**掘金量化API说明**

目录

[**掘金量化API说明** 1](#_Toc510024130)

[**快速开始** 2](#_Toc510024131)

[下载SDK 2](#_Toc510024132)

[安装配置 2](#_Toc510024133)

[我的第一个策略 2](#_Toc510024134)

[**Python API范例** 3](#_Toc510024135)

[策略接口范例 3](#_Toc510024136)

[数据接口范例 7](#_Toc510024137)

[交易接口范例 12](#_Toc510024138)

[错误与版本等接口范例 23](#_Toc510024139)

[**Python API接口** 24](#_Toc510024140)

[枚举常量定义 24](#_Toc510024141)

[交易数据类型 26](#_Toc510024142)

[行情数据类型 30](#_Toc510024143)

[策略初始化方法 35](#_Toc510024144)

[策略启停方法 39](#_Toc510024145)

[行情订阅方法 39](#_Toc510024146)

[数据提取方法 41](#_Toc510024147)

[交易接口方法 50](#_Toc510024148)

[行情事件方法 62](#_Toc510024149)

[交易事件方法 63](#_Toc510024150)

[其他方法 65](#_Toc510024151)

**快速开始**

下载SDK

掘金量化平台提供策略开发服务包(SDK)用于策略开发者实现自己的策略。SDK下载地址请点击[这里](http://www.myquant.cn/downloads/)。 Python SDK支持Windows + Python2.7/3.6 + 32位/64位、Linux Python2/3 x64共六种版本，下载时找到对应本地安装的Python版本的SDK包。需要注意的是，此处所指的32位/64位不是指系统的版本，而是本地Python的版本。

安装配置

* 在Windows系统，需要解压下载的SDK包，双击安装程序进行安装。在安装过程中，安装程序会自行寻找Python安装路径，若提示找不到Python，需要手动添加注册信息或重新安装Python。若Python3.6不是安装在个人文件夹下，需要以管理员权限运行SDK安装程序。
* 在Linux系统，解压ZIP文件，运行pip install xxxx.whl命令进行安装；python3使用pip3 install xxxxx.whl 。Wheel文件名称以发布为准。

我的第一个策略

方式一，直接使用例子运行

解压SDK包，使用examples中的例子项目。

方式二，创建自己的策略

1. 参考[这里](https://www.myquant.cn/gm2/docs/terminal/)注册账号，登录终端。
2. 参考[这里](https://www.myquant.cn/gm2/docs/terminal/#_4)创建策略基本架构。编程语言选择Python，订阅上交所浦发银行(证券代码：600000)的Tick行情。
3. 编写自己的策略逻辑。

以下是一个完整的策略代码示例，策略逻辑：每收到一笔Tick行情，以最新价买入100股。

**from** gmsdk.api **import** StrategyBase

**class** **MyStrategy**(StrategyBase):

**def** **\_\_init\_\_**(self, \*args, \*\*kwargs):

super(MyStrategy, self).\_\_init\_\_(\*args, \*\*kwargs)

**def** **on\_tick**(self, tick):

self.open\_long(tick.exchange, tick.sec\_id, tick.last\_price, 100)

print("OpenLong: exchange %s, sec\_id %s, price %s" %

(tick.exchange, tick.sec\_id, tick.last\_price))

**if** \_\_name\_\_ == '\_\_main\_\_':

ret = MyStrategy(

username='username',

password='password',

strategy\_id='strategy\_2',

subscribe\_symbols='SHSE.600000.tick',

mode=2

).run()

print(('exit code: ', ret))

4.编译策略并运行。策略运行起来后，控制台打印策略的每一笔下单记录，在掘金终端的模拟交易中可查看该策略的运行详情。

**Python API范例**

策略接口范例

策略构建

如何构建策略

构建策略的三种方式：

1.掘金终端构建自己的策略，然后在构建的策略类中重写基类方法

2.修改SDK包中的例子，构建自己的策略类

3.参考API定义文档，自定义策略类

策略运行

策略类应继承自StrategyBase基类，并以方法重写的方式满足策略开发需求。然后调用子类实例的run()方法，运行策略。

**class** **Strategy**(StrategyBase):

...

strategy.run()

如何初始化策略

策略在运行前，必须先初始策略对象。需要指定挖金子账户、密码、策略ID、订阅代码和运行模式。

示例：策略初始化并订阅上交所浦发银行的实时[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据和1min的[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据

方式一：以参数方式初始化策略

ret = Strategy(

username='demo@myquant.cn',

password='123456',

strategy\_id='strategy\_2',

subscribe\_symbols='SHSE.600000.bar.60',

mode=2

)

策略支持四种运行模式及对应参数值：

* 1.不接收行情流：1
* 2.接收实时行情：2
* 3.模拟行情模式：3
* 4.回测模式：4

方式二：以配置文件方式初始化策略

策略初始化配置可以保存在\*.ini为文件中，然后以文件路径为参数初始化策略。

ret = Strategy(config\_file='strategy.ini')

strategy.ini文件配置示例：

**[strategy]**

*;掘金用户名*

username=-

*;掘金密码*

password=-

*;策略ID*

strategy\_id=3c7e70be-7e02-11e5-b293-5ec5d48ea63a

*;订阅证券代码或合约代码列表*

subscribe\_symbols=SHSE.600000.tick,SHSE.600000.bar.60

*;行情模式，2-实时行情模式，3-模拟行情模式，4-回放行情模式*

mode=2

*;交易服务地址，使用掘金终端交易时，地址为localhost:8001, 如果此项配置为空，则订单发往掘金云服务器*

td\_addr=localhost:8001

如何运行策略

完善策略逻辑后，先初始化策略，然后运行策略。策略运行后，策略开始接收数据并执行策略逻辑

strategy.run()

如何停止策略

1.直接关闭程序来停止策略

2.调用api方式停止策略

strategy.stop()

策略运行模式

策略支持三种运行模式：

1.实时行情模式：订阅行情服务器推送的实时行情，也就是交易所的实时行情，只在交易时段提供。适用的场景是策略仿真交易和实盘交易阶段。

2.模拟行情模式：模拟行情是近期的历史行情，行情服务器将7\*24小时不间断循环推送，推送频率近似实时行情。适用的场景是策略开发阶段，主要就是能方便随时随地都能有数据，能开发/调试策略，不受交易时段的限制。

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

如何回测策略

策略使用回测模式初始化，并配置回测相关参数，即进入回测。

示例：回测上交所浦发银行的daily数据, 回测时间为 2015-04-15 09:00:00到 2015-05-15 15:00:00, 策略初始资金为1,000,000, 委托成交为全部成交, 手续费率为零, 滑点比率为零, 数据使用前复权.

方式一：在策略初始化后调用backtest\_config()方法

strategy = Strategy(

username='demo@myquant.cn',

password='123456',

strategy\_id='strategy\_2',

subscribe\_symbols='SHSE.600000.bar.daily',

mode=4,

td\_addr='localhost:8001')

strategy.backtest\_config(

start\_time='2015-04-15 9:00:00',

end\_time='2015-05-15 15:00:00',

initial\_cash=1000000,

transaction\_ratio=1,

commission\_ratio=0,

slippage\_ratio=0,

price\_type=1,

bench\_symbol='SHSE.000300')

方式二：在配置文件中指定参数

strategy = Strategy(config\_file='strategy.ini')

在strategy.ini配置文件中添加backtest节点

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

*;策略回测参数配置*

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

**[backtest]**

*;历史数据回放开始时间*

start\_time=2015-04-15 09:00:00

*;历史数据回放结束时间*

end\_time=2015-05-15 15:00:00

*;策略初始资金*

initial\_cash=1000000

*;委托量成交比率，默认=1（每个委托100%成交）*

transaction\_ratio=1

*;手续费率，默认=0（不计算手续费）*

commission\_ratio=0

*;滑点比率，默认=0（无滑点）*

slippage\_ratio=0

*;行情复权模式,0=不复权,1=前复权*

price\_type=1

*;基准*

bench\_symbol=SHSE.000300

如何在非交易时间段调试策略

首先需[关联模拟交易通道](https://www.myquant.cn/gm2/docs/terminal/#_11)。

示例：订阅上交所浦发银行[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情和1min的[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)的模拟行情

方式一，通过参数的方式设置：

ret = strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.tick,SHSE.600000.bar.60",

3,

"localhost:8001")

方式二，通过配置方式初始化策略，策略的配置文件subscribe\_symbols和mode节点设置如下：

*;订阅证券代码或合约代码列表*

subscribe\_symbols=SHSE.600000.tick,SHSE.600000.bar.60

*;行情模式，2-实时行情模式，3-模拟行情模式，4-回放行情模式*

mode=3

如何设置策略仿真交易

首先需[关联仿真交易通道](https://www.myquant.cn/gm2/docs/terminal/#_11)。

示例：订阅上交所浦发银行[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情和1min的[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)的实时行情

方式一，通过参数的方式设置：

ret = strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.tick,SHSE.600000.bar.60",

2,

"localhost:8001")

方式二，通过配置方式初始化策略，策略的配置文件strategy节点设置如下：

*;订阅证券代码或合约代码列表*

subscribe\_symbols=SHSE.600000.tick,SHSE.600000.bar.60

*;行情模式，2-实时行情模式，3-模拟行情模式，4-回放行情模式*

mode=2

如何设置策略实盘交易

1.[关联实盘交易通道](https://www.myquant.cn/gm2/docs/terminal/#_14)。

2.参考如何设置策略仿真交易，接收实时行情即可开始实盘交易。

策略事件

策略提供哪些事件

策略的基类提供策略3类事件：

1.登录事件：策略初始化时触发

2.行情数据事件：接收实时行情数据时触发，主要有[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情事件和[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)行情事件

3.交易相关事件：交易时触发，主要有下单、撤单、订单回报事件

用户重写自己关注事件的回调方法完善策略逻辑。

策略登录

如何处理策略登录事件

策略在初始化时将触发登录事件，可以在自己的策略类中重写策略基类的[on\_login](https://www.myquant.cn/gm2/docs/api/python/#on_login)方法，以便策略初始化时进行自定义操作

**class** **Strategy**(StrategyBase):

**def** **on\_login**(self):

...

数据接口范例

行情类型

掘金SDK提供回测行情、模拟行情、实时行情及历史行情数据

1.回测行情、模拟行情和实时行情在订阅了symbol后，在策略的[on\_tick](https://www.myquant.cn/gm2/docs/api/python/#on_tick)方法和[on\_bar](https://www.myquant.cn/gm2/docs/api/python/#on_bar)方法中接收行情数据

2.各频率的历史行情数据提供对应的API提取。

数据定义

数据类型分为tick数据和bar数据

1.[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情是指按交易所实际发送的行情数据

2.[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据是指各种频率的行情数据，可订阅 1分、15分、60分的实时Bar数据。

3.日频数据提供[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据类型，日频仅在回测和历史数据提取时可用，不能实时订阅。

行情订阅方法

如何初始化行情接口

方式一，策略对象也提供行情接口，请参考如何初始化策略

方式二，获取行情对象实例，然后调用行情接口获取数据，该方式下仅获取数据

**from** gmsdk **import** md

ret = md.init("your user name", "your password")

如何订阅指定证券代码的tick行情

订阅上交所浦发银行和深交所平安银行的[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情数据，订阅的证券代码格式定义详细介绍请点击[这里](#forum.myquant.cn/t/topic/71)。

方式一，通过策略类对象订阅实时行情

1.策略初始化时订阅

ret = Strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.tick,SZSE.000001.tick",

2,

"localhost:8001")

2.策略程序运行中订阅

ret = strategy.subscribe("SHSE.600000.tick,SHSE.600000.bar.60")

方式二，通过行情类对象订阅实时行情，该方式仅提供行情数据

1.行情类对象初始化时订阅

**from** gmsdk **import** md

ret = md.init("your user name",

"your password",

2,

"SHSE.600000.tick,SZSE.000001.tick")

2.程序运行过程中订阅

ret = md.subscribe("SHSE.600000.tick,SZSE.000001.tick")

如何订阅指定证券代码的Bar行情

示例：订阅上交所浦发银行的1min [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)行情和深交所平安银行的30s [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)行情

方式一，通过策略类对象订阅实时行情

1.策略初始化时订阅

ret = strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.bar.60,SZSE.000001.bar.30",

3,

"localhost:8001")

2.策略运行过程中订阅

ret = strategy.subscribe("SHSE.600000.bar.60,SZSE.000001.bar.30")

方式二，通过行情类对象订阅实时行情，该方式仅提供行情数据

1.行情类对象初始化时订阅

ret = md.init("your user name",

"your password",

2,

"SHSE.600000.bar.60,SZSE.000001.bar.30")

2.程序运行过程中订阅

ret = md.subscribe("SHSE.600000.bar.60,SZSE.000001.bar.30")

如何订阅指定证券代码的日频行情

日频行情([DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar))仅在策略回测时订阅使用，其他策略运行模式下订阅将接收不到日频行情

示例：订阅上交所浦发银行的日频行情和深交所平安银行的日频行情

通过策略类对象订阅实时行情

1.策略初始化时订阅

ret = strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.bar.daily,SZSE.000001.bar.daily",

3,

"localhost:8001")

2.策略运行中订阅

ret = strategy.subscribe("SHSE.600000.daily,SZSE.000001.daily")

如何退订指定证券代码的行情

示例：退订上交所浦发银行的tick行情和深交所平安银行的1min [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)行情

方式一，通过策略类对象退订：

ret = strategy.unsubscribe("SHSE.600000.tick,SZSE.000001.bar.60")

方式二，通过行情类对象退订：

ret = md.unsubscribe("SHSE.600000.tick,SZSE.000001.bar.60")

历史数据提取

如何提取指定时间的Tick数据

示例：提取上交所浦发银行和深交所平安银行 2015-10-29 9:30:00 到 2015-10-29 15:00:00 的[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

方式一，通过策略类对象接口提取

ticks = strategy.get\_ticks("SHSE.600000,SZSE.000001",

"2015-10-29 9:30:00",

"2015-10-29 15:00:00")

方式二，通过行情类对象接口提取

ticks = md.get\_ticks("SHSE.600000,SZSE.000001",

"2015-10-29 9:30:00",

"2015-10-29 15:00:00")

如何提取指定时间的Bar数据

示例：提取上交所浦发银行和深交所平安银行2015-10-29 10:00:00到2015-10-29 15:00:00的1min [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据,提取30s、5min频率的bar数据修改参数bar\_type为30、300即可。

方式一，通过策略类对象接口提取

bars = strategy.get\_bars("SHSE.600000,SZSE.000001",

60,

"2015-10-29 10:00:00",

"2015-10-29 15:00:00")

方式二，通过行情类对象接口提取

bars = md.get\_bars("SHSE.600000,SZSE.000001",

60,

"2015-10-29 10:00:00",

"2015-10-29 15:00:00")

如何提取指定日期的日频数据

示例：提取上交所浦发银行和深交所平安银行2014-10-29到2015-10-29 的日频数据([DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar))。

方式一，通过策略类对象接口提取

daily\_bars = strategy.get\_dailybars("SHSE.600000,SZSE.000001", "2014-10-29", "2015-10-29")

方式二，通过行情类对象接口提取

daily\_bars = md.get\_dailybars("SHSE.600000,SZSE.000001","2014-10-29", "2015-10-29")

如何提取最近N笔tick数据

示例：提取上交所浦发银行和深交所平安银行最近100笔[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

方式一，通过策略类对象接口提取

ticks = strategy.get\_last\_n\_ticks("SHSE.600000,SZSE.000001", 100)

方式二，通过行情类对象接口提取

ticks = md.get\_last\_n\_ticks("SHSE.600000,SZSE.000001", 100)

如何提取最近N个Bar数据

示例：提取上交所浦发银行和深交所平安银行最近20笔1min频率的[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据，提取30s、5min频率的bar数据修改参数bar\_type为30、300即可。

方式一，通过策略类对象接口提取

bars = strategy.get\_last\_n\_bars("SHSE.600000,SZSE.000001", 60, 20)

方式二，通过行情类对象接口提取

bars = md.get\_last\_n\_bars("SHSE.600000,SZSE.000001", 60, 20)

如何提取最近N个交易日的日频数据

示例：提取上交所浦发银行和深交所平安银行最近20笔日频数据([DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar))

方式一，通过策略类对象接口提取

daily\_bars = strategy.get\_last\_n\_dailybars("SHSE.600000,SZSE.000001", 20)

方式二，通过行情类对象接口提取

daily\_bars = md.get\_last\_n\_dailybars("SHSE.600000,SZSE.000001", 20)

如何提取最近1笔tick数据

示例：提取上交所浦发银行和深交所平安银行最近1笔[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

方式一，通过策略类对象接口提取

ticks = strategy.get\_last\_ticks("SHSE.600000,SZSE.000001")

方式二，通过行情类对象接口提取

ticks = md.get\_last\_ticks("SHSE.600000,SZSE.000001")

如何提取最近1个Bar数据

示例：提取上交所浦发银行和深交所平安银行最近1个1min频率的[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据，提取30s、5min频率的bar数据修改参数bar\_type为30、300即可。

方式一，通过策略类对象接口提取

bars = strategy.get\_last\_bars("SHSE.600000,SZSE.000001", 60)

方式二，通过行情类对象接口提取

bars = md.get\_last\_bars("SHSE.600000,SZSE.000001", 60)

如何提取最近1笔日频数据

示例：提取上交所浦发银行和深交所平安银行最近1笔日频数据([DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar))

方式一，通过策略类对象接口提取

daily\_bars = strategy.get\_last\_dailybars("SHSE.600000,SZSE.000001")

方式二，通过行情类对象接口提取

daily\_bars = md.get\_last\_dailybars("SHSE.600000,SZSE.000001")

行情事件方法

如何处理tick事件和bar事件

tick、bar事件在接收实时行情时触发,在tick和bar事件的回调方法[on\_tick](https://www.myquant.cn/gm2/docs/api/python/#on_tick)、[on\_bar](https://www.myquant.cn/gm2/docs/api/python/#on_bar)中可接收订阅的[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)行情和[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)行情

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_tick](https://www.myquant.cn/gm2/docs/api/python/#on_tick)、[on\_bar](https://www.myquant.cn/gm2/docs/api/python/#on_bar)方法。

**class** **Strategy**(StrategyBase):

**def** **on\_tick**(self, tick):

...

**def** **on\_bar**(self, bar):

...

方式二，在纯行情模式中处理

1.编写tick事件和bar事件的回调方法

**def** **on\_tick**(tick):

...

**def** **on\_bar**(bar):

...

2.订阅tick事件和bar事件

md.ev\_tick += on\_tick

md.ev\_bar += on\_bar

如何处理行情状态事件

在自己的策略类中重写策略基类的[on\_md\_event](https://www.myquant.cn/gm2/docs/api/python/#on_md_event)方法，行情状态事件在开盘、收盘、回放行情结束时触发

**class** **Strategy**(StrategyBase):

**def** **on\_md\_event**(self, evt):

...

交易接口范例

交易相关数据类型

交易涉及到持仓[Position](https://www.myquant.cn/gm2/docs/api/python/#position)、委托[Order](https://www.myquant.cn/gm2/docs/api/python/#order)、成交回报[ExecRpt](https://www.myquant.cn/gm2/docs/api/python/#execrpt)、资金[Cash](https://www.myquant.cn/gm2/docs/api/python/#cash)、绩效指标[Indicator](https://www.myquant.cn/gm2/docs/api/python/#indicator)等数据类型。

委托

如何开多仓

示例：市价买入1000股上交所浦发银行股票，市价开1手IF1512的多单

注意：股票、基金等现货品种只有买入和卖出，对应下单类型是开多仓open\_long和平多仓close\_long；

方式一，调用开多仓的异步接口[open\_long](https://www.myquant.cn/gm2/docs/api/python/#open_long)

1.在策略类的方法中调用，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.open\_long("SHSE", "600000", 0, 1000)

self.open\_long("CFFEX", "IF1512", 0, 1)

Strategy(...).run()

2.通过交易服务类对象调用,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**():

td.open\_long("SHSE", "600000", 0, 1000)

td.open\_long("CFFEX", "IF1512", 0, 1)

td.ev\_some\_event += on\_some\_event

td.init(...)

td.run()

方式二，调用开多仓的同步接口[open\_long\_sync](https://www.myquant.cn/gm2/docs/api/python/#open_long_sync)，无需策略执行Run方法

1.通过策略类对象调用

strategy = Strategy(...)

strategy.open\_long\_sync("SHSE", "600000", 0, 1000)

strategy.open\_long\_sync("CFFEX", "IF1512", 0, 1)

2.通过交易服务类对象调用

**from** gmsdk **import** td

td.init(...)

td.open\_long\_sync("SHSE", "600000", 0, 1000)

td.open\_long\_sync("CFFEX", "IF1512", 0, 1)

如何平多仓

示例：市价卖出1000股上交所浦发银行股票，市价平1手IF1512的多单

注意：股票、基金等现货品种只有买入和卖出，对应下单类型是开多仓open\_long和平多仓close\_long；

方式一，调用平多仓的异步接口[close\_long](https://www.myquant.cn/gm2/docs/api/python/#close_long)

1.在策略类的方法中调用接口，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.close\_long("SHSE", "600000", 0, 1000)

self.close\_long("CFFEX", "IF1512", 0, 1)

Strategy(...).run()

2.通过交易服务类对象调用接口,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

td.close\_long("SHSE", "600000", 0, 1000)

td.close\_long("CFFEX", "IF1512", 0, 1)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

方式二，调用平多仓的同步接口[close\_long\_sync](https://www.myquant.cn/gm2/docs/api/python/#close_long_sync)，无需策略执行Run方法

1.通过策略类对象调用接口

strategy = Strategy(...)

strategy.close\_long\_sync("SHSE", "600000", 0, 1000)

strategy.close\_long\_sync("CFFEX", "IF1512", 0, 1)

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

td.init(...)

td.close\_long\_sync("SHSE", "600000", 0, 1000)

td.close\_long\_sync("CFFEX", "IF1512", 0, 1)

如何开空仓

示例：市价开1手IF1512的空单

方式一，调用开空仓的异步接口[open\_short](https://www.myquant.cn/gm2/docs/api/python/#open_short)

1.在策略类的方法中调用接口，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.open\_short("CFFEX", "IF1512", 0, 1)

Strategy(...).run()

2.通过交易服务类对象调用接口,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

td.open\_short("CFFEX", "IF1512", 0, 1)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

方式二，调用开空仓的同步接口[open\_short\_sync](https://www.myquant.cn/gm2/docs/api/python/#open_short_sync)，无需策略执行Run方法

1.通过策略类对象调用接口

strategy = Strategy(...)

order\_ret = strategy.open\_short\_sync("CFFEX", "IF1512", 0, 1)

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

td.init(...)

td.open\_short\_sync("CFFEX", "IF1512", 0, 1)

如何平空仓

示例：市价平1手IF1512的空单

方式一，调用平空仓的异步接口[close\_short](https://www.myquant.cn/gm2/docs/api/python/#close_short)

1.在策略类的方法中调用接口，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.close\_short("CFFEX", "IF1512", 0, 1)

Strategy(...).run()

2.通过交易服务类对象调用接口,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

td.close\_short("CFFEX", "IF1512", 0, 1)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

方式二，调用平空仓的同步接口[close\_short\_sync](https://www.myquant.cn/gm2/docs/api/python/#close_short_sync)，无需策略执行Run方法

1.通过策略类对象调用接口

strategy = Strategy(...)

order\_ret = strategy.close\_short\_sync("CFFEX", "IF1512", 0, 1)

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

td.init(...)

td.close\_short\_sync("CFFEX", "IF1512", 0, 1)

如何自定义下单

示例：市价开1手IF1512的多单

定义委托单对象

order = Order()

order.exchange = "CFFEX"

order.sec\_id = "IF1512"

order.side = 1

order.position\_effect = 1

order.price = 0

order.volume = 1

说明：

order.side: 设置买卖方向[OrderSide](https://www.myquant.cn/gm2/docs/api/python/#orderside)

order.position\_effect：设置开平类型[PositionEffect](https://www.myquant.cn/gm2/docs/api/python/#positioneffect),上期所可设置平今平昨

order.price：价格为０则表示市价单，否则为限价单

方式一，调用原生下单的异步接口[place\_order](https://www.myquant.cn/gm2/docs/api/python/#place_order)

1.在策略类的方法中调用接口，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.place\_order(order)

Strategy(...).run()

2.通过交易服务类对象调用接口,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

td.place\_order(order)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

方式二，调用原生下单的同步接口[place\_order\_sync](https://www.myquant.cn/gm2/docs/api/python/#place_order_sync)，无需策略执行Run方法

1.通过策略类对象调用接口

strategy = Strategy(...)

strategy.place\_order\_sync(order)

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

td.init(...)

td.place\_order\_sync(order);

如何撤回未成交订单

方式一，调用异步撤单接口[cancel\_order](https://www.myquant.cn/gm2/docs/api/python/#cancel_order)

1.在策略类的方法中调用接口，必须在策略执行了Run方法后调用有效

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

self.cancel\_order(order\_id)

Strategy(...).run()

2.通过交易服务类对象调用接口,必须执行对象的Run方法后调用有效

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

td.cancel\_order(order\_id)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

方式二，调用同步撤单接口[cancel\_order\_sync](https://www.myquant.cn/gm2/docs/api/python/#cancel_order_sync)，无需策略执行Run方法

1.通过策略类对象调用接口

strategy = Strategy(...)

strategy.cancel\_order(order\_id)

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

td.init(...)

td.cancel\_order(order\_id)

如何查询指定委托当前详情

1.在策略类的方法中调用接口

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

order = self.get\_order(order\_id)

Strategy(...).run()

2.通过交易服务类对象调用接口

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

order = td.get\_order(order\_id)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

回报

订单的回报以交易事件的方式给出，当前提供的交易事件有：

1. 委托执行回报事件
2. 订单拒绝事件
3. 订单被柜台接受事件
4. 订单状态变更事件
5. 订单全部成交事件
6. 订单部分成交事件
7. 订单停止成交事件
8. 订单撤单成功事件
9. 订单撤单拒绝事件

具体事件处理请参考交易事件处理章节

资金

如何查询资金

调用[get\_cash](https://www.myquant.cn/gm2/docs/api/python/#get_cash)方法查询当前策略的资金信息。

1. 在策略类的方法中直接调用接口

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

cash = self.get\_cash()

Strategy(...).run()

1. 通过交易服务类对象调用接口

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

cash = td.get\_cash()

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

持仓

如何获取指定证券代码的持仓

调用[get\_position](https://www.myquant.cn/gm2/docs/api/python/#get_position)方法查询单一交易品种的持仓情况。

示例：获取上交所浦发银行持仓情况

1. 在策略类的方法中直接调用持仓接口

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

postion = self.get\_position("SHSE", "600000", 1)

Strategy(...).run()

1. 通过交易服务类对象调用持仓接口

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

postion = td.get\_position("SHSE", "600000", 1)

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

如何获取当前策略的全部持仓信息

调用[get\_positions](https://www.myquant.cn/gm2/docs/api/python/#get_positions)方法获取当前策略的全部持仓信息。

1. 在策略类的方法中直接调用接口

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

pos\_list = self.get\_positions()

Strategy(...).run()

1. 通过交易服务类对象调用接口

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

pos\_list = td.get\_positions()

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

绩效

如何获取当前策略的绩效指标

调用[get\_indicator](https://www.myquant.cn/gm2/docs/api/python/#get_indicator)方法获取当前策略的绩效指标。

1. 在策略类的方法中直接调用接口

**class** **Strategy**(StrategyBase):

**def** **on\_some\_event**(self): *## eg. def on\_tick(self, tick):*

indicator = self.get\_indicator()

Strategy(...).run()

1. 通过交易服务类对象调用接口

**from** gmsdk **import** td

**def** **on\_some\_event**(): *##eg. def on\_tick(tick):*

indicator = td.get\_indicator()

td.ev\_some\_event += on\_some\_event *##eg. td.ev\_tick += on\_tick*

td.init(...)

td.run()

交易事件

当我们在程序下一笔委托单到交易服务时，将产生一系列交易事件

如何处理订单执行回报事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_execrpt](https://www.myquant.cn/gm2/docs/api/python/#on_execrpt)方法，[on\_execrpt](https://www.myquant.cn/gm2/docs/api/python/#on_execrpt)方法在委托执行时触发，订单的任何执行回报都会触发本事件

**class** **Strategy**(StrategyBase):

**def** **on\_execrpt**(self, rpt):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写委托执行的回调方法

**def** **on\_execrpt**(rpt):

...

1. 订阅委托执行事件

**from** gmsdk **import** td

td.ev\_execrpt += on\_execrpt

td.init(...).run()

如何处理订单拒绝事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_rejected](https://www.myquant.cn/gm2/docs/api/python/#on_order_rejected)方法，[on\_order\_rejected](https://www.myquant.cn/gm2/docs/api/python/#on_order_rejected)方法在订单拒绝时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_rejected**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写委订单拒绝的回调方法

**def** **on\_order\_rejected**(order):

...

1. 订阅订单拒绝事件

**from** gmsdk **import** td

td.ev\_order\_rejected += on\_order\_rejected

td.init(...)

td.run()

如何处理订单被交易所接收事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_new](https://www.myquant.cn/gm2/docs/api/python/#on_order_new)方法，当订单已被交易所接受时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_new**(self, order):

...

ret = Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单接收的回调方法

**def** **on\_order\_new**(order):

...

1. 订阅订单接收事件

**from** gmsdk **import** td

td.ev\_order\_new += on\_order\_new

td.init(...)

td.run()

如何处理订单状态更新事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_status](https://www.myquant.cn/gm2/docs/api/python/#on_order_status)方法，当订单状态更新时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_status**(self, order):

...

ret = Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单接收的回调方法

**def** **on\_order\_status**(order):

...

1. 订阅订单状态更新事件

**from** gmsdk **import** td

td.ev\_order\_status += on\_order\_status

td.init(...)

td.run()

如何处理订单全部成交事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_filled](https://www.myquant.cn/gm2/docs/api/python/#on_order_filled)方法,当一笔订单完全成交时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_filled**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单全成的回调方法

**def** **on\_order\_filled**(order):

...

1. 订阅订单全成事件

**from** gmsdk **import** td

td.ev\_order\_filled += on\_order\_filled

td.init(...)

td.run()

如何处理委托单部分成交的事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_partially\_filled](https://www.myquant.cn/gm2/docs/api/python/#on_order_partially_filled)方法,当一笔订单有部分成交时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_partially\_filled**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单部分成交的回调方法

**def** **on\_order\_partially\_filled**(order):

...

1. 订阅订单部分成交事件

**from** gmsdk **import** td

td.ev\_order\_partially\_filled += on\_order\_partially\_filled

td.init(...)

td.run()

如何处理停止订单执行事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_stop\_executed](https://www.myquant.cn/gm2/docs/api/python/#on_order_stop_executed)方法,当订单停止执行时触发，例如限价单收市时还未成交

**class** **Strategy**(StrategyBase):

**def** **on\_order\_stop\_executed**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单停止执行的回调方法

**def** **on\_order\_stop\_executed**(order):

...

1. 订阅订单停止执行事件

**from** gmsdk **import** td

td.ev\_order\_stop\_executed += on\_order\_stop\_executed

td.init(...)

td.run()

如何处理撤单拒绝事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_cancel\_rejected](https://www.myquant.cn/gm2/docs/api/python/#on_order_cancel_rejected)方法,当撤单拒绝时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_cancel\_rejected**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写订单停止执行的回调方法

**def** **on\_order\_cancel\_rejected**(rpt):

...

1. 订阅订单撤单拒绝事件

**from** gmsdk **import** td

td.ev\_order\_cancel\_rejected += on\_order\_cancel\_rejected

td.init(...)

td.run()

如何处理委托撤单成功事件

方式一，在策略类中处理

在自己的策略类中重写策略基类的[on\_order\_cancelled](https://www.myquant.cn/gm2/docs/api/python/#on_order_cancelled)方法,当委托单的状态发生变化时触发

**class** **Strategy**(StrategyBase):

**def** **on\_order\_cancelled**(self, order):

...

Strategy(...).run()

方式二，通过交易服务类对象处理

1. 编写委托单状态变化的回调方法

**def** **on\_order\_cancelled**(order):

...

1. 订阅撤单成功事件

**from** gmsdk **import** td

td.ev\_order\_cancelled += on\_order\_cancelled

td.init(...)

td.run()

错误与版本等接口范例

如何获取API版本信息

调用[get\_version](https://www.myquant.cn/gm2/docs/api/python/#get_version)方法。

**import** gmsdk

gmsdk.get\_version()

如何获取API返回值及错误事件参数值的文本信息

API返回值及错误事件的参数值都有统一的定义，调用[get\_strerror](https://www.myquant.cn/gm2/docs/api/python/#get_strerror)方法获取[返回值](https://www.myquant.cn/gm2/docs/api/python/#_63)对应的文本信息。

如何获取策略运行中的错误信息

在自己的策略类中重写策略基类的[on\_error](https://www.myquant.cn/gm2/docs/api/python/#on_error)方法,当策略执行过程有任何错误都会回调该方法，通过参数获取错误信息

**class** **Strategy**(StrategyBase):

**def** **on\_error**(self, error\_code, error\_msg):

...

Strategy(...).run()

**Python API接口**

枚举常量定义

OrderStatus

订单状态。

OrderStatus\_New = 1, *#已报*

OrderStatus\_PartiallyFilled = 2, *#部成*

OrderStatus\_Filled = 3, *#已成*

OrderStatus\_DoneForDay = 4, *#*

OrderStatus\_Canceled = 5, *#已撤*

OrderStatus\_PendingCancel = 6, *#待撤*

OrderStatus\_Stopped = 7, *#停止*

OrderStatus\_Rejected = 8, *#已拒绝*

OrderStatus\_Suspended = 9, *#挂起*

OrderStatus\_PendingNew = 10, *#待报*

OrderStatus\_Calculated = 11, *#计算*

OrderStatus\_Expired = 12, *#已过期*

OrderStatus\_AcceptedForBidding = 13, *#接受竞价*

OrderStatus\_PendingReplace = 14 *#待修改*

OrderRejectReason

订单拒绝原因。

OrderRejectReason\_UnknownReason = 1, *#未知原因*

OrderRejectReason\_RiskRuleCheckFailed = 2, *#不符合风控规则*

OrderRejectReason\_NoEnoughCash = 3, *#资金不足*

OrderRejectReason\_NoEnoughPosition = 4, *#仓位不足*

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\_NoMatchedTradingChannel = 9, *#没有匹配的交易通道*

OrderRejectReason\_AccountForbidTrading = 10, *#交易账号被禁止交易*

OrderRejectReason\_TradingChannelNotConnected = 11, *#交易通道未连接*

OrderRejectReason\_StrategyForbidTrading = 12, *#策略不允许交易*

OrderRejectReason\_NotInTradingSession = 13 *#非交易时段*

CancelOrderRejectReason\_OrderFinalized = 101 *#订单已是最终状态*

CancelOrderRejectReason\_UnknownOrder = 102 *#未知订单*

CancelOrderRejectReason\_BrokerOption = 103 *#柜台拒绝*

CancelOrderRejectReason\_AlreadyInPendingCancel = 104 *#重复撤单*

OrderSide

订单方向。

OrderSide\_Bid = 1 *## 多方向*

OrderSide\_Ask = 2 *## 空方向*

OrderType

订单类型。

OrderType\_LMT = 0, *## 限价委托(limit)*

OrderType\_BOC = 1, *## 对方最优价格(best of counterparty)*

OrderType\_BOP = 2, *## 己方最优价格(best of party)*

OrderType\_B5TC = 3, *## 最优五档剩余撤销(best 5 then cancel)*

OrderType\_B5TL = 4, *## 最优五档剩余转限价(best 5 then limit)*

OrderType\_IOC = 5, *## 即时成交剩余撤销(immediately or cancel)*

OrderType\_FOK = 6, *## 即时全额成交或撤销(fill or kill)*

OrderType\_AON = 7, *## 全额成交或不成交(all or none)*

OrderType\_MTL = 8, *## 市价剩余转限价(market then limit)*

OrderType\_EXE = 9 *## 期权行权(option execute)*

ExecType

订单执行回报类型。

ExecType\_New = 1 *## 交易所已接受订单*

ExecType\_DoneForDay = 4

ExecType\_Canceled = 5 *## 已撤*

ExecType\_PendingCancel = 6 *## 待撤*

ExecType\_Stopped = 7 *## 已停*

ExecType\_Rejected = 8 *## 已拒绝*

ExecType\_Suspended = 9 *## 暂停*

ExecType\_PendingNew = 10 *## 待接受*

ExecType\_Calculated = 11 *## 已折算*

ExecType\_Expired = 12 *## 过期*

ExecType\_Restated = 13 *## 重置*

ExecType\_PendingReplace = 14 *## 待修改*

ExecType\_Trade = 15 *## 交易*

ExecType\_TradeCorrect = 16 *## 交易更正*

ExecType\_TradeCancel = 17 *## 交易取消*

ExecType\_OrderStatus = 18 *## 更新订单状态*

ExecType\_CancelRejected = 19 *## 撤单被拒绝*

PositionEffect

开平仓类型。

PositionEffect\_Open = 1 *## 开仓*

PositionEffect\_Close = 2 *## 平仓*

PositionEffect\_CloseToday = 3 *## 平今仓*

PositionEffect\_CloseYesterday = 4 *## 平昨仓*

交易数据类型

交易数据类型主要包括委托，执行回报，资金，持仓，绩效等数据类型。

Order

委托订单。

**class** **Order**(object):

**def** **\_\_init\_\_**(self):

self.strategy\_id = '' *## 策略ID*

self.account\_id = '' *## 交易账号*

self.cl\_ord\_id = '' *## 客户端订单ID*

self.order\_id = '' *## 柜台订单ID*

self.ex\_ord\_id = '' *## 交易所订单ID*

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.position\_effect = 0 *## 开平标志*

self.side = 0 *## 买卖方向*

self.order\_type = 0 *## 订单类型*

self.order\_src = 0 *## 订单来源*

self.status = 0 *## 订单状*

self.ord\_rej\_reason = 0 *## 订单拒绝原因*

self.ord\_rej\_reason\_detail = '' *## 订单拒绝原因描述*

self.price = 0.0 *## 委托价*

self.stop\_price = 0.0; *## 止损价*

self.volume = 0.0 *## 委托量*

self.filled\_volume = 0.0 *## 已成交量*

self.filled\_vwap = 0.0 *## 已成交均价*

self.filled\_amount = 0.0 *## 已成交额*

self.sending\_time = 0.0 *## 委托下单时间*

self.transact\_time = 0.0 *## 最新一次成交时间*

ExecRpt

委托执行回报。

**class** **ExecRpt**(object):

**def** **\_\_init\_\_**(self):

self.strategy\_id = '' *## 策略ID*

self.cl\_ord\_id = '' *## 客户端订单ID*

self.order\_id = '' *## 交易所订单ID*

self.exec\_id = '' *## 订单执行回报ID*

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.position\_effect = 0 *## 开平标志*

self.side = 0 *## 买卖方向*

self.ord\_rej\_reason = 0 *## 订单拒绝原因*

self.ord\_rej\_reason\_detail = '' *## 订单拒绝原因描述*

self.exec\_type = 0 *## 订单执行回报类型*

self.price = 0.0 *## 成交价*

self.volume = 0.0 *## 成交量*

self.amount = 0.0 *## 成交额*

self.transact\_time = 0.0 *## 交易时间*

Cash

资金。

**class** **Cash**(object):

**def** **\_\_init\_\_**(self):

self.strategy\_id = '' *## 策略ID*

self.account\_id = '' *## 账户id*

self.currency = 0 *## 币种*

self.nav = 0.0 *## 资金余额*

self.fpnl = 0.0 *## 浮动收益*

self.pnl = 0.0 *## 净收益*

self.profit\_ratio = 0.0 *## 收益率*

self.frozen = 0.0 *## 持仓冻结金额*

self.order\_frozen = 0.0 *## 挂单冻结金额*

self.available = 0.0 *## 可用资金*

self.cum\_inout = 0.0 *## 累计出入金*

self.cum\_trade = 0.0 *## 累计交易额*

self.cum\_pnl = 0.0 *## 累计收益*

self.cum\_commission = 0.0 *## 累计手续费*

self.last\_trade = 0.0 *## 最新一笔交易额*

self.last\_pnl = 0.0 *## 最新一笔交易收益*

self.last\_commission = 0.0 *## 最新一笔交易手续费*

self.last\_inout = 0.0 *## 最新一次出入金*

self.change\_reason = 0 *## 变动原因*

self.transact\_time = 0.0 *## 交易时间*

Position

持仓。

**class** **Position**(object):

**def** **\_\_init\_\_**(self):

self.strategy\_id = '' *## 策略ID*

self.account\_id = '' *## 账户id*

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.side = 0 *## 买卖方向*

self.volume = 0.0 *## 持仓量*

self.volume\_today = 0.0 *## 今仓量*

self.amount = 0.0 *## 持仓额*

self.vwap = 0.0 *## 持仓均价*

self.price = 0.0 *## 当前行情价格*

self.fpnl = 0.0 *## 持仓浮动盈亏*

self.cost = 0.0 *## 持仓成本*

self.order\_frozen = 0.0 *## 挂单冻结仓位*

self.available = 0.0 *## 可平仓位*

self.available\_today = 0.0 *## 可平今仓位(volume\_today-order\_frozen\_today)*

self.avaliable\_yesterday = 0.0 *## 可平昨仓位(available - available\_today)*

self.order\_frozen\_today = 0.0 *## 挂单冻结今仓*

self.last\_price = 0.0 *## 上一笔成交价*

self.last\_volume = 0.0 *## 上一笔成交量*

self.init\_time = 0.0 *## 初始建仓时间*

self.transact\_time = 0.0 *## 上一仓位变更时间*

Indicator

绩效。

**class** **Indicator**(object):

**def** **\_\_init\_\_**(self):

self.strategy\_id = '' *## 策略ID*

self.account\_id = '' *## 账号ID*

self.nav = 0.0 *## 净值(cum\_inout + cum\_pnl + fpnl - cum\_commission)*

self.pnl = 0.0 *## 净收益(nav-cum\_inout)*

self.profit\_ratio = 0.0 *## 收益率(pnl/cum\_inout)*

self.profit\_ratio\_bench = 0.0 *## 基准收益率*

self.sharp\_ratio = 0.0 *## 夏普比率*

self.risk\_ratio = 0.0 *## 风险比率*

self.trade\_count = 0 *## 交易次数*

self.win\_count = 0 *## 盈利次数*

self.lose\_count = 0 *## 亏损次数*

self.win\_ratio = 0.0 *## 胜率*

self.max\_profit = 0.0 *## 最大收益*

self.min\_profit = 0.0 *## 最小收益*

self.max\_single\_trade\_profit = 0.0 *## 最大单次交易收益*

self.min\_single\_trade\_profit = 0.0 *## 最小单次交易收益*

self.daily\_max\_single\_trade\_profit = 0.0 *## 今日最大单次交易收益*

self.daily\_min\_single\_trade\_profit = 0.0 *## 今日最小单次交易收益*

self.max\_position\_value = 0.0 *## 最大持仓市值或权益*

self.min\_position\_value = 0.0 *## 最小持仓市值或权益*

self.max\_drawdown = 0.0 *## 最大回撤*

self.daily\_pnl = 0.0 *## 今日收益*

self.daily\_return = 0.0 *## 今日收益率*

self.annual\_return = 0.0 *## 年化收益率*

self.cum\_inout = 0.0 *## 累计出入金*

self.cum\_trade = 0.0 *## 累计交易额*

self.cum\_pnl = 0.0 *## 累计平仓收益(没扣除手续费)*

self.cum\_commission = 0.0 *## 累计手续费*

self.transact\_time = 0.0 *## 指标计算时间*

BrokerAccount

柜台账户

**class** **BrokerAccount**(object):

**def** **\_\_init\_\_**(self):

self.account\_id = '' *# 柜台账号ID*

self.username = '' *# 柜台账号*

self.permissible = 0 *# 允许交易*

self.status = 0 *# 账号当前状态*

StrategyParameter

策略参数

**class** **StrategyParameter**(object):

**def** **\_\_init\_\_**(self):

self.name = '' *# 参数名*

self.value = 0.0 *# 参数值*

self.min = 0.0 *# 可设置的最小值*

self.max = 0.0 *# 可设置的最大值*

self.readonly = false *# 是否只读*

self.group = '' *# 组名*

self.intro = '' *# 参数说明*

StrategySymbol

策略交易标的

**class** **StrategySymbol**(object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *# 交易代码*

self.exchange = '' *# 交易所代码*

self.sec\_id = '' *# 证券ID*

行情数据类型

行情数据类型有Tick，Bar，DailyBar。

Tick

逐笔行情数据。

**class** **Tick**(object):

**def** **\_\_init\_\_**(self):

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.utc\_time = 0.0 *## 行情时间戳*

self.strtime = '' *## 可视化时间*

self.last\_price = 0.0 *## 最新价*

self.open = 0.0 *## 开盘价*

self.high = 0.0 *## 最高价*

self.low = 0.0 *## 最低价*

self.cum\_volume = 0.0 *## 成交总量/最新成交量,累计值*

self.cum\_amount = 0.0 *## 成交总金额/最新成交额,累计值*

self.cum\_position = 0.0 *## 合约持仓量(期),累计值*

self.last\_volume = 0 *## 瞬时成交量(中金所提供)*

self.last\_amount = 0.0 *## 瞬时成交额*

self.upper\_limit = 0.0 *## 涨停价*

self.lower\_limit = 0.0 *## 跌停价*

self.settle\_price = 0.0 *## 今日结算价*

self.trade\_type = 0 *## (保留)交易类型,对应多开,多平等类型 0:'上一个tick没有成交量', 1:'双开', 2: '双平', 3: '多开', 4:'空开', 5: '空平', 6:'多平', 7:'多换', 8:'空换'*

self.pre\_close = 0.0 *## 昨收价*

self.bids = [] *## [(price, volume), (price, volume), ...] ## 1-5档买价,量*

self.asks = [] *## [(price, volume), (price, volume), ...] ## 1-5档卖价,量*

Bar

各种周期的Bar数据。

**class** **Bar**(object):

**def** **\_\_init\_\_**(self):

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.bar\_type = 0 *## bar类型，以秒为单位，比如1分钟bar, bar\_type=60*

self.strtime = '' *## Bar开始时间*

self.utc\_time = 0.0 *## Bar开始时间*

self.strendtime = '' *## Bar结束时间*

self.utc\_endtime = 0.0 *## Bar结束时间*

self.open = 0.0 *## 开盘价*

self.high = 0.0 *## 最高价*

self.low = 0.0 *## 最低价*

self.close = 0.0 *## 收盘价*

self.volume = 0.0 *## 成交量*

self.amount = 0.0 *## 成交额*

self.pre\_close *## 前收盘价*

self.position; *## 持仓量*

self.adj\_factor *## 复权因子*

self.flag *## 除权出息标记*

DailyBar

日频数据，在Bar数据的基础上，还包含结算价，涨跌停价等静态数据。

**class** **Dailybar**(object):

**def** **\_\_init\_\_**(self):

self.exchange = '' *## 交易所代码*

self.sec\_id = '' *## 证券ID*

self.bar\_type = 0 *## bar类型*

self.strtime = '' *## 可视化时间*

self.utc\_time = 0.0 *## 行情时间戳*

self.open = 0.0 *## 开盘价*

self.high = 0.0 *## 最高价*

self.low = 0.0 *## 最低价*

self.close = 0.0 *## 收盘价*

self.volume = 0.0 *## 成交量*

self.amount = 0.0 *## 成交额*

self.position = 0.0 *## 仓位*

self.settle\_price = 0.0 *## 结算价*

self.upper\_limit = 0.0 *## 涨停价*

self.lower\_limit = 0.0 *## 跌停价*

self.pre\_close *## 前收盘价*

self.adj\_factor *## 复权因子*

self.flag *## 除权出息，停牌等标记*

Instrument

交易代码数据类型

**class** **Instrument**(object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *## 交易代码*

self.sec\_type = 0 *## 代码类型*

self.sec\_name = '' *## 代码名称*

self.multiplier = 0.0 *## 合约乘数*

self.margin\_ratio = 0.0 *## 保证金比率*

self.price\_tick = 0.0 *## 价格最小变动单位*

self.upper\_limit = 0.0 *## 当天涨停板*

self.lower\_limit = 0.0 *## 当天跌停板*

self.is\_active = 0 *## 当天是否交易*

self.update\_time = '' *## 更新时间*

Constituent

成份股数据类型

**class** **Constituent**(object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *## 交易代码*

self.weight = 0.0 *## 代码权重*

FinancialIndex

财务指标

**class** **FinancialIndex**(object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *#股票代码*

self.pub\_date = '' *#公告日期*

self.eps = 0.0 *#每股收益*

self.bvps = 0.0 *#每股净资产*

self.cfps = 0.0 *#每股现金流*

self.afps = 0.0 *#每股公积金*

self.total\_asset = 0.0 *#总资产*

self.current\_asset = 0.0 *#流动资产*

self.fixed\_asset = 0.0 *#固定资产*

self.liability = 0.0 *#负债合计*

self.current\_liability = 0.0 *#流动负债*

self.longterm\_liability = 0.0 *#长期负债*

self.equity = 0.0 *#所有者权益*

self.income = 0.0 *#主营业务收入*

self.operating\_profit = 0.0 *#主营业务利润*

self.net\_profit = 0.0 *#净利润*

ShareIndex

股本指标

**class** **ShareIndex**(Object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *#股票代码*

self.pub\_date = '' *#公告日期*

self.total\_share = 0.0 *#总股本*

self.flow\_a\_share = 0.0 *#流通A股*

self.nonflow\_a\_share = 0.0 *#限售流通A股*

MarketIndex

市场指标

**class** **MarketIndex**(Object):

**def** **\_\_init\_\_**(self):

self.symbol = '' *#股票代码*

self.pub\_date = '' *#公告日期*

self.pe\_ratio = 0.0 *#市盈率*

self.pb\_ratio = 0.0 *#市净率*

self.ps\_ratio = 0.0 *#市销率*

self.market\_value = 0.0 *#市值*

self.market\_value\_flow = 0.0 *#流通市值*

TradeDate

交易日类型

**class** **TradeDate**(object):

**def** **\_\_init\_\_**(self):

self.utc\_time = 0.0 *## UTC时间戳[带毫秒]*

self.strtime = '' *## 交易日*

StockAdjustFactor

复权因子

**class** **StockAdjustFactor**(object):

**def** **\_\_init\_\_**(self):

self.symbol *##股票代码*

self.trade\_date *##交易日*

self.adj\_factor *##复权因子*

StockDivident

分红送股事件明细

**class** **StockDivident**(object):

**def** **\_\_init\_\_**(self):

self.symbol *##股票代码*

self.div\_date *##除权除息日*

self.cash\_div *##每股派现*

self.share\_div\_ratio *##每股送股比例*

self.share\_trans\_ratio *##每股转增股比例*

self.allotment\_ratio *##每股配股比例*

self.allotment\_price *##配股价*

VirtualContract

虚拟合约明细

**class** **VirtualContract**(object):

**def** **\_\_init\_\_**(self):

self.vsymbol *##主力合约或连接合约代码*

self.symbol *##真实symbol*

self.trade\_date *##交易日*

策略初始化方法

init

初始化策略，设置服务地址，账号密码，策略ID，以及需要订阅的symbol列表。策略自动连接服务，并登陆，订阅对应的行情，准备好运行。

**函数原型：**

\_\_init\_\_(self,

username,

password,

strategy\_id,

subscribe\_symbols='',

mode=2,

td\_addr='',

config\_file=**None**,

config\_file\_encoding='utf-8')

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| username | string | 掘金终端账号 |
| password | string | 掘金终端密码 |
| strategy\_id | string | 策略ID |
| subscribe\_symbols | string | 行情订阅的代码列表 |
| mode | int | 枚举类型，行情模式 |
| td\_addr | string | 交易服务器uri, 如设置为localhost:8001，则终端用户指向本地客户端 , 如果设置为空, 则使用掘金云端服务 |

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

初始化时订阅上交所浦发银行的tick和1min bar实时行情

ret = Strategy("your user name",

"your password",

"strategy id",

"SHSE.600000.tick,SHSE.600000.bar.60",

2,

"localhost:8001")

**注意项：**

symbol\_list 订阅代码表,参数格式如下:

订阅串有三节或四节组成,用'.'分隔，格式：

对应交易所exchange.代码code.数据类型data\_type.周期类型 bar\_type

只有订阅bar数据时, 才用到第四节, 周期类型才有作用 交易所exchange统一四个字节:

CFFEX-中金所 SHFE-上期所 DCE-大商所 CZCE-郑商所 SHSE-上交所 SZSE-深交所

支持6种格式的订阅,使用如下:

* SHSE.\* : 上交所,所有数据
* SHSE.600000.\* : 上交所,600000,所有数据
* SHSE.600000.tick : 上交所,600000, tick数据
* SHSE.600000.bar.60: 上交所, 600000, 1分钟(60秒)Bar数据
* SHSE.600000.\*,SHSE.600004.\* : 上交所,600000和600004所有数据(订阅多个代码)

init\_with\_config

使用配置文件初始化策略，配置文件已设置服务地址，账号密码，策略ID，行情模式以及需要订阅的symbol列表

**函数原型：**

\_\_init\_\_(self,

username,

password,

strategy\_id,

subscribe\_symbols='',

mode=2,

td\_addr='',

config\_file=**None**,

config\_file\_encoding='utf-8')

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| config\_file | string | 策略配置文件路径 |
| config\_file\_encoding | string | 策略配置文件编码 |

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

strategy.ini文件配置：

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

*;策略基本配置*

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

**[strategy]**

*;掘金用户名*

username=-

*;掘金密码*

password=-

*;策略ID*

strategy\_id=3c7e70be-7e02-11e5-b293-5ec5d48ea63a

*;订阅证券代码或合约代码列表*

subscribe\_symbols=SHSE.600000.tick,SHSE.600000.bar.60

*;行情模式，2-实时行情模式，3-模拟行情模式，4-回放行情模式*

mode=2

*;交易服务地址，使用掘金终端交易时，地址为localhost:8001, 如果此项配置为空，则订单发往掘金云服务器*

td\_addr=localhost:8001

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

*;策略回测参数配置*

*;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;*

**[backtest]**

*;历史数据回放开始时间*

start\_time=2014-10-25 09:30:00

*;历史数据回放结束时间*

end\_time=2015-10-29 15:15:00

*;策略初始资金*

initial\_cash=1000000

*;委托量成交比率，默认=1（每个委托100%成交）*

transaction\_ratio=1

*;手续费率，默认=0（不计算手续费）*

commission\_ratio=0.0008

*;滑点比率，默认=0（无滑点）*

slippage\_ratio=0.246

*;行情复权模式,0=不复权,1=前复权*

price\_type=1

*;基准*

bench\_symbol=SHSE.000300

ret = Strategy(config\_file="strategy.ini")

backtest\_config

回测参数设置, 仅在回测模式下有效

**函数原型：**

backtest\_config(self,

start\_time,

end\_time,

initial\_cash=1000000,

transaction\_ratio=1,

commission\_ratio=0,

slippage\_ratio=0,

price\_type=1,

bench\_symbol='SHSE.000300',

check\_cache=1)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| start\_time | string | 回放行情开始时间，格式：yyyy-mm-dd HH:MM:SS |
| end\_time | string | 回放行情结束时间，格式：yyyy-mm-dd HH:MM:SS |
| initial\_cash | float | 回测初始资金，默认1000000 |
| transaction\_ratio | float | 委托量成交比率,默认1，按委托量全部成交 |
| commission\_ratio | float | 手续费率，默认0，无手续费 |
| slippage\_ratio | float | 滑点比率，默认0，无滑点 |
| price\_type | int | 复权方式，0-不复权，1-前复权 |
| bench\_symbol | string | 基准代码 |
| check\_cache | int | 回测时是否使用本地缓存数据，0-不使用，1-使用, 默认为值为 1 |

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

策略启停方法

run

运行策略

**函数原型：**

run()

**参数：**

无

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

stop

停止策略

**参数：**

无

**返回值：**

无

行情订阅方法

subscribe

订阅行情,策略类和行情服务了都提供该接口。

**函数原型:**

subscribe(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 订阅代码列表 |

**返回值:**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

通过策略对象订阅上交所浦发银行[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)和1分钟[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)实时数据

strategy.subscribe("SHSE.600000.tick,SHSE.600000.bar.60")

**注意项：**

symbol\_list 订阅代码表的[参数格式](https://www.myquant.cn/gm2/docs/api/concept/#_4) 当在回测模式下，需要显式关闭本地缓存api才能正常工作, 如何关闭本地缓存，请参见 backtest\_config

unsubscribe

退订指定行情，策略类和行情服务了都提供该接口。

**函数原型:**

unsubscribe(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 订阅代码列表 |

**返回值:**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

通过策略对象退订上交所浦发银行[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)和1分钟[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)实时数据

strategy.unsubscribe("SHSE.600000.tick,SHSE.600000.bar.60")

**注意项：**

symbol\_list 订阅代码表的[参数格式](https://www.myquant.cn/gm2/docs/api/concept/#_4) 当在回测模式下，需要显式关闭本地缓存api才能正常工作, 如何关闭本地缓存，请参见 backtest\_config

resubscribe

重置订阅条件，相当于先退订原来所有行情，再重新订阅指定的行情

**函数原型:**

resubscribe(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 订阅代码列表 |

**返回值:**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)。

symbol\_list 订阅代码表的[参数格式](https://www.myquant.cn/gm2/docs/api/concept/#_4) 当在回测模式下，需要显式关闭本地缓存api才能正常工作, 如何关闭本地缓存，请参见 backtest\_config

数据提取方法

get\_ticks

提取指定时间段的历史[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据，支持单个代码提取或多个代码组合提取。策略类和行情服务类都提供该接口。

**函数原型:**

get\_ticks(symbol\_list, begin\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |
| begin\_time | string | 开始时间, 如2015-10-30 09:30:00 |
| end\_time | string | 结束时间, 如2015-10-30 15:00:00 |

**返回值：**

Tick列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行2015-10-30 09:30:00到2015-10-30 15:00:00时间段的所有[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

ticks = strategy.get\_ticks("SHSE.600000,SZSE.000001",

"2015-10-30 09:30:00",

"2015-10-30 15:00:00")

get\_bars

提取指定时间段的历史[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据，支持单个代码提取或多个代码组合提取。策略类和行情服务类都提供该接口。

**函数原型:**

get\_bars(symbol\_list, bar\_type, begin\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |
| bar\_type | int | bar周期，以秒为单位，比如60即1分钟bar |
| begin\_time | string | 开始时间, 如2015-10-30 09:30:00 |
| end\_time | string | 结束时间, 如2015-10-30 15:00:00 |

**返回值：**

Bar列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行2015-10-30 09:30:00到2015-10-30 15:00:00时间段的1分钟[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据

bars = strategy.get\_bars("SHSE.600000,SZSE.000001",

60,

"2015-10-30 09:30:00",

"2015-10-30 15:00:00")

get\_dailybars

提取指定时间段的历史日周期[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据，支持单个代码提取或多个代码组合提取。[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)比[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)多了部分静态数据，如结算价，涨跌停等。策略类和行情服务类都提供该接口。

**函数原型:**

get\_dailybars(symbol\_list, begin\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |
| begin\_time | string | 开始日期, 如2015-10-19 |
| end\_time | string | 结束日期, 如2015-10-30 |

**返回值：**

DailyBar列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行2015-10-19到2015-10-30时间段的[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据

var dailybars = strategy.get\_dailybars("SHSE.600000,SZSE.000001",

"2015-10-19 00:00:00",

"2015-10-30 00:00:00")

get\_last\_ticks

提取最新的1条[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据，支持单个代码提取或多个代码组合提取。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_ticks(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |

**返回值：**

Tick列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行的最近一笔[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

ticks = strategy.get\_last\_ticks("SHSE.600000,SZSE.000001")

get\_last\_bars

提取最新1条[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据，支持单个代码提取或多个代码组合提取。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_bars(symbol\_list, bar\_type)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |
| bar\_type | int | bar周期，以秒为单位，比如60即1分钟bar |

**返回值：**

Bar列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行的最新的1笔1min [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据

bars = strategy.get\_last\_bars("SHSE.600000,SZSE.000001", 60)

get\_last\_dailybars

提取最新1条[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据，支持单个代码提取或多个代码组合提取。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_dailybars(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000，同时支持多只代码 |

**返回值：**

DailyBar列表

**示例：**

通过策略对象提取上交所浦发银行和深交所平安银行的最近一笔[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据

dailybars = strategy.get\_last\_dailybars("SHSE.600000,SZSE.000001")

get\_last\_n\_ticks

提取单个代码最新n条[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据,策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_ticks(symbol, n, end\_time='')

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000 |
| n | int | 提取的数据条数 |
| end\_time | string | 指定截止时间, 如2015-10-30 15:00:00 |

**返回值：**

Tick列表

**示例：**

通过策略对象提取上交所浦发银行的最近10笔[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据

dailybars = strategy.get\_last\_n\_ticks("SHSE.600000")

get\_last\_n\_bars

提取单个代码的最新n条[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据,策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_bars(symbol, bar\_type, n, end\_time='')

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000 |
| bar\_type | int | bar周期，以秒为单位，比如60即1分钟bar |
| n | int | 提取的数据条数 |
| end\_time | string | 指定截止时间, 如2015-10-30 15:00:00 |

**返回值：**

Bar列表

**示例：**

通过策略对象提取上交所浦发银行的最近10笔1分钟[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据

dailybars = strategy.get\_last\_n\_bars("SHSE.600000", 60， 10)

get\_last\_n\_dailybars

提取单个代码的最新n条[DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据, 策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_dailybars(symbol, n, end\_time='')

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 证券代码, 带交易所代码以确保唯一，如SHSE.600000 |
| n | int | 提取的数据条数 |
| end\_time | string | 指定截止时间, 如2015-10-30 15:00:00 |

**返回值：**

Bar列表

**示例：**

通过策略对象提取上交所浦发银行的最近10笔1min [DailyBar](https://www.myquant.cn/gm2/docs/api/python/#dailybar)数据

dailybars = strategy.get\_last\_n\_dailybars("SHSE.600000", 10)

get\_instruments

提取交易代码。策略类和行情服务类都提供该接口。

**函数原型:**

get\_instruments(exchange, sec\_type, is\_active)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码 |
| sec\_type | int | 代码类型:1 股票，2 基金，3 指数，4 期货，5 ETF |
| is\_active | int | 当天是否交易：1 是，0 否 |

**返回值：**

Instrument对象列表

get\_instruments\_by\_name

根据期货品种提取交易代码。策略类和行情服务类都提供该接口。

**函数原型:**

get\_instruments\_by\_name(name)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| name | string | 期货品种，如'ag', 'ic' |

**返回值：**

Instrument对象列表

get\_constituents

提取指数的成分股代码。策略类和行情服务类都提供该接口。

**函数原型:**

get\_constituents(index\_symbol)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| index\_symbol | string | 指数代码 |

**返回值：**

Constituent对象列表

get\_financial\_index

按时间周期提取FinancialIndex,按时间升序排列。策略类和行情服务类都提供该接口。

**函数原型:**

get\_financial\_index(symbol, t\_begin, t\_end)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 品种代码, 如SHSE.600000 |
| t\_begin | string | 开始时间, 如2013-8-14 00:00:00 |
| t\_end | string | 结束时间, 如2013-8-15 00:00:00 |

**返回值：**

FinancialIndex对象列表

get\_last\_financial\_index

提取快照, 即最新的FinancialIndex，支持一次性提取多个代码的快照。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_financial\_index(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 多个品种代码列表, 如SHSE.600000,SZSE.000001 |

**返回值：**

FinancialIndex对象列表

get\_last\_n\_financial\_index

提取最近n条FinancialIndex。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_financial\_index(symbol, n)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 代码, 如SHSE.600000 |
| n | string | 数据个数 |

**返回值：**

FinancialIndex对象列表

get\_share\_index

按时间周期提取ShareIndex,按时间升序排列。策略类和行情服务类都提供该接口。

**函数原型:**

get\_share\_index(symbol, t\_begin, t\_end)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 品种代码, 如SHSE.600000 |
| t\_begin | string | 开始时间, 如2013-8-14 00:00:00 |
| t\_end | string | 结束时间, 如2013-8-15 00:00:00 |

**返回值：**

ShareIndex对象列表

get\_last\_share\_index

提取快照, 即最新的ShareIndex，支持一次性提取多个代码的快照。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_share\_index(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 多个品种代码列表, 如SHSE.600000,SZSE.000001 |

**返回值：**

ShareIndex对象列表

get\_last\_n\_share\_index

提取最近n条ShareIndex。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_share\_index(symbol, n)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 代码, 如SHSE.600000 |
| n | string | 数据个数 |

**返回值：**

ShareIndex对象列表

get\_market\_index

按时间周期提取MarketIndex,按时间升序排列。策略类和行情服务类都提供该接口。

**函数原型:**

get\_market\_index(symbol, t\_begin, t\_end)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 品种代码, 如SHSE.600000 |
| t\_begin | string | 开始时间, 如2013-8-14 00:00:00 |
| t\_end | string | 结束时间, 如2013-8-15 00:00:00 |

**返回值：**

MarketIndex对象列表

get\_last\_market\_index

按时间周期提取MarketIndex,按时间升序排列。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_market\_index(symbol\_list)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol\_list | string | 多个品种代码列表, 如SHSE.600000,SZSE.000001 |

**返回值：**

MarketIndex对象列表

get\_last\_n\_market\_index

按时间周期提取MarketIndex,按时间升序排列。策略类和行情服务类都提供该接口。

**函数原型:**

get\_last\_n\_market\_index(symbol, n)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 代码, 如SHSE.600000 |
| n | string | 数据个数 |

**返回值：**

MarketIndex对象列表

get\_calendar

获取交易所交易日历。策略类和行情服务类都提供该接口。

**函数原型:**

get\_calendar(exchange, start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所, 如SHSE |
| start\_time | string | 开始时间, 如2016-01-01 |
| end\_time | string | 结束时间, 如2016-03-15 |

**返回值：**

TradeDate对象列表

get\_stock\_adj

查询复权因子

**函数原型:**

get\_stock\_adj(symbol, start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 如:SZSE.000001 |
| start\_time | string | 查询开始时间 |
| end\_time | string | 查询结束时间 |

**返回值：**

StockAdjustFactor 列表

get\_divident

查询分红送股明细

**函数原型:**

get\_divident(symbol, start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| symbol | string | 如:SZSE.000001 |
| start\_time | string | 查询开始时间 |
| end\_time | string | 查询结束时间 |

**返回值：**

StockDivident 列表

get\_virtual\_contract

查询虚拟合约和真实合约对应关系

**函数原型:**

get\_virtual\_contract(vsymbol, start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| vsymbol | string | 如:CFFEX.IF, CFFEX.IF00 |
| start\_time | string | 查询开始时间 |
| end\_time | string | 查询结束时间 |

**返回值：**

返回VirtualContract列表

交易接口方法

open\_long

异步开多仓，以参数指定的symbol、价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口

**函数原型:**

open\_long(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上交所SHSE |
| sec\_id | string | 证券代码,如浦发银行600000 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 委托量 |

**返回值：**

委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

市价买入1000股上交所浦发银行

order = open\_long("SHSE", "600000", 0, 1000)

open\_short

异步开空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

open\_short(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如金融期货交易所 CFFEX |
| sec\_id | string | 证券代码,如股指期货合约1511 IF1511 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 委托量 |

**返回值：**

委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

在策略类的方法中以市价开1手股指期货合约IF1511的空单

order = open\_short("CFFEX", "IF1511", 0, 1)

**注意事项：**

1. 该接口为异步下单接口，需策略Run之后才能正常运行
2. 平仓接口只对期货有效，现货不存在空仓单

close\_long

异步平多仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

close\_long(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上交所 SHSE |
| sec\_id | string | 证券代码,如浦发银行 600000 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

在策略类的方法中以市价卖出1000股上交所浦发银行

order = close\_long("SHSE", "600000", 0, 1000)

**注意事项：**

该接口为异步下单接口，需策略Run之后才能正常运行

close\_long\_yesterday

异步平昨多仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。 此api用于平上期所昨仓，策略类和交易服务类都提供该接口。

**函数原型:**

close\_long\_yesterday(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上期所SHFE |
| sec\_id | string | 证券代码 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

以市价平1手白银合约ag1512的多单,

order = close\_long\_yesterday("SHFE", "ag1512", 0, 1)

**注意事项：**

1. 该接口为异步下单接口，需策略Run之后才能正常运行
2. 平仓接口只对期货有效，现货不存在平仓

close\_short

异步平空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

close\_short(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如股指期货交易所CFFEX |
| sec\_id | string | 证券代码,如股指期货合约IF1511 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

在策略类的方法中以市价平1手股指期货合约IF1511的空单

order = close\_short("CFFEX", "IF1511", 0, 1)

**注意事项：**

1. 该接口为异步下单接口，需策略Run之后才能正常运行
2. 平仓接口只对期货有效，现货不存在平仓

close\_short\_yesterday

异步平昨空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。 此api用于平上期所昨仓，策略类和交易服务类都提供该接口。

**函数原型:**

close\_short\_yesterday(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码，如上期所SHFE |
| sec\_id | string | 证券代码 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

以市价平1手白银合约ag1512的空单

order = close\_short\_yesterday("SHFE", "ag1512", 0, 1)

**注意事项：**

1. 该接口为异步下单接口，需策略Run之后才能正常运行
2. 平仓接口只对期货有效，现货不存在平仓

place\_order

异步下单原生函数，需要创建[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象，填充对应字段，一般异步下单接口建议使用open\_long、open\_short、close\_long、close\_short 4个快捷下单接口。如果价格price字段为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

place\_order(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | Order | 委托Order对象 |

**返回值：**

直接返回参数[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

在策略类的方法中以市价开1手IF1511的多单

order = Order()

order.exchange = "CFFEX"

order.sec\_id = "IF1511"

order.side = 1

order.position\_effect = 1

order.price = 0

order.volume = 1

order\_ret = place\_order(order)

**注意事项：**

该接口为异步下单接口，需策略Run之后才能正常运行

cancel\_order

异步撤单接口，根据参数cl\_ord\_id指定的客户端订单ID，撤销之前的下单委托。撤单是否成功取决于订单当前的状态。

**函数原型:**

cancel\_order(cl\_ord\_id)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| cl\_ord\_id | string | 委托订单的客户方id |

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

ret = cancel\_order(order.cl\_ord\_id)

**注意事项：**

1. 该接口为异步下单接口，需策略Run之后才能正常运行
2. 执行的结果由on\_execrpt, on\_order\_cancelled, on\_order\_cancel\_rejected回调方法返回

open\_long\_sync

同步开多仓接口，以参数指定的symbol, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

open\_long\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上交所SHSE |
| sec\_id | string | 证券代码,如浦发银行600000 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 委托量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

通过策略对象调用该接口，以市价买入1000股上交所浦发银行

order = strategy.open\_long\_sync("SHSE", "600000", 0, 1000)

open\_short\_sync

同步开空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

open\_short\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如金融期货交易所 CFFEX |
| sec\_id | string | 证券代码,如股指期货1511合约 IF1511 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 委托量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

通过策略对象调用该接口，以市价开1手股指期货IF1511合约的空单

strategy.open\_short\_sync("CFFEX", "IF1511", 0, 1)

**注意事项：**

平仓接口只对期货有效，现货不存在空仓单

close\_long\_sync

同步平多仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

close\_long\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上交所SHSE |
| sec\_id | string | 证券代码,如浦发银行 600000 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

通过策略对象调用该接口，以市价卖出1000股上交所浦发银行

strategy.close\_long("SHSE", "600000", 0, 1000)

close\_long\_yesterday\_sync

同步平昨多仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。 此api用于平上期所昨仓，策略类和交易服务类都提供该接口。

**函数原型:**

close\_long\_yesterday\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码，如上期所SHFE |
| sec\_id | string | 证券代码，如浦发银行 600000 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

以市价平1手白银合约ag1512的多单

order = close\_long\_yesterday\_sync("SHFE", "ag1512", 0, 1)

**注意事项：**

1.平仓接口只对期货有效，现货不存在平仓

close\_short\_sync

同步平空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

close\_short\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如股指期货交易所CFFEX |
| sec\_id | string | 证券代码,如股指期货合约IF1511 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

通过策略对象调用该接口，在策略类的方法中以市价平1手股指期货合约IF1511的空单

strategy.close\_short\_sync("CFFEX", "IF1511", 0, 1)

**注意事项：**

平仓接口只对期货有效，现货不存在平仓

close\_short\_yesterday\_sync

同步平昨空仓接口，以参数指定的exchange, 证券代码sec\_id, 价和量下单。如果价格为0，为市价单，否则为限价单。此api用于平上期所昨仓，策略类和交易服务类都提供该接口。

**函数原型:**

close\_short\_yesterday\_sync(exchange, sec\_id, price, volume)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码, 如上期所SHFE |
| sec\_id | string | 证券代码,如股指期货合约IF1511 |
| price | float | 委托价，如果price=0,为市价单，否则为限价单 |
| volume | float | 平仓量 |

**返回值：**

返回委托下单生成的[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

以市价平1手白银合约ag1512的空单

strategy.close\_short\_yesterday\_sync("SHFE", "ag1512", 0, 1)

**注意事项：**

平仓接口只对期货有效，现货不存在平仓

place\_order\_sync

同步下单原生函数，需要创建Order对象，填充对应字段，一般同步下单接口建议使用open\_long\_sync、open\_short\_sync、close\_long\_sync、close\_short\_sync 4个快捷下单接口。如果价格price字段为0，为市价单，否则为限价单。策略类和交易服务类都提供该接口。

**函数原型:**

place\_order\_sync(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | Order | 委托Order对象 |

**返回值：**

直接返回参数[Order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

**示例：**

通过策略对象调用该接口，以市价开1手IF1511的多单

order = Order()

order.exchange = "CFFEX"

order.sec\_id = "IF1511"

order.side = 1

order.position\_effect = 1

order.price = 0

order.volume = 1

order\_ret = place\_order\_sync(order)

cancel\_order\_sync

同步撤单接口，根据参数clordid指定的客户端订单ID，撤销之前的下单委托。撤单是否成功取决于订单当前的状态。策略类和交易服务类都提供该接口。

**函数原型:**

cancel\_order\_sync(cl\_ord\_id)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| cl\_ord\_id | string | 委托订单的客户方id |

**返回值：**

[返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3)

**示例：**

ret = cancel\_order\_sync(order.cl\_ord\_id)

**注意事项：**

执行的结果由[on\_execrpt](https://www.myquant.cn/gm2/docs/api/python/#on_execrpt), [on\_order\_cancelled](https://www.myquant.cn/gm2/docs/api/python/#on_order_cancelled), [on\_order\_cancel\_rejected](https://www.myquant.cn/gm2/docs/api/python/#on_order_cancel_rejected)回调方法返回

get\_order

查询单个委托信息，策略类和交易服务类都提供该接口。

**函数原型:**

get\_order(cl\_ord\_id)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| cl\_ord\_id | string | 委托订单的客户方id |

**返回值：**

[order](https://www.myquant.cn/gm2/docs/api/python/#order)对象

get\_orders

按时间段查询委托信息列表，策略类和交易服务类都提供该接口。

**函数原型:**

get\_orders(start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| start\_time | string | 开始时间，如2016-01-01 00:00:00 |
| end\_time | string | 开始时间，如2016-01-02 00:00:00 |

**返回值：**

[order](https://www.myquant.cn/gm2/docs/api/python/#order)对象列表

get\_orders\_by\_symbol

按时间段和代码查询委托信息列表，策略类和交易服务类都提供该接口。

**函数原型:**

get\_orders\_by\_symbol(exchange, sec\_id, start\_time, end\_time)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所名称， 如SHSE |
| sec\_id | string | 证券代码，如600000 |
| start\_time | string | 开始时间，如2016-01-01 00:00:00 |
| end\_time | string | 开始时间，如2016-01-02 00:00:00 |

**返回值：**

[order](https://www.myquant.cn/gm2/docs/api/python/#order)对象列表

get\_unfinished\_orders

查询未完成委托信息，策略类和交易服务类都提供该接口。

**函数原型:**

get\_unfinished\_orders()

**参数：**

无

**返回值：**

[order](https://www.myquant.cn/gm2/docs/api/python/#order)对象列表

get\_cash

查询当前策略的资金信息，策略类和交易服务类都提供该接口。

**函数原型:**

get\_cash()

**参数：**

无

**返回值：**

[Cash](https://www.myquant.cn/gm2/docs/api/python/#cash)对象，当前策略的资金信息

**示例：**

在策略类的方法中查询当前策略的资金信息

cash = get\_cash()

get\_position

查询当前策略指定symbol（由交易所代码和证券ID组成）和买卖方向的持仓信息。策略类和交易服务类都提供该接口。

**函数原型:**

get\_position(exchange, sec\_id, side);

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| exchange | string | 交易所代码 |
| sec\_id | string | 证券代码 |
| side | int | 买卖方向 |

**返回值：**

[Position](https://www.myquant.cn/gm2/docs/api/python/#position)对象，持仓信息

**示例：**

在策略类的方法中查询买入浦发银行的持仓信息

postion = get\_position("SHSE", "600000", 1)

get\_positions

查询当前策略的全部持仓信息。策略类和交易服务类都提供该接口。

**函数原型:**

get\_positions()

**参数：**

无

**返回值：**

当前策略全部持仓列表

**示例：**

在策略类的方法中查询当前策略全部持仓信息

postion = get\_positions()

get\_indicator

查询当前策略的绩效信息。策略类和交易服务类都提供该接口。

**函数原型:**

get\_indicator()

**参数：**

无

**返回值：**

当前策略的绩效信息[Indicator](https://www.myquant.cn/gm2/docs/api/python/#indicator)

**示例：**

在策略类的方法中查询当前策略的绩效信息

indicator = get\_indicator()

get\_broker\_accounts

获取柜台交易账号列表。策略类和交易服务类都提供该接口。

**函数原型:**

get\_broker\_accounts()

**参数：**

无

**返回值：**

BrokerAccount对象列表

get\_broker\_cash

获取柜台交易账号资金。策略类和交易服务类都提供该接口。

**函数原型:**

get\_broker\_cash()

**参数：**

无

**返回值：**

Cash对象列表

get\_broker\_positions

获取柜台交易账号持仓。策略类和交易服务类都提供该接口。

**函数原型:**

get\_broker\_positions()

**参数：**

无

**返回值：**

Position对象列表

行情事件方法

on\_tick

响应Tick事件，收到[Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2)数据后本函数被调用。

**函数原型:**

on\_tick(tick)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| tick | [Tick](https://www.myquant.cn/gm2/docs/api/python/#tick_2) | Tick数据 |

**返回值：**

无

on\_bar

响应Bar事件，收到[Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2)数据后本函数被调用。

**函数原型:**

on\_bar(bar)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| bar | [Bar](https://www.myquant.cn/gm2/docs/api/python/#bar_2) | Bar数据 |

**返回值：**

无

交易事件方法

on\_execrpt

响应委托执行回报事件，收到Execution数据后本函数被调用。

**函数原型:**

on\_execrpt(rpt)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| rpt | [ExecRpt](https://www.myquant.cn/gm2/docs/api/python/#execrpt) | 执行回报数据 |

**返回值：**

无

on\_order\_rejected

响应订单被拒绝事件，收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_rejected(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_new

响应订单被柜台接收事件，收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_new(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_status

响应订单状态更新事件，Order状态变更后本函数被调用。

**函数原型:**

on\_order\_status(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_filled

响应订单完全成交事件，收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_filled(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_partially\_filled

响应订单部分成交事件，收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_partially\_filled(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_stop\_executed

响应订单停止执行事件，比如，限价单到收市仍然未能成交。收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_stop\_executed(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_cancelled

响应订单撤单成功事件，收到Order变更数据后本函数被调用。

**函数原型:**

on\_order\_cancelled(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

on\_order\_cancel\_rejected

响应订单撤单请求被拒绝事件，收到Execution数据后本函数被调用。ord\_rej\_reason说明为什么撤单失败。

**函数原型:**

on\_order\_cancel\_rejected(order)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| order | [Order](https://www.myquant.cn/gm2/docs/api/python/#order) | 最新的订单数据 |

**返回值：**

无

其他方法

on\_login

策略登录事件，初始化策略时本函数被调用。

**函数原型:**

on\_login()

**参数：**

无

**返回值：**

无

on\_error

响应错误事件，策略内部出现错误时，比如行情或交易连接断开，数据错误，超时等，将触发本函数。

**函数原型:**

on\_error(error\_code, error\_msg)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| error\_code | int | [返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3) |
| error\_msg | string | 错误信息 |

**返回值：**

无

on\_backtest\_finish

回测结束事件，在回测结束时触发。

**函数原型:**

on\_backtest\_finish(indicator)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| indicator | object | 回测的绩效 |

**返回值：**

无

set\_timer

设置策略定时事件的时间间隔，单位为毫秒。

**函数原型:**

set\_timer(interval)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| interval | int | 定时器时间间隔 |

**返回值：**

无

unset\_timer

解除该时间间隔对应的定时器，单位为毫秒。

**函数原型:**

unset\_timer(interval)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| interval | int | 定时器时间间隔 |

**返回值：**

无

on\_timer

策略定时事件，按设定的时间间隔定时调用。

**函数原型:**

on\_timer(interval)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| interval | int | 定时器时间间隔 |

**返回值：**

无

set\_timeout\_val

设置同步API的超时时间，系统默认为30秒

**函数原型:**

set\_timeout\_val(seconds)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| seconds | int | 超时时间，单位为秒 |

**返回值：**

无

get\_timeout\_val

获取同步API的超时时间

**函数原型:**

get\_timeout\_val()

**参数：**

无

**返回值：**

超时时间，单位为秒

get\_strerror

根据错误码错误详细错误信息

**函数原型:**

get\_strerror(err\_code)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| err\_code | int | [返回值编码](https://www.myquant.cn/gm2/docs/api/errors/#_3) |

**返回值：**

错误信息

on\_md\_event

响应行情状态事件，收到MarketDataEvent数据后本函数被调用。

**函数原型:**

on\_md\_event(md\_event)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| md\_event | [MDEvent](https://www.myquant.cn/gm2/docs/api/python/#_67) | 开盘，收盘，回放行情结束等事件 |

**返回值：**

无

get\_version

获取SDK版本号。

**函数原型:**

get\_version()

**参数：**

无

**返回值：**

返回当前版本号信息

to\_dict

类型转换函数，将GMSDK内置类型对象转换为dict对象; 使用前从gmsdk引入,即from gmsdk import to\_dict。

**函数原型:**

to\_dict(obj)

**参数：**

| **参数名** | **类型** | **说明** |
| --- | --- | --- |
| obj | object | GMSDK内置类型对象 |

**返回值：**

dict对象

[下一章](https://www.myquant.cn/gm2/docs/api/matlab/)[上一章](https://www.myquant.cn/gm2/docs/api/csharp/)