专业·创新·增值

金程教育
GOLDEN FUTURE

20180318AQF模拟题讲解

量化投资项目组

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

专业来自101%的投入!

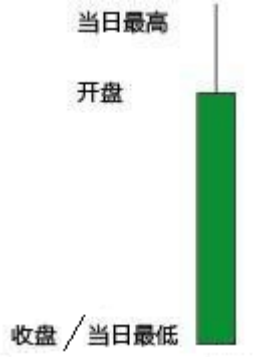
- **1.1 技术分析是重要的投资分析方法之一。其中，起源于日本德川幕府时代的“K线图”（又称蜡烛图、阴阳线）是常用的技术分析方法。当我们发现某交易日的K线为无下影线阴线时，那么该K线实体的上边线表示（ ）？**

A. 最高价

B. 收盘价

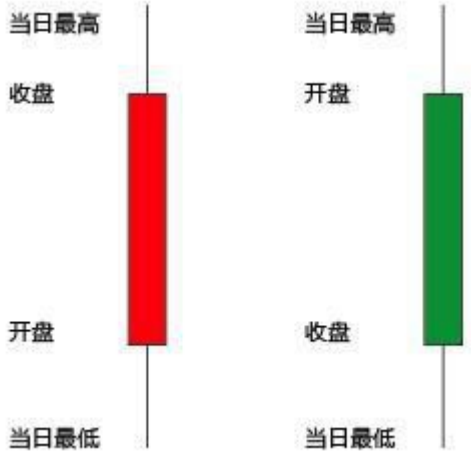
C. 最低价

D. 开盘价



- **参考答案：D**
- **解析：**
  - K线收盘价低于开盘价称为阴线。当K线无下影线时，表示收盘价即为最低价，此题的K线下边线既是收盘价也是最低价，而上边线则表示开盘价。

- **知识点： K线图**
  - K线图的画法包含四个数据信息，即开盘价、最高价、最低价、收盘价，所有的k线都是围绕这四个数据信息展开；
  - 反映特定时期内的状况和价格信息 ，具体将得到日K线图、周K线图、月K线图、季度K线图，以及年度K线图等 。



- **1.2 以下关于各大量化投资交易策略的描述中，不正确的是（ ）？**

A. 在多因子策略中，一般而言，所选因子的相关性越低越好

B. 一般而言，资金流对个股的短期波动影响更大

C. 动量反转策略利用的是价格均值回归的特性

D. 趋势跟踪策略本质上是一种追涨杀跌的策略，因此并不具有任何盈利的可能性

- **参考答案：D**
- **解析：**
  - 趋势跟踪作为一种经典的量化投资策略，实证检验中在不同的市场、不同的交易时期其策略有效性都得到了一定的验证。因此，‘不具备任何盈利可能性’的描述存在问题。

- **知识点：趋势跟踪策略、多因子策略、动量反转策略**
  - 趋势跟踪策略
    - ✓ 趋势跟踪策略的理念为：顺应趋势，待时而动，其具体做法为，追求长线趋势的投资，涨势买进跌势卖出。
  - 多因子策略
    - ✓ 多因子策略是通过回归法找到某些和收益率最相关的指标，并根据该指标建立一个投资组合，所选因子的相关性越低，则分散性也会越好。
  - 动量反转策略
    - ✓ 动量反转策略是利用动量及反转效应，即一段时间内的股价偏离都会有回复均值的需要，来赚取收益。

- **1.3 李明，AQF，某量化基金经理，他在量化投资交易的过程中，发现回测收益往往会大幅高于实盘收益，研究发现造成这种现象的原因有很多，其中，常见的一种原因是在回测的过程中使用了未来数据，未来数据会使得回测收益虚高。那么在下列量化策略研究过程中，哪个选项最有可能没有使用到未来数据（ ）？**

A. 在策略回测的过程中，采用某天的最低价作为当天的买入成本价

B. 使用整个样本数据对策略参数进行寻优后，使用该优化后的参数进行策略回测，并对该策略进行有效性评估

C. 以发出交易信号后的下一天的开盘价作为策略的成交价，未设置滑点

D. 以当前沪深300成分股为研究对象，研究过去10年沪深300成分股的选股策略

- **参考答案：C**
- **解析：**
  - 误用未来数据是回测常见问题之一。
  - 选项A中，某日最低价在当日进行交易时是无法确定的，因此在当日盘中，最低价属于未来数据。
  - 选项B中，在策略回测时，由于使用了整个样本数据对参数进行了优化，因此进行策略有效性的评估时所使用的参数包含了所在时间点以后的信息，所以使用到了未来数据。
  - 选项D中，在对过去10年进行研究过程中，当前沪深300组成成分的信息属于未来数据。

- **知识点：未来数据/未来函数**
  - 未来函数：A和B存在依赖关系，即B=f(A), 使用时序上靠后的B影响时序上靠前的A会产生未来函数
  - 常见的未来函数出现的情况：
    - ✓ 策略测试过程中，使用一天的最低价作为当天交易的成交价。真实交易中只有一天结束后才能确定当天最低点，因此最低价产生的时间晚于当天每一个成交数据
    - ✓ 使用优化后的参数对产生优化的数据样本做测试。优化参数的时间晚于训练数据对应的时间段，因此使用优化后的参数决定训练时间段内的交易属于使用未来数据
    - ✓ 使用当前沪深300成分股，测试过去针对沪深300的选股策略。当前的指数成分，在时序上晚于过去交易的产生，使用当前的成分股数据决定过去时间点的交易使用了未来数据。

- **知识拓展：**
  - 其他可能导致模型回测表现优于实盘表现的原因：
    - ✓ 针对训练数据的过度拟合
    - ✓ 未考虑滑点、冲击成本、交易成本等
    - ✓ 幸存者偏差（沪深300成分股也可以用幸存者偏差解释）
    - ✓ 测试数据过少或产生交易信号过少，两者都使回测结果缺少统计意义

- **1.4 李明，AQF，某量化基金经理，正在研究事件驱动型套利策略，以下哪个选项描述了事件驱动型套利策略（ ）？**

A. 基金经理持有目前或者预期会发生诸如以下交易事项的公司的金融产品：（包括但不限于）合并、重组、财务危机、股权收购、股东回购、发行债务交换、证券发行或其他资本结构调整

B. 基金经理根据潜在的宏观经济变量及其对股票、固定资产、货币和大宗商品市场的影响，进行相关交易

C. 基金经理基于多个证券估值差异及其关系的理论进行交易

D. 基金经理在现货市场和衍生品市场进行方向相反的操作

- **参考答案：A**

➢ **解析：**

- 事件驱动型套利策略是指基金经理持有目前或者预期会发生诸如以下交易事项的公司的金融产品：（包括但不限于）合并、重组、财务危机、股权收购、股东回购、发行债务交换、证券发行或其他资本结构调整。
- 选项B中，宏观套利策略是指基金经理根据潜在的宏观经济变量及其对股票、固定资产、货币和大宗商品市场的影响，进行相关交易。
- 选项C中，相对价值套利策略是指基金经理基于多个证券估值差异及其关系的理论进行交易。
- 选项D中，股票对冲策略是指基金经理在现货市场和衍生品市场进行方向相反的操作。

➢ **知识点：事件驱动型套利策略**

- 事件驱动型套利策略(Events Driven) ：指针对上市公司目前或者将来发生的重大事件（如：并购和重组）， 预期其正面或者负面的深远影响，利用发现的价格差别，来进行套利的交易策略。

- **1.5 字符串格式化是量化投资策略编写过程中常用的方法，那么以下哪种代码可以用来实现浮点数格式化（ ）？**

  A. %c
  B. %d
  C. %f
  D. %s

- **参考答案：C**
- **解析：**
  - 选项A中，%c表示格式化字符及ASCII码，如：print('%c' % 'd')，输出 d，print('%c' % 65)，输出A，A对应的ASCII码为65;
  - 选项B中，%d表示格式化整数;
  - 选项C中，%f表示格式化浮点数，可以指定小数位数，如：print('%.2f' % 1.23879) 输出 1.24;
  - 选项D中，%s表示格式化字符串。

➢ **知识点：字符串格式化占位符**

- 常用格式化占位符

占位符	功能
%c	格式化字符及其ASCII码
%s	格式化字符串
%d	格式化整数
%f	格式化浮点数字，可指定小数点后的精度
%e	用科学计数法格式化浮点数

- ASCII码

ASCII（American Standard Code for Information Interchange）是早期计算机使用的一套编码系统，使用一个字节表示一个字符。一个字节能最多只能存储2^8种不同字符，所以随着计算机的推广, ASCII编码逐渐被取代，当前比较常用的是Unicode编码，使用两个字节表示一个字符，可表示的字符数量扩充到2^16个。

ASCII码对照表：

http://tool.oschina.net/commons?type=4

- **知识拓展：**
  - 字符串格式化的常用方式
    - ✓ 使用%格式化字符串，e.g. 'Output: %s, %f' % ('apple', 1.65)
    - ✓ 使用.format方法格式化字符串， e.g. 'Output: {},{}'.format ('apple', 1.65)
  - 字符串常用方法

方法	功能
str.capitalize()	把字符串的第一个字符大写
str.count(sub[, start[, end]])	返回 sub在 str里面出现的次数, start和end 指定统计的范围
str.encode(encoding='UTF-8')	以 encoding 指定的编码格式编码 str
str.find(sub[, start[, end]])	检测 sub 是否包含在 str 中，beg 和end 指定查找的范围，返回值为sub在str中起始位置的索引，没有找到的情况返回-1

- **1.6 李明，AQF，某量化基金经理，在1.2308做空欧元/美元的差价合约，之后欧元/美元汇率跌至1.2133，李明可以通过以下哪个类型的委托单来实现继续持有空头仓位的同时控制回撤风险（ ）？**

  A. 市价买单
  B. 市价卖单
  C. 限价买单
  D. 止损买单

- **参考答案：D**
- **解析：**
  - 题目要求继续持仓，所以选项A不正确，选项B会导致空头头寸增加，不能起到对冲的作用，所以选项B不正确，选项C中限价买单是在低于市场价的位置买入的指令，不能起到控制回撤的作用。选项D中止价买单是在市场价以上设置买单，当价格突破设置的价位时买入，所以如果价格上升时可以实现平仓，保护获利；如果价格并未升破止价买单的价位，则可以继续持仓。所以选项D正确。

- **知识点：常用订单类型**
  - 常用订单类型及应用场景

订单类型	功能	应用场景
市价单	以当前市场价格成交的订单，不能指定成交价格	希望尽快成交时使用
限价单	只能在特定价格或更好的价格上成交的订单	希望获得更好的成交价格，对成交价格要求严格
止损单	达到或超过特定的止损价格后，发出市价	预期行情在突破关键点位后会出现爆发，或需要将突破作为下单条件的情况
跟踪止损单	止损单价格随市场单向浮动，行情向有利方向发展时相应调整止损单价位，行情向不利方向发展不调整，直到止损单被触及，发出与头寸方向相反的市价单	行情向有利方向发展后，希望保护部分浮盈，同时能够继续跟随有利行情。常用于预期中的行情尾段，以实现获利保护

- **1.7 当横线处填入（ ）时，代码打印输出的结果是列表中所有的深交所上市的股票代码？（注意：上交所代码以6开头，深交所代码以0或3开头）**

```
for stock_code in [ '002003', '600015', '300001', '002300']:
    if stock_code.startswith('6'):
        _____
    print(stock_code)
```

A. raise
B. continue
C. pass
D. break

- **参考答案：B**

- **解析：**
  - 选项A中，raise触发异常关键字，常用在try...except...或条件判断结构中，用于抛出异常。
  - 选项B中，continue关键字，用在循环结构中，在continue出现的位置中断当前循环，直接进入下一次循环，当continue触发时，continue以下的循环体代码不会继续执行。
  - 选项C中，pass关键字，通常用于占位，保持程序结构的完整性，不执行任何操作。
  - 选项D中，break关键字，用于循环结构中，作用是终止并跳出循环，这样就只能输出'002003'这一个股票代码。

知识点：循环控制语句

- 常用循环控制关键字

关键字	功能
break	终止当前循环
continue	中断当前循环，直接进入下一次循环

- 其他常用关键字

关键字	功能
pass	保持语法结构完整，不做任何操作，常用在函数和方法中作为占位语句
raise	用于触发异常，出现异常后，后续代码不会继续执行

- **知识拓展：**
  - 常用控制结构
    - ✓ 条件语句
    - ✓ 循环语句
      - ◆ for 循环
      - ◆ while 循环
    - ✓ 异常处理语句
  - 股票代码编码规则：
    - ✓ 上海证券交易所上市股票以6开头
    - ✓ 深圳证券交易所上市股票
      - ◆ 主板以 000 开头
      - ◆ 中小板以 002 开头
      - ◆ 创业板以 300 开头

- **1.8 李明，AQF，某量化基金经理，他在进行策略研究时需要从DataFrame数据类型stock_base_data中提取2018-01-03至2018-01-05（含01-05）时间段中的股票PE、CLOSE数据，该DataFrame如下，则李明提取数据时可以使用的代码为（ ）？**

A. stock_base_data.iloc['2018-01-03':'2018-01-05', ['PE', 'CLOSE']]

B. stock_base_data.loc['2018-01-03':'2018-01-05', ['PE', 'CLOSE']]

C. stock_base_data.loc[['PE', 'CLOSE'], '2018-01-03':'2018-01-05']

D. stock_base_data.iloc[2:4, ['PE', 'CLOSE']]

	PB	PE	ROE	CLOSE
2018-01-01	1.5	10	0.05	35
2018-01-02	1.5	10	0.05	36
2018-01-03	1.5	10	0.05	34
2018-01-04	1.4	10	-0.10	32
2018-01-05	1.4	10	-0.10	33
2018-01-06	1.4	12	-0.10	30
2018-01-07	1.4	12	-0.10	29

- **参考答案：B**

➢ **解析：**

- 选项A中，使用iloc应为对位置进行索引；
- 选项C中，loc进行索引时，应先写明行索引，再写明列索引；
- 选项D中，iloc不能混用位置索引和标签索引。

- **知识点：**
  - pandas.loc/iloc[] 数据切片
    - ✓ loc标签切片
      - ◆ 选择单一行：df.loc['000004']
      - ◆ 选择不连续的列：df.loc[:,['PE', 'LCAP']]
      - ◆ 同时对行列进行选择：df.loc['000001':'000004', ['PE', 'LCAP']]
    - ✓ iloc索引切片
      - ◆ 选择单一行：df.iloc[3]
      - ◆ 选择不连续的列：df.iloc[:,[2,6]]
      - ◆ 同时对行列进行选择：df.iloc[1:4, 2:6]

- **知识拓展：**
  - 对筛选结果赋值
    - ✓ df.loc[df['PE']<0, 'PE'] = 1000　　# 将所有PE小于0的PE数据赋值为1000
    - ✓ df[df<0] = -df　　# 将所有小于0的数据都转化为相反数
    - ✓ df.loc[df['PE']<0, 'new_column'] = 0　　# 根据已有数据做判别，在原数据表中加入一列新数据 ‘new_column’

➢ **1.9. 李明，AQF，某量化基金经理，想要评估长期持有的某只股票在过去三天的总体表现。已知该股票在过去三个交易日的日收益率为0.02, 0.05, 0.06，李明编写了如下代码：**

```
import pandas as pd

equity_return = pd.Series([0.02, 0.05, 0.06])
```

请问以下哪行代码可正确计算该股票在过去三个交易日的累计收益率（ ）？

A. equity_return.sum()
B. equity_return.mean()
C. (equity_return+1).cumprod()[2]-1
D. equity_return.ptp()

- **参考答案：C**
- **解析：**
  - 累计收益率 = (1 + 0.02)(1 + 0.05)(1 + 0.06) - 1
  - 选项A计算序列的总和，选项B计算序列的均值，选项D计算序列的极差。
  - 所以选项C正确。
- **知识点：累计收益率**
  - 两个率的概念：

    每日收益率 =（当日收盘价 - 前日收盘价）/前日收盘价 ；

    累计收益率 =（期末收盘价 - 期初收盘价）/期初收盘价。
  - 据此，累计收益率 = (1 + 0.02)(1 + 0.05)(1 + 0.06) – 1，通过简单的数学运算我们得到选项C。
  - 选项中涉及到了常见的描述性统计相关的函数，包括sum()（求和）、mean()（均值）、cumprod()（累乘）、ptp()（极差）。

- **1.10 李明，AQF，某量化基金经理，他目前采取的交易策略包括：同时做多通用汽车和做空福特汽车、同时做多可口可乐和做空百事可乐，请问该基金经理所采用的策略类型最有可能是（ ）？**

  A. 主观交易策略
  B. 事件驱动策略
  C. 时间序列策略
  D. 配对交易策略

- **参考答案：D**
- **解析：**
  - 配对交易基本原理是寻找相关性较高的两个证券，假设两者在未来同样存在较高相关性，当两者价格出现不同走势时，认为该行情会在未来纠正，由此产生套利机会。
  - 题目选取同行业的两只股票，具有高相关性可能性较高，同时采用相反方向操作，符合配对交易的交易方式，因此最可能是配对交易。

➢ **知识点**

- 配对交易策略
  - ✓ 寻找相关性较高的投资目标证券，基于均值回归的原理进行交易。
  - ✓ 举例：
    - ◆ 国内白银期货和国外白银期货

- 事件驱动策略
  - ✓ 基于能够造成投资目标价格波动的事件进行交易。
  - ✓ 举例：
    - ◆ 买入发布业绩预期盈利增长公告的股票

- **1.11 机器学习算法目前正在高速的发展过程中，有些算法已经被应用于量化投资交易，那么下列机器学习算法中，不属于监督学习的是（ ）？**

  A. K近邻算法

  B. 决策树

  C. K均值聚类算法

  D. 支持向量机

- **参考答案：C**
- **解析：**
  - 机器学习中常见的算法分类方法为，根据是否需要使用标记数据，将机器学习中的算法分为监督学习和无监督学习。常见的监督学习算法包括：K近邻算法、决策树、支持向量机、逻辑回归等；常见的无监督学习算法包括：K均值聚类。

- **知识点：机器学习的算法分类**
- **需要明白的概念**
  - 监督学习和无监督学习
  - **输入数据是否有标签（label），有→监督学习；无→无监督学习**
  - **K近邻算法：**如果一个样本在特征空间中的k个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别。
  - **决策树：**决策树(decision tree)是一种基本的分类与回归方法。决策树模型呈树形结构，在分类问题中，表示基于特征对实例进行分类的过程。
  - **K均值聚类算法：**K均值聚类算法是先随机选取K个对象作为初始的聚类中心。然后计算每个对象与各个种子聚类中心之间的距离，把每个对象分配给距离它最近的聚类中心。聚类中心以及分配给它们的对象就代表一个聚类。一旦全部对象都被分配了，每个聚类的聚类中心会根据聚类中现有的对象被重新计算。这个过程将不断重复直到满足某个终止条件，例如：没有（或最小数目）对象被重新分配给不同的聚类，没有（或最小数目）聚类中心再发生变化，误差平方和局部最小。

- **需要明白的概念（续）**
  - **支持向量机：**监督学习中算法的一种。给定训练样本，支持向量机建立一个超平面作为决策曲面，使得正例和反例（即两类样本）的隔离边界最大化。
  - **逻辑回归：**监督学习中回归算法的一种。 逻辑回归分类算法对数据集建立回归公式，并以此进行分类。

- **1.12.下列关于 Python 常用模块的说法中，不正确是（ ）？**
  - A. 模块的程序代码和数据可以封装重用
  - B. Pandas是科学计算的常用基础模块，模块中常见的数据结构包括Ndarray
  - C. Gensim是Python的自然语言处理模块，包括自然语言的主题模型，用于文本的主题挖掘
  - D. 用于机器学习的模块有scikit-learn

- **参考答案： B**
- **解析：**
  - Pandas是科学计算的常用基础模块，常见的数据结构包括Series、DataFrame，而Ndarray为Numpy模块中的常见数据结构，因此，B选项错误。

- **知识点：模块、 Pandas、 Numpy**
  - 模块：只要使用文本编辑器，把一些Python代码输入至文本文件中，然后以“.py”为后缀名进行保存，任何此类文件都会被自动认为是Python模块。
  - Pandas：pandas是Pannel Data Analysis（面板数据分析）的缩写，其基于numpy构建的，为时间序列分析提供了很好的支持。pandas中有两个主要的数据结构，一个是Series，另一个是DataFrame。
  - Numpy：numpy是Python科学计算的核心库，它提供了高性能多维数组对象，以及使用这些数组的工具。
- **知识拓展：自然语言处理模块、机器学习模块**
  - **机器学习模块**
    - ✓ Python机器学习开源项目有： Scikit-learn、 Tensorflow、 Caffe；
    - ✓ Caffe由伯克利视觉和学习中心（BVLC）与社区贡献者开发，是一个深入学习的框架，具备速度快和模块化的特点。
    - ✓ Scikit-learn 是基于NumPy，SciPy和matplotlib，用于数据挖掘和数据分析的简单而有效的工具

- **1.13 李明，AQF，某量化基金经理，正在使用Ndarray数据结构储存最近10日的股票收盘价数据并命名为close_data，如果李明现在想取出最后五天的收盘价数据，则使用的切片代码为（ ）？**

  A. close_data[::-5]

  B. close_data[-5:]

  C. close_data[6:]

  D. close_data[5:9]

- **参考答案：B**
- **解析：**
  - 选项A以5为步长反向取出收盘价数据。
  - 选项B取出最后五天的收盘价数据，因此，B选项正确。
  - 选项C取出最后四天的收盘价数据。
  - 选项D取出最后5天中除最后一天的收盘价数据，不满足题干要求。

- **知识点：**
  - Ndarray数据类型的切片（写法类似list）

```
data[start:end:step]
#     start: 切片的起始位置
#     end：切片的终止位置（不包含该位置元素）
#     step: 步长
```

```
# 常用写法
data[:]                    # 切片为data全部数据
data[::2]                  # 切片为data从头至尾以2为步长获取元素
data[1:4]                  # 第二个到第四个元素（含）
data[1:4:2]                # 第二、四元素
data[-4:]                  # 后四个元素
```

- **1.14 李明，AQF，某量化基金经理，正在使用Series储存某只股票日频的交易量数据并命名为volume_data，行index为日期，交易量部分数据如下图。现希望计算每月的交易量之和，则使用的代码为（ ）？**

A. volume_data.resample('m').sum()

B. volume_data.groupby('m').sum()

C. volume_data.groupby('m').prod()

D. volume_data.resample('m').prod()

```
2018-01-28    100
2018-01-29    120
2018-01-30    310
2018-01-31    240
2018-02-01    130
2018-02-02    420
2018-02-03    350
Freq: D, dtype: int64
```

- **参考答案： A**

- **解析：**

  选项B和选项C选项groupby为以列为依据进行聚合，与本题要求不同。

  选项D选项prod()为乘积。

- **知识点：**
  - pandas中的Groupby技术（详见3.14）
  - pandas中的resample采样

- **知识点及拓展：**
  - pandas中的resample采样常用参数

```
假设用于resample采样的数据名为data，其index为DatetimeIndex格式，数据类型为Series或DataFrame
data.resample(rule, axis=0, closed=None)

# parameters:
#     rule: 转换的目标频率
#     axis：轴向，0表示对列进行处理，1表示对行进行处理
#     closed: 时间区间的闭合方式，left表示前闭， right表示后闭
```

```
# 假设用于resample采样的数据名为data，其index为DatetimeIndex格式，数据类型为
# Series，在代码运行时返回DatetimeIndexResampler对象

data.resample('m')
```

```
DatetimeIndexResampler[freq=<MonthEnd>, axis=0, closed=right,
label=right, convention=start, base=0]
```

- **知识点及拓展：**
  - pandas中的DatetimeIndexResampler常用方法

方法名称	作用
.count()	重采样后各组内数据的数量
.max() .min() .mean() .median() .sum() .std() .var()	重采样后各组内数据的最大值、最小值、均值、总和、标准差、方差
.ohlc()	返回类似K线的开高低收数据

- **知识点及拓展：**
  - pandas中的resample采样例子

```
import pandas as pd
import numpy as np
# 创建用于resample采样的数据
data = pd.Series(
    np.random.randn(100),
    index=pd.date_range('2018-01-01', periods=100)
)
data
```

（部分数据）

```
2018-04-04   -0.353089
2018-04-05    1.915180
2018-04-06    0.417589
2018-04-07   -0.607410
2018-04-08   -0.098125
2018-04-09   -0.964248
2018-04-10   -0.645061
Freq: D, Length: 100, dtype: float64
```

- **知识点及拓展：**
  - pandas中的resample采样例子

```
data.resample('m').sum()
```

```
2018-01-31   -0.807612
2018-02-28   -2.331638
2018-03-31   -6.423468
2018-04-30   -1.116032
Freq: M, dtype: float64
```

```
data.resample('m').ohlc()
```

	open	high	low	close
2018-01-31	-1.291989	1.814639	-1.965325	-1.965325
2018-02-28	0.388459	2.090161	-2.263213	-0.640748
2018-03-31	-1.943919	2.239457	-2.462102	-0.657979
2018-04-30	-0.230669	1.933635	-1.992761	-1.468141

- **1.15 李明，AQF，某量化基金经理，正在编写策略回测代码，按惯例导入Numpy模块后读取数据的代码如下：**

```
pe_data = np.array([10,15,30])
pe = pe_data[3]
```

则上述代码运行时，报告错误的类型与产生错误的原因为（ ）？

A. IndexError, 代码索引超出范围

B. ImportError, 未导入相应模块

C. SyntaxError, 代码语法错误

D. KeyError, 字典的键不存在

- **参考答案：A**
- **解析：**
  - pe_data为Ndarray类型，最大索引为2，pe_data[3]代码超出索引范围，因此，A选项正确。

➢ **知识点：**

- IndexError（序列中没有此索引时）

```
list_ = [1, 2, 3, 4]
list_[5]    # 此列表进行索引时，最大索引为4，当使用list_[5]时，索引不存在
```

```
…
IndexError: list index out of range
```

- ImportError（导入不存在模块或从模块中无法找到该名称）

```
from pandas import no_exist_object    # pandas中无对象名为no_exist_object
```

```
…
ImportError: cannot import name 'no_exist_object'
```

- **知识点：**
  - SyntaxError（无效语法）

```
for i in range(5) print(i)     # 代码for循环后未加’:’号
```

```
…
SyntaxError: invalid syntax
```

  - KeyError（映射的键无法找到）

```
dict_ = {'a':1}
dict_['b']     # 从字典中获得’b’对应的值，但该映射关系不存在
```

```
…
KeyError: 'b'
```

- **常见异常汇总**
  - AttributeError：属性、方法错误，特性引用和赋值失败时会引发属性错误
  - NameError：试图访问的变量名不存在
  - SyntaxError：语法错误，代码形式错误
  - Exception：所有异常的基类，因为所有python异常类都是基类Exception的其中一员，异常都是从基类Exception继承的，并且都在exceptions模块中定义。
  - IOError：一般常见于打开不存在文件时会引发IOError错误，也可以解理为输出输入错误
  - KeyError：使用了映射中不存在的关键字（键）时引发的关键字错误
  - IndexError：索引错误，使用的索引不存在，常索引超出序列范围，什么是索引
  - TypeError：类型错误，内建操作或是函数应于在了错误类型的对象时会引发类型错误
  - ZeroDivisonError：除数为0，在用除法操作时，第二个参数为0时引发了该错误
  - ValueError：值错误，传给对象的参数类型不正确，像是给int()函数传入了字符串数据类型的参数。

- **1.16 列表（List）是量化交易策略中一种常见的数据类型，关于列表的说法中错误的是（ ）？**

A. List是不可变对象

B. List是有序数据类型

C. 使用List时，下标可以是负数

D. List可以存放任意类型的元素

- **参考答案：A**
- **解析：**
  - 列表是Python中的常用数据结构之一。列表可变：可对元素进行修改、添加、插入、删除等操作。选项A中的描述错误。

- **知识点：列表数据类型**
  - 列表结构的特点
    - ✓ 列表是有序结构 （ vs 无序结构，e.g. 集合 ）
    - ✓ 列表索引从0开始
    - ✓ 一个列表可以存储不同类型的数据
    - ✓ 列表中元素可更新 （ vs 不可更新的序列，e.g. 元组 ）
  - 列表访问和切片

语法	功能
l[5], l[-5]	使用索引序号访问列表元素，-n表示倒数第n个元素
l[3:6]	返回索引号为3到5的元素，即不含右端位置6的元素，注：索引从0开始
l[:10:3]	在列表前10个元素中，每隔3个元素取一个元素，第三个参数为步长

- **知识拓展：**
  - 常用列表相关函数及列表方法
    - ✓ 列表相关函数
      - ◆ len(list)：返回列表长度
      - ◆ max/min(list)：返回最大值/最小值
      - ◆ list(seq)：将序列转化为列表
    - ✓ 列表方法
      - ◆ list.append(obj)：在列表末尾添加元素
      - ◆ list.count(obj)：统计某元素在列表中出现的次数
      - ◆ list.reverse()：倒序排列列表元素，注：直接修改原列表，因此没有返回值
      - ◆ list.sort()：排序原列表，注：直接修改原列表，因此没有返回值

- **1.17 在研究量化策略时，各只股票缩写以字符串形式保存在一个list中，如：close_data = ['aaPL', 'Ibm', 'GM']，由于数据质量问题，该组股票名称大小写不统一，现需要将股票缩写调整为大写，则使用的字符串方法是（ ）？**

  A. str.isdigit()

  B. str.upper()

  C. str.lower()

  D. str.capitalize()

- **参考答案：B**
- **解析：**
  - 选项 A判断字符串中字符是否全部为十进制数字；
  - 选项 B将字符串内字符全部调整为大写；
  - 选项 C将字符串内字符全部调整为小写；
  - 选项 D将字符串首字母大写；
  - 因此，正确选项为B。

- **知识拓展：**
  - 字符串判断

字符串方法	作用
isdigit	判断字符串是否全部为十进制数字
isalpha	判断字符串是否全部为字母
isalnum	判断字符串是否全部为数字或字母
isspace	判断字符串是否为空格
istitle	判断字符串中每个单词首字母是否大写且其他字母为小写
isupper	判断字符串中单词是否全部大写
islower	判断字符串中单词是否全部小写

- **1.18 在量化交易的低风险交易策略中，针对同一种类的期货品种，在不同交易所进行的方向相反、数量相同的交易行为，属于哪一种交易策略类型（ ）？**

A. 跨市场套利
B. 跨期套利
C. 多品种套利
D. 期现套利

- **参考答案：A**
- **解析：**
  - 选项A中，符合题目描述，正确。
  - 选项B中，跨期套利是针对相同交易品种，建立月份不同，方向相反的头寸。
  - 选项C中，多品种套利是针对不同交易品种建立方向相反的头寸。
  - 选项D中，期现套利是利用交易品种的期货和现货之间的价差进行的套利。

➢ **知识点：套利交易**

- 期货套利交易分类

套利交易类型	功能
期限套利	利用同品种期货价格和现货价格间的异常价格偏离实施套利
跨期套利	利用同品种期限不同的衍生品之间的异常价格偏离实施套利
跨市场套利	利用同品种在不同市场间的异常价格偏离实施套利
跨品种套利	利用相似品种间的异常价格偏离实施套利

- 统计套利：
- 期权套利：

- **1.19 当try语句中没有出现异常，以下一定不会执行的语句是（ ）？**

  A. try
  B. except
  C. finally
  D. else

- **参考答案：B**
- **解析：**
  - 选项A中，try 内部语句一定会执行。
  - 选项B中，except 语句仅在对应的异常产生时执行。
  - 选项C中，finally语句无论是否产生异常都会执行。
  - 选项D中，else语句仅在不产生异常时执行。

- **知识点：异常处理**
  - 异常处理分支结构

```
try:
    # code 正常执行代码
except ExceptionA:
    # code 异常A发生时的处置代码
except ExceptionB:
    # code 异常B发生时的处置代码
else:
    # code 无异常发生时的处置代码
finally:
    # code 有异常或无异常发生都会最终执行的代码
```

- **1.20 下列哪个代码是进行除法求余数的运算符（ ）？**
  - A. /
  - B. //
  - C. %
  - D. divmod(a,b)

- **参考答案：C**
- **解析：**
  - 选项A为除法运算符。
  - 选项B为除法后取整数。
  - 选项C为模运算，除法后取余数。
  - 选项D为返回商和余数的元组。

- **知识点：算数运算符**
  - 常用算数运算符

运算符	功能
+	两个数求和
-	两个数做差
*	两个数求乘积
/	两个数求商，返回结果为浮点数
%	取模 - 返回除法的余数
**	幂运算
//	取整除法 ，返回浮点数除法向数轴左边取整数以后的结果，运算符两边都是整数则结果为整数，运算符两边出现至少一个浮点数，则结果为浮点数

- **知识拓展：**
  - 其他常用运算符
    - ✓ 比较运算符：用来判断两个对象的值是否相同，e.g. !=, >= ....
    - ✓ 身份运算符：用来判断两个标识符是否表示同一对象（指向内存上的同一地址），e.g. is, is not
    - ✓ 赋值运算符：是来实现变量赋值，e.g. =, *=, //=, **=....
    - ✓ 逻辑运算符：用来实现逻辑运算，e.g. and, or, not
    - ✓ 成员运算符：测试一个对象是否在一个序列中， e.g. in, not in
  - 比较运算符 vs 身份运算符
    - ✓ Python对象包含三个基本元素：id（身份标识）， type（数据类型），value（值）
    - ✓ 比较运算符是对value属性的比较
    - ✓ 身份运算符是对id属性的比较

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

专业来自101%的投入!

- **2.1 李明，AQF，某量化基金经理，正在编写多因子量化交易策略，以下哪些步骤是在研究多因子交易策略的过程中一般需要进行的（ ）？**

  A. 找到候选因子
  B. 筛选有效因子
  C. 因子数据处理
  D. 剔除冗余因子
  E. 因子打分筛选股票

- **参考答案：ABCDE**
- **解析：**

  多因子策略研究过程一般包含确定候选因子、收集因子数据、因子数据处理、筛选有效因子、剔除冗余因子、因子打分、股票组合构建、回测评价与优化等步骤。

- **知识点：**
  - 多因子策略应用:多因子策略实现的一般步骤：
    - ✓ 寻找候选因子：测试单一因子对收益率的影响，挑选对收益率有明显影响的因子
    - ✓ 数据收集：收集并处理因子数据，剔除无效值，空值处理等
    - ✓ 因子筛选：剔除因子中相关性比较强的因子，仅保留相关性尽可能低的因子
    - ✓ 根据因子数据选股：
      - ◆ 排序法
      - ◆ 打分法
    - ✓ 构建投资组合

- **2.2 李明，AQF，某量化基金经理，目前手上持有大量大盘股股票现货头寸，李明担心短期内股票价格可能会出现下跌，那么他可以采取的衍生品风险对冲策略有（ ）？**

A. 直接卖出手上股票
B. 做空股指期货
C. 做多股指期货
D. 做多股指看跌期权
E. 做空股指看跌期权

- **参考答案：BD**

➢ **解析：**

- 选项A中，直接卖股票不属于衍生品对冲策略，所以不正确。
- 选项B中，做空股指期货，如果未来价格下跌，做空头寸会产生正收益，能够对冲现货下跌的损失。
- 选项C中，期货多头和现货多头的收益是正相关关系，如果现货价格下跌，期货多头同样产生亏损，不能起到对冲的作用。
- 选项D中，做多看跌期权，等同于获得一个以一定价格卖出标的资产的权利，在一定范围内看跌期权的价格和基础资产的价格负相关，因此在资产价格下跌时可以起到对冲作用。
- 选项E与D选项相反，D正确所以E错误。

- **知识点：对冲策略**
  - 对冲：用一种资产的价格波动抵消另一种资产的价格波动对整体收益的影响
  - 常用对冲工具
    - ✓ 期货：期货的本质是对未来价格的锁定，期货价格与现货价格存在正相关关系，当现货价格下降时，期货价格一般也会有一定程度的下降，因此当预期现货价格未来会有回调时，可以通过做空期货合约实现对冲
    - ✓ 期权：
      - ◆ 期权的本质是在未来某个时间，以约定的价格买入或卖出资产的权利
      - ◆ 期权买入方获得合约约定的权利，期权卖出方承诺履行合约中规定的义务
      - ◆ 根据认购和认沽期权的分类和买卖方向的不同，期权会产生四种基本收益结构
      - ◆ 使用期权实现对冲需要根据具体交易方向结合期权品种分析

- **知识点：衍生品、 对冲策略**
  - 衍生品
    - ✓ 金融衍生品是指其价值依赖于基础资产（underlyings）价值变动的合约（contracts）。 典型的金融衍生品工具包括远期，期货、期权和互换等。
  - 对冲策略
    - ✓ 对冲策略是指同时在现货市场（基础资产市场）和衍生品市场（远期、期货、期权和互换等）上进行数量相当、方向相反的交易，通过两个市场的盈亏相抵，来锁定既得利润（或成本），可以达到规避系统性风险的目的。具体做法是，已持有股票组合的投资者，预期股市面临下跌风险，可以在衍生品市场上卖空一定数量的股指期货或买入一定数量的股指看跌期权，如果大盘下跌，股指期货或股指看跌期权交易中的收益可以弥补股票组合下跌的损失，规避股票市场下跌引起的风险。

- **2.3 高频交易目前在量化投资领域占据越来越重要的地位，那么下列关于高频交易的说法中正确的有（ ）？**

A. 高频交易一般利用硬件提升即时处理能力，快速发出交易指令
B. 高频交易一般即时下单，追求快速达成交易
C. 高频交易一般可以在几秒甚至毫秒之内完成交易
D. 所有采用高频交易策略的基金一定会维持很高的持仓比例
E. 高频交易一定程度上可以提高金融市场的流动性

- **参考答案：ABCE**
- **解析：**
  - 高频交易和高持仓比例没有必然联系，因此D选项不正确。

- **知识点：高频交易**
  - 高频交易由于对交易速度要求高，因此需要高性能硬件配合
  - 高频交易使用的系统一般部署在交易所主机的附近，以减少数据传输过程中的时间损耗
  - 高频交易单笔的盈亏都比较小，主要依靠大成交量累计盈利
  - 高频交易交易因为成交量巨大，因此可以为市场提供大量流动性
  - 同样由于成家量巨大，高频交易程序的错误也可能对市场造成严重冲击

- **2.4 李明，AQF，某量化基金经理，在进行蒙特卡洛模拟时想要产生500行10列服从标准正态分布的随机数，以下选项正确的是（ ）？**

  A. random.random(size=(500, 10))
  B. numpy.random.randn(500, 10)
  C. numpy.random.normal(size=(500, 10))
  D. numpy.random.sample(size=(500, 10))
  E. numpy.random.exponential(size=(500, 10))

- **参考答案： BC**
- **解析：**
  - 常见的随机数产生方式有：Python下的random模块和numpy模块中的随机数生成函数。

Numpy下的随机数

Function	Description
seed	Seed the random number generator
permutation	Return a random permutation of a sequence, or return a permuted range
shuffle	Randomly permute a sequence in place
rand	Draw samples from a uniform distribution
randint	Draw random integers from a given low-to-high range
randn	Draw samples from a normal distribution with mean 0 and standard deviation 1 (MATLAB-like interface)
binomial	Draw samples a binomial distribution
normal	Draw samples from a normal (Gaussian) distribution
beta	Draw samples from a beta distribution
chisquare	Draw samples from a chi-square distribution
gamma	Draw samples from a gamma distribution
uniform	Draw samples from a uniform [0, 1) distribution

- **2.5 Python语言在量化投资领域已经占据了半壁江山，那么以下关于Python语言进行了描述哪些是正确的（ ）？**

  A. 允许通过一个简单的语句完成复杂的操作
  B. 语句分组由缩进完成而不是首尾括号等
  C. 变量不需要声明
  D. 一般而言，使用Python和C++完成同样一件事情，Python代码运行比C++快
  E. 开源，被称为“胶水语言”，且有大量丰富的第三方库

- **参考答案：ABCE**
- **解析：**
  - 选项A中，Python可以通过简单语句完成复杂操作，如sort函数等。
  - 选项B中，Python使用缩进标记代码块，括号标记是其他语言的特点。
  - 选项C中，C++,Java等需要先声明变量类型再使用，Python直接赋值就可以使用。
  - 选项D中，Python是解释语言，C++是编译语言，编译语言一般比解释语言快。
  - 选项E中，胶水特性：能够轻易调用其他语言编写的库。

- **知识点：**
  - Python 语言相关
    - ✓ 高级语言相对低级语言，抽象程度更高，能用简单的语句实现复杂的功能
    - ✓ Python完全通过缩进标记代码块，C++和Java等语言通过{}标记代码块
    - ✓ C++、Java等语言中，变量使用前需要事先声明变量类型，之后才能赋值使用；Python解释器会自行检测变量值并推测变量类型，这样做的代价是内存使用效率和运算速度的下降
    - ✓ C++属于编译语言，Python属于解释语言，两者区别在于编译语言在编译过程中生成针对CPU的指令，解释语言在运行过程中才生成目标平台指令。一般编译语言的运行效率高于解释语言
    - ✓ Python可以调用其他常见语言编写的模块，因此被称为是一种胶水语言

- **2.6 高数据质量是研究和执行量化投资策略的重要前提。为了得到高质量的结构化金融数据，我们需要进行数据清洗。其中，对于缺失值的处理是数据清洗中的重要一环。以下对于缺失值处理的做法中正确的是（ ）？**

A. 当我们确信删除缺失值对研究结果影响不大时，可以删除缺失值所在的该条样本数据

B. 缺失值可以填入某个反映已有样本分布的统计量。例如：中位数、平均数、众数等

C. 回归法，通过构建合理回归模型得到缺失值的估计值

D. 插值法，如最邻近值插值法

E. 缺失值一定是无效的，所以即使当缺失值在样本中占到了很高比例，也应一律删除

- **参考答案：ABCD**
- **解析：**
  - 在对金融数据清洗过程中，对缺失值进行处理是必要的一个环节。其中，E选项中，缺失值所在的样本数据是否无效应结合样本量大小、数据需求、缺失值填充对具体问题的影响程度等实际情况决定。

- **知识点：**
  - 常用缺失值处理方法
    - ✓ 删除样本：当缺失值数量不大时可以采取删除操作；缺失值数量过多时简单删除会导致样本大量损失
    - ✓ 填充：
      - ◆ 特殊值填充：e.g. 对空值补零
      - ◆ 使用统计特征填充：均值，中位数...
      - ◆ 就近填充：使用前值或后值填充
      - ◆ K均值填充：使用最近的K个临近值的均值填充
      - ◆ 回归填充：使用已有数据建立回归模型，根据回归模型填充空值

➢ **2.7 假设当前有600001, 600002两只股票的ROE数据如下：**

	600001
2017-03-31	0.01
2017-06-30	0.05
2017-12-30	0.20

	600002
2017-03-31	0.04
2017-09-30	0.02
2017-12-30	-0.01

如果将两个储存数据的DataFrame以日期为参照进行横向并集合并，则使用的代码应为（ ）？

A. pd.concat(df1, df2, axis=1)

B. pd.concat([df1, df2], axis=1)

C. pd.concat([df1, df2])

D. pd.concat(df1, df2)

E. pd.merge(df1, df2, left_index=True, right_index=True, how='outer')

多选题

- **参考答案：BE**

- **解析：**
  - 选项 A、D都未将待合并数据存入一个可迭代对象的问题， pd.concat无法进行操作;
  - 选项 C为纵向合并，未以日期为参照;
  - 选项 B、E以日期为参照进行横向并集合并，为正确选项;

- **知识点：**
  - pd.concat()函数合并DataFrame
  - pd.merge()函数合并DataFrame

- **知识点：**
  - pd.concat()函数合并DataFrame及常用参数

```
pd.concat(objs, axis=0, join='outer')
# parameter:
#    objs: 待合并对象的序列
#    axis: 合并方向，0为纵向，1为横向，默认为纵向合并
#    join: 合并方式，outer为取并集，inner为取交集,默认取并集

# 举例
# 纵向拼接两个DataFrame
pd.concat([df1, df2])
# 横向拼接两个DataFrame,结果取并集
pd.concat([df1, df2], axis=1)
# 横向拼接两个DataFrame,结果取交集
pd.concat([df1, df2], axis=1, join='inner')
```

- **知识点：**
  - pd.merge()函数合并DataFrame及常用参数

```
pd.merge(left, right, how='inner', on=None, left_on=None,
right_on=None, left_index=False, right_index=False)

# parameter:
#    left: 参与合并的DataFrame
#    right: 参与合并的另一个DataFrame
#    how:合并方式，outer为取并集，inner为取交集,默认取交集
#    on:合并参照的列名（在两个DataFrame中都应有该列）
#    left_on:左侧DataFrame合并时参照的列
#    right_on:右侧DataFrame合并时参照的列
#    left_index:如果为True，则左侧DataFrame合并时参照index
#    right_index:如果为True，则右侧DataFrame合并时参照index
```

- **2.8 量化交易策略评价体系中，常用来衡量策略的风险的指标有（ ）？**

  A. α

  B. β

  C. 波动率

  D. 最大回撤率

  E. 平均收益

- **参考答案：BCD**

- **解析：**
  - 策略风险衡量时，常用的指标包括波动率、β、最大回撤率、连续回撤次数等。

➢ **知识点：波动率、 β、最大回撤率、连续回撤次数**

- 波动率
  - ✓ 波动率（Volatility）是金融资产价格的波动程度，是对资产收益率不确定性的衡量，用于反映金融资产的风险水平
  - ✓ 波动率越高，金融资产价格的波动越剧烈，资产收益率的不确定性就越强；波动率越低，金融资产价格的波动越平缓，资产收益率的确定性就越强。
- β
  - ✓ β系数也称为贝塔系数（Beta coefficient），是一种风险指数，用来衡量个别股票或股票基金相对于整个股市的价格波动情况。
- 最大回撤率
  - ✓ 最大回撤率（Drawdown）是在选定周期内任一历史时点往后推，产品净值走到最低点时的收益率回撤幅度的最大值，即可能出现的最糟糕的情况。
  - ✓ 计算公式可以表达为：$Drawdown = Max\left(\frac{D_i - D_j}{D_i}\right)$
  - ✓ 备注：$Drawdown$为最大回撤率；D_i、D_j为某一天的产品净值。
- 连续回撤次数
  - ✓ 连续回撤次数是指在某一段时期内出现产品净值从高点开始回落到低点的次数。

- **2.9 一般而言，以下说法正确的是（ ）？**

  A. 套利可以分为风险套利和无风险套利

  B. 套利一定程度上可以减少资产价格的大幅波动

  C. 套利可以提升市场流动性

  D. 统计套利属于无风险套利

  E. 统计套利的优势是不依赖市场趋势，因此可以规避系统性风险

- **参考答案：ABCE**
- **解析：**
  - 选项A中，套利可以分为风险套利和无风险套利，正确。
  - 选项B中，价格波动增加的情况下可能出现套利机会，套利交易者反向操作会促进市场回归均衡值。
  - 选项C中，套利交易者加入可以提供更多的市场流动性。
  - 选项D中，统计套利的基础是历史统计规律，因此统计套利仍然存在风险，故选项D错误。
  - 选项E中，统计套利的依据是交易标的之间的相对关系，不依赖市场趋势。

- **知识点：套利的分类、套利的优点、统计套利**
  - 套利的分类
    - ✓ 套利是指投资者同时利用两地利息率的差价和货币汇率的差价，流动资本以赚取利润。套利分为风险套利（非抵补套利）和无风险套利（抵补套利）两种。其中，风险套利（非抵补套利）是指套利者仅仅利用两种不同货币所带有的不同利息的差价，而将利息率较低的货币转换成利息率较高的货币以赚取利润，未通过相应的衍生品工具规避汇率变动的风险；无风险套利（抵补套利）是套利者在赚取两种不同货币所带有的不同利息的差价的同时，利用相应的衍生品工具规避汇率变动的风险。
  - 套利的优点
    - ✓ 优点：减少资产价格的大幅波动、提升市场流动性。
  - 统计套利
    - ✓ 统计套利是将套利建立对历史数据进行统计分析的基础之上，估计相关变量的概率分布，并结合基本面数据进行分析用以指导套利交易。另，相对价格（价差或比价）的走势由其本身的因素决定，对于来自外界的突发性因素对价差影响不大，因此相对价格的走势的分析往往可以忽略外围的不确定性因素，即规避了系统性风险。

➢ **知识拓展：**

- 金融市场的参与者
  - ✓ 金融市场的参与者主要有：投资者（投机者）、筹资者、套期保值者、套利者，以及调控和监管者五大类。
- 套利交易类型
  - ✓ 套利交易可分为两种类型：
  - ✓ 一是期现套利，即在期货和现货之间进行套利；
  - ✓ 二是对期货市场不同月份之间、不同品种之间、不同市场之间的价差进行套利，被称为价差交易，其中，根据操作对象的不同，价差交易又可分为跨期套利、跨品种套利和跨市套利三种。

- **2.10 在策略执行的过程中，常见的止盈策略有（ ）？**
  - A. 百分比止盈
  - B. 跟踪移动止盈
  - C. 关键价位/点位止盈
  - D. 目标价位止盈
  - E. 技术指标止盈
- **参考答案：ABCDE**
- **解析：**
  - 选项A中，需要在一定盈利比例止盈时使用。
  - 选项B中，跟踪价格波动，当价格向有利方向浮动时相应调整止盈价位，当价格向不利方向运动且超出容忍程度时平仓了结。
  - 选项C中，根据支撑、阻力等关键价位设置止盈目标。
  - 选项D中，当达到事先确定的目标价格时止盈。
  - 选项E中，根据技术指标发出的信号决定止盈时机。

- **知识点：**
  - 常用止盈/止损策略
    - ✓ 百分比：按照成本价的一定比例计算止盈/止损位置
    - ✓ 移动跟踪止盈：通过设置浮动止损指令，在价格发生有利变化时跟随市场调整止损单位置，价格发生反向波动时保持止损单位置，按照此方式可以一直持仓，直到止损单价格被触及则平仓离场，
    - ✓ 根据技术分析或基本面分析，确定关键点位或目标价值实现止盈/止损
    - ✓ 根据技术指标发出的信号止盈/止损

CONTENTS

PROFESSIONAL · LEADING · VALUE-CREATING

专业来自101%的投入!

- **3.1 已知某天股票市场横截面数据为DataFrame数据类型并命名为stock_data, 其index为股票代码，values为股票当天各因子数据，部分数据如下图。现希望打印输出该数据中股票的数量，所使用的代码是？**

	PE
600001	6.68
600004	21.75
600005	75.85
600006	66.71
600007	23.77

- **参考答案：**

```
len(stock_data)
```

- **参考知识与拓展：**
  - 常用内置函数

函数	作用
tuple	创建或转化为元组
str	创建或转化为字符串
list	创建或转化为列表
dict	创建或转化为字典
int	转化为整数
float	转化为浮点
type	检查对象类型
round	返回浮点数四舍五入的值
len	返回对象长度
print	输出指定文字

函数	作用
zip	用于循环
enumerate	
reversed	
map	可用于替代循环
filter	
open	打开文件
sorted	排序
format	字符串格式化
help	显示帮助文档

- **3.2 已知某只银行股票2016年10月1日至2016年10月31日的每日收盘价数据：**

{'2016/10/10':12.57, '2016/10/11':12.58, '2016/10/12':12.53, '2016/10/13':12.47, '2016/10/14':12.51, '2016/10/17':12.35, '2016/10/18':12.41, '2016/10/19':12.35, '2016/10/20':12.37, '2016/10/21':12.38, '2016/10/24':12.59, '2016/10/25':12.41, '2016/10/26':12.39, '2016/10/27':12.26, '2016/10/28':12.32, '2016/10/31':12.35}

请使用Pandas的Series数据结构编写程序计算并打印输出该股票在该段时间的5日移动平均线数据。

- **参考答案：**

```
import pandas as pd
data = pd.Series(
{'2016/10/10':12.57, '2016/10/11':12.58, '2016/10/12':12.53, '2016/10/13':12.47,
'2016/10/14':12.51, '2016/10/17':12.35, '2016/10/18':12.41, '2016/10/19':12.35,
'2016/10/20':12.37, '2016/10/21':12.38, '2016/10/24':12.59, '2016/10/25':12.41,
'2016/10/26':12.39, '2016/10/27':12.26, '2016/10/28':12.32, '2016/10/31':12.35}
)
print(data.rolling(5).mean())
```

- **知识点：**
  - pandas.Series 数据构建：
    - ✓ 语法：pandas.Series(data, index)
    - ✓ 主要参数data参数通常是array-like或dict
  - rolling方法使用：
    - ✓ 语法：pandas.Series.rolling(window).apply_func()
    - ✓ window参数说明移动窗口的大小
    - ✓ rolling方法本身返回Rolling对象，进一步计算需要配合不同的apply_function使用
    - ✓ 常用的apply_function：sum() / mean() …
    - ✓ 可以结合apply()方法使用自定义函数
  - 技术指标计算
    - ✓ 简单指标运算可以通过pandas自带功能实现，比如移动平均线计算
    - ✓ 复杂指标可以通过talib或其他指标计算工具包实现

- **知识拓展：**
  - pandas基本数据结构
    - ✓ Series
    - ✓ DataFrame
  - 其他pandas类似方法
    - ✓ groupby()
    - ✓ resample()
    - ✓ 特点：都需要配合其他聚合函数使用

- **3.3 多因子策略是量化交易策略中最为常见的策略之一，李明，AQF，某量化基金经理，正在执行多因子选股策略，已知df为pandas.DataFrame数据，部分数据如下图：**

	LCAP	PE	REVS20
000001	25.4104	7.3031	0.9322
000002	25.1213	5.9634	0.9236
000004	20.7556	311.7743	1.0415
000005	21.4881	335.2534	0.9438
000006	22.4981	9.2826	0.8973

现要筛选出PE<=20且LCAP>=20且LCAP<=25的股票代码，请编写相应代码，并将结果储存在chosen_stock_codes变量中。

参考答案：

```
chosen_stock_codes = df[(df.PE<=20)
                        & (df.LCAP>=20)
                        & (df.LCAP<=25)].index
```

➢ **解析：**

pandas 数据切片，条件之间用&、|连接，条件要放在括号里，区间的表达要分开两部分，因此可以写成：(df.LCAP>=20) & (df.LCAP<=25)，题目要求获取股票代码，所以最后要通过.index获取。

➢ **知识点**

- pandas.DataFrame上使用bool值索引
- 逻辑运算符号& |

- **3.4 固定收益相对价值套利是很多量化投资基金常用的交易策略，请问在如下实际的市场交易环境中，应该利用哪种相对价值策略进行套利？**

某国当前市场环境：10年期长期债券的收益率过低，短期债券收益率过高，请写出具体可行的相对价值交易策略。

- **参考答案：**

长期收益率低，代表其价格过高，所以策略应该做空该国长期债券；短期收益率高，代表其价格过低，应该同时做多该国短期债券进行相对价值策略交易。

- **3.5 已知data.txt文件中存储数据如下：**

  date,open,close,high,low,volume,code
  2015-04-10,10.806,11.236,11.261,10.742,4515434.0,600000
  2015-04-13,11.531,11.556,11.845,11.293,5814065.0,600000
  2015-04-14,11.556,11.466,11.563,11.255,3728370.0,600000
  ……
  ……
  请写一段代码将文件中的数据读入一个pandas.DataFrame中并设置日期列为index。

- **参考答案：**

```
import pandas as pd
df = pd.read_csv('data.txt', index_col=0)
```

- **解析：**
  - pd.read_csv函数可用于txt和csv文件的读取，读取结果返回DataFrame数据类型。
  - 使用pd.read_csv函数时，可通过index_col参数设置将哪一列作为DataFrame的index，此题index_col=0含义是取第一列，即date列作为index。

- **知识点：import、pandas.read_csv()函数**
  - import
    - ✓ import是可执行的语句，将整个模块对象赋值给一个变量名，而不是编译期间的声明，而且可以嵌套在if测试中，出现在函数def之中等，直到执行程序时，Python执行到这些语句，才会进行解析，被导入的模块和变量名，直到所对应的import或from语句执行后，才可以使用。结合关键字‘as’一起使用，可以给模块定义一个别名
  - pandas.read_csv()函数中常用参数介绍
    - ✓ 函数原型：pandas.read_csv (filepath_or_buffer, sep= ‘, ’ , delimiter=None, header= ‘infer’ , names=None, index_col=None, usecols=None)
    - ✓ filepath_or_buffer： 文件路径或数据缓存地址

- pandas.read_csv()函数中常用参数介绍
  - ✓ sep： 指定分隔符，如果不指定参数，默认使用逗号分隔；
  - ✓ delimiter：定界符，备选分隔符，如果指定该参数，则sep参数失效；
  - ✓ header：指定行数用来作为列名，数据开始行数。如果文件中没有列名，则默认为0，否则设置为None。如果明确设定 header=0 就会替换掉原来存在列名；
  - ✓ names： 用于结果的列名列表，如果数据文件中没有列标题行，就需要执行 header = None；
  - ✓ index_col：用作行索引的列编号或者列名；
  - ✓ usecols ： 返回一个数据子集，该列表中的值必须可以对应到文件中的位置（数字可以对应到指定的列）或者是字符传为文件中的列名。

- **知识拓展：import和from的差别、 DataFrame**
  - import和from的差别
    - ✓ import会读取整个模块，必须进行定义后才能读取相应的变量名；from将获取（或者说是复制）模块特定的变量名
  - DataFrame
    - ✓ DataFrame表示一个表格型的数据结构，并提供有序的列和不同类型的列值；

- **3.6 请定义一个函数calculate_information_ratio(return_data),参数return_data是pandas.DataFrame类型数据，return_data中daily_return列包含了某资产的日收益率信息, benchmark_daily_return列包含了计算信息比率时参照基准的日收益率信息，return_data可能的数据结构如下图：**

	benchmark_daily_return	daily_return
2018-01-01	0.01	0.02
2018-01-02	-0.02	0.06
2018-01-03	0.01	0.10
2018-01-04	0.03	0.03

- **参考答案：**

```
def calculate_information_ratio(return_data):
        std = (return_data['daily_return']-return_data['benchmark_daily_return']).std()
        return (return_data['daily_return'] -
return_data['benchmark_daily_return']).mean()/std
```

- **解析：**
  - 函数定义要点包括： def关键字、函数名、参数、冒号、缩进、返回值。

知识点：函数、信息比率（Information Ratio）

- 函数

```
def  <name>( arg1, arg2, ... argN ):
        ...
       return <value>
```

- 函数是一个通用的程序结构部件，在定义函数时，def 语句将创建一个函数对象并将其赋值给一个变量名，def 语句一般的格式如上所示，def 的首行定义了函数名，赋值给了函数对象，并在括号中包含了0个或以上的参数（有些时候称为是形参），首行以冒号终止后跟随一个代码块，这个代码块通常都会缩进（或者就是在冒号后边简单的一句），利用行缩进分界代码块。return 语句可以在函数主体中的任何地方出现，表示函数调用的结束，并将结果返回至函数调用处，return语句是可选的，如果没有出现，那么函数将会在控制流执行完函数主体时结束。

知识拓展：lambda

- lambda 表达式
  - ✓ lambda 创建一个对象但将其作为结果返回，也可以用 lambda 表达式创建函数，这一功能允许把函数定义内联到语法上一条 def 语句不能工作的地方。这也是一个比较高级的概念。
  - ✓ lambda的一般形式是关键字 lambda，之后是一个或多个参数（与一个 def 头部内用括号括起来的参数列表极其相似），紧跟的是一个冒号，之后是一个表达式：
  - ✓ lambda argument1, argument2,... argumentN : expression using arguments

- **3.7 请用pandas一次性生成日期以用于后续作为DataFrame的Index，要求：时间频率：按天，起始时间：2018年3月18日，时间长度：5000天。（已按惯例导入Pandas和Numpy）**

- **参考答案：**

```
pd.date_range('2018-03-18', periods=5000)
```

知识点及拓展

- pandas.date_range函数及常用参数

  用于生成指定范围频率的DatetimeIndex

```
pd.date_range(start, end, periods, freq, closed)
# parameter:
#     start: 起始日期
#     end: 截止日期
#     periods: 日期数
#     freq: 频率，默认为’D’

# 举例
# 产生2018-01-01到2018-01-08日频DatetimeIndex
pd.date_range(start='2018-01-01', end='2018-01-08')
pd.date_range(start='2018-01-01', periods=8)
# 产生2018-01-01到2018-01-02 2hour频率的DatetimeIndex
pd.date_range(start='2018-01-01', end='2018-01-02', freq='2H')
```

- **3.8 已知signal['MA_5']列和signal['MA_20']分别存储了某标的5日和20日移动平均线的值，signal部分数据如下图。现要求产生一列新数据作为交易信号保存在signal新建的一列上，列名为position，当5日均线值大于20日均线值时，信号列数据为1；当5日均线值小于等于20日均线值时，信号列数据为-1。请写出实现代码（已按惯例导入Pandas和Numpy）。**

- **参考答案：**

```
signal['position'] = np.where(signal['MA_5'] >
signal['MA_20'], 1, -1)
```

- **解析：**

numpy.where()函数的功能是根据条件对数据进行填充，常用于回测过程中产生交易信号。

	MA_5	MA_20
2018-01-01	15	10
2018-01-02	13	11
2018-01-03	10	12
2018-01-04	10	13
2018-01-05	13	12

- **知识点：**
  - numpy.where()函数：
    - ✓ 语法：numpy.where(condition, x,y]
    - ✓ 功能：根据condition的真值情况，当condition为真时返回x中元素, 否则返回y中元素
    - ✓ 应用场景：需要根据条件对数据表进行填充或产生新数据时
  - pandas.DataFrame 插入一列新值
    - ✓ 语法：df[‘new_column’] = new_value
    - ✓ 插入一列新值必须使用[]而不能使用df.new_column

- **3.9 在研究过程中，某因子数据读取后是字符串类型并命名为factor_data，内容形式如下：**
  '\n6.23,5.34,7.23, ... ,3.65,8.93\n'
  现希望将其转化为如下格式：
  ['6.23', '5.34', '7.23', ... , '3.65', '8.93']
  则使用的代码为？

- **参考答案：**

```
factor_data.strip().split(',')
```

知识点及拓展

- 字符串调整方法

方法	作用
string.capitalize()	首字母大写
string.lower()	字符串变为小写
string.expandtabs()	将字符串中\t转化为空格
string.replace()	用新字符替换字符串中旧字符
string.split()	通过指定分隔符将字符串分割
string.strip()	移除字符串头尾指定的字符

- **3.10 请补全函数sort_change(return_data)的定义，参数return_data是一个pandas.Series 数据，索引是股票代码字符串，数据是股票对应的涨跌幅数据，部分数据如下图。函数的功能是返回当日跌幅最大10只股票股票代码并保存在列表中。**

```
def sort_change(return_data):
  # your code here
```

600001	0.0100
600002	-0.0100
600003	0.0700
600004	0.0500
600005	0.0200
600006	0.0190
600007	-0.0121

- **参考答案：**

```
def sort_change(return_data):
  sorted_change =
return_data.sort_values(ascending=True)
  return list(sorted_change.index[:10])
```

- **知识点：**
  - pandas 数据结构排序
    - ✓ .sort_values()方法
    - ✓ 排序方法一般默认升序，需要降序排序可以指定参数ascending=False
  - 提取索引或列名序列：df.index
  - pandas数据切片
  - 使用内置函数list()将pandas数据结构转化为list

- **知识拓展：**
  - 其他pandas排序方法
    - ✓ df.sort_values(by = ['col1', 'col2'])　# 根据多列进行排序
    - ✓ df.sort_index()　# 根据索引排序

- **3.11 假设某因子截面数据以字典形式储存，字典键为股票代码，字典值为因子数据，**

  factor_data = {
  '600001':10, '600002':-5, '600003':0,
  '600004':-1, '600005':3, '600006':5,
  }
  请编写代码选取出因子值大于0的股票并将股票名称储存在一个list中。

- **参考答案：**

```
chosen_stocks = [code for code in factor_data if factor_data[code] > 0]
```

- **解析：**

  此题有多种解法，例如：

```
chosen_stocks = []
for code in factor_data:
    if factor_data[code] > 0:
        chosen_stocks.append(code)
```

- **知识点：**
  - 字典索引
  - 列表解析

- **知识点：**
  - 字典索引

通过字典键获取相应值

```
# 创建字典
dict_ = {'600001':1, '600002':2}
# 获取键’600001’对应值
dict_['600001']
```

```
1
```

- **知识点：**
  - 列表解析

列表解析可通过在循环末尾加if的方式

```
# 方式1 通过列表解析
chosen_stocks = [code for code in factor_data if factor_data[code] > 0]
```

```
# 列表解析原理与以下代码运行逻辑类似（原模拟题解析）
chosen_stocks = []
for code in factor_data:
    if factor_data[code] > 0:
        chosen_stocks.append(code)
```

- **知识拓展：**
  - 其他解决方案

    借助pandas中Series数据类型的bool值索引实现

```
import pandas as pd

# 创建Series
factor_data = {
    '600001':10, '600002':-5, '600003':0,
    '600004':-1, '600005':3, '600006':5,
}

factor_data_series = pd.Series(factor_data)

# 进行筛选获取股票代码
chosen_stocks = list(factor_data_series[factor_data_series>0].index)
```

- **3.12 李明，AQF，某量化基金经理，认为两只股票价格和收益率之间可能存在某种联系。已知一只股票是纺织行业，另一只股票是服装行业。**

以下为2018年1月上半月该纺织行业股票每日收盘价数据：

{'2018/01/02':7.74, '2018/01/03':7.88, '2018/01/04':8.12,
'2018/01/05':7.92, '2018/01/08':7.95, '2018/01/09':7.9,
'2018/01/10':7.82, '2018/01/11':7.81, '2018/01/12':7.74,
'2018/01/15':7.56}

以下为2018年1月上半月该服装行业股票每日收盘价数据：

{'2018/01/02':9.81, '2018/01/03':9.73, '2018/01/04':9.75,
'2018/01/05':9.77, '2018/01/08':10.01, '2018/01/09':9.97,
'2018/01/10':9.69, '2018/01/11':9.72, '2018/01/12':9.73,
'2018/01/15':9.94}

（1）请编写绘制两只股票股价折线图的代码（请注明横轴刻度、图例和标题）；

（2）请编写计算两只股价pearson相关系数的代码。

➢ **参考答案：**

```
# 第一题参考答案
import pandas as pd
import matplotlib.pyplot as plt

# 数据准备
data_textile = pd.Series(
    {'2018/01/02':7.74, '2018/01/03':7.88, '2018/01/04':8.12,
     '2018/01/05':7.92, '2018/01/08':7.95, '2018/01/09':7.9,
     '2018/01/10':7.82, '2018/01/11':7.81, '2018/01/12':7.74,
     '2018/01/15':7.56}
)
data_textile.index = pd.to_datetime(data_textile.index)
data_clothing = pd.Series(
    {'2018/01/02':9.81, '2018/01/03':9.73, '2018/01/04':9.75,
     '2018/01/05':9.77, '2018/01/08':10.01, '2018/01/09':9.97,
     '2018/01/10':9.69, '2018/01/11':9.72, '2018/01/12':9.73,
     '2018/01/15':9.94}
)
data_clothing.index = pd.to_datetime(data_clothing.index)
```

➢ **参考答案：**

```
# 绘图
plt.plot(data_textile, 'ro-',label='XX stock in textile industry')
plt.plot(data_clothing, 'bo-',label='XX stock in clothing industry')
plt.legend()
plt.xticks(rotation=45)
plt.show()

# 第二题参考答案
data_textile.corr(data_clothing)
```

➢ **知识点：**

- 字典创建Series及index转化为DatetimeIndex
- 绘图函数及其参数设置
- 图像设置
- Series.corr方法计算相关系数

- **知识点：**
  - 字典创建Series及index转化为DatetimeIndex

通过向pd.Series()函数中传入字典创建Series，创建时字典的键将转化为Series的index，字典的值将转化为Series的values

```
# 创建Series
data_textile = pd.Series(
    {'2018/01/02':7.74, '2018/01/03':7.88,
'2018/01/04':8.12,
     '2018/01/05':7.92, '2018/01/08':7.95,
'2018/01/09':7.9,
     '2018/01/10':7.82, '2018/01/11':7.81,
'2018/01/12':7.74,
     '2018/01/15':7.56}
)
```

```
2018/01/02    7.74
2018/01/03    7.88
2018/01/04    8.12
2018/01/05    7.92
2018/01/08    7.95
2018/01/09    7.90
2018/01/10    7.82
2018/01/11    7.81
2018/01/12    7.74
2018/01/15    7.56
dtype: float64
```

- **知识点：**
  - 字典创建Series及index转化为DatetimeIndex

通过向pd.to_datetime()函数中传入一个序列，将表示时间的字符串序列转化为DatetimeIndex并作为索引

```
# 将data_textile.index通过pd.to_datetime函数转化为DatetimeIndex后通过等号
# 赋值的方式修改Series的index
data_textile.index = pd.to_datetime(data_textile.index)
```

- 绘图函数及其参数设置

```
# 绘图
# 通过‘ro-’、‘bo-’字段设置线形和颜色
# 通过label函数设置图例名称
plt.plot(data_textile, 'ro-',label='XX stock in textile industry')
plt.plot(data_clothing, 'bo-',label='XX stock in clothing industry
```

- **知识点：**
  - 图像设置

```
# 设置显示图例
plt.legend()
# 设置显示X轴刻度，通过rotation参数设置刻度逆时针旋转45度防止重叠
plt.xticks(rotation=45)
# 显示图像
plt.show()
```

  - Series.corr方法计算相关系数

```
# 计算相关系数
data_textile.corr(data_clothing)
```

- **知识拓展：**
  - 绘图函数plt.plot()常用参数

参数名称	作用
alpha	设置透明度
label	设置图例名称
color	设置颜色
linestyle	设置线形
marker	设置折线数据点形状

- color参数：使用单个字符串进行设置，如’r’:red, ‘g’:green, ‘b’:blue。
- linestyle参数：使用特定字符串设置线形，如’-’:实线，‘--’:虚线
- marker参数：使用特定字符串设置折线数据点形状，如’o’:圆点，’v’:尖头向下的三角形

- **知识拓展：**
  - 使用pyplot模块不同函数绘制不同类型图
  - 使用pyplot设置图像显示

函数名称	作用
plt.plot	绘制折线图
plt.bar	绘制条形图
scatter	绘制散点图
plt.hist	绘制直方图
plt.pie	绘制饼状图
plt.boxplot	绘制箱形图

函数名称	作用
plt.legend	显示图例
plt.xticks	设置x轴刻度
plt.show	显示图像
plt.grid	显示网格
plt.xlim	设置x轴范围

- **知识拓展：**

  由于绘图代码编写比较灵活，因此存在多种代码编写方式实现。

```
# 方法1：数据创建部分
data_textile = {
    '2018/01/02':7.74, '2018/01/03':7.88, '2018/01/04':8.12,
    '2018/01/05':7.92, '2018/01/08':7.95, '2018/01/09':7.9,
    '2018/01/10':7.82, '2018/01/11':7.81, '2018/01/12':7.74,
    '2018/01/15':7.56
}

data_clothing = {
    '2018/01/02':9.81, '2018/01/03':9.73, '2018/01/04':9.75,
    '2018/01/05':9.77, '2018/01/08':10.01, '2018/01/09':9.97,
    '2018/01/10':9.69, '2018/01/11':9.72, '2018/01/12':9.73,
    '2018/01/15':9.94
}
```

- **知识拓展：**

  由于绘图代码编写比较灵活，因此存在多种代码编写方式实现。

```
# 方法1：绘图部分
import matplotlib.pyplot as plt

plt.plot(
    data_textile.keys(),
    data_textile.values(),
    label='XX stock in textile industry'
)
plt.plot(
    data_clothing.keys(),
    data_clothing.values(),
    label='XX stock in clothing industry'
)
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

➢ **知识拓展：**

由于绘图代码编写比较灵活，因此存在多种代码编写方式实现。

```
# 方法2：数据创建部分
import pandas as pd

data_textile = pd.Series(
    {'2018/01/02':7.74, '2018/01/03':7.88, '2018/01/04':8.12,
     '2018/01/05':7.92, '2018/01/08':7.95, '2018/01/09':7.9,
     '2018/01/10':7.82, '2018/01/11':7.81, '2018/01/12':7.74,
     '2018/01/15':7.56}
)

data_clothing = pd.Series(
    {'2018/01/02':9.81, '2018/01/03':9.73, '2018/01/04':9.75,
     '2018/01/05':9.77, '2018/01/08':10.01, '2018/01/09':9.97,
     '2018/01/10':9.69, '2018/01/11':9.72, '2018/01/12':9.73,
     '2018/01/15':9.94}
)
```

- **知识拓展：**

  由于绘图代码编写比较灵活，因此存在多种代码编写方式实现。

```
# 方法2：绘图部分
import matplotlib.pyplot as plt

data_textile.plot(
    xticks=range(10),
    label='XX stock in textile industry'
)
data_clothing.plot(
    xticks=range(10),
    label='XX stock in clothing industry'
)
plt.legend()
plt.xticks(rotation=45)
plt.show()
```

- **3.13 请补全Position类定义的代码。要求该类创建的实例拥有_trade_records属性。_trade_records数据结构为列表，列表元素为表示交易记录的元组，元组结构为(code, amount, time) 分别对应股票代码，交易数量和成交日期，其中交易数量正为买入，负为卖出。Position对象有一个方法add_record，add_record(self，record)方法将交易记录record添加到_trade_records属性中。**

```
class Position():
    def __init__(self):
     # your codes here

    def add_record(self, record):
     # your codes here
```

- **参考答案：**

```
class Position():
    def __init__(self):
        self._trade_records = []

    def add_record(self, record):
        self._trade_records.append(record)
```

- **知识点：**
  - 类定义、__init__方法定义、方法定义
  - list.append方法

- **知识点：**
  - 类定义、__init__方法定义、方法定义、实例属性获取

```
# 类定义，使用class语句定义类
class Position():

    # __init__内的代码块在创建实例时运行，通常在其中写明实例的初始化设置过程
    # 该函数后第一个传入self参数，代表创建的实例
    def __init__(self):

        # 实例属性的设置和调取都可以通过self.attribute的方式实现
        # 初始化实例的_trade_records属性，设置为空列表
        self._trade_records = []

    # 定义实例的方法时，类似函数的定义过程，区别是实例方法定义时第一个参数应该
    # 为self
    # 使用self.attribute获取
    def add_record(self, record):
        self._trade_records.append(record)
```

- **3.14 李明，AQF，某量化基金经理，在进行策略研究时需要对股票进行筛选，当前已知股票数据stock_data如下图：**

现要使用DataFrame中groupby技术从数据stock_data中取出各行业PE的最大值和各行业ROE最小值分别命名为pe_max和roe_min。请编写相应代码。

	industry	PE	ROE
stock1	industry1	30	0.05
stock2	industry1	15	0.07
stock3	industry2	5	-0.01
stock4	industry1	20	0.10
stock5	industry2	8	0.03

- **参考答案：**

```
pe_max = stock_data.groupby('industry').apply(lambda
df:df['PE'].max())
roe_min = stock_data.groupby('industry').apply(lambda
df:df['ROE'].min())
```

- **解析：**
  - 首先通过DataFrame.groupby()方法创建Groupby对象时，通过第一个参数设置作为分组标准的列，此处选择'industry'列作为分组标准。
  - 创建Groupby对象后，使用Groupby.apply()方法通过传入函数的方式对分组后的子数据进行运算，此处传入函数为使用lambda创建的匿名函数。

- **知识点：**
  - Groupby方法原理
  - Groupby对象创建
  - Groupby.apply()方法的使用
  - Lambda匿名函数的创建
  - DataFrame的索引及运算

- **知识点：**
  - Groupby方法原理

	industry	PE	ROE
stock1	industry1	30	0.05
stock2	industry1	15	0.07
stock3	industry2	5	-0.01
stock4	industry1	20	0.10
stock5	industry2	8	0.03

按'industry'列拆分

	industry	PE	ROE
stock1	industry1	30	0.05
stock2	industry1	15	0.07
stock4	industry1	20	0.10

	industry	PE	ROE
stock3	industry2	5	-0.01
stock5	industry2	8	0.03

先求各分组下'PE'最大值，然后合并

industry1	30
industry2	8

➢ **知识点：**

- Groupby对象创建

```
import pandas as pd
stock_data = pd.DataFrame(
    [['industry1', 30, 0.05],
     ['industry1', 15, 0.07],
     ['industry2', 5, -0.01],
     ['industry1', 20, 0.10],
     ['industry2', 8, 0.03]],
    index=['stock1', 'stock2', 'stock3', 'stock4', 'stock5'],
    columns=['industry', 'PE', 'ROE']
)

stock_data.groupby('industry')    # 通过输入列名作为分组依据
```

```
<pandas.core.groupby.DataFrameGroupBy object at
0x000000000B07EA20>
```

➢ **知识点：**

- Groupby对象常用方法

方法分类	Groupby对象方法
统计计算类方法	.describe()
	.mean()
	.median()
	.std()
	.min()
一般操作方法	.apply()
	.agg()

其中：

- Groupby.apply(func, *args, **kwargs)
  可以通过传入一个函数func及其参数的方式对分组后的数据进行处理。func所需参数可以通过tuple或字典在apply中进行设置。

- **知识点：**
  - lambda创建匿名函数：可用于创建简单函数

```
# 乘方
lambda x: x**2

# 对pandas的DataFrame数据结构通过索引获得PE列的数据
lambda df: df['PE']

# 对pandas的DataFrame数据结构通过索引获得PE列数据后取该列最大值
Lambda df: df['PE'].max()
```

- **知识拓展：**
  - Groupby对象分组方式

```
import pandas as pd
stock_data = pd.DataFrame(
    [['industry1', 30, 0.05],
     ['industry1', 15, 0.07],
     ['industry2', 5, -0.01],
     ['industry1', 20, 0.10],
     ['industry2', 8, 0.03]],
    index=['stock1', 'stock2', 'stock3', 'stock4', 'stock5'],
    columns=['industry', 'PE', 'ROE']
)

stock_data.groupby('industry')    # 通过输入列名作为分组依据
stock_data.groupby(stock_data['industry'])    # 以某一列数据作为分组依据
stock_data.groupby(['A', 'B', 'B', 'B', 'A'])    # 传入外部数据进行分组
```

➢ **知识拓展：**

- Groupby对象可进行索引

```
import pandas as pd
stock_data = pd.DataFrame(
    [['industry1', 30, 0.05],
     ['industry1', 15, 0.07],
     ['industry2', 5, -0.01],
     ['industry1', 20, 0.10],
     ['industry2', 8, 0.03]],
    index=['stock1', 'stock2', 'stock3', 'stock4', 'stock5'],
    columns=['industry', 'PE', 'ROE']
)

stock_data.groupby('industry')['PE']     # 对Groupby对象进行索引获得PE列，
                                         # 此时仍为Groupby对象
```

- **知识拓展：**
  - Groupby对象可进行索引，因此答案可以有多种方式实现。

```
# 以各行业PE的最大值为例

# 方式1：答案
# 在匿名函数中完成PE数据的获取及最大值计算
pe_max = stock_data.groupby('industry').apply(lambda df:df['PE'].max())


# 方式2：在groupby对象上进行索引
# 在groupby对象上进行索引后计算各分组的最大值
pe_max = stock_data.groupby('industry')['PE'].max()


# 方式3：先计算各分组最大值之后对拼接后的数据进行索引
pe_max = stock_data.groupby('industry').max()['PE']
```

- **3.15 已知df为pandas中的DataFrame数据，df['close']中存储了某资产的每日价格数据，df索引为日期时间索引。**

（1）请在df中添加一列名为change的数据，计算资产每日收益率后储存到该列；

（2）请根据（1）中结果，计算资产在取样期的日波动率；

（3）计算资产的每月收益率，可以使用（1）问题的计算结果；

（4）计算资产的日夏普比率，已知年化无风险利率4%， 每年按252天计算。

- **参考答案：**

（1）df['change'] = df['close'].pct_change()

（2）std = df['change'].std()

（3）month_change = (df['change']+1).resample('M').prod()-1

（4）sharpe_ratio = (df['change'].mean() - 0.04/252)/std

- **解析：**

  （1）日收益率 =（当日价格 – 前一日价格）/ 前一日价格。

  （2）标准差计算，使用.std()方法。

  （3）resample()函数，小周期到大周期聚合要结合具体情况选择合适的聚合函数。

  （4）sharpe_ratio计算，要使用相同频率的收益率，波动率和无风险利率。

- **知识点：**
  - 日收益率计算、月收益率计算（理论）
  - 变动百分比、标准差计算（代码）
  - resample重采样（代码）
  - sharpe ratio计算（理论）

➢ **知识点：**

- 日收益率和月收益率之间的关系（假设每月有30天）
- $R_{month\ i}+1=\frac{P_{month\ i}}{P_{month\ i-1}}=\frac{P_{month\ i,30}}{P_{month\ i-1,30}}$

$$=\frac{P_{month\ i,1}}{P_{month\ i-1,30}}\cdot\frac{P_{month\ i,2}}{P_{month\ i,1}}\cdot\frac{P_{month\ i,3}}{P_{month\ i,2}}\cdots\frac{P_{month\ i,29}}{P_{month\ i,28}}\cdot\frac{P_{month\ i,30}}{P_{month\ i,29}}$$

$$=\left(1+r_{\text{month }i,1}\right)\cdot\left(1+r_{month\ i,2}\right)\cdot\left(1+r_{month\ i,3}\right)\cdots\left(1+r_{month\ i,29}\right)\cdot\left(1+r_{month\ i,30}\right)$$

其中，$R_{month\ i}$ 表示月份 i 的月收益率

$P_{month\ i,1}$ 表示月份 i 第 1 日的收盘价，其他相似符号以此类推

$r_{month\ i,1}$ 表示月份 i 第 1 日的日收益率，其他相似符号以此类推

- **知识点：**
  - 变动百分比、标准差计算（代码）

```
# 准备用于计算的数据
import pandas as pd
import numpy as np
df = pd.DataFrame(
    np.random.randn(100),
    index=pd.date_range('2018-01-01', periods=100),
    columns=['close'],
)

# 计算百分比可以采用答案所示的.pct_change()方法或使用.shift()方法参照日收益率
# 的公式进行计算

# 方法1 按照答案计算日收益率
df['change'] = df['close'].pct_change()
# 方法2 按照公式计算日收益率
df['change2'] = df['close']/df['close'].shift() - 1
```

- **知识点：**
  - 变动百分比、标准差计算（代码）

```
# 方法3 按照公式计算日收益率,此时使用.div方法代替/除法运算符
df ['change2'] = df['close'].div(df['close'].shift()) - 1

# 以上3种方法均可实现日收益率计算，但因实现机制不同，方法3运行速度最快


# 标准差计算方法
# 使用.std()方法计算日收益率标准差
std = df['change'].std()
```

  - .resample重采样

    详见1.14讲解

➢ **知识点：**

- Sharpe Ratio计算（理论）

$$SharpeRatio = \frac{E(R_P) - R_f}{\sigma_p}$$

- $E(R_P)$表示投资组合的预期收益率
- R_f表示无风险利率
- σ_p 表示投资组合收益率标准差
- 注意，此处投资组合的预期收益率期限要与无风险利率相对应，如投资组合预期日收益率与无风险利率的日收益率对应。

Thank you!