为什么需要回测

搭建回测框架是为了测试、评估和优化金融交易策略。它可以模拟市场情景并执行交易，从而对交易策略的性能进行分析和评估，判断其在真实市场中所能实现的风险和回报。在实际交易之前，回测框架可以帮助交易者识别和解决潜在的问题，避免损失。

以下是搭建回测框架的主要好处：

1. 无需实际进行交易：回测框架可以在模拟市场中执行交易和策略测试，无需实际进行交易。这样可以将风险降至最小，并避免在实际交易中犯错误。

2. 分析策略性能：回测框架可以在历史数据上运行交易策略，并输出有关策略表现的统计信息。这些信息可以帮助交易者评估策略的性能，并识别市场偏差和潜在问题。

3. 优化策略：回测框架可以帮助交易者调整和优化交易策略。通过逐步修改和测试策略参数，交易者可以为其策略寻找最佳组合，从而在实际交易中实现更好的表现。

搭建回测框架是金融交易者的一个关键工具，可以帮助他们测试、跟踪和优化其交易策略，从而提高其交易表现并减少风险。

使用开源量化平台还是自己搭回测框架?

使用开源量化平台的好处是可以快速开始，并且有很多文档和社区支持，可以直接使用先进的交易算法和库。开源量化平台通常具有一个可视化界面和各种财务指标，可以更轻松地分析和了解投资组合和策略。

然而，如果需要更高的灵活性和控制权，或者想要访问更低级别的模块和数据源的话，自己搭建回测框架会更好。

自己搭建回测框架，虽然可能需要更多的学习和开发时间，但是可以更好地掌握细节，以至于能够更深入地理解所使用的算法和模型，以及更精细的控制数据源和模型参数的细节。此外，自己搭建回测框架也可以更好地适应交易系统变化和挑战。

方案	优点	缺点
开源量化平台	快速开始、支持度高、数据易获取	不够灵活、平台系统性风险
自己搭	可扩展性强，适合有想法的人	上手有一定的难度，数据源要自己想办法

主流回测框架

量化回测框架	Github Star	维护情况
Backtrader	10k	☆☆☆☆
Zipline	16k	☆☆
vnpy	21k	☆☆☆☆☆

backtrader

文档：https://www.backtrader.com/docu/

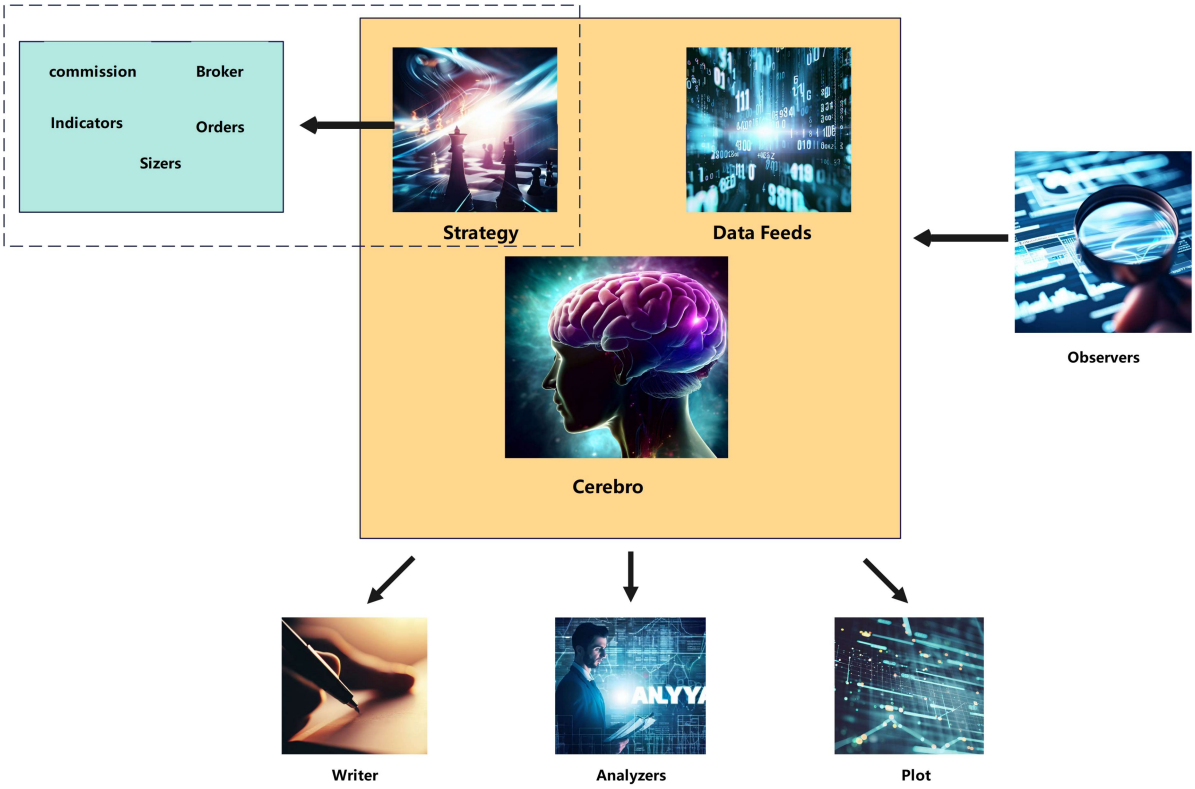
Github : https://github.com/mementum/backtrader

安装：pip install backtrader

1.1基本概念

- Cerebro: 回测框架调度中心，包括加载数据、运行策略、执行观察、结果分析及记录等。
- DataFeeds：回测数据加载。
- Strategy： 即交易策略，在这里面实现自己的交易策略。
    - Broker : 经纪人，用于管理资金账户和交易佣金等。Commission：管理佣金、保证金等信息。
    - Indicators : 指标是构建策略的基础，如移动平均线，RSI等，相当于交易信号。
    - Orders : 订单管理。
    - Sizer： 仓位管理。
- Cerebro.run()： 运行策略
- Analyzers： 用于分析回测结果，包括夏普率、最大回撤、年化收益率等。
- Plot： 绘图，实现回测结果可视化。
- Writer： 用于将策略、分析结果、指标等信息保存下来。

1.2整体框架图



1.3回测数据的加载和引用

- 数据格式要求：

假设已经有一个名为df的数据框，数据列名要求为：

```
datetime, open, high, low, close, volume, openinterest
```

- 数据加载：

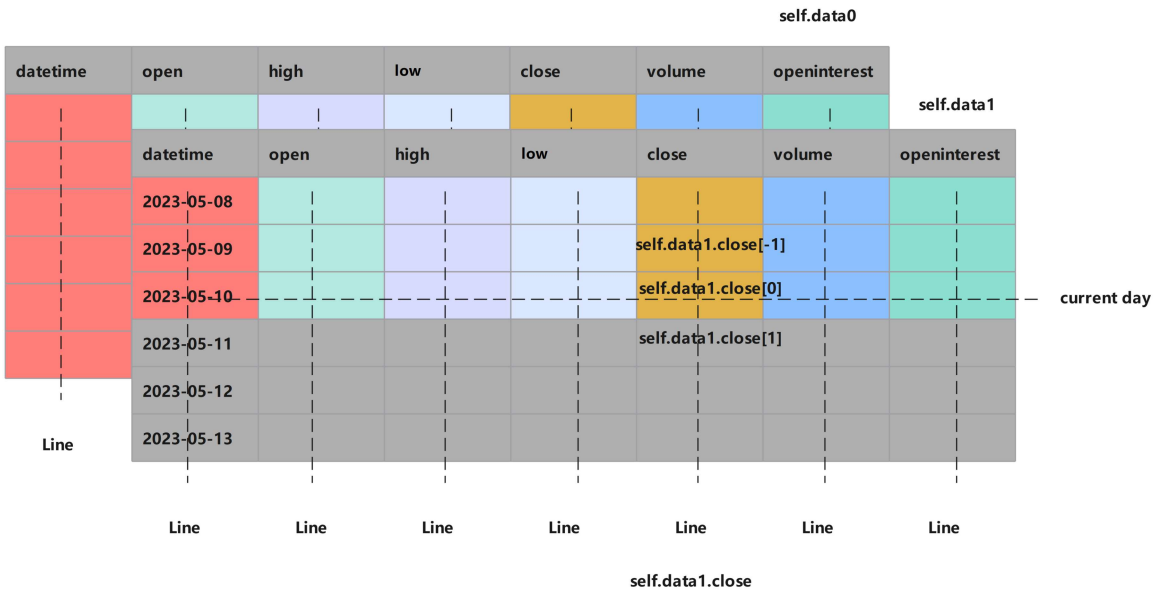
```
import backtrader as bt
cerebro = bt.Cerebro()
data = bt.feeds.PandasData(dataname=df,timeframe=bt.TimeFrame.Days) #加载数据，频率为日频
cerebro.adddata(data, name='stockname')
```

- 数据引用：在策略中引用数据。

  在backtrader中，数据是三维的，首先需要选择数据表，在表中，表的每1列为一个line。

```
class TestStrategy(bt.Strategy):
    def __init__(self):
        pass
    def next(self):
        dt = self.data0.close[0] #数据源0当前的收盘价
```

注意：line中的序号与python 列表序号的区别：line中0是当前时刻，-1是前一时刻，依次类推。



1.4 策略编写

```
class TestStrategy(bt.Strategy):
    # 设置回测参数
    #params = (('period5',5),('period20',20), ) #元组形式
    params = dict(period5=5,period20=20)
    def __init__(self):
        """
        必选，可以进行一些初始化工作
        """
        pass
    def next(self):
        """
        必选，编写交易策略
        每个交易时间调用一次
        """

        pass
    def log(self):
        """
        可选，打印日志信息
        """

    def notify_order(self, order):
        """
        可选，传入有变更的订单，可打印订单信息
```

```
        """
        pass

    def notify_trade(self, trade):
        """
        可选，传入成交信息，可打印成交信息
        """
        pass
```

1.5 quick start

- 策略流程