交易策略回测框架技术文档

版本号	更新时间	编写人
V1.0.0	2022/01/28	席源嘉

目录

框架简介

设计目标

1 根据给定交易标的在给定历史时间段的历史行情数据，模拟给定投资组合在给定交易策略下的交易行为，并由此统计投资组合的收益表现；

2 参考与对标目前已有的第三方量化回测平台，如同花顺 MindGo、开源的 VN.PY 等；

3 自主开发回测框架可以作为第三方量化回测平台的有效补充；

4 回测框架采用的是事件驱动模式，由事件优先队列接收与分配处理各类事件，如行情信息、交易信号、交易委托、委托成交等；

5 现阶段以股票为主要交易品种，后期逐步纳入其他交易品种，如债券、期货、期权等；

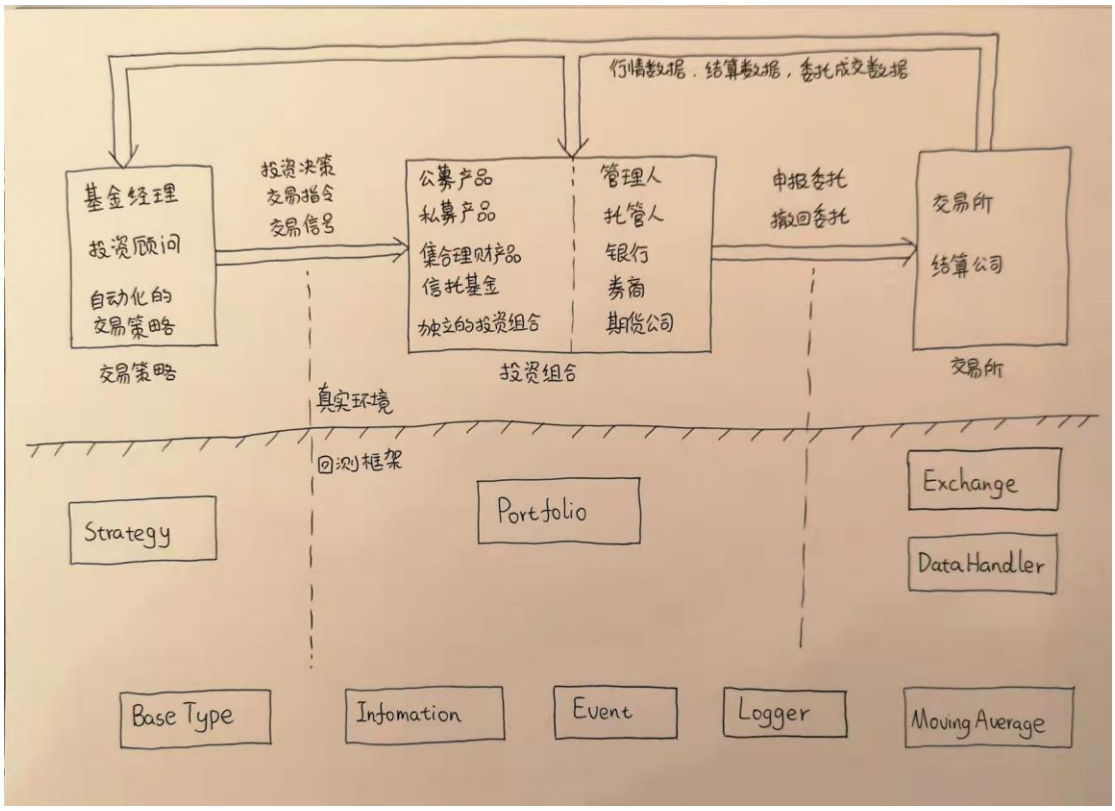
6 现阶段以券商普通账户为交易主体，带有简单的借款还款功能，支持 A 股普通交易、港股通交易，后期逐步纳入其他类型的交易主体和交易类型，如券商信用、衍生品账户、场外期权、股权投资等；

7 现阶段以连续竞价交易为主要的交易所交易形式，后期逐步纳入其他交易形式，如集合竞价、盘后交易等；

8 现阶段以人民币为唯一结算与交收货币，后期逐步纳入其他绞肉货币，如美元、港币等。

设计结构

如图所示



1 设计结构参考目前中国 A 股市场交易和以二级市场交易为主的基金的业务模式；

2 交易策略（Strategy）：

2.1 该模块主要模拟真实环境中的以下主体：

2.1.1 基金经理，

2.1.2 投资顾问，

2.1.3 自动化的交易策略；

2.2 该模块会模拟上述主体向投资组合（Portfolio）做出的投资决策、发出的交易指令或交易信号；

3 投资组合（Portfolio）：

3.1 该模块主要模拟真实环境中的以下投资组合：

3.1.1 公募基金、私募基金，

3.1.2 集合理财产品，

3.1.3 信托基金，

3.1.4 独立的投资组合；

3.2 该模块主要模拟上述投资组合的申购赎回、借款还款、交易行为，计算并记录估值变动情况以及其他相关信息；

3.3 该模块会模拟上述投资组合向交易所(Exchange)申报或撤回交易委托；

3.4 根据真实环境中的行为主体区分，该模块可以模拟以下主体的各种行为：

3.4.1 管理人，

3.4.2 托管人，

3.4.3 银行，

3.4.4 券商；

4 交易所（Exchange)：

4.1 该模块主要模拟真实环境中的以下主体：

4.1.1 交易所，

4.1.2 结算公司；

4.2 该模块主要模拟上述主体提供连续竞价交易的交易场所，以及提供整体市场的交易结算、清算、交收所需的数据；

4.3 该模块会模拟上述主体向交易策略(Strategy)和投资组合

（Portfolio）广播行情数据和结算数据，以及回报委托成交数据；

5 对应情况：

5.1 交易策略(Strategy)对应于回测框架中的 Strategy 包；

5.2 投资组合（Portfolio）对应于回测框架中的 Portfolio 包；

5.3 交易所（Exchange）对应于回测框架中的 Exchange 包和 DataHandler 包；

6 回测框架中的其他包：

6.1 BastType：用于回测框架中使用的基础类、常量、通用函数的设计；

6.2 Information:用于回测框架中传递信息所使用的各种信息类的设计；

6.3 Event：用于回测框架中事件生成与处理所使用的事件类、事件队列、各种接口类的设计；

6.4 Logger：用于回测框架中记录数据所使用的类的设计；

6.5 MovingAverage：回测框架提供的测试样例。

模块设计

事件队列

1 回测框架采用事件（Event）驱动模式；

2 事件（Event）会被统一放入回测框架的事件优先队列（EVENT_QUEUE）中，并根据事件的类型（type）分发给不同的模块进行处理；

3 基本流程：

3.1 数据处理模块（DataHandler）从数据源读取数据，并向事件队列放入 Bar 事件，

3.2 Bar 事件交由交易所模块（Exchange）和交易策略模块（Strategy）处理，

3.3 交易所模块将收到的 Bar 事件切片为 Price 事件，并放入事件队列，

3.4 Price 事件交由交易所模块、交易策略模块和投资组合模块（Portfolio）处理，

3.5 交易策略模块根据收到的 Bar 事件和 Price 事件，将发出的交易指令以 Signal 事件的形式放入事件队列，

3.6 Signal 事件交由投资组合模块处理，

3.7 投资组合模块根据收到的 Signal 事件，将申报的交易委托以 Order 事件的形式放入事件队列，

3.8 另外，投资组合模块撤回的交易委托以 Cancel 事件的形式放入事件队列，

3.9 Order 事件和 Cancel 事件均交由交易所模块处理，

3.10 交易所模块根据收到的 Price 事件、Order 事件和 Cancel 事件判断交易委托的成交情况，将成交回报以 Fill 事件的形式放入事件队列，

3.11 Fill 事件交由交易策略模块和投资组合模块处理，

3.12 交易策略模块和投资组合模块分别处理收到的 Fill 事件，并根据更新后的情况放入 Signal 事件、Order 事件或 Cancel 事件，

3.13 如果交易所模块收到的 Bar 事件产生了交易日变更，则将结算、清算所需的数据以 Clear 事件的形式放入放入事件队列，

3.14 Clear 事件交由交易策略模块和投资组合模块处理，

3.15 交易策略模块和投资组合模块分别处理收到的 Clear 事件，执行各项交易日变更时的任务，

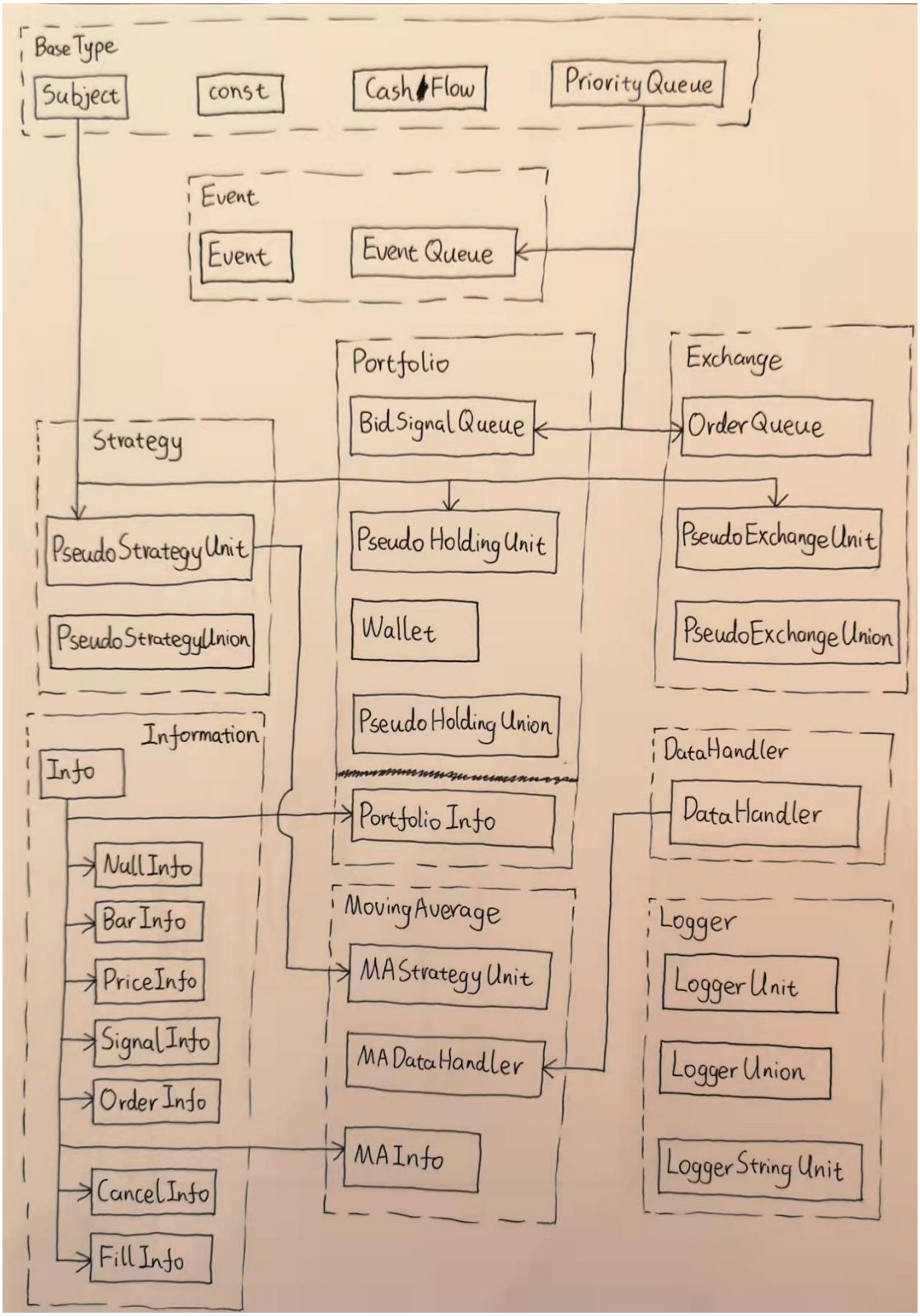
3.16 DEFAULT 事件和 END 事件作为特殊备用事件，优先级分别为最低和最高，目前主要由事件优先队列自行根据条件生成；

4 事件的优先级（值越大优先级越高，默认值为 0）、事件的生成和处理、各模块的对应情况见下表：

	事件	DEFAULT	Bar	Price	Cancel	Fill	Order	Signal	Clear	END
模块	**优先级**	-1	10	20	30	40	50	60	70	80
事件优先队列		生成/处理								生成/处理
数据处理模块			生成							
交易所	**模块**		处理	处理	处理		处理		生成	处理
	单位交易		处理	生成/处理	处理	生成	处理			处理
交易策略	**模块**		处理	处理		处理			处理	处理
	单位策略		处理	处理		处理		生成	处理	处理
投资组合	**模块**			处理	生成	处理	生成	处理	处理	处理
	单位持仓			处理		处理			处理	

类的继承关系

如图所示


Base Type
Subject
const
Cash Flow
Priority Queue
Event
Event
Event Queue
Portfolio
Bid Signal Queue
Exchange
Order Queue
Strategy
Pseudo Strategy Unit
Pseudo Strategy Union
Pseudo Holding Unit
Wallet
Pseudo Holding Union
Pseudo Exchange Unit
Pseudo Exchange Union
Information
Info
Null Info
Bar Info
Price Info
Signal Info
Order Info
Cancel Info
Fill Info
Portfolio Info
Moving Average
MA Strategy Unit
MA Data Handler
MA Info
DataHandler
DataHandler
Logger
Logger Unit
Logger Union
Logger String Unit

1 Subject：回测框架中使用的各类实体对象的基类；

1.1 PseudoStrategyUnit：交易策略（Strategy）体系中使用的以单一策略为单位的交易模块，

1.1.1 MAStrategyUnit：移动均线策略的单位策略模块，

1.2 PseudoHoldingUnit：投资组合（Portfolio）体系中使用的以标的为单位的交易模块，

1.3 PseudoExchangeUnit：交易所（Exchange）体系中使用的以标的为单位的交易模块；

2 PriorityQueue：基于最大堆（max heap）实现的优先队列（priority queue）类；

2.1 EventQueue：回测框架使用的事件优先队列，

2.2 BidSignalQueue：投资组合（Portfolio）进行买入信号资金分配时使用的信号（Signal）优先队列，

2.3 OrderQueue：交易所（Exchange）进行委托撮合时使用的委托优先队列；

3 Info：回测框架中使用的信息类的基类，此类为抽象类；

3.1 NullInfo：空信息，主要用于事件无需传递信息的情形，

3.2 BarInfo：Bar 信息，用于 Bar 事件中传递以“标的在一定时间段内的报价成交数据”为单位的市场信息，

3.3 PriceInfo：Price 信息，用于 Price 事件中传递以“标的在一个时刻的价格数据”为单位的市场信息，

3.4 SignalInfo：Signal 信息，用于 Signal 事件中传递由

交易策略（strategy）发出的交易信号的相关信息，

3.5 OrderInfo：Order 信息，用于 Order 事件中传递由持仓组合（Portfolio）发出的委托的相关信息，

3.6 CancelInfo：Cancel 信息，用于 Cancel 事件中传递由持仓组合（portfolio）发出的撤回委托的相关信息，

3.7 Fillinfo：Fill 信息，用于 Fill 事件中传递由交易所（exchange）返回的委托成交的相关信息，

3.8 PortfolioInfo：Portfolio 信息，用于投资组合(Portfolio)体系中的投资组合/基金/集合理财产品的相关信息，

3.9 MAInfo：MAInfo 信息，用于记录移动均线策略（MovingAverage）的相关信息；

4 DataHandler：回测框架中使用的处理输入数据的基类，此类为抽象类；

4.1 MADataHandler：移动均线策略的输入数据处理模块

接口类

1 回测框架中，针对事件（Event）的处理设计了一系列接口类（interface）；

2 定义类时继承目标事件的接口类，并实现对应的抽象方法，并在初始化方法中向事件队列（EVENT_QUEUE）注册目标事件和实现方法；

3 事件处理（EventHandler）接口类明细：

3.1 DEFAULTHandler：

针对处理 DEFAULT 事件的接口类

抽象方法 on_default(event: Event) -> None

3.2 ENDHandler：

针对处理 END 事件的接口类

抽象方法 on_end(event: Event) -> None

3.3 BarHandler：

针对处理 Bar 事件的接口类

抽象方法 on_bar(event: Event) -> None

3.4 PriceHandler：

针对处理 Price 事件的接口类

抽象方法 on_price(event: Event) -> None

3.5 ClearHandler：

针对处理 Clear 事件的接口类

抽象方法 on_clear(event: Event) -> None

3.6 SignalHandler：

针对处理 Signal 事件的接口类

抽象方法 on_signal(event: Event) -> None

3.7 OrderHandler：

针对处理 Order 事件的接口类

抽象方法 on_order(event: Event) -> None

3.8 CancelHandler：

针对处理 Cancel 事件的接口类

抽象方法 on_cancel(event: Event) -> None

3.9 FillHandler：

针对处理 Fill 事件的接口类

抽象方法 on_fill(event: Event) -> None

类、常量与函数的设计

BaseType

Const.py

1 用于设计和保存回测框架中使用的常量

2 const(object)：用于保存回测框架使用的各种常量的类

3 定义 const 类的实例 CONST，作为全局变量

3.1 CURRENCY（默认货币)：人民币（CNY)

3.2 START_TIME（默认起始时间)：1900/01/01 00:00:00

3.3 END_TIME（默认结束时间)：2099/12/31 23:59:59

3.4 DEFAULT_QUEUE_SIZE（默认队列长度)：16

3.5 ENCODING（默认文件编码)：GB2312

3.6 RESAMPLE（默认合并 tick 时间周期)：1 分钟

3.7 THIS_PATH（项目本地地址)

3.8 QUEUE_PATH（事件队列记录地址)：

/log/QueueLog.csv

3.9 PORTFOLIO_PATH（Portfolio 信息记录地址)：

/log/PortfolioLog.csv

3.10 STRATEGY_PATH（Strategy 信息记录地址)：

/log/StrategyLog.csv

3.11 SYMBOL（默认标的代码)：NULL

3.12 EXCHANGE (默认交易所): NULL

3.13 PER_HAND (默认每手数量): 100

3.14 PER_PRICE (默认报价单位): 0.01

3.15 BID_COMMISSION (默认买入费用定额): 0.0

3.16 BID_COMMISSION_RATE(默认买入费用费率):0.00015

3.17 BID_TAX (默认买入缴税定额): 0.0

3.18 BID_TAX_RATE (默认买入缴税税率): 0.0

3.19 ASK_COMMISSION (默认卖出费用定额): 0.0

3.20 ASK_COMMISSION_RATE(默认卖出费用费率):0.00015

3.21 ASK_TAX (默认卖出缴税定额): 0.0

3.22 ASK_TAX_RATE (默认卖出缴税税率): 0.001

3.23 CRT_PRICE (默认单位现价): 0.0

3.24 NET_PRICE (默认单位净值): 0.0

3.25 BOOK_PRICE (默认单位面值): 1.0

3.26 VOLUME (默认数量): 0.0

3.27 MULTIPLIER (默认乘数): 1

3.28 MARGIN_RATE (默认保证金比例): 1

3.29 TIME_OFFSET (默认时间流逝单位): seconds

3.30 TIME_OFFSET_TIMES (默认时间流逝量): 1

Subject.py

1 用于设计回测框架中各类实体对象的基类

2 Subject(object)：回测框架中使用的各类实体对象的基类

2.1 __init__()：初始化

@symbol_(str)：标的代码

@exchange_(str)：交易所

@last_datetime_(pandas.Timestamp)：最新时间戳

@per_hand_(int)：每手数量

@per_price_(int)：报价单位

@bid_commission_(float)：买入费用定额

@bid_commission_rate_(float)：买入费用费率

@ask_commission_(float)：卖出费用定额

@ask_commission_rate_(float)：卖出费用费率

@bid_tax_(float)：买入缴税定额

@bid_tax_rate_(float)：买入缴税税率

@ask_tax_(float)：卖出缴税定额

@ask_tax_rate_(float)：卖出缴税税率

@crt_price_(float)：单位现价

@net_price_(float)：单位净值

@book_value_(float)：单位面值

@volume_(float)：当前数量

@multiplier_(int)：乘数

@margin_rate_(float)：保证金比率

@currency_(str)：货币代码

2.2 其他成员

2.2.1 crt_amount：现值

2.2.2 net_amount：净值

2.2.3 book_amount：面值

2.3 refresh()：更新以人民币（CNY）为单位的对象现值（crt_amount）、净值（net_amount）、面值(book_amount)

@return(None)

2.4 amount_to_volume()：对于给定金额的人民币（CNY)、给定交易价格和给定交易方向，计算交易数量

@amount_(float)：给定给定金额的人民币（CNY)

@price_(float)：给定交易价格

@direction_(int)：给定交易方向

@return(float)：交易数量，按每手数量取整

当交易方向为买入时，计算给定金额的人民币（CNY）可买入的最大数量，按每手数量取整

当交易方向为卖出时,计算获得给定金额的人民币(CNY)所需卖出的最小数量，按每手数量取整

2.5 cash_flow_to_volume()：对于给定的现金流、给定交易价格和给定交易方向，计算交易数量

@cash_flow_(CashFlow)：给定的现金流

@price_(float)：给定交易价格

@direction_(int)：给定交易方向

@return(float)：交易数量，按每手数量取整

将给定的现金流转换为以对象的货币代码为单位的金额

当交易方向为买入时，计算给定的现金流可买入的最大数量，按每手数量取整

当交易方向为卖出时，计算获得给定的现金流所需卖出的最小数量，按每手数量取整

2.6 volume_to_amount()：对于给定的交易数量、给定交易价格和给定交易方向，计算以人民币（CNY）为单位的交易金额

@volume_(float)：给定的交易数量

@price_(float)：给定交易价格

@direction_(int)：给定交易方向

@return(float)：以人民币（CNY）为单位的交易金额

当交易方向为买入时，计算买入给定数量的标的所需的以人民币（CNY）为单位的金额

当交易方向为卖出时，计算卖出给定数量的标的获得的以人民币（CNY）为单位的金额，至少为 0

2.7 volume_to_amount()：对于给定的交易数量、给定交易价格和给定交易方向，计算以对象的货币代码为单位的现金流

@volume_(float)：给定的交易数量

@price_(float)：给定交易价格

@direction_(int)：给定交易方向

@return(CashFlow)：以对象的货币代码为单位的现金流

当交易方向为买入时，现金流的金额为买入给定数量的标的所需的以对象的货币代码为单位的金额

当交易方向为卖出时，现金流的金额为卖出给定数量的标的获得的以对象的货币代码为单位的金额，至少为 0

2.8 set()：设置对象的属性

args(dict)：需要修改的属性的(name, value)键值对

return(None)

2.9 time_offset()：对象内的最新时间戳的时间流逝

@offset(str)：时间流逝的单位颗粒

@times(int)：时间流逝的颗粒数量

@return(None)

调用 pandas.tseries.offsets.DateOffset 方法

PriorityQueue.py

1 用于设计通用的给予最大堆的优先队列，回测框架中其他优先队列的基类

2 PriorityQueue(object)：基于最大堆（max heap）实现的优先队列（priority queue）类

2.1 __init__()：初始化

@factory_(Callable[None，队列元素类])：队列元素的默认生成函数

2.2 其他成员：

2.2.1 size：最大堆的当前容量，初始化为CONST["DEFAULT_QUEUE_SIZE"]

2.2.2 heap：列表形式的最大堆

2.2.3 max_index：堆中最后一个元素的下标

2.3 is_empty()：判断优先队列当前是否为空

@return(bool)：最大堆是否为空

2.4 clear()：清空当前的优先队列

@return(None)

2.5 first()：查询优先队列当前的第一个元素，队列为空时报错

@return(队列元素类)：优先队列当前的第一个元素

2.6 put()：将给定元素放入优先队列

@obj(队列元素类)：放入优先队列的元素

@return(None)

2.7 pop()：对于给定下标，弹出最大堆中对应下标的元素，下标无效时报错

@i(int)：给定的结点下标

@return(队列元素类)：最大堆中对应下标的元素

2.8 get()：取出优先队列当前的第一个元素，队列为空时报错，等价于调用 pop()

@return(队列元素类)： 优先队列当前的第一个元素

ExchangeRate.py

1 用于回测框架中与汇率和货币兑换有关的设计

FROM_CNY：记录各个货币的购汇汇率的全局变量

TO_CNY：记录各个货币的结汇汇率的全局变量

默认人民币（CNY）汇率为 1

设置默认港币（HKD）汇率

2 set_exchange_rate()： 设置给定货币的购汇和/或结汇汇率

@currency_(str)： 货币代码

@from_cny_(float)： 购汇汇率，默认为 None

@to_cny_(float)： 结汇汇率，默认为 None

return(None)

3 is_valid_currency()： 判断给定货币是否有效

@currency_(str)： 货币代码

return(bool)： FROM_CNY、TO_CNY 均有 currency_记录

4 amount_from_cny()： 对于给定金额的给定货币，计算购汇所需的人民币（CNY）金额

@currency_(str)： 货币代码

@amount_(float)：给定金额

return(float)：购汇所需的人民币（CNY）金额，保留 2 位小数

5 from_amount_of_cny()：对于给定金额的人民币（CNY），计算购汇所得的给定货币金额

@currency_(str)：货币代码

@amount_(float)：给定金额

return(float)：购汇所得的给定货币金额，保留 2 位小数

6 amount_to_cny()：对于给定金额的给定货币，计算结汇所得的人民币（CNY）金额

@currency_(str)：货币代码

@amount_(float)：给定金额

return(float)：结汇所得的人民币（CNY）金额，保留 2 位小数

7 to_amount_of_cny()：对于给定金额的人民币（CNY），计算结汇所需的给定货币金额

@currency_(str)：货币代码

@amount_(float)：给定金额

return(float)：结汇所需的给定货币金额，保留 2 位小数

CashFlow.py

1 用于回撤框架中与现金流动相关的设计

2 CashFlow：回测框架中用于表示现金流（cashflow）数据的类

2.1 __init__()：初始化

@currency_(str)：现金流的货币代码（已在回测框架设置）

@amount_(float)：现金流的金额（非负）

2.2 to_cny()：对于当前现金流，计算结汇所得的人民币(CNY)金额

return(float)：结汇所得的人民币（CNY）金额，保留 2 位小数

2.3 __gt__()：比较两笔现金流结汇所得的人民币（CNY）金额

3 cashflow_exchange()：将给定的现金流，换汇成给定货币的现金流

@cash_flow_(CashFlow)：给定的现金流，默认为 None

@currency_(str)：货币代码，默认为 CNY

@return(Optional[CashFlow])：如果换汇成功，则返回一笔现金流（CashFlow），失败则返回 None

Information

Info.py

1 用于回测框架中使用的信息类的设计

2 Info：回测框架中使用的信息类的基类，此类为抽象类

2.1 __repr__()：

@abstractmethod

强制要求子类实现__repr__()方法

2.2 type(str)：用于表示信息类型的成员

3 NullInfo(Info)：空信息，主要用于事件无需传递信息的情形

3.1 type = None

4 BarInfo(Info)：Bar 信息，用于 Bar 事件中传递以"标的在一定时间段内的报价成交数据"为单位的市场信息

4.1 type = "Bar"

4.2 __init__()：初始化

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@open_(float)：开盘价

@high_(float)：最高价

@low_(float)：最低价

@close_(float)：收盘价

@volume_(float)：成交数量

@turnover_(float)：成交金额

5 PriceInfo(Info)：Price 信息，用于 Price 事件中传递以“标的在一个时刻的价格数据”为单位的市场信息

5.1 type = "Price"

5.2 __init__()：初始化

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@crt_price_(float)：现价

@last_price_(float)：上一个价格，默认为 0

@volume_(float)：成交数量，默认为 0

6 SIGNAL_PRIORITY：记录各类信号的优先级别的全局变量，值越大优先级越高，默认值为 0

信号按照对于信号所包含的交易行为的处理方式进行分类：

FOK (Fill or Kill)：即时，或者全部转化为委托，或者全部作废，默认值为 10

IOC (Immediate or Cancel)：即时，最大限度转化为委托，其余作废，默认值为 20

FOW (Fill or Wait)：等待直到某一时刻，或者全部转化为委托，或者主动作废，默认值为 30

TBF (to be Fill)：持续有效，直至信息全部转化为委托，或者主动作废，默认值为 40

7 SignalInfo(Info)：Signal 信息，用于 Signal 事件中传递

由交易策略（strategy）发出的交易信号的相关信息

7.1 type = "Signal"

7.2 __init__()：初始化

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@direction_(int)：交易方向，买入为 1，卖出为-1

@open_or_close_(int)：开平仓标志，开仓为 1，平仓为 -1

@price_(float)：交易价格

@volume_(float)：交易数量

@amount_(float)：预算交易金额，默认为 0

@currency_(str)：交易货币代码，默认为 CNY

@signal_type_(str)：信号分类，默认为 FOW

@uid_(uuid.UUID)：信号 ID，默认为 None

7.3 __gt__()：比较信号的优先级：分类优先、预算金额较少优先

8 SIGNAL_MAP_ORDER：记录由各类信号生成各类委托时的类型映射关系

委托按照交易所撮合系统对于委托的处理方式进行分类：

FOK（Fill or Kill）：即时，或者全部成交，或者全部撤回

IOC（Immediate or Cancel）：即时，最大限度成交，其余撤回

TBF（to be Fill)：持续有效，直至全部成交，或者主动撤回

GFD（good for day)：当日有效，持续至全部成交，或者主动撤回，或者当日清算时撤回

考虑到回测系统连续运作的便利性，暂不自动生成 GFD 委托

9 OrderInfo(Info)：Order 信息，用于 Order 事件中传递由持仓组合（Portfolio）发出的委托的相关信息

9.1 type = "Order"

9.2 __init__()：初始化

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@direction_(int)：交易方向，买入为 1，卖出为-1

@open_or_close_(int)：开平仓标志，开仓为 1，平仓为-1

@price_(float)：交易价格

@volume_(float)：交易数量

@uid_(uuid.UUID)：委托 ID，默认为 None

@order_type_(str)：委托分类，默认为 TBF

9.3 get_uuid()：获取委托信息的 ID

@return(uuid.UUID)：委托信息的 ID

9.4 __gt__()：比较委托的优先级：价格优先、时间优先

10 CancelInfo(Info)：Cancel 信息，用于 Cancel 事件中传递

由持仓组合（portfolio）发出的撤回委托的相关信息

10.1 type = "Cancel"

10.2 __init__()：初始化

@uid_(uuid.UUID)：委托 ID

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@direction_(int)：交易方向，买入为 1，卖出为-1

10.3 get_uuid()：获取撤回委托的 ID

@return(uuid.UUID)：撤回委托的 ID

11 FillInfo(Info)：Fill 信息，用于 Fill 事件中传递由交易所（exchange）返回的委托成交的相关信息

11.1 type = “Fill”

11.2 __init__()：初始化

@uid_(uuid.UUID)：委托 ID

@symbol_(str)：标的代码

@datetime_(pandas.Timestamp)：信息时间戳

@direction_(int)：交易方向，买入为 1，卖出为-1

@open_or_close_(int)：开平仓标志，开仓为 1，平仓为 -1

@filled_price_(float)：成交价格

@volume_(float)：成交数量

@partial_(bool)：是否部分成交，默认为 False

11.3 get_uuid()：获取成交委托的 ID

@return(uuid.UUID)：成交委托的 ID

Event

Event.py

1 EVENT_PRIORITY：记录各类事件的优先级别的全局变量，值越大优先级越高，默认值为 0

2 EVENT_PRIORITY：记录不包含信息的空事件的类型标签

默认包含{"DEFAULT", "Clear", "END"}

3 Event(object)：回测框架所使用的事件类

3.1 __init__()：初始化

@type_(str)：事件分类

@datetime_(pandas.Timestamp)：信息时间戳

@info_(Info)：事件包含的信息（信息分类应与事件分类一致)

除非事件属于不包含信息的空事件，否则事件分类标签和信息分类标签必须一致

3.2 __gt__()：比较事件的优先级：分类优先、时间优先

EventHandler.py

1 DEFAULTHandler：针对处理 DEFAULT 事件的接口类

1.1 on_default(self, event: Event) -> None:

@abstractmethod

强制要求包含 DEFAULTHandler 接口的子类实现 on_default 方法

2 ENDHandler：针对处理 END 事件的接口类

2.1 on_end(self, event: Event) -> None:

@abstractmethod

强制要求包含 ENDHandler 接口的子类实现 on_end 方法

3 BarHandler：针对处理 Bar 事件的接口类

3.1 on_bar(self, event: Event) -> None:

@abstractmethod

强制要求包含 BarHandler 接口的子类实现 on_bar 方法

4 PriceHandler：针对处理 Price 事件的接口类

4.1 on_price(self, event: Event) -> None:

@abstractmethod

强制要求包含 PriceHandler 接口的子类实现 on_price 方法

5 ClearHandler：针对处理 Clear 事件的接口类

5.1 on_clear(self, event: Event) -> None:

@abstractmethod

强制要求包含 ClearHandler 接口的子类实现 on_clear 方法

6 SignalHandler：针对处理 Signal 事件的接口类

6.1 on_signal(self, event: Event) -> None:

@abstractmethod

强制要求包含 SignalHandler 接口的子类实现 on_signal 方法

7 OrderHandler：针对处理 Order 事件的接口类

7.1 on_order(self, event: Event) -> None：

@abstractmethod

强制要求包含 OrderHandler 接口的子类实现 on_order 方法

8 CancelHandler：针对处理 Cancel 事件的接口类

8.1 on_cancel(self, event: Event) -> None：

@abstractmethod

强制要求包含 CancelHandler 接口的子类实现 on_cancel 方法

9 FillHandler：针对处理 Fill 事件的接口类

9.1 on_fill(self, event: Event) -> None：

@abstractmethod

强制要求包含 FillHandler 接口的子类实现 on_fill 方法

EventQueue.py

1 用于回测框架中使用的事件优先队列的设计

2 HANDLER_TYPE(Event -> None)：对于事件处理接口所实现的函数类型的定义

3 IGNORE_LIST：忽略记录的事件标签

默认包含{"Price"}

4 EventQueue(PriorityQueue, DEFAULTHandler, ENDHandler)：

回测框架使用的事件优先队列

可处理事件：DEFAULT、END

4.1 __init__()：初始化

@default_handler(HANDLER_TYPE)：自定义 DEFAULT 事件处理方法，默认为 None

@end_handler(HANDLER_TYPE)：自定义 END 事件处理方法，默认为 None

如果未提供自定义 DEFAULT 事件处理方法，则使用 self.on_default 方法

如果未提供自定义 END 事件处理方法，则使用 self.on_end 方法

4.2 register()：将给定事件处理方法，加入给定事件分类标签的处理方法列表中

@event_type_(str)：给定事件分类标签

@handler_(HANDLER_TYPE)：给定事件处理方法

@return(None)

4.3 process_next()：处理下一事件，根据事件的分类标签，依次应用于标签对应的处理方法列表中的方法

@return(None)

如果下一事件的分类不在忽略记录的列表中，则在事件记录模块中记录事件

4.4 process_through()：处理事件，直至事件队列为空

@return(None)

4.5 run()：根据给定的事件迭代器（如有），运行事件队列

@iter_(Iterator)：给定的事件（Event）迭代器，默认为 None

@return(None)

如果提供了事件迭代器，则每次向事件队列中装入一个迭代器提供的事件，并运行至事件队列为空

如果未提供事件迭代器，或提供的事件迭代器已处理完，则装入一个 DEFAULT 事件，并运行至事件队列为空

4.6 run_until()：运行事件队列，直至给定的时间戳之前

@datetime_(pandas.Timestamp)：处理截止时间戳

@return(None)

停止标准为事件队列已空，或队列中下一个事件的时间戳晚于给定的时间戳

4.7 on_default()：处理给定 DEFAULT 事件的方法

@event(Event)：给定的 DEFAULT 事件

@return(None)

4.8 on_end()：处理给定 END 事件的方法

@event(Event)：给定的 END 事件

@return(None)

5 EVENT_QUEUE：EventQueue 类的实例，作为全局变量

EventLogger.py

1 用于回测框架中事件记录的设计

2 EVENT_LOGGER 为类 LoggerStringUnit 的实例，回测框架使用的事件记录模块，作为全局变量

DataHandler

DataHandler.py

1 用于回测框架中输入数据处理的设计

2 DataHandler：回测框架中使用的处理输入数据的基类，此类为抽象类

2.1 publish_bar()：将处理输入数据得到的 Bar 事件放入回测框架的事件队列（EVENT_QUEUE）

@abstractmethod

强制要求子类实现 publish_bar()方法

2.2 bar_iterator()：将处理输入数据得到的 Bar 事件装入迭代器（Iterator）并返回

@return(Iterator)：包含处理输入数据得到的 Bar 事件的迭代器

@abstractmethod

强制要求子类实现 bar_iterator()方法

2.3 __iter__()：

使得 DataHandler 类满足 Iterable 的要求，默认返回 bar_iterator()方法的结果

Exchange

OrderQueue.py

1 用于交易所(Exchange)中用于集中竞价交易撮合的委托优先队列的设计

2 OrderQueue(PriorityQueue)：交易所(Exchange)进行委托撮合时使用的委托优先队列

2.1 __init__()：初始化

@symbol_(str)：标的代码

@direction_(str)：交易方向，买入/卖出

2.2 put(self, o: OrderInfo) -> None：

仅当委托标的代码、交易方向均与委托队列一致时，才将委托放入撮合队列

2.3 cancel()：根据给定的委托ID撤销对应委托

@uid_(uuid.UUID)：委托ID

@return(None)

2.4 cross()：根据给定的现价进行交易撮合

@crt_price_(float)：现价

@return(Generator)：包含成交委托的生成器，根据成交顺序排列

仅根据现价与委托价格判断是否成交，暂不考虑委托数量与当前盘口的关系

Exchange.py

1 用于回测框架内与交易所（Exchange）有关的设计

2 day_bar_slicer()：将给定的标的以交易日为单位的报价成交数据切片，转换成多个包含“标的在一个时刻的价格数据”信息的 Price 事件

@bar(Info.BarInfo)： 给定的标的以交易日为单位的报价成交数据

@return(Generator)： 包含“标的在一个时刻的价格数据”信息的 Price 事件的生成器，以时间戳顺序排列

将开盘价、收盘价、最高价、最低价依次指定给交易日的时点：09:30、11：30、13:00、15:00

如果开盘价不高于收盘价，则采用开盘价、最低价、最高价、收盘价的顺序

如果开盘价高于收盘价，则采用开盘价、最高价、最低价、收盘价的顺序

3 minute_bar_slicer()：将给定的标的以 1 分钟为单位的报价成交数据切片，转换成多个包含“标的在一个时刻的价格数据”信息的 Price 事件

@bar(Info.BarInfo)： 给定的标的以 1 分钟为单位的报价成交数据

@return(Generator)： 包含“标的在一个时刻的价格数据”信息的 Price 事件的生成器，以时间戳顺序排列

将开盘价、收盘价、最高价、最低价依次指定给 1 分钟内的时点：+0S、+15S、+30S、+45S

如果开盘价不高于收盘价，则采用开盘价、最低价、最高价、收盘价的顺序

如果开盘价高于收盘价，则采用开盘价、最高价、最低价、收盘价的顺序

4 order_to_fill()：根据给定的交易委托信息(OrderInfo)、成交时间、成交价格、成交数量，生成委托成交信息(FillInfo)

@order_(Info.OrderInfo)：给定的交易委托信息(OrderInfo)

@datetime_(pandas.Timestamp)：给定的成交时间

@filled_price_(float)：给定的成交价格，默认为 None

@volume_(float)：给定的成交数量，默认为 None

@return(Info.FillInfo)：生成委托成交信息(FillInfo)

如果没有提供成交价格或成交数量，则使用交易委托信息(OrderInfo)中的成交价格或成交数量

5 PseudoExchangeUnit(Subject, BarHandler, PriceHandler, OrderHandler, CancelHandler, ClearHandler, ENDHandler)：

回测框架中，交易所(Exchange)体系中使用的以标的为单位的交易模块

可处理事件：Bar、Price、Order、Cancel、Clear、END

除了常规的通过参数进行初始化的方式外，提供了三种简化的初始化方式：

分别通过 BarInfo、PriceInfo、OrderInfo 进行初始化

5.1 __init__()：初始化

@bar(Info.BarInfo)：用于初始化的 Bar 信息，默认为 None

@price(Info.PriceInfo)：用于初始化的 Price 信息，默认为 None

@order(Info.OrderInfo)：用于初始化的 Order 信息，默认为 None

@bar_slicer_(BarInfo->Iterable[Event])： Bar 信息到 Price 事件的切片器，默认为 minute_bar_slicer

如果提供了 Bar 信息，则通过 Bar 信息初始化

如果提供了 Price 信息，则通过 Price 信息初始化

如果提供了 Order 信息，则通过 Order 信息初始化，同时记录对 Order 事件的处理

如果上述信息均未提供，则进行常规的参数初始化

如果提供了 Order 信息，则将 Order 信息放入委托撮合队列

5.2 cross()：当前交易模块进行交易撮合，成交结果作为 Fill 事件放入事件队列

@return(None)

如果没有现价（crt_price）数据，则不进行撮合

如果现价(crt_price)低于前一价格(last_price)，则对买入委托队列进行撮合

如果现价(crt_price)高于前一价格(last_price)，则对卖出委托队列进行撮合

5.3 on_bar()：接收并处理 Bar 事件

@event(Event)：接收的 Bar 事件

@return(None)

使用 bar_slicer 方法，将 Bar 事件包含的信息拆分为若干个 Price 事件并放入事件队列

5.4 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

5.5 on_order()：接收并处理 Order 事件，注意默认设计不区分委托类别，统一按照 TBF 类型处理

@event(Event)：接收的 Order 事件

@return(None)

如果 Order 信息包含的委托可以即时成交，则将成交结果作为 Fill 事件放入事件队列

否则，根据交易方向放入对应的交易委托队列，等待撮合

5.6 on_cancel()：接收并处理 Cancel 事件

@event(Event)：接收的 Cancel 事件

@return(None)

根据 Cancel 事件中包含的交易委托 ID，从交易委托队列中撤回对应委托

5.7 on_clear()：接收并处理 Clear 事件

@event(Event)：接收的 Clear 事件

@return(None)

5.8 on_end()：接收并处理 END 事件

@event(Event)：接收的 END 事件

@return(None)

5.9 cancel_all()：撤回当前交易委托队列中的所有委托

@return(None)

6 ExchangeUnion(BarHandler, PriceHandler, OrderHandler, CancelHandler, ClearHandler, ENDHandler)：

回测框架中，交易所（Exchange）体系中使用的以交易所为单位的交易模块

可处理事件：Bar、Price、Order、Cancel、Clear、END

如果有继承 PseudoExchangeUnit 类的自定义单位交易模块，初始化时需要提交单位交易模块的初始化方法

6.1 __init__()：初始化

@factory_（单位交易模块初始化方法）：继承 PseudoExchangeUnit 类的自定义单位交易模块，默认为

PseudoExchangeUnit

在 EVENT_QUEUE 中注册交易所（Exchange）体系中的事件处理方法

6.2 register()：当前的交易所模块中，注册给定的单位交易模块

@unit(PseudoExchangeUnit)：给定的单位交易模块

@return(None)

6.3 on_bar()：接收并处理 Bar 事件

@event(Event)：接收的 Bar 事件

@return(None)

如果标的代码（symbol）未注册，则通过事件中的 BarInfo 生成单位交易模块并注册

如果交易日发生变更，则向事件队列放入前一交易日的 Clear 事件

将 Bar 事件交给标的代码（symbol）对应的单位交易模块处理

6.4 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

如果标的代码（symbol）未注册，则通过事件中的 PriceInfo 生成单位交易模块并注册

将 Price 事件交给标的代码（symbol）对应的单位交易

模块处理

6.5 on_order()：接收并处理 Order 事件，注意默认设计不区分委托类别，统一按照 TBF 类型处理

@event(Event)：接收的 Order 事件

@return(None)

如果标的代码（symbol）未注册，则通过事件中的 OrderInfo 生成单位交易模块并注册，同时处理 OrderInfo 中包含的交易委托

否则，将 Order 事件交给标的代码（symbol）对应的单位交易模块处理

6.6 on_cancel()：接收并处理 Cancel 事件

@event(Event)：接收的 Cancel 事件

@return(None)

如果标的代码（symbol）已注册，将 Cancel 事件交给标的代码（symbol）对应的单位交易模块处理

6.7 on_clear()：接收并处理 Clear 事件

@event(Event)：接收的 Clear 事件

@return(None)

将 Clear 事件交给已注册的所有单位交易模块处理

6.8 on_end()：接收并处理 END 事件

@event(Event)：接收的 END 事件

@return(None)

向事件队列放入当前易日的 Clear 事件

6.9 cancel_all()：撤回所有单位交易模块当前交易委托队列中的所有委托

@return(None)

Strategy

Strategy.py

1 用于回测框架内与交易策略（Strategy）有关的设计

2 PseudoStrategyUnit(Subject, PriceHandler, BarHandler, FillHandler, ClearHandler, ENDHandler)：

回测框架中，交易策略（Strategy）体系中使用的以单一策略为单位的交易模块

可处理事件：Bar、Price、Fill、Clear、END

除了常规的通过参数进行初始化的方式外，提供了通过 FillInfo 进行初始化的简化方式

可以视为一个抽象类，回测中使用的交易策略可以装入继承这个类的子类中，并实现各种方法

2.1 __init__()：初始化

@init_fill(Info.FillInfo)：用于初始化的 Fill 信息，默认为 None

如果未提供 Fill 信息，则进行常规的参数初始化

如果提供了 Fill 信息，则通过 Fill 信息初始化

2.2 put_signals(self) -> None：

@abstractmethod

强制继承 PseudoStrategyUnit 类的子类实现

put_signals 方法

回测框架使用的交易策略单位模块，应该至少实现一种生成信号（Signal）的方法

3 StrategyUnion(PriceHandler, BarHandler, FillHandler, ClearHandler, ENDHandler)：

回测框架中，交易策略（Strategy）体系中使用的以投资顾问/基金经理为单位的交易模块

可处理事件：Bar、Price、Fill、Clear、END

如果有继承 PseudoStrategyUnit 类的自定义单位策略模块，初始化时需要提交单位策略模块的初始化方法

3.1 __init__()：初始化

@factory_(单位策略模块初始化方法)：继承 PseudoStrategyUnit 类的自定义单位策略模块，默认为 PseudoStrategyUnit

在 EVENT_QUEUE 中注册交易策略（Strategy）体系中的事件处理方法

3.2 register()：当前的投资顾问/基金经理模块中，注册给定的单位策略模块

@strategy(PseudoStrategyUnit)：给定的单位策略模块

@return(None)

3.3 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

将 Price 事件交给标的代码（symbol）对应的单位交易模块处理

3.4 on_bar()：接收并处理 Bar 事件

@event(Event)：接收的 Bar 事件

@return(None)

将 Bar 事件交给标的代码（symbol）对应的单位交易模块处理

3.5 on_fill()：接收并处理 Fill 事件

@event(Event)：接收的 Fill 事件

@return(None)

如果标的代码（symbol）未注册，且 Fill 事件为买入开仓委托成交，则通过事件中的 FillInfo 生成单位策略模块并注册

将 Fill 事件交给标的代码（symbol）对应的单位策略模块处理

3.6 on_clear()：接收并处理 Clear 事件

@event(Event)：接收的 Clear 事件

@return(None)

将 Clear 事件交给已注册的所有单位交易模块处理

3.7 on_end()：接收并处理 END 事件

@event(Event)： 接收的 END 事件

@return(None)

将 END 事件交给已注册的所有单位策略模块处理

Portfolio

Wallet.py

1 用于投资组合（Portfolio）体系中用于进行现金管理的设计

2 Wallet(object)：回测框架中，投资组合（Portfolio）体系中用于进行现金管理的模块

Wallet 类中管理的现金统一以人民币（CNY）的形式存在

对于其他货币的现金流入，以即时的结汇汇率进行结汇

对于其他货币的现金流出，以即时的购汇汇率进行购汇

2.1 get_total()：获取当前以人民币（CNY）为单位的现金余额，包括可用资金和处于冻结状态的资金

@return(float)：当前以人民币（CNY）为单位的现金余额

2.2 has_available()：对于给定金额的给定货币的现金流出，判断是否有足够的以人民币（CNY）计算的可用资金

@currency_(str)：货币代码

@amount_(float)：给定金额

@return(bool)：是否有足够的以人民币（CNY）计算的可用资金

2.3 input()：处理给定的现金流入

@cash_flow_(CashFlow)：给定的现金流入

@return(None)

将给定的现金流入结汇成人民币（CNY）后，计入可用资金

2.4 output()：处理给定金额的给定货币的现金流出需求

@currency_(str)：货币代码

@amount_(float)：给定金额

@return(Optional[CashFlow])：对现金流出需求的处理结果，成功则返回一笔现金流（CashFlow），失败则返回None

如果有足够的以人民币（CNY）计算的可用资金，则从可用资金中扣除现金流出需求，并返回一笔现金流（CashFlow）

否则，则返回 None

2.5 freeze()：根据给定的委托 ID、标的代码、货币代码和委托金额，冻结可用的人民币（CNY）

@uid_(uuid.UUID)：委托 ID

@symbol_(str)：标的代码

@currency_(str)：货币代码

@amount_(float)：给定金额

@return(None)

2.6 release()：根据给定的委托 ID、标的代码，释放对应的冻结资金

@uid_(uuid.UUID)：委托 ID

@symbol_(str)：标的代码

@return(None)

如果委托 ID、标的代码不存在对应的冻结资金，则不作处理

2.7 release_all()：释放所有冻结资金

@return(None)

2.8 process_partial_fill()：根据给定的部分成交的委托成交信息、给定的现金流，处理资金变动

@fill(Info.FillInfo)：给定的部分成交的委托成交信息

@cash_flow_(CashFlow)：给定的现金流

@return(None)

如果部分成交的是买入委托，且存在委托 ID、标的代码对应的冻结资金，则直接扣减冻结资金

否则，直接增减可用资金

2.9 process_full_fill()：根据给定的完全成交的委托成交信息、给定的现金流，处理资金变动

@fill(Info.FillInfo)：给定的部分成交的委托成交信息

@cash_flow_(CashFlow)：给定的现金流

@return(None)

如果成交的是买入委托，且存在委托 ID、标的代码对应的冻结资金，则先释放冻结资金

BidSignalQueue.py

1 用于投资组合（Portfolio）中用于买入信号资金分配的优先队列的设计

2 BidSignalQueue(PriorityQueue)：投资组合(Portfolio)进行买入信号资金分配时使用的信号（Signal）优先队列

2.1 put(self, s: SignalInfo) -> None:

仅当信号的交易方向为买入时，才将信号放入分配队列

2.2 cancel()：根据给定的信号 ID 撤销对应信号

@uid_(uuid.UUID)：信号 ID

@return(None)

Holding.py

1 用于回测框架内与投资组合（Portfolio）有关的设计

2 PortfolioInfo(Info.Info)：

Portfolio 信息，用于投资组合（Portfolio）体系中的投资组合/基金/集合理财产品的相关信息

2.1 __init__()：初始化

@cash_(float)：现金余额

@amount_(float)：持仓总额

@asset_(float)：总资产

@debt_(float)：总负债

@net_asset_(float)：净资产

@share_(float)：当前份额

@net_price_(float)：当前净值

3 PORTFOLIO_LOGGER 为类 LoggerStringUnit 的实例，回测框架使用的投资组合记录模块，作为全局变量

4 PseudoHoldingUnit(Subject, PriceHandler, FillHandler)：

回测框架中，投资组合（Portfolio）体系中使用的以标的为单位的交易模块

可处理事件：Price、Fill

除了常规的通过参数进行初始化的方式外，提供了通过 FillInfo 进行初始化的简化方式

4.1 __init__()：初始化

@init_fill(Info.FillInfo)：用于初始化的 Fill 信息，默认为 None

如果未提供 Fill 信息，则进行常规的参数初始化

如果提供了 Fill 信息，则通过 Fill 信息初始化

4.2 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

4.3 on_fill()：接收并处理 Fill 事件

@event(Event)：接收的 Fill 事件

@return(None)

更新加权成本价和持仓数量

5 HoldingUnion(PriceHandler, SignalHandler, FillHandler, ClearHandler, ENDHandler)：

回测框架中，投资组合（Portfolio）体系中使用的以投资组合/基金/集合理财产品为单位的交易模块

可处理事件：Price、Signal、Fill、Clear、END

如果有继承 PseudoHoldingUnit 类的自定义单位交易模块，初始化时需要提交单位交易模块的初始化方法

5.1 __init__()：初始化

@factory_(单位交易模块初始化方法)：继承 PseudoHoldingUnit 类的自定义单位交易模块，默认为 PseudoHoldingUnit

在 EVENT_QUEUE 中注册投资组合（Portfolio）体系中的事件处理方法

5.2 time_offset()：模块内的最新时间戳的时间流逝

@offset(str)：时间流逝的单位颗粒

@times(int)：时间流逝的颗粒数量

@return(None)

调用 pandas.tseries.offsets.DateOffset 方法

5.3 refresh()：更新模块的现金余额（cash）、持仓总额（amount）、总资产（asset）、净资产（net_asset）、净

值（net_price）

@return(None)

现金余额为现金管理模块中以人民币（CNY）为单位的现金余额

持仓总额为包含的所有标的单位模块的以人民币（CNY）为单位的现值总和

总资产为现金余额和持仓总额的求和，保留 2 位小数

净资产为总资产减去投资组合的当前负债

净值为净资产除以当前份额，保留 4 位小数

5.4 get_info()：提取当前交易模块的信息（PortfolioInfo）

@return(PortfolioInfo)：提取的交易模块信息

5.5 reset_price()：将模块的净值重置为 1.0000，当前份额设置为当前净资产的值

@return(None)

5.6 set_share()：根据给定的份额数据，设置当前份额

@share_(float)：给定的份额数据

@return(None)

如果给定的份额数据非正数，则不作处理

5.7 subscribe()：使用给定金额的给定货币的现金申购当前投资组合

@amount_(float)：给定金额

@currency_(str)：货币代码，默认为 CNY

@return(None)

以当前净值计算申购所增加的份额，保留 2 位小数

5.8 redeem_amount()：从当前投资组合中赎回给定金额的给定货币

@amount_(float)：给定金额

@currency_(str)：货币代码，默认为 CNY

@return(Optional[CashFlow])：对赎回需求的处理结果，成功则返回一笔现金流 (CashFlow)，失败则返回 None

如果当前可用资金不足用于赎回需求，则赎回失败

如果赎回成功，以当前净值计算赎回所减少的份额，保留 2 位小数

5.9 redeem_share()：从当前投资组合中赎回给定份额，并以给定货币为单位

@share_(float)：给定赎回份额

@currency_(str)：货币代码，默认为 CNY

@return(Optional[CashFlow])：对赎回需求的处理结果，成功则返回一笔现金流 (CashFlow)，失败则返回 None

以当前净值计算赎回所需的资金，如果当前可用资金不足用于赎回需求，则赎回失败

5.10 borrow()：投资组合借入给定金额的给定货币的现金

@amount_(float)：给定金额

@currency_(str)：货币代码，默认为 CNY

@return(None)

5.11 repay()：投资组合偿还负债，以给定金额的给定货币支付

@amount_(float)：给定金额

@currency_(str)：货币代码，默认为 CNY

@return(Optional[CashFlow])：对偿还需求的处理结果，成功则返回一笔现金流（CashFlow），失败则返回 None

如果当前可用资金不足用于偿还需求，则偿还负债失败

5.12 register()：当前的投资组合/基金/集合理财产品模块中，注册给定的单位持仓模块

@holding(PseudoHoldingUnit)：给定的单位持仓模块

@return(None)

5.13 get_holding()：获取给定标的代码对应的单位持仓模块

@symbol_(str)：给定的标的代码

@return(PseudoHoldingUnit)：给定标的代码对应的单位持仓模块

如果标的代码（symbol）未注册，则根据给定的标的代码生成单位持仓模块并注册

否则，获取标的代码（symbol）对应的单位持仓模块

5.14 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

将 Price 事件交给标的代码（symbol）对应的单位持仓模块处理

5.15 cancel()：根据给定的委托 ID、标的代码，撤回对应的交易委托

@uid_(uuid.UUID)：委托 ID

@symbol_(str)：标的代码

@return(None)

根据给定的委托 ID、标的代码，释放对应的冻结资金

如果存在委托 ID（uid）的进行中的委托，且进行中的委托中包含标的代码（symbol），则撤回对应的交易委托

5.16 cancel_symbol()：根据给定的标的代码，撤回对应的交易委托

@symbol_(str)：标的代码

@return(None)

5.17 cancel_all()：撤回当前投资组合的所有交易委托

@return(None)

释放所有冻结资金

5.18 on_fill()：接收并处理 Fill 事件

@event(Event)：接收的 Fill 事件

@return(None)

如果标的代码（symbol）未注册，且 Fill 事件为买入开

仓委托成交，则通过事件中的 FillInfo 生成单位持仓模块并注册

否则，将 Fill 事件交给标的代码（symbol）对应的单位持仓模块处理

获取标的代码（symbol）对应的单位持仓模块

根据给定单位持仓模块和事件中的 FillInfo，计算成交所涉及的现金变动

根据委托成交是部分成交/完全成交，分别处理现金变动

如果委托成交是完全成交，则撤回同一委托 ID（uid）且同一标的代码（symbol）对应的交易委托

5.19 put_bid_order()：根据给定的买入交易委托信息和预计冻结金额，发出交易委托并冻结相应资金

@order(Info.OrderInfo)：给定的买入交易委托信息

@amount_(float)：预计冻结金额

@return(None)

向交易所（Exchange）发出买入交易委托

按照买入委托预计占用的金额冻结资金

5.20 process_bid_signal()：根据给定的单位持仓模块，处理给定的买入信号（Signal）信息

@signal(Info.SignalInfo)：给定的买入信号(Signal)信息

@holding(PseudoHoldingUnit)：给定的单位持仓模块

@return(None)

根据给定的单位持仓模块，计算买入信号（Signal）预计占用的资金

如果投资组合有足够的可用资金用于买入信号（Signal），则发出相应的买入委托

否则，如果买入信号（Signal）分类为 TBF 或 IOC，则根据可用资金计算可申报的买入数量，并申报相应的买入委托

未申报的买入数量计入余量

如果有余量，且买入信号（Signal）分类为 TBF 或 FOW，则将买入信号（Signal）剩余部分放入买入信号优先队列（BidSignalQueue）

5.21 process_bid_signal_queue()：处理买入信号优先队列（BidSignalQueue）中的买入信号

@return(None)

持续申报买入委托，直至买入信号优先队列（BidSignalQueue）为空，或者投资组合可用资金不足以用于当前优先第一的买入信号

获取当前优先第一的买入信号包含的标的代码（symbol）对应的单位持仓模块，并根据可用资金计算可申报的买入数量

如果有可申报买入数量，且买入信号（Signal）分类为

TBF，则申报相应的买入委托

5.22 process_ask_signal()：根据给定的单位持仓模块，处理给定的卖出信号（Signal）信息

@signal(Info.SignalInfo)：给定的卖出信号(Signal)信息

@holding(PseudoHoldingUnit)：给定的单位持仓模块

@return(None)

当单位持仓模块中的数量少于卖出信号（Signal）的需求，如果卖出信号（Signal）分类为 TBF 或 IOC，则按照剩余数量申报卖出委托

否则，忽略卖出信号（Signal）

5.23 on_signal()：接收并处理 Signal 事件

@event(Event)：接收的 Signal 事件

@return(None)

获取标的代码（symbol）对应的单位持仓模块

根据 Signal 事件中包含的交易方向，以及标的代码（symbol)对应的单位持仓模块，处理信号(SignalInfo)

5.24 on_clear()：接收并处理 Clear 事件

@event(Event)：接收的 Clear 事件

@return(None)

5.25 on_end()：接收并处理 END 事件

@event(Event)：接收的 END 事件

```
@return(None)
```

Logger

Logger.py

1 用于回测框架中的记录功能的设计

2 LoggerUnit：回测框架中，用于记录的单位模块的抽象类，记录结果的存储方式为 pandas.DataFrame

2.1 log()：根据给定记录者提交的记录对象，记录一行数据

@obj(object)：提交的记录对象

@committer(str)：给定的记录者

@return(None)

@abstractmethod

强制要求子类实现 log()方法

2.2 to_csv()：以给定的编码方式，将保存的记录结果输出到给定的.csv 文件

@path_(str)：给定.csv 输出文件地址

@encoding_(str)：给定.csv 输出文件编码方式，默认为 GB2312

@return(None)

3 Logger：回测框架中，用于记录的模块，管理多个单位记录模块

3.1 register()：根据给定的记录类型及其对应的单位记录模块，注册模块及其.csv 输出文件路径和文件编码方式

@log_type_(str)：给定的记录类型

@unit(LoggerUnit)：对应的单位记录模块

@path_(str)：给定.csv 输出文件地址

@encoding_(str)：给定.csv 输出文件编码方式，默认为 GB2312

@return(None)

3.2 log()：根据给定记录者提交的记录对象，由给定的记录类型对应的单位记录模块记录一行数据

@obj(object)：提交的记录对象

@log_type_(str)：给定的记录类型

@committer(str)：给定的记录者

@return(None)

如果记录类型对应的单位记录模块未注册，则不做记录

3.3 to_csv()：将已注册的所有单位记录模块，以注册的编码方式，将保存的记录结果输出到注册的.csv 文件

@return(None)

4 LoggerStringUnit：回测框架中，用于记录的单位模块，记录结果的存储方式为字符串

4.1 __init__()：初始化

@head_(str)：输出结果的首行，默认为“info”

4.2 log()：根据给定记录者在给定时间提交的记录对象，记录一行数据

@obj(object)：提交的记录对象

@committer(str)：给定的记录者

@datetime_(pandas.Timestamp)：给定的记录时间

@return(None)

4.3 to_file()：以给定的编码方式，将保存的记录结果输出到给定的文件

@path_(str)：给定输出文件地址

@encoding_(str)：给定输出文件编码方式，默认为GB2312

@return(None)

MovingAverage

MADataHandler.py

1 用于移动平均测试样例中数据处理部分的设计

2 DEFAULT_COLUMN：默认的读取数据文件的列

默认包含["Symbol", "Date", "Time", "Open", "High", "Low", "Close", "Volume", "Turnover"]

3 series_to_bar()：根据给定的一行数据(pandas.Series)，生成一个Bar事件

@row(pandas.Series)：给定的一行数据(pandas.Series)

@return(Event)：生成的Bar事件

4 MADataHandler(DataHandler)：移动均线策略的输入数据处理模块

4.1 load_file()：根据给定的.csv文件路径和文件编码方式，读取行情数据

@file_(str)：给定.csv文件地址

@encoding_(str)：给定.csv文件编码方式，默认为GB2312

@return(None)

MAStrategy.py

1 用于移动平均测试样例中交易策略部分的设计

2 定义默认使用 5 日、20 日移动平均值

3 MAInfo(Info.Info)：MAInfo 信息，用于记录移动均线策略（MovingAverage）的相关信息

3.1 4.1 __init__()：初始化

@crt_price_(float)：当前价格

@short_ma_(float)：当前短周期均线值

@long_ma_(float)：当前长周期均线值

@crt_direction(int)：当前交易方向

4 STRATEGY_LOGGER 为类 LoggerStringUnit 的实例，是移动均线策略使用的记录模块，作为全局变量

5 MAStrategyUnit(PseudoStrategyUnit)：移动均线策略的单位策略模块

5.1 __init__()：初始化

@short_(int)：短周期均线的周期，默认为 CONST["SHORT"]

@long_(int)：长周期均线的周期，默认为 CONST["LONG"]

5.2 update_price()：根据给定的现价，更新策略的数据

@price_(float)：给定的现价

@return(None)

当价格数据达到一个长周期之后，激活策略

5.3 get_info()：提取当前单位策略模块的信息（MAInfo）

@return(MAInfo)：提取的单位策略模块的信息

5.4 on_price()：接收并处理 Price 事件

@event(Event)：接收的 Price 事件

@return(None)

仅当策略已激活时执行

首次触发，成交数量减半执行

仅当短周期均线上穿或下穿长周期均线时发出交易信号

5.5 on_clear()：接收并处理 Clear 事件

@event(Event)：接收的 Clear 事件

@return(None)

test.py

1 用于移动均线策略的回测设计

2 定义回测样例的起始时间：2021-01-01 00:00:00

3 定义投资组合的起始资金：1,000,000.00

4 bar_slicer()：

@bar(Info.BarInfo)：给定的标的以交易日为单位的报价成交数据

@return(Generator)：包含“标的在一个时刻的价格数据”信息的 Price 事件的生成器

```
    # 由于回测样例采用日线级别数据，则仅适用收盘价生成 Price
    事件
5   test()：
    # 初始化数据处理模块
    # 读入输入数据，并将生成的 Bar 事件放入事件优先队列
    # 初始化交易所、投资组合、投资顾问模块
    # 投资组合注入起始资金
    # 初始化回测样例使用的标的的单位模块
    # 投资组合买入标的的起始持仓
    # 运行事件队列
    # 保存结果
```