银河启睿策略中心

约定函数¶

init - 策略初始化¶

`init`(*context*)

初始化方法 - 在回测和实时模拟交易只会在启动的时候触发一次。你的算法会使用这个方法来设置你需要的各种初始化配置。 context 对象将会在你的算法的所有其他的方法之间进行传递以方便你可以拿取到。

Parameters:

context (`StrategyContext` object) -- 策略上下文

Example:

```
def init(context):
    # cash_limit的属性是根据用户需求自己定义的，你可以定义无限多种自己随后需要的属性，系统默认只是会占用context.portfolio的关键字来调用策略的投资组合信息
    context.cash_limit = 5000
```

handle_bar - k 线数据更新¶

`handle_bar`(*context*, *bar_dict*)¶

bar数据的更新会自动触发该方法的调用。策略具体逻辑可在该方法内实现，包括交易信号的产生、订单的创建等。 在实时模拟交易中，该函数在交易时间内会每分钟被触发一次。

Parameters:

- **context** (`StrategyContext` object) -- 策略上下文
- **bar_dict** (Dict[`BarObject`]) -- key 为 market_code，value 为 bar 对象

Example:

```
def handle_bar(context, bar_dict):
    # put all your algorithm main logic here.
    # ...
    order_shares('000001.SZ', 500)
    # ...
```

handle_tick - 快照数据更新¶

`handle_tick`(*context*, *tick*)¶

在 tick 级别的策略中，已订阅快照数据的更新会自动触发该方法的调用。策略具体逻辑可在该方法内实现，包括交易信号的产生、订单的创建等。 若订阅了多个合约，不同合约快照数据的更新会分别触发该方法。(触发时间包括集合竞价和连续交易时段)。

Parameters:

- **context** (`StrategyContext` object) -- 策略上下文

- **tick** (TickObject object) -- key为market_code，value为bar数据。当前合约池内所有合约的bar数据信息都会更新在bar_dict里面

Example:

```
def handle_tick(context, tick):
    # put all your algorithm main logic here.
    # ...
    order_shares(tick.market_code, tick.last)
    # ...
```

open_auction - 集合竞价¶

open_auction(*context*, *bar_dict*)¶

盘前集合竞价发生时会触发该函数的调用，在该函数内发出的订单会以当日开盘价被撮合。 tick级别回测频率不触发集合竞价事件。

Parameters:

- **context** (StrategyContext object) -- 策略上下文
- **bar_dict** (Dict[BarObject]) -- key 为 market_code，value 为 **不完整的** bar 对象，该对象仅有 open, limit_up, limit_down 等字段，没有 close 等字段

Example:

```
def open_auction(context, bar_dict):
    # put all your algorithm main logic here.
    # ...
    market_code = "000001.SZ"
    order_shares(market_code, bar_dict[market_code].open)
    # ...
```

before_trading - 盘前¶

before_trading(*context*)¶

每天在策略开始交易前会被调用。不能在这个函数中发送订单。需要注意，该函数的触发时间取决于用户当前所订阅合约的交易时间。

举例来说，如果用户订阅的合约中存在有夜盘交易的期货合约，则该函数可能会在前一日的20:00触发，而不是早晨08:00.

Parameters:

context (StrategyContext object) -- 策略上下文

Example:

```
def before_trading(context):
    logger.info("This is before trading")
```

after_trading - 盘后¶

after_trading(*context*)¶

每天在收盘后被调用。不能在这个函数中发送订单。您可以在该函数中进行当日收盘后的一些计算。

在实时模拟交易中，该函数会在每天15:30触发。

Parameters:

context (StrategyContext object) -- 策略上下文

Next Previous