳数量必须是10张的整数倍，也即上海要10万元一个单位的买，深圳一千元一个单位的买。

bond_convertible_call - 可转债转股

仅在实盘中可以使用

```
bond_convertible_call(symbol, volume, price=0.0, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量

price	float	价格
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict[Text, Any]]`

bond_convertible_put - 可转债回售

仅在实盘中可以使用

```
1. bond_convertible_put(symbol, volume, price=0.0, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量
price	float	价格
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict[Text, Any]]`

bond_convertible_put_cancel - 可转债回售撤销

仅在实盘中可以使用

```
1. bond_convertible_put_cancel(symbol, volume, account_id='')
```

参数:

参数名	类型	说明
symbol	str	标的代码
volume	int	认购数量
account_id	str	账户ID，不指定则使用默认账户

返回值 `List[Dict[Text, Any]]`

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。
注意：
1、交易账户重连后，会重新推送一遍交易账户登录成功后查询回来的所有委托
2、撤单拒绝，会推送撤单委托的最终状态

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 init(context):
    # 记录委托id
    context.cl_ord_id = {}
    order_list = get_orders()
    if order_list:
        context.cl_ord_id = {i['cl_ord_id']: {'status': i['status'], 'filled_volume': i['filled_volume']} for i in order_list}


def on_bar(context, bars):
    # 记录下单后对应的委托id，方便在on_order_status里追踪，实时模式下下单后会立刻返回10待报状态，其他状态需要通过on_order_status事件监控
    order = order_volume(symbol=symbol, volume=volume, side=OrderSide_Sell, order_type=OrderType_Limit,
                         position_effect=PositionEffect_CloseToday,
                         price=price_1)
    context.cl_ord_id[order[0]['cl_ord_id']] = {}
    context.cl_ord_id[order[0]['cl_ord_id']]['status'] = order[0]['status']
    context.cl_ord_id[order[0]['cl_ord_id']]['filled_volume'] = order[0]['filled_volume']


def on_order_status(context, order):
    # 过滤非此策略下单的委托和重复状态的委托
    if order.strategy_id == context.strategy_id and order.cl_ord_id in
```

```
    context.cl_ord_id.keys():
        if order.status != context.cl_ord_id[order.cl_ord_id]['status']:
            context.cl_ord_id[order.cl_ord_id]['status'] = order['status']
            context.cl_ord_id[order.cl_ord_id]['filled_volume'] =
order['filled_volume']
        else:
            # 部分成交状态时，根据已成交量判断order是不是最新的
            if order.status == 2 and order.filled_volume !=
context.cl_ord_id[order.cl_ord_id]['filled_volume']:
                context.cl_ord_id[order.cl_ord_id]['filled_volume'] =
order['filled_volume']
            else:
                return

    # 可在后面执行其他处理逻辑
    print(order)
```

输出：

```
{'strategy_id': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'account_id': 'd7443a53-f65b-
11ea-bb9d-484d7eaefe55', 'cl_ord_id': '000000000', 'symbol': 'SHSE.600000', 'side': 1,
'position_effect': 1, 'position_side': 1, 'order_type': 1, 'status': 3, 'price': 11.0,
'order_style': 1, 'volume': 18181800, 'value': 200000000.0, 'percent': 0.1,
'target_volume': 18181800, 'target_value': 199999800.0, 'target_percent': 0.1,
'filled_volume': 18181800, 'filled_vwap': 11.0011, 'filled_amount': 200019799.98,
'created_at': datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'updated_at':
datetime.datetime(2020, 9, 1, 9, 40, tzinfo=tzfile('PRC')), 'filled_commission':
20001.979998, 'account_name': '', 'order_id': '', 'ex_ord_id': '', 'algo_order_id': '',
'order_business': 0, 'order_duration': 0, 'order_qualifier': 0, 'order_src': 0,
'position_src': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'stop_price': 0.0}
```

on_execution_report - 委托执行回报事件

响应委托被执行事件，委托成交或者撤单拒绝后被触发。
注意：
1、交易账户重连后，会重新推送一遍交易账户登录成功后查询回来的所有执行回报
2、撤单拒绝后，会推送撤单拒绝执行回报，可以根据exec_id区分

函数原型：

```
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：

```
def init(context):
    # 记录成交id
    context.exec_id = []
    exec_rpt_list = get_execution_reports()

    if exec_rpt_list:
        context.exec_id = [i['exec_id'] for i in exec_rpt_list]

def on_execution_report(context, execrpt):
    # 过滤掉非此策略和成交类型不是15成交的回报
    if execrpt.exec_type == 15 and execrpt.exec_id not in context.exec_id and execrpt.strategy_id == context.strategy_id:
        context.exec_id.append(execrpt.exec_id)
        # 后面可以执行其他处理逻辑
        print(execrpt)
```

输出:

```
{'strategy_id': 'e41dec22-2d72-11eb-b043-00ff5a669ee2', 'cl_ord_id': '000000247', 'symbol': 'SHSE.600000', 'position_effect': 1, 'side': 1, 'exec_type': 15, 'price': 12.640000343322754, 'volume': 200, 'amount': 2528.000068664551, 'account_id': '', 'account_name': '', 'order_id': '', 'exec_id': '', 'order_business': 0, 'ord_rej_reason': 0, 'ord_rej_reason_detail': '', 'commission': 0.0, 'cost': 0.0, 'created_at': None}
```

on_account_status - 交易账户状态更新事件

响应交易账户状态更新事件，交易账户状态变化时被触发。
函数原型:

```
def on_account_status(context, account):
    print(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),status(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。

示例:

```
def on_account_status(context, account):
    print(account)
```

输出:

```
{'account_id': '4946cc09-fefd-11ea-ac6a-52560acd7da0', 'account_name': '', 'status':
```

```
    {'state': 5, 'error': {'code': 0, 'type': '', 'info': ''}}}
```

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

动态参数仅在仿真交易和实盘交易下生效， 可在终端设置和修改。

动态参数通过策略调用接口实现策略和掘金界面参数交互， 在不停止策略运行的情况下，界面修改参数（移开光标，修改就会生效）会对策略里的指定变量做修改

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. context.k_value = 80
```

```
add_parameter(key='k_value', value=context.k_value, min=0, max=100, name='k值阀值',
intro='调整k值', group='1', readonly=False)
```

set_parameter - 修改已经添加过的动态参数

注意：需要保持key键名和添加过的动态参数的key一致，否则不生效，无报错

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：

```
context.k_xl = 0.3
set_parameter(key='k_value', value=context.k_xl, min=0, max=1, name='k值斜率', intro='调
整k值斜率', group='1', readonly=False)
```

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：

```
{'key': 'k_value', 'value': 80.0, 'max': 100.0, 'name': 'k值阀值', 'intro': '调整k值', 'group': '1', 'min': 0.0, 'readonly': False}
```

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

示例：

```
print(context.parameters)
```

输出：

```
{'k_value': {'key': 'k_value', 'value': 80.0, 'max': 100.0, 'name': 'k值阀值', 'intro': 'k值阀值', 'group': '1', 'min': 0.0, 'readonly': False}, 'd_value': {'key': 'd_value', 'value': 20.0, 'max': 100.0, 'name': 'd值阀值', 'intro': 'd值阀值', 'group': '1', 'min': 0.0, 'readonly': False}}
```

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

注意：
token输入错误会报错“错误或无效的token”。

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。

4.重启终端日志记录会自动清除，需要记录日志到本地的，可以使用Python的logging库

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)
```

输出：

```
b'\xe6\x97\xa0\xe6\xb3\x95\xe8\x8e\xb7\xe5\x8f\x96\xe6\x8e\x98\xe9\x87\x91\xe6\x9c\x8d\xe
```

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()
print(version)
```

输出：

```
3.0.127
```

其他事件

on_backtest_finished - 回测结束事件

描述:
在回测模式下，回测结束后会触发该事件，并返回回测得到的绩效指标对象

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': 'd7443a53-f65b-11ea-bb9d-484d7eaefe55', 'pnl_ratio':
-0.007426408687162637, 'pnl_ratio_annual': -1.3553195854071813, 'sharp_ratio':
-15.034348187048744, 'max_drawdown': 0.0009580714324989177, 'risk_ratio':
0.10010591267452242, 'open_count': 1, 'close_count': 1, 'lose_count': 1,
'calmar_ratio': -1414.6331259164358, 'win_count': 0, 'win_ratio': 0.0, 'created_at':
None, 'updated_at': None}
```

on_error - 错误事件

描述:
当发生异常情况，比如断网时、终端服务崩溃是会触发

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):
    print('code:{}, info:{}'.format(code, info))
    stop()
```

返回:

```
code:1201, info:实时行情服务连接断开
```

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

描述:
实时行情网络连接时触发，比如策略实时运行启动后会触发、行情断连又重连后会触发

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 ('实时行情网络连接成功')
```

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

描述:
目前监控SDK的交易和终端的链接情况，终端之后部分暂未做在内。账号连接情况可通过终端内账户连接指示灯查看

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 ('交易通道网络连接')
```

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

函数原型：

描述：
实时行情网络断开时触发，比如策略实时运行行情断连会触发

```
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 ('实时行情网络连接')
```

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

描述：
目前监控SDK的交易和终端的链接情况，终端交易服务崩溃后会触发，终端之后部分暂未做在内。账号连接情况可通过终端内账户连接指示灯查看

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 ('交易通道网络连接失败')
```

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 （无效）
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  （还不支持）
```

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，请参考

```
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)
OrderDuration_AHT = 8           # 盘后定价交易(after hour trading)
```

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，请参考

```
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)
```

OrderBusiness - 委托业务类型

```
OrderBusiness_NORMAL = 0           # 普通交易。默认值为空，以保持向前兼容


OrderBusiness_CREDIT_BOM = 200      # 融资买入(buying on margin)
OrderBusiness_CREDIT_SS = 201       # 融券卖出(short selling)
OrderBusiness_CREDIT_RSBBS = 202    # 买券还券(repay share by buying share)
OrderBusiness_CREDIT_RCBSS = 203    # 卖券还款(repay cash by selling share)
OrderBusiness_CREDIT_DRS = 204      # 直接还券(directly repay share)

# 直接还款: 不通过委托，参考接口...
OrderBusiness_CREDIT_BOC = 207      # 担保品买入(buying on collateral)
OrderBusiness_CREDIT_SOC = 208      # 担保品卖出(selling on collateral)
OrderBusiness_CREDIT_CI = 209       # 担保品转入(collateral in)
OrderBusiness_CREDIT_CO = 210       # 担保品转出(collateral out)
OrderBusiness_BOND_CONVERTIBLE_CALL = 402        #可转债转股
OrderBusiness_BOND_CONVERTIBLE_PUT = 403         #可转债回售
OrderBusiness_BOND_CONVERTIBLE_PUT_CANCEL = 404  #可转债回售撤销
```

ExecType - 执行回报类型

```
ExecType_Unknown = 0
ExecType_Trade = 15                 # 成交
```

```
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

（仿真有效，实盘需要参考具体的拒绝原因）

```
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 - 委托风格

```
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_CREDIT = 6                             # 信用交易
SEC_TYPE_BOND = 7                               # 债券
SEC_TYPE_BOND_CONVERTIBLE = 8                   # 可转债
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
```

PositionSrc - 头寸来源(仅适用融券融券)

```
PositionSrc_Unknown = 0
PositionSrc_L1 = 1                      #普通池
PositionSrc_L2 = 2                      #专项池
```

AlgoOrderStatus 算法单状态,暂停/恢复算法单时有效

```
AlgoOrderStatus_Unknown = 0,
AlgoOrderStatus_Resume = 1,                    #恢复母单
AlgoOrderStatus_Pause = 2,                     #暂停母单
AlgoOrderStatus_PauseAndCancelSubOrders = 3   #暂停母单并撤子单
```

错误码

错误码	描述	解决方法
0	成功	
1000	错误或无效的token	检查下token是否有误
1001	无法连接到终端服务	检查是否开启了掘金终端
1010	无法获取掘金服务器地址列表	检查是否开启了掘金终端
1013	交易服务调用错误	检查终端是否正常或重启掘金终端
1014	历史行情服务调用错误	在微信群或者QQ群通知技术支持
1015	策略服务调用错误	检查终端是否正常或重启掘金终端
1016	动态参数调用错误	检查动态参数设置
1017	基本面数据服务调用错误	在微信群或者QQ群通知技术支持
1018	回测服务调用错误	重启掘金终端、重新运行策略
1019	交易网关服务调用错误	检查终端是否正常或重启掘金终端
1020	无效的ACCOUNT_ID	检查账户id是否填写正确
1021	非法日期格式	对照帮助文档修改日期格式， 检查run()回测日期是否正确
1027	输入参数有误	检查定时任务的频率参数，实时模式只支持1d
1100	交易消息服务连接失败	检查终端是否正常或重启掘金终端
1101	交易消息服务断开	一般不用处理，等待自动重连
1200	实时行情服务连接失败	一般不用处理，等待自动重连
1201	实时行情服务连接断开	一般不用处理，等待自动重连
1202	实时行情订阅失败	订阅代码标的数量超过账户权限，联系商务咨询权限
1300	初始化回测失败	检查终端是否启动或策略是否连接到终端
1301	回测时间区间错误	检查回测时间是否超出范围