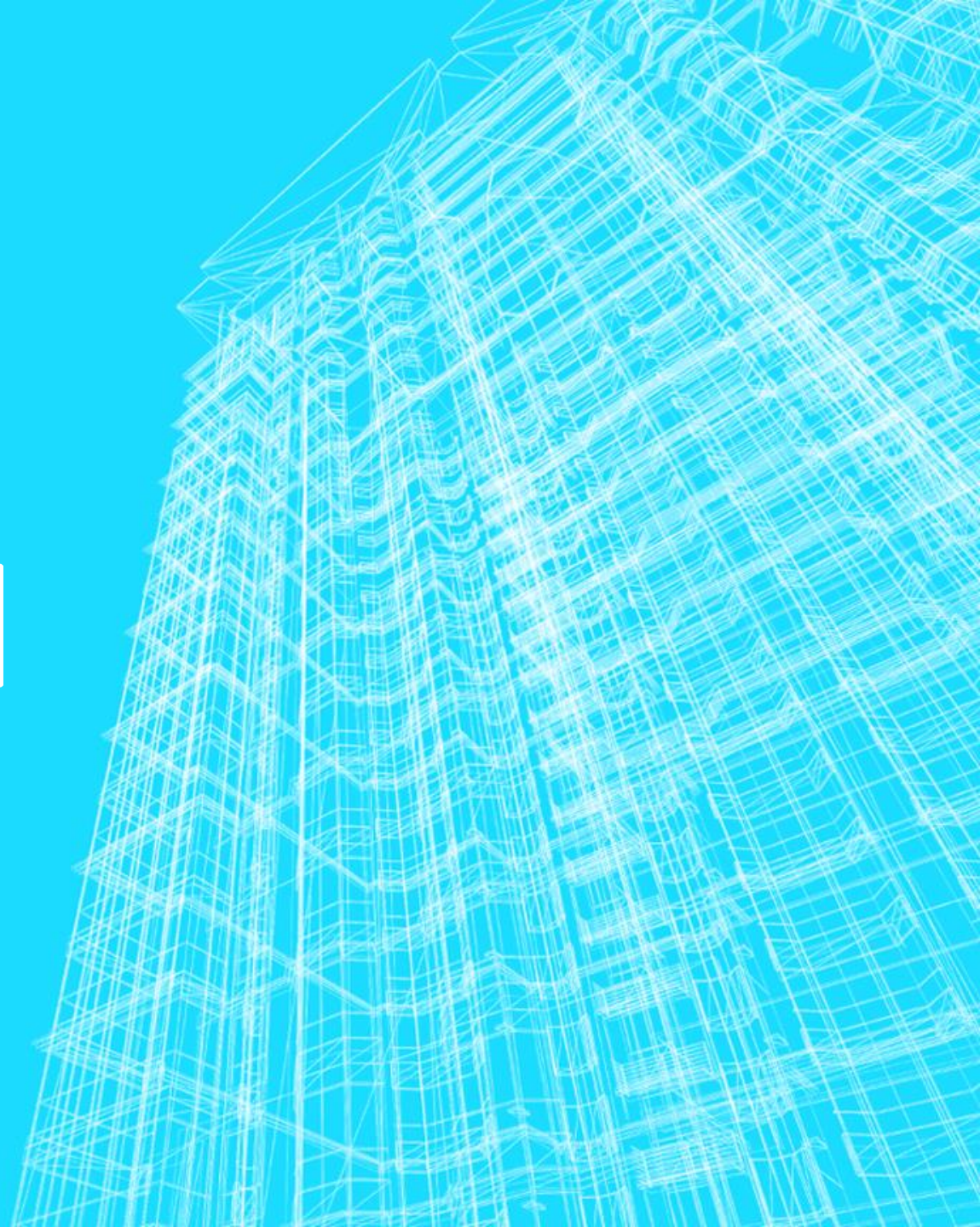
量化交易实战培训

基于开源框架vn.py

Day 4/6

李来佳 QQ/WeChat 28888502

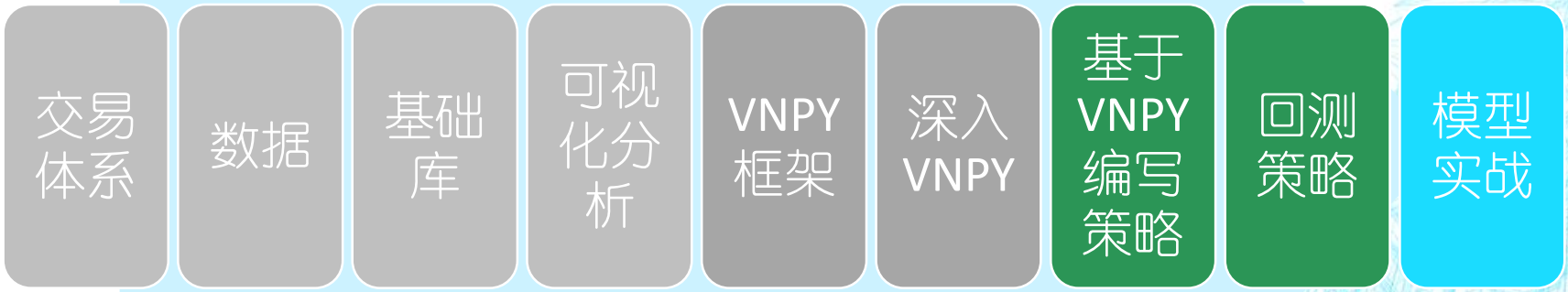


1 内　容：　较为完整地讲解量化交易体系，并通过vn.py实战量化策略。

2 听　众：　具有一定编程基础的量化从业人员。

3 讲　师：　一群爱好交易的程序员。

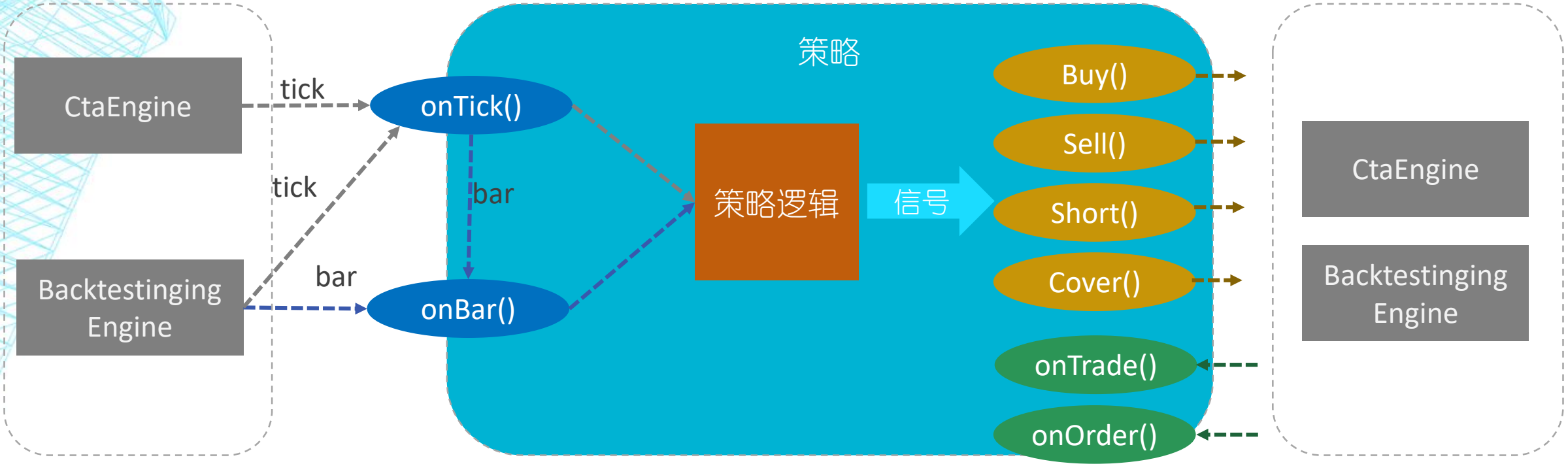
4 感　谢：　vn.py的创始人陈晓优和他的开源社区



大纲

- 7 基于VNPY编写策略
  - 7.1策略运行逻辑　讲解策略在vnpy中的运行逻辑，包括tick级别、分钟级别、混合等
  - 7.2策略模板　讲解策略模板，如何扩展策略模板实现自己的想法。
  - 7.3策略初始化　讲解策略的初始化要注意哪些，有那些手段
  - 7.4运行数据持久化　如何应对各种运行风险，数据如何持久化和重载
  - 7.5策略风控　如何针对单一策略，单一实例进行风控，如何与账号风控共同协作
  - 7.6策略状态监控　讲解状态监控的原理，如何扩展自己的策略状态
  - 7.7日内策略逻辑　针对日内策略，给出实现的逻辑建议
  - 7.8隔日策略逻辑　针对隔日策略，给出实现的逻辑建议，如何避开假期?
- 8 回测策略
  - 8.1回测数据准备　本地、云端和外部供应商的数据选取，tick数据，分钟数据
  - 8.2回测原理　讲解回测原理
  - 8.3回测陷阱　如何理解回测中的陷阱
  - 8.4回测优化　介绍若干中回测优化的手段与实例

7.1 策略运行逻辑

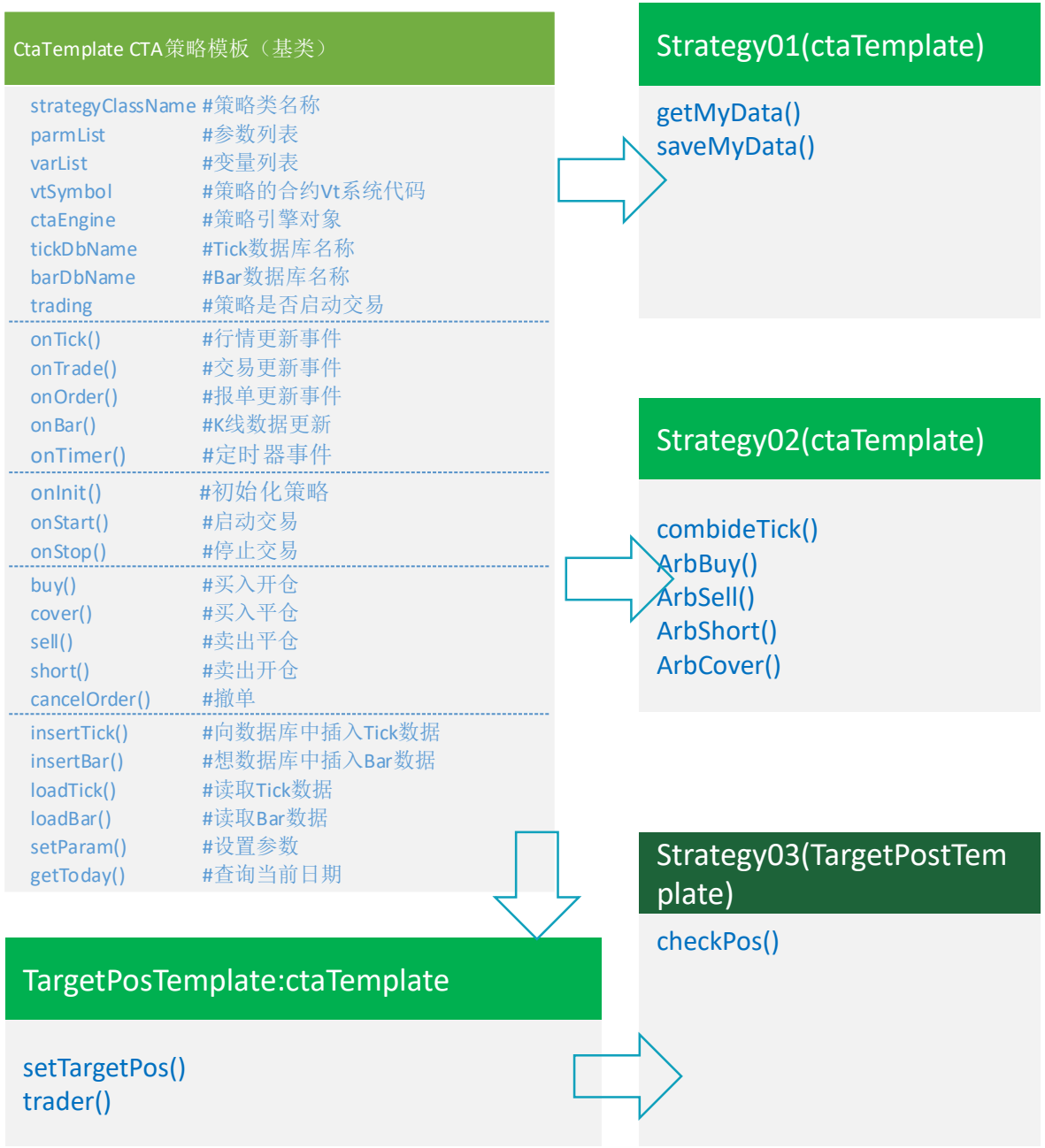


- 策略输入Tick/Bar，输出信号
- 策略接受交易反馈

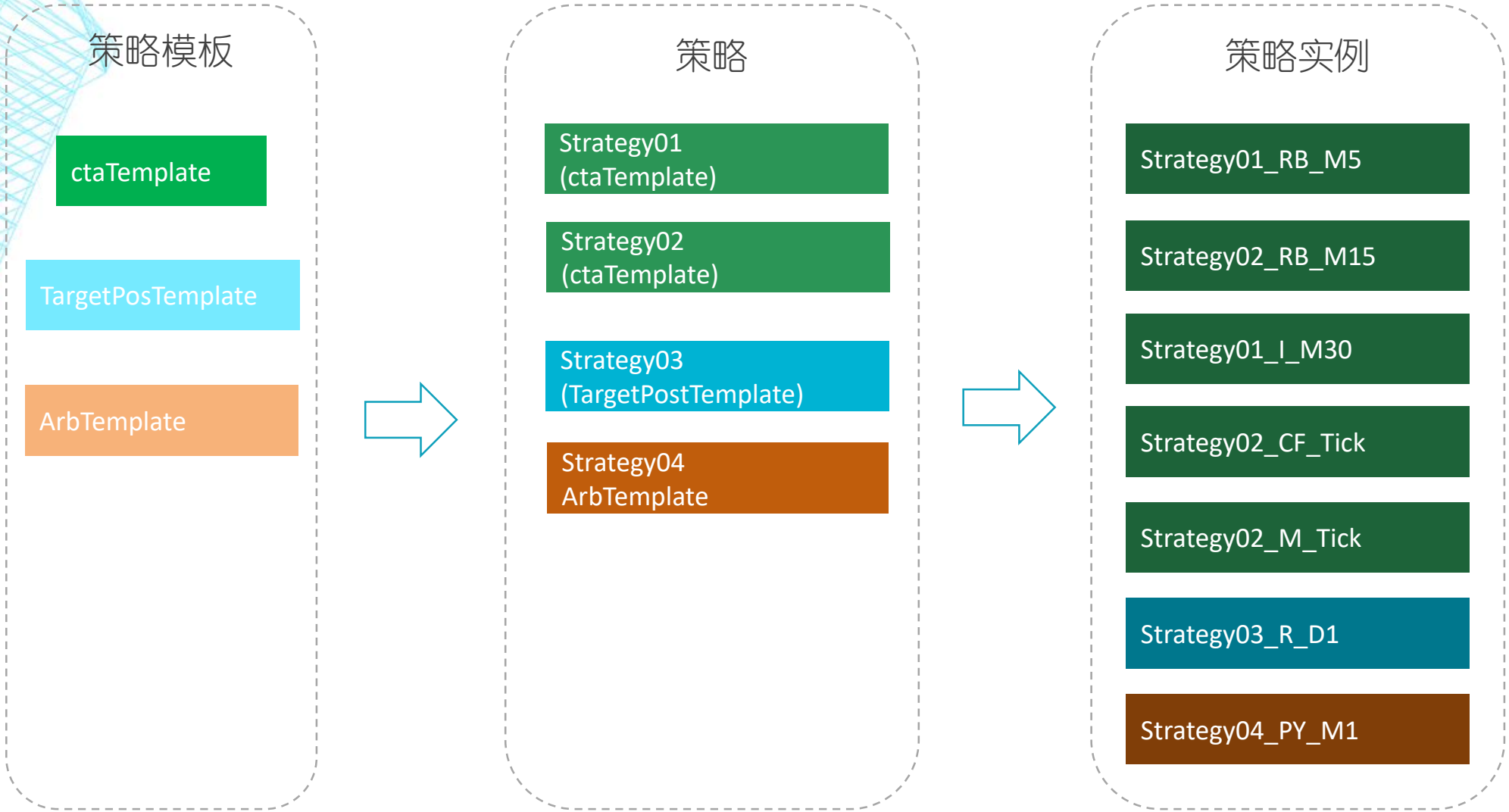
- Tick驱动（Bar内运算）：每一个tick都执行策略逻辑判断，计算/产生信号。
- Bar驱动 （Bar外运算）：每产生新Bar时，执行策略逻辑判断，计算/产生信号。
- 混合驱动（同时运算）

7.2 策略模板

- 模板=基类，包括：
  - 参数
  - 变量
  - 外部容器调用函数
  - 公用的函数
- 具体策略继承模板，在其基础上扩展自己的逻辑
- 在模板的基础上再扩展出某一类特定需求的模板，如套利模板、目标持仓模板等



7.2 模板\策略\实例关系



Vn.trader/catAlgo下放置策略模板

Vn.trader/catAlgo/strategy/下放置策略

Vn.trader/catAlgo/cta_setting.json配置策略实例

7.3 策略加载/初始化

- V1.4之前的清单加载：
  - vn.trader/catAlgo/ctaSetting.py，添加:
    - from strategy22_demo import Strategy22
    - STRATEGY_CLASS['Strategy22'] = Strategy22
- V1.4之后的清单加载:
  - 需要加载的策略，统一放置在vn.trader/catAlgo/strategy目录下
  - __init__.py自动加载
- 策略（实例）加载/启动
  - 人工操作：vn.trader->算法->CTA策略->加载策略，启动策略
  - 无人值守：
    - mainEngine.ctaEngine.loadSetting() # 加载cta的配置
    - mainEngine.ctaEngine.initStrategy(u'S22_DEMO') # 初始化策略实例，如果多个，则需要逐一初始化多个

7.3 策略加载与初始化

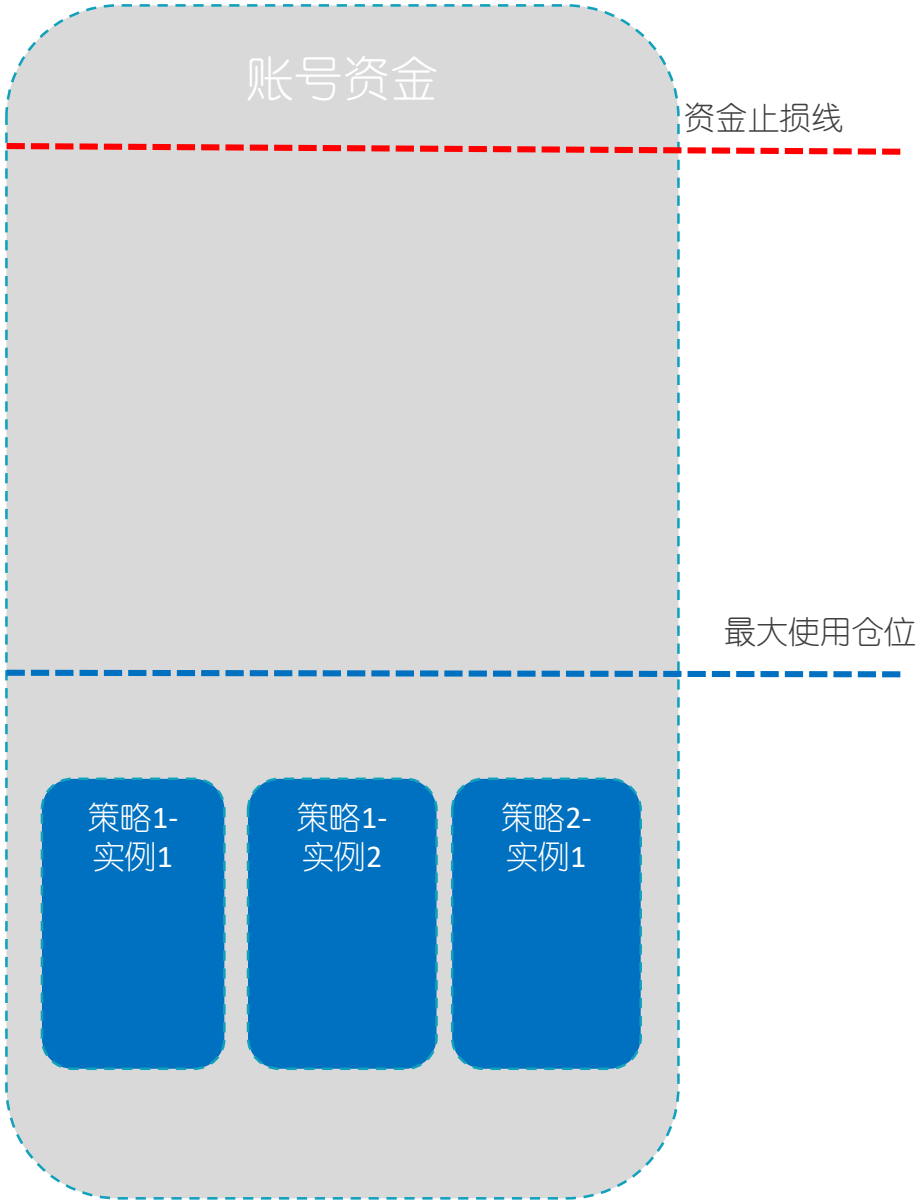
- 初始化顺序：
  - CtaTemplate.__init__()
  - Strategy.__init__()
  - Strategy.Init()
- 初始化的内容：
  - 初始化各类参数
  - 加载前置数据，推送到OnTick或者OnBar
  - 加载上一状态的持久化数据（从本地JSON文件加载或 catEngine.loadPosition()
- 一些问题
  - 初始化失败，如何处理 （扩展接口/按钮，强制重新初始化）
  - 初始化时间较长，如何处理

7.4 运行数据持久化

- 策略运行的风险
  - 程序异常（代码报错/gateway异常/Python环境崩溃）
  - 成交不一致
  - 成仓位不一致
  - 人为干预
- 数据持久化
  - 保存至本地Json文件（策略实例名.json)
  - 保存内容：持仓数量、开仓价格、止损价格等必要数据
  - 保存至MongoDB。ctaEngine. savePosition()
- 其他数据
  - Bar数据（传统的k线/自定义的Renko Bar/RangeBar等）
  - 特殊的策略配置（policy，网格）

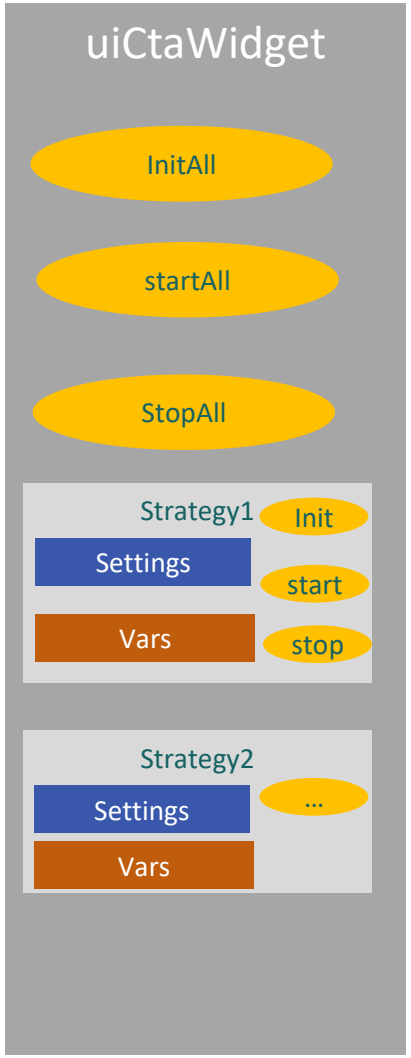
7.5 策略风控

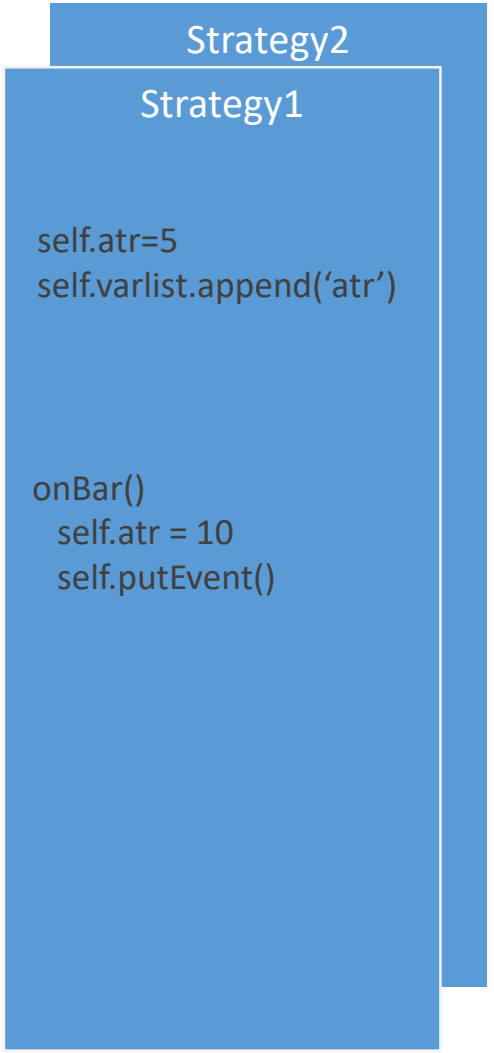
- 单一策略：
  - 仓位控制：
    - 增加最大仓位数量
    - 开仓前检查剩余可开仓数
    - 动态根据配置参数/ATR/VAR等更新最大开仓数
  - 止损
    - 固定止损
    - 跟随止损
- 资金风控
  - 资金最大使用仓位
  - 资金止损线



7.6 策略状态监控

- 状态包括
  - 仓位：合约/今仓/昨仓
  - 动态参数：最大仓位/ATR
  - 连续亏损次数
- 添加状态
  - varList.append(变量名)
  - 变量须是策略内部的变量名
- 更新变量的频率
  - onBar
  - onOrder
  - onTimer
- Self.putEvent()





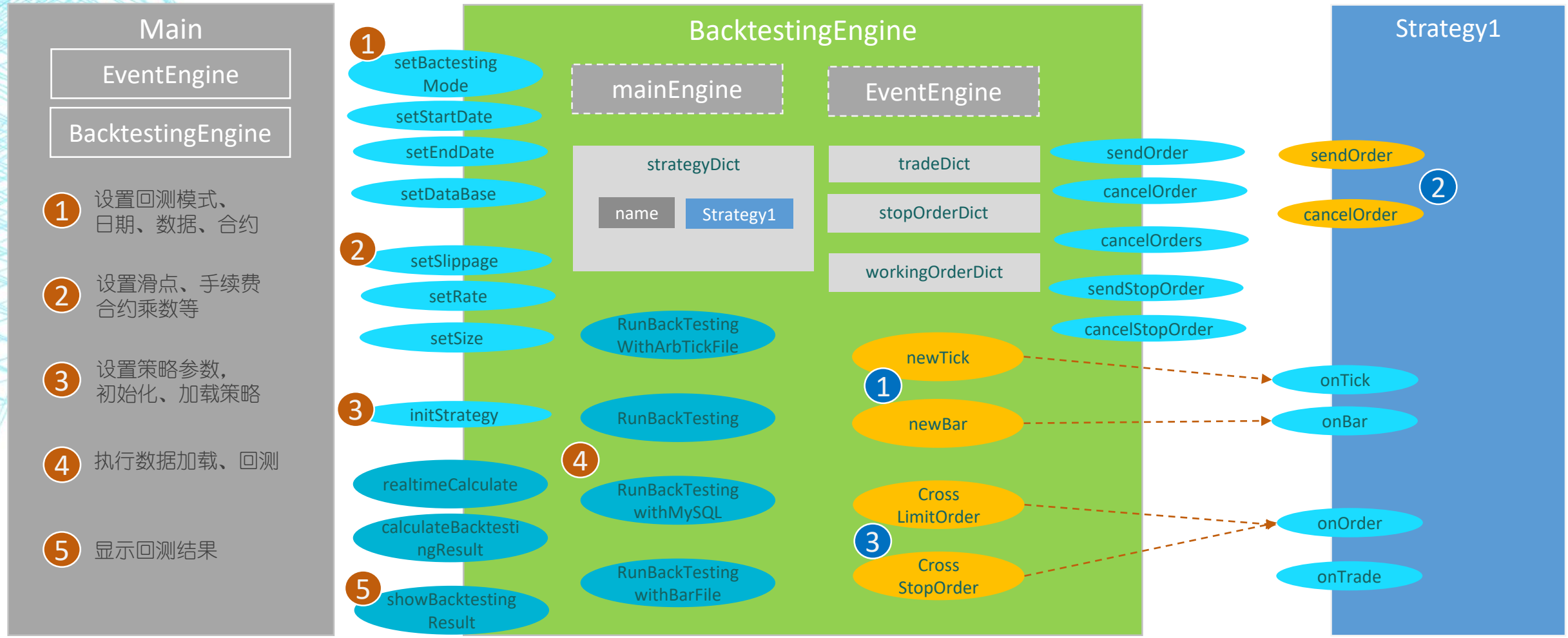
7.7 日内策略逻辑

- 经典的日内策略
  - R-Breaker
  - Dual-Thrust
- 策略初始化
  - 获取日线级别数据，计算各个阈值
  - 重新初始化当日tick数据（任意时点初始化）
- 开盘时刻
  - 波动较大，可自定义openWindow来进行特殊处理
- 收盘时刻
  - 自定义closeWindow，撤销开仓单，强制平仓

7.8 隔日策略逻辑

- 一般是15分钟周期以上，持仓超过1天
- 期货合约换月风险
  - 配置中，预先设置策略合约的开启、结束日期
  - 到达日期后，进行合约更换，平近期合约，开远期合约
  - 更新止损位、开仓价等
- 期货假期保证金/波动风险
  - 预先配置假期、仓位降低比例
  - 到达日期时，执行仓位降低
  - 假期结束后，调整仓位比例
- 股票复权风险
- 可参考ricequant的策略模板：
  - before_Trading()
  - after_Trading()
  - scheduler.run_daily()

8. 回测策略



模拟过程 1 推送tick/bar 2 策略逻辑运算，产生交易信号 3 下一Tick/bar执行信号，更新交易记录

8. 1 回测数据准备

- Tick数据回测
  - 保证是日期和时间连续
  - 单一合约，主力合约？
  - 文件tick：先循环日期，寻找对应合约文件后加载；先日盘，再到夜盘
  - 数据库：分批提取，分批灌送
  - 标准套利合约tick：先生成套利tick，再推送
- 分钟以上级别数据回测
  - 通过供应商接口获取并导出csv
  - 回测程序导入csv，分批灌送
  - 注意分钟数据的datatime:是bar开始时间/结束时间?

https://github.com/msincenselee/vnpy/blob/master/vn.trader/ctaAlgo/ctaBacktesting.py

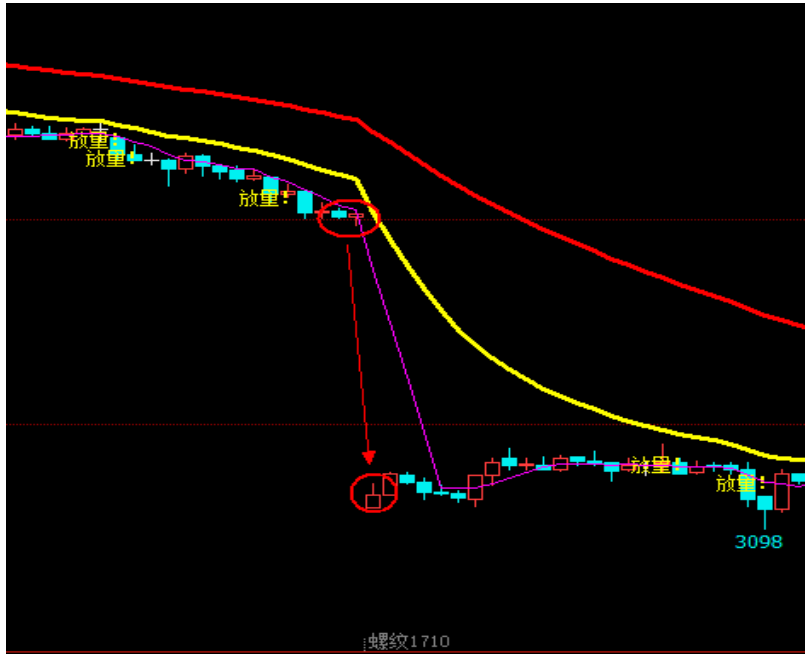
8. 2 回测原理

- 撮合方式
  - 下一根Bar/下一个Tick
  - crossLimitOrder() 撮合限价单
  - crossStopOrder() 撮合停止单
  - onBar()/onTick() 推送到策略

撮合成交价格	Bar	Tick
buyCrossPrice	bar.low	tick.askPrice1
sellCrossPrice	bar.high	tick.bidPrice1
buyBestCrossPrice	bar.open	tick.askPrice1
sellBestCrossPrice	bar.open	tick.bidPrice1

8.3 回测陷阱

- 采用主力合约回测时，跨月产生的利润/亏损，是不真实的。
- 若资金规模大，要考虑下单时的成交量是否满足。



8.4 回测优化

- 策略中，使用资金比例来实时计算仓位，
  - 回测引擎需支持实时计算realtimeCalculate()
  - getAccountInfo()
- 策略中，需要区别回测和实盘的微妙处理
  - 使用self.backtesting
- 交易分析
  - 导出交易记录exportTradeResult()
  - K线回放交易记录

```
def __coverLogic(self, price, volume):
    """平空单逻辑"""

    # 如果只有一手或回测，直接平
    if volume == 1 or self.backtesting:
        self.cover(price, volume, orderTime=self.curDateTime)
        return
```

```
if self.backtesting:
    openPrice = bar.close - self.minDiff
else:
    openPrice = self.curTick.bidPrice1
```

课后作业

- 使用第一堂课的作业，存入MongoDB的数据，对demo策略进行回测。
- 跟踪查看策略的OnTick，OnOrder，OnTrade
- 把策略改为5分钟级别，直接使用Bar数据进行回测

THANKS

微信：28888502