

XtQuant.XtData 行情模块

xtdata是xtquant库中提供行情相关数据的模块，本模块旨在提供精简直接的数据满足量化交易者的数据需求，作为python库的形式可以被灵活添加到各种策略脚本中。

主要提供行情数据（历史和实时的K线和分笔）、财务数据、合约基础信息、板块和行业分类信息等通用的行情数据。

版本信息

- 2020-09-01
 - 初稿
- 2020-09-07
 - 添加获取除权数据的接口 `get_divid_factors`，附录添加除权数据字段说明
 - 获取合约信息、获取合约类型接口完善
 - 获取交易日列表接口 `get_trading_dates` 支持指定日期范围
- 2020-09-13
 - 添加财务数据接口，调整获取和下载财务数据接口的说明，添加财务数据报表字段列表
 - 将“补充”字样调整为“下载”，“supply”接口调整为“download”
- 2020-09-13
 - 将 `voolumn` 拼写错误修正为 `volume`，影响范围：
 - `tick` 和 `l2quote` 周期行情数据 - 成交量字段
 - 合约基础信息 - 总股本、流通股本
- 2020-11-23
 - 合约基础信息 `CreateDate` `OpenDate` 字段类型由 `int` 调整为 `str`
 - 添加数据字典部分，添加`level2`数据字段枚举值说明
- 2021-07-20
 - 添加新版下载数据接口
 - 下载行情数据 `download_history_data2`
 - 下载财务数据 `download_financial_data2`
- 2021-12-30
 - 数据字典调整
 - 委托方向、成交类型添加关于上交所、深交所撤单信息的区分说明
- 2022-06-27
 - 数据字典调整
 - K线添加前收价、停牌标记字段
- 2022-09-30
 - 添加交易日历相关接口
 - 获取节假日数据 `get_holidays`
 - 获取交易日历 `get_trading_calendar`
 - 获取交易时段 `get_trade_times`
- 2023-01-04
 - 添加千档行情获取
- 2023-01-31

- 可转债基础信息的下载 `download_cb_data`
 - 可转债基础信息的获取 `get_cb_info`
- 2023-02-06
 - 添加连接到指定ip端口的接口 `reconnect`
- 2023-02-07
 - 支持QMT的本地Python模式
 - 优化多个QMT同时存在的场景，自动选择xtdata连接的端口
- 2023-03-27
 - 新股申购信息获取 `get_ipo_info`
- 2023-04-13
 - 本地python模式下运行VBA函数
- 2023-07-27
 - 文档部分描述修改
- 2023-08-21
 - 数据接口支持投研版特色数据
 - 参考 `接口概述` - `常用类型说明` - `周期` - `投研版` - `特色数据`
 - 获取合约基础信息 `get_instrument_detail` 返回字段调整
 - 增加 `ExchangeCode` `Unicode`
 - 添加获取可用周期列表的接口 `get_period_list`
- 2023-10-11
 - `get_market_data_ex` 支持获取ETF申赎清单数据
 - 数据字典添加 现金替代标志

接口概述

运行逻辑

xtdata提供和MiniQmt的交互接口，本质是和MiniQmt建立连接，由MiniQmt处理行情数据请求，再把结果回传返回到python层。使用的行情服务器以及能获取到的行情数据和MiniQmt是一致的，要检查数据或者切换连接时直接操作MiniQmt即可。

对于数据获取接口，使用时需要先确保MiniQmt已有所需要的数据，如果不足可以通过补充数据接口补充，再调用数据获取接口获取。

对于订阅接口，直接设置数据回调，数据到来时会由回调返回。订阅接收到的数据一般会保存下来，同种数据不需要再单独补充。

接口分类

- 行情数据（K线数据、分笔数据，订阅和主动获取的接口）
 - 功能划分（接口前缀）
 - `subscribe_ / unsubscribe_` 订阅/反订阅
 - `get_` 获取数据
 - `download_` 下载数据
 - 常见用法
 - level1数据的历史部分用 `download_history_data` 补充，实时部分用 `subscribe_xxx` 订阅，使用 `get_xxx` 获取

- level2数据实时部分用 `subscribe_xxx` 订阅，用 `get_l2_xxx` 获取。level2函数无历史数据存储，跨交易日后数据清理
- 财务数据
- 合约基础信息
- 基础行情数据板块分类信息等基础信息

常用类型说明

- `stock_code` - 合约代码
 - 格式为 `code.market`，例如 `000001.SZ` `600000.SH` `000300.SH`
- `period` - 周期，用于表示要获取的周期和具体数据类型
 - level1数据
 - `tick` - 分笔数据
 - `1m` - 1分钟线
 - `5m` - 5分钟线
 - `15m` - 15分钟线
 - `30m` - 30分钟线
 - `1h` - 1小时线
 - `1d` - 日线
 - level2数据
 - `l2quote` - level2实时行情快照
 - `l2order` - level2逐笔委托
 - `l2transaction` - level2逐笔成交
 - `l2quoteaux` - level2实时行情补充（总买总卖）
 - `l2orderqueue` - level2委买委卖一档委托队列
 - `l2thousand` - level2千档盘口
 - 投研版 - 特色数据
 - `warehousereceipt` - 期货仓单
 - `futureholderrank` - 期货席位
 - `interactiveqa` - 互动问答
 - 逐笔成交统计
 - `transactioncount1m` - 逐笔成交统计1分钟级
 - `transactioncount1d` - 逐笔成交统计日线
 - `delistchangebond` - 退市可转债信息
 - `replacechangebond` - 待发可转债信息
 - `specialtreatment` - ST 变更历史
 - 港股通（深港通、沪港通）资金流向
 - `northfinancechange1m` - 港股通资金流向1分钟级
 - `northfinancechange1d` - 港股通资金流向日线
 - `dividendplaninfo` - 红利分配方案信息
 - `historycontract` - 过期合约列表
 - `optionhistorycontract` - 期权历史信息
 - `historymaincontract` - 历史主力合约

- 时间范围，用于指定数据请求范围，表示的范围是 `[start_time, end_time]` 区间（包含前后边界）中最后不多于 `count` 个数据
 - `start_time` - 起始时间，为空则认为是最早的起始时间
 - `end_time` - 结束时间，为空则认为是最新的结束时间
 - `count` - 数据个数，大于0为正常限制返回个数，等于0为不需要返回，-1为返回全部
 - 通常以 `[start_time = '', end_time = '', count = -1]` 表示完整数据范围，但数据请求范围过大会导致返回时间变长，需要按需裁剪请求范围
- `dividend_type` - 除权方式，用于K线数据复权计算，对 `tick` 等其他周期数据无效
 - `none` 不复权
 - `front` 前复权
 - `back` 后复权
 - `front_ratio` 等比前复权
 - `back_ratio` 等比后复权
- 其他依赖库 `numpy`、`pandas`会在数据返回的过程中使用
 - 本模块会尽可能减少对`numpy`和`pandas`库的直接依赖，以允许使用者在不同版本的库之间自由切换
 - `pandas`库中旧的三维数据结构`Panel`没有被使用，而是以`dict`嵌套`DataFrame`代替（后续可能会考虑使用`xarray`等的方案，也欢迎使用者提供改进建议）
 - 后文中会按常用规则分别简写为`np`、`pd`，如`np.ndarray`、`pd.DataFrame`

请求限制

- 全推数据是市场全部合约的切面数据，是高订阅数场景下的有效解决方案。持续订阅全推数据可以获取到每个合约最新分笔数据的推送，且流量和处理效率都优于单股订阅
- 单股订阅行情是仅返回单股数据的接口，建议单股订阅数量不超过50。如果订阅数较多，建议直接使用全推数据
- 板块分类信息等静态信息更新频率低，无需频繁下载，按周或按日定期下载更新即可

接口说明

行情接口

订阅单股行情

```
subscribe_quote(stock_code, period='1d', start_time='', end_time='', count=0, callback=None)
```

- 释义
 - 订阅单股的行情数据，返回订阅号
 - 数据推送从`callback`返回，数据类型和`period`指定的周期对应
 - 数据范围代表请求的历史部分的数据范围，数据返回后会进入缓存，用于保证数据连续，通常情况仅订阅数据时传 `count = 0` 即可
- 参数
 - `stock_code` - string 合约代码
 - `period` - string 周期
 - `start_time` - string 起始时间
 - `end_time` - string 结束时间

- count - int 数据个数
- callback - 数据推送回调
 - 回调定义形式为 `on_data(datas)`，回调参数 `datas` 格式为 `{ stock_code : [data1, data2, ...]}`

```
def on_data(datas):
    for stock_code in datas:
        print(stock_code, datas[stock_code])
```

- 返回
 - 订阅号，订阅成功返回 大于0，失败返回 -1
- 备注
 - 单股订阅数量不宜过多，详见 接口概述-请求限制

订阅全推行情

```
subscribe_whole_quote(code_list, callback=None)
```

- 释义
 - 订阅全推行情数据，返回订阅号
 - 数据推送从callback返回，数据类型为分笔数据
- 参数
 - code_list - 代码列表，支持传入市场代码或合约代码两种方式
 - 传入市场代码代表订阅全市场，示例： `['SH', 'SZ']`
 - 传入合约代码代表订阅指定的合约，示例： `['600000.SH', '000001.SZ']`
 - callback - 数据推送回调
 - 回调定义形式为 `on_data(datas)`，回调参数 `datas` 格式为 `{ stock1 : data1, stock2 : data2, ... }`

```
def on_data(datas):
    for stock_code in datas:
        print(stock_code, datas[stock_code])
```

- 返回
 - 订阅号，订阅成功返回 大于0，失败返回 -1
- 备注
 - 订阅后会首先返回当前最新的全推数据

反订阅行情数据

```
unsubscribe_quote(seq)
```

- 释义
 - 反订阅行情数据
- 参数
 - seq - 订阅时返回的订阅号
- 返回

- 无
- 备注
 - 无

阻塞线程接收行情回调

run()

- 释义
 - 阻塞当前线程来维持运行状态，一般用于订阅数据后维持运行状态持续处理回调
- 参数
 - seq - 订阅时返回的订阅号
- 返回
 - 无
- 备注
 - 实现方式为持续循环sleep，并在唤醒时检查连接状态，若连接断开则抛出异常结束循环

获取行情数据

```
get_market_data(field_list=[], stock_list=[], period='1d', start_time='',
end_time='', count=-1, dividend_type='none', fill_data=True)
```

- 释义
 - 从缓存获取行情数据，是主动获取行情的主要接口
- 参数
 - field_list - list 数据字段列表，传空则为全部字段
 - stock_list - list 合约代码列表
 - period - string 周期
 - start_time - string 起始时间
 - end_time - string 结束时间
 - count - int 数据个数
 - 默认参数，大于等于0时，若指定了start_time，end_time，此时以end_time为基准向前取count条；若start_time，end_time缺省，默认取本地数据最新的count条数据；若start_time，end_time，count都缺省时，默认取本地全部数据
 - dividend_type - string 除权方式
 - fill_data - bool 是否向后填充空缺数据
- 返回
 - period为 1m 5m 1d 等K线周期时
 - 返回dict { field1 : value1, field2 : value2, ... }
 - field1, field2, ... : 数据字段
 - value1, value2, ... : pd.DataFrame 数据集，index为stock_list，columns为time_list
 - 各字段对应的DataFrame维度相同、索引相同
 - period为 tick 分笔周期时
 - 返回dict { stock1 : value1, stock2 : value2, ... }
 - stock1, stock2, ... : 合约代码
 - value1, value2, ... : np.ndarray 数据集，按数据时间戳 time 增序排列
- 备注
 - 获取lv2数据时需要数据终端有lv2数据权限

- 时间范围为闭区间

获取本地行情数据

```
get_local_data(field_list=[], stock_code=[], period='1d', start_time='',
end_time='', count=-1,
                dividend_type='none', fill_data=True, data_dir=data_dir)
```

- 释义
 - 从本地数据文件获取行情数据，用于快速批量获取历史部分的行情数据
- 参数
 - field_list - list 数据字段列表，传空则为全部字段
 - stock_list - list 合约代码列表
 - period - string 周期
 - start_time - string 起始时间
 - end_time - string 结束时间
 - count - int 数据个数
 - dividend_type - string 除权方式
 - fill_data - bool 是否向后填充空缺数据
 - data_dir - string MiniQmt配套路径的userdata_mini路径，用于直接读取数据文件。默认情况下xtdata会通过连接向MiniQmt直接获取此路径，无需额外设置。如果需要调整，可以将数据路径作为 data_dir 传入，也可以直接修改 xtdata.data_dir 以改变默认值
- 返回
 - period为 1m 5m 1d K线周期时
 - 返回dict { field1 : value1, field2 : value2, ... }
 - field1, field2, ... : 数据字段
 - value1, value2, ... : pd.DataFrame 数据集，index为stock_list，columns为time_list
 - 各字段对应的DataFrame维度相同、索引相同
 - period为 tick 分笔周期时
 - 返回dict { stock1 : value1, stock2 : value2, ... }
 - stock1, stock2, ... : 合约代码
 - value1, value2, ... : np.ndarray 数据集，按数据时间戳 time 增序排列
- 备注
 - 仅用于获取level1数据

获取全推数据

```
get_full_tick(code_list)
```

- 释义
 - 获取全推数据
- 参数
 - code_list - 代码列表，支持传入市场代码或合约代码两种方式
 - 传入市场代码代表订阅全市场，示例： ['SH', 'SZ']
 - 传入合约代码代表订阅指定的合约，示例： ['600000.SH', '000001.SZ']
- 返回
 - dict 数据集 { stock1 : data1, stock2 : data2, ... }
- 备注

- 无

获取除权数据

```
get_divid_factors(stock_code, start_time='', end_time='')
```

- 释义
 - 获取除权数据
- 参数
 - stock_code - 合约代码
 - start_time - string 起始时间
 - end_time - string 结束时间
- 返回
 - pd.DataFrame 数据集
- 备注
 - 无

获取level2行情快照数据

```
get_l2_quote(field_list=[], stock_code='', start_time='', end_time='', count=-1)
```

- 释义
 - 获取level2行情快照数据
- 参数
 - field_list - list 数据字段列表，传空则为全部字段
 - stock_code - string 合约代码
 - start_time - string 起始时间
 - end_time - string 结束时间
 - count - int 数据个数
- 返回
 - np.ndarray 数据集，按数据时间戳 time 增序排列
- 备注
 - 需要缓存中有接收过的数据才能获取到

获取level2逐笔委托数据

```
get_l2_order(field_list=[], stock_code='', start_time='', end_time='', count=-1)
```

- 释义
 - 获取level2逐笔委托数据
- 参数
 - field_list - list 数据字段列表，传空则为全部字段
 - stock_code - string 合约代码
 - start_time - string 起始时间
 - end_time - string 结束时间
 - count - int 数据个数
- 返回
 - np.ndarray 数据集，按数据时间戳 time 增序排列

- 备注
 - 需要缓存中有接收过的数据才能获取到

获取level2逐笔成交数据

```
get_l2_transaction(field_list=[], stock_code='', start_time='', end_time='', count=-1)
```

- 释义
 - 获取level2逐笔成交数据
- 参数
 - field_list - list 数据字段列表, 传空则为全部字段
 - stock_code - string 合约代码
 - start_time - string 起始时间
 - end_time - string 结束时间
 - count - int 数据个数
- 返回
 - np.ndarray 数据集, 按数据时间戳 time 增序排列
- 备注
 - 需要缓存中有接收过的数据才能获取到

下载历史行情数据

```
download_history_data(stock_code, period, start_time='', end_time='')
```

- 释义
 - 补充历史行情数据
- 参数
 - stock_code - string 合约代码
 - period - string 周期
 - start_time - string 起始时间
 - end_time - string 结束时间
- 返回
 - 无
- 备注
 - 同步执行, 补充数据完成后返回

```
download_history_data2(stock_list, period, start_time='', end_time='', callback=None)
```

- 释义
 - 补充历史行情数据, 批量版本
- 参数
 - stock_list - list 合约列表
 - period - string 周期
 - start_time - string 起始时间
 - end_time - string 结束时间

- callback - func 回调函数
 - 参数为进度信息dict
 - total - 总下载个数
 - finished - 已完成个数
 - stockcode - 本地下载完成的合约代码
 - message - 本次信息
 - ```
def on_progress(data):
 print(data)
 # {'finished': 1, 'total': 50, 'stockcode': '000001.SZ',
 'message': ''}
```
- 返回
  - 无
- 备注
  - 同步执行，补充数据完成后返回
  - 有任务完成时通过回调函数返回进度信息

## 获取节假日数据

```
get_holidays()
```

- 释义
  - 获取截止到当年的节假日日期
- 参数
  - 无
- 返回
  - list, 为8位的日期字符串格式
- 备注
  - 无

## 获取交易日历

```
get_trading_calendar(market, start_time = '', end_time = '', tradetimes = False)
```

- 释义
  - 获取指定市场交易日历
- 参数
  - market - str 市场
  - start\_time - str 起始时间，8位字符串。为空表示当前市场首个交易日时间
  - end\_time - str 结束时间，8位字符串。为空表示当前时间
  - tradetimes - bool 是否包含日内交易时段
- 返回
  - tradetimes 为True，返回dict，key为str类型的交易日日期，value为list类型的交易时间段
  - tradetimes 为False，返回list，完整的交易日列表
- 备注
  - 无

## 获取交易时段

```
get_trade_times(stockcode)
```

- 释义
  - 返回指定代码的交易时段
- 参数
  - stockcode - str 合约代码（例如 600000.SH）或市场代码（例如 SH）
- 返回
  - list, 合约的交易时段
- 备注
  - 无

## 可转债基础信息的下载

```
download_cb_data()
```

- 释义
  - 下载全部可转债信息
- 参数
  - 无
- 返回
  - 无
- 备注
  - 无

## 获取可转债基础信息

```
get_cb_info(stockcode)
```

- 释义
  - 返回指定代码的可转债信息
- 参数
  - stockcode - str 合约代码（例如 600000.SH）
- 返回
  - dict, 可转债信息
- 备注
  - 需要先下载可转债数据

## 获取新股申购信息

```
get_ipo_info(start_time, end_time)
```

- 释义
  - 返回所选时间范围的新股申购信息
- 参数
  - start\_time: 开始日期（如：'20230327'）
  - end\_time: 结束日期（如：'20230327'）
  - start\_time 和 end\_time 为空则返回全部数据

- 返回
  - list[dict], 新股申购信息
  - ```

securityCode - string 证券代码
codeName - string 代码简称
market - string 所属市场
actIssueQty - int 发行总量, 单位: 股
onlineIssueQty - int 网上发行量, 单位: 股
onlineSubCode - string 申购代码
onlineSubMaxQty - int 申购上限, 单位: 股
publishPrice - float 发行价格
isProfit - int 是否已盈利 0: 上市时尚未盈利 1: 上市时已盈利
industryPe - float 行业市盈率
afterPE - float 发行后市盈率
          
```

获取可用周期列表

```
get_period_list()
```

- 释义
 - 返回可用周期列表
- 参数
 - 无
- 返回
 - list 周期列表

财务数据接口

获取财务数据

```
get_financial_data(stock_list, table_list=[], start_time='', end_time='',
report_type='report_time')
```

- 释义
 - 获取财务数据
- 参数
 - stock_list - list 合约代码列表
 - table_list - list 财务数据表名称列表
 - | | |
|-------------------|---------|
| 'Balance' | #资产负债表 |
| 'Income' | #利润表 |
| 'CashFlow' | #现金流量表 |
| 'Capital' | #股本表 |
| 'Holdernum' | #股东数 |
| 'Top10holder' | #十大股东 |
| 'Top10flowholder' | #十大流通股东 |
| 'Pershareindex' | #每股指标 |
 - start_time - string 起始时间
 - end_time - string 结束时间
 - report_type - string 报表筛选方式

- `'report_time'` #截止日期
`'announce_time'` #披露日期

- 返回
 - dict 数据集 { stock1 : datas1, stock2 : data2, ... }
 - stock1, stock2, ... : 合约代码
 - datas1, datas2, ... : dict 数据集 { table1 : table_data1, table2 : table_data2, ... }
 - table1, table2, ... : 财务数据表名
 - table_data1, table_data2, ... : pd.DataFrame 数据集, 数据字段详见附录 - 财务数据字段列表
- 备注
 - 无

下载财务数据

```
download_financial_data(stock_list, table_list=[])
```

- 释义
 - 下载财务数据
- 参数
 - stock_list - list 合约代码列表
 - table_list - list 财务数据表名列表
- 返回
 - 无
- 备注
 - 同步执行, 补充数据完成后返回

```
download_financial_data2(stock_list, table_list=[], start_time='', end_time='',  
callback=None)
```

- 释义
 - 下载财务数据
- 参数
 - stock_list - list 合约代码列表
 - table_list - list 财务数据表名列表
 - start_time - string 起始时间
 - end_time - string 结束时间
 - 以 `m_anntime` 披露日期字段, 按 `[start_time, end_time]` 范围筛选
 - callback - func 回调函数
 - 参数为进度信息dict
 - total - 总下载个数
 - finished - 已完成个数
 - stockcode - 本地下载完成的合约代码
 - message - 本次信息

- ```
def on_progress(data):
 print(data)
 # {'finished': 1, 'total': 50, 'stockcode': '000001.sz',
 'message': ''}
```

- 返回
  - 无
- 备注
  - 同步执行，补充数据完成后返回

## 基础行情信息

### 获取合约基础信息

```
get_instrument_detail(stock_code)
```

- 释义
  - 获取合约基础信息
- 参数
  - stock\_code - string 合约代码
- 返回
  - dict 数据字典，{ field1 : value1, field2 : value2, ... }, 找不到指定合约时返回 None
  - ```
ExchangeID - string 合约市场代码  
InstrumentID - string 合约代码  
InstrumentName - string 合约名称  
ProductID - string 合约的品种ID(期货)  
ProductName - string 合约的品种名称(期货)  
ExchangeCode - string 交易所代码  
Unicode - string 统一规则代码  
CreateDate - str 上市日期(期货)  
OpenDate - str IPO日期(股票)  
ExpireDate - int 退市日或者到期日  
PreClose - float 前收盘价格  
SettlementPrice - float 前结算价格  
UpStopPrice - float 当日涨停价  
DownStopPrice - float 当日跌停价  
FloatVolume - float 流通股本  
TotalVolume - float 总股本  
LongMarginRatio - float 多头保证金率  
ShortMarginRatio - float 空头保证金率  
PriceTick - float 最小价格变动单位  
VolumeMultiple - int 合约乘数(对期货以外的品种，默认是1)  
MainContract - int 主力合约标记，1、2、3分别表示第一主力合约，第二主力合约，第三主力合约  
LastVolume - int 昨日持仓量  
InstrumentStatus - int 合约停牌状态  
IsTrading - bool 合约是否可交易  
IsRecent - bool 是否是近月合约
```
- 备注
 - 可用于检查合约代码是否正确

- 合约基础信息 `CreateDate` `OpenDate` 字段类型由 `int` 调整为 `str`

获取合约类型

```
get_instrument_type(stock_code)
```

- 释义
 - 获取合约类型
- 参数
 - `stock_code` - string 合约代码
- 返回
 - dict 数据字典, { `type1`: `value1`, `type2`: `value2`, ... }, 找不到指定合约时返回 `None`
 - `type1`, `type2`, ... : string 合约类型
 - `value1`, `value2`, ... : bool 是否为该类合约
 - | | |
|----------------------|------|
| <code>'index'</code> | #指数 |
| <code>'stock'</code> | #股票 |
| <code>'fund'</code> | #基金 |
| <code>'etf'</code> | #ETF |
- 备注
 - 无

获取交易日列表

```
get_trading_dates(market, start_time='', end_time='', count=-1)
```

- 释义
 - 获取交易日列表
- 参数
 - `market` - string 市场代码
 - `start_time` - string 起始时间
 - `end_time` - string 结束时间
 - `count` - int 数据个数
- 返回
 - list 时间戳列表, [`date1`, `date2`, ...]
- 备注
 - 无

获取板块列表

```
get_sector_list()
```

- 释义
 - 获取板块列表
- 参数
 - 无
- 返回
 - list 板块列表, [`sector1`, `sector2`, ...]

- 备注
 - 需要下载板块分类信息

获取板块成分股列表

```
get_stock_list_in_sector(sector_name)
```

- 释义
 - 获取板块成分股列表
- 参数
 - sector_name - string 板块名称
- 返回
 - list 成分股列表, [stock1, stock2, ...]
- 备注
 - 需要板块分类信息

下载板块分类信息

```
download_sector_data()
```

- 释义
 - 下载板块分类信息
- 参数
 - 无
- 返回
 - 无
- 备注
 - 同步执行，下载完成后返回

创建板块目录节点

```
create_sector_folder(parent_node, folder_name, overwrite)
```

- 释义
 - 创建板块目录节点
- 参数
 - parent_node - string 父节点, '为 '我的' (默认目录)
 - folder_name - string 要创建的板块目录名称
 - overwrite- bool 是否覆盖, 如果目标节点已存在, 为True时跳过, 为False时在 folder_name后增加数字编号, 编号为从1开始自增的第一个不重复的值。默认为True
- 返回
 - folder_name2 - string 实际创建的板块目录名
- 备注
 - 无

创建板块

```
create_sector(parent_node, sector_name, overwrite)
```

- 释义
 - 创建板块
- 参数
 - parent_node - string 父节点, ''为 '我的' (默认目录)
 - sector_name - string 板块名称
 - overwrite- bool 是否覆盖, 如果目标节点已存在, 为True时跳过, 为False时在 sector_name后增加数字编号, 编号为从1开始自增的第一个不重复的值。
默认为True
- 返回
 - sector_name2 - string 实际创建的板块名
- 备注
 - 无

添加自定义板块

```
add_sector(sector_name, stock_list)
```

- 释义
 - 添加自定义板块
- 参数
 - sector_name - string 板块名称
 - stock_list - list 成分股列表
- 返回
 - 无
- 备注
 - 无

移除板块成分股

```
remove_stock_from_sector(sector_name, stock_list)
```

- 释义
 - 创建板块
- 参数
 - sector_name - string 板块名称
 - stock_list- list 成分股列表
- 返回
 - result - bool 操作成功为True, 失败为False
- 备注
 - 无

移除自定义板块

```
remove_sector(sector_name)
```

- 释义

- 移除自定义板块
- 参数
 - sector_name - string 板块名称
- 返回
 - 无
- 备注
 - 无

重置板块

```
reset_sector(sector_name, stock_list)
```

- 释义
 - 重置板块
- 参数
 - sector_name - string 板块名称
 - stock_list- list 成分股列表
- 返回
 - result - bool 操作成功为True，失败为False
- 备注
 - 无

获取指数成分权重信息

```
get_index_weight(index_code)
```

- 释义
 - 获取指数成分权重信息
- 参数
 - index_code - string 指数代码
- 返回
 - dict 数据字典, { stock1 : weight1, stock2 : weight2, ... }
- 备注
 - 需要下载指数成分权重信息

下载指数成分权重信息

```
download_index_weight()
```

- 释义
 - 下载指数成分权重信息
- 参数
 - 无
- 返回
 - 无
- 备注
 - 同步执行，下载完成后返回

附录

行情数据字段列表

tick - 分笔数据

'time'	#时间戳
'lastPrice'	#最新价
'open'	#开盘价
'high'	#最高价
'low'	#最低价
'lastClose'	#前收盘价
'amount'	#成交总额
'volume'	#成交总量
'pvolume'	#原始成交总量
'stockStatus'	#证券状态
'openInt'	#持仓量
'lastSettlementPrice'	#前结算
'askPrice'	#委卖价
'bidPrice'	#委买价
'askVol'	#委卖量
'bidVol'	#委买量

1m / 5m / 1d - K线数据

'time'	#时间戳
'open'	#开盘价
'high'	#最高价
'low'	#最低价
'close'	#收盘价
'volume'	#成交量
'amount'	#成交额
'settlementPrice'	#今结算
'openInterest'	#持仓量
'preClose'	#前收价
'suspendFlag'	#停牌标记 0 - 正常 1 - 停牌 -1 - 当日起复牌

除权数据

'interest'	#每股股利（税前，元）
'stockBonus'	#每股红股（股）
'stockGift'	#每股转增股本（股）
'allotNum'	#每股配股数（股）
'allotPrice'	#配股价格（元）
'gugai'	#是否股改，对于股改，在算复权系数时，系统有特殊算法
'dr'	#除权系数

l2quote - level2实时行情快照

'time'	#时间戳
'lastPrice'	#最新价
'open'	#开盘价

'high'	#最高价
'low'	#最低价
'amount'	#成交额
'volume'	#成交总量
'pvolume'	#原始成交总量
'openInt'	#持仓量
'stockStatus'	#证券状态
'transactionNum'	#成交笔数
'lastClose'	#前收盘价
'lastSettlementPrice'	#前结算
'settlementPrice'	#今结算
'pe'	#市盈率
'askPrice'	#多档委卖价
'bidPrice'	#多档委买价
'askVol'	#多档委卖量
'bidVol'	#多档委买量

l2order - level2逐笔委托

'time'	#时间戳
'price'	#委托价
'volume'	#委托量
'entrustNo'	#委托号
'entrustType'	#委托类型
'entrustDirection'	#委托方向

l2transaction - level2逐笔成交

'time'	#时间戳
'price'	#成交价
'volume'	#成交量
'amount'	#成交额
'tradeIndex'	#成交记录号
'buyNo'	#买方委托号
'sellNo'	#卖方委托号
'tradeType'	#成交类型
'tradeFlag'	#成交标志

l2quoteaux - level2实时行情补充（总买总卖）

'time'	#时间戳
'avgBidPrice'	#委买均价
'totalBidQuantity'	#委买总量
'avgOffPrice'	#委卖均价
'totalOffQuantity'	#委卖总量
'withdrawBidQuantity'	#买入撤单总量
'withdrawBidAmount'	#买入撤单总额
'withdrawOffQuantity'	#卖出撤单总量
'withdrawOffAmount'	#卖出撤单总额

l2orderqueue - level2委买委卖一档委托队列

'time'	#时间戳
'bidLevelPrice'	#委买价
'bidLevelVolume'	#委买量
'offerLevelPrice'	#委卖价
'offerLevelVolume'	#委卖量
'bidLevelNumber'	#委买数量
'offLevelNumber'	#委卖数量

数据字典

证券状态

0,10 - 默认为未知
11 - 开盘前S
12 - 集合竞价时段C
13 - 连续交易T
14 - 休市B
15 - 闭市E
16 - 波动性中断V
17 - 临时停牌P
18 - 收盘集合竞价U
19 - 盘中集合竞价M
20 - 暂停交易至闭市N
21 - 获取字段异常
22 - 盘后固定价格行情
23 - 盘后固定价格行情完毕

委托类型

- level2逐笔委托 - `entrustType` 委托类型
- level2逐笔成交 - `tradeType` 成交类型

0 - 未知
1 - 正常交易业务
2 - 即时成交剩余撤销
3 - ETF基金申报
4 - 最优五档即时成交剩余撤销
5 - 全额成交或撤销
6 - 本方最优价格
7 - 对手方最优价格

委托方向

- level2逐笔委托 - `entrustDirection` 委托方向
 - 注：上交所的撤单信息在逐笔委托的委托方向，区分撤买撤卖

1 - 买入
2 - 卖出
3 - 撤买（上交所）
4 - 撤卖（上交所）

成交标志

- level2逐笔成交 - `tradeFlag` 成交标志

- 注：深交所的在逐笔成交的成交标志，只有撤单，没有方向

- 0 - 未知
- 1 - 外盘
- 2 - 内盘
- 3 - 撤单（深交所）

现金替代标志

- ETF申赎清单成份股现金替代标志

- 0 - 禁止现金替代（必须有股票）
- 1 - 允许现金替代（先用股票，股票不足的话用现金替代）
- 2 - 必须现金替代
- 3 - 非沪市（股票）退补现金替代
- 4 - 非沪市（股票）必须现金替代
- 5 - 非沪深退补现金替代
- 6 - 非沪深必须现金替代
- 7 - 港市退补现金替代（仅适用于跨沪深ETF产品）
- 8 - 港市必须现金替代（仅适用于跨沪深港ETF产品）

财务数据字段列表

Balance - 资产负债表

'm_anntime'	#披露日期
'm_timetag'	#截止日期
'internal_shoule_recv'	#内部应收款
'fixed_capital_clearance'	#固定资产清理
'should_pay_money'	#应付分保账款
'settlement_payment'	#结算备付金
'receivable_premium'	#应收保费
'accounts_receivable_reinsurance'	#应收分保账款
'reinsurance_contract_reserve'	#应收分保合同准备金
'dividends_payable'	#应收股利
'tax_rebate_for_export'	#应收出口退税
'subsidies_receivable'	#应收补贴款
'deposit_receivable'	#应收保证金
'apportioned_cost'	#待摊费用
'profit_and_current_assets_with_deal'	#待处理流动资产损益
'current_assets_one_year'	#一年内到期的非流动资产
'long_term_receivables'	#长期应收款
'other_long_term_investments'	#其他长期投资
'original_value_of_fixed_assets'	#固定资产原值
'net_value_of_fixed_assets'	#固定资产净值
'depreciation_reserves_of_fixed_assets'	#固定资产减值准备
'productive_biological_assets'	#生产性生物资产
'public_welfare_biological_assets'	#公益性生物资产
'oil_and_gas_assets'	#油气资产
'development_expenditure'	#开发支出
'right_of_split_share_distribution'	#股权分置流通权
'other_non_mobile_assets'	#其他非流动资产
'handling_fee_and_commission'	#应付手续费及佣金
'other_payables'	#其他应交款
'margin_payable'	#应付保证金
'internal_accounts_payable'	#内部应付款

'advance_cost'	#预提费用
'insurance_contract_reserve'	#保险合同准备金
'broker_buying_and_selling_securities'	#代理买卖证券款
'acting_underwriting_securities'	#代理承销证券款
'international_ticket_settlement'	#国际票证结算
'domestic_ticket_settlement'	#国内票证结算
'deferred_income'	#递延收益
'short_term_bonds_payable'	#应付短期债券
'long_term_deferred_income'	#长期递延收益
'undetermined_investment_losses'	#未确定的投资损失
'quasi_distribution_of_cash_dividends'	#拟分配现金股利
'provisions_not'	#预计负债
'cust_bank_dep'	#吸收存款及同业存放
'provisions'	#预计流动负债
'less_tsy_stk'	#减:库存股
'cash_equivalents'	#货币资金
'loans_to_oth_banks'	#拆出资金
'tradable_fin_assets'	#交易性金融资产
'derivative_fin_assets'	#衍生金融资产
'bill_receivable'	#应收票据
'account_receivable'	#应收账款
'advance_payment'	#预付款项
'int_rcv'	#应收利息
'other_receivable'	#其他应收款
'red_monetary_cap_for_sale'	#买入返售金融资产
'agency_bus_assets'	#以公允价值计量且其变动计入当期损益的金融资产
'inventories'	#存货
'other_current_assets'	#其他流动资产
'total_current_assets'	#流动资产合计
'loans_and_adv_granted'	#发放贷款及垫款
'fin_assets_avail_for_sale'	#可供出售金融资产
'held_to_mty_invest'	#持有至到期投资
'long_term_eqy_invest'	#长期股权投资
'invest_real_estate'	#投资性房地产
'accumulated_depreciation'	#累计折旧
'fix_assets'	#固定资产
'constru_in_process'	#在建工程
'construction_materials'	#工程物资
'long_term_liabilities'	#长期负债
'intang_assets'	#无形资产
'goodwill'	#商誉
'long_deferred_expense'	#长期待摊费用
'deferred_tax_assets'	#递延所得税资产
'total_non_current_assets'	#非流动资产合计
'tot_assets'	#资产总计
'shortterm_loan'	#短期借款
'borrow_central_bank'	#向中央银行借款
'loans_oth_banks'	#拆入资金
'tradable_fin_liab'	#交易性金融负债
'derivative_fin_liab'	#衍生金融负债
'notes_payable'	#应付票据
'accounts_payable'	#应付账款
'advance_peceipts'	#预收账款
'fund_sales_fin_assets_rp'	#卖出回购金融资产款
'empl_ben_payable'	#应付职工薪酬
'taxes_surcharges_payable'	#应交税费
'int_payable'	#应付利息

'dividend_payable'	#应付股利
'other_payable'	#其他应付款
'non_current_liability_in_one_year'	#一年内到期的非流动负债
'other_current_liability'	#其他流动负债
'total_current_liability'	#流动负债合计
'long_term_loans'	#长期借款
'bonds_payable'	#应付债券
'longterm_account_payable'	#长期应付款
'grants_received'	#专项应付款
'deferred_tax_liab'	#递延所得税负债
'other_non_current_liabilities'	#其他非流动负债
'non_current_liabilities'	#非流动负债合计
'tot_liab'	#负债合计
'cap_stk'	#实收资本(或股本)
'cap_rsrv'	#资本公积
'specific_reserves'	#专项储备
'surplus_rsrv'	#盈余公积
'prov_nom_risks'	#一般风险准备
'undistributed_profit'	#未分配利润
'cnvd_diff_foreign_curr_stat'	#外币报表折算差额
'tot_shrhldr_eqy_excl_min_int'	#归属于母公司股东权益合计
'minority_int'	#少数股东权益
'total_equity'	#所有者权益合计
'tot_liab_shrhldr_eqy'	#负债和股东权益总计

Income - 利润表

'm_anntime'	#披露日期
'm_timetag'	#截止日期
'revenue_inc'	#营业收入
'earned_premium'	#已赚保费
'real_estate_sales_income'	#房地产销售收入
'total_operating_cost'	#营业总成本
'real_estate_sales_cost'	#房地产销售成本
'research_expenses'	#研发费用
'surrender_value'	#退保金
'net_payments'	#赔付支出净额
'net_withdrawal_ins_con_res'	#提取保险合同准备金净额
'policy_dividend_expenses'	#保单红利支出
'reinsurance_cost'	#分保费用
'change_income_fair_value'	#公允价值变动收益
'futures_loss'	#期货损益
'trust_income'	#托管收益
'subsidize_revenue'	#补贴收入
'other_business_profits'	#其他业务利润
'net_profit_excl_merged_int_inc'	#被合并方在合并前实现净利润
'int_inc'	#利息收入
'handling_chrg_comm_inc'	#手续费及佣金收入
'less_handling_chrg_comm_exp'	#手续费及佣金支出
'other_bus_cost'	#其他业务成本
'plus_net_gain_fx_trans'	#汇兑收益
'il_net_loss_disp_noncur_asset'	#非流动资产处置收益
'inc_tax'	#所得税费用
'unconfirmed_invest_loss'	#未确认投资损失
'net_profit_excl_min_int_inc'	#归属于母公司所有者的净利润
'less_int_exp'	#利息支出
'other_bus_inc'	#其他业务收入

'revenue'	#营业总收入
'total_expense'	#营业成本
'less_taxes_surcharges_ops'	#营业税金及附加
'sale_expense'	#销售费用
'less_geri_admin_exp'	#管理费用
'financial_expense'	#财务费用
'less_impair_loss_assets'	#资产减值损失
'plus_net_invest_inc'	#投资收益
'incl_inc_invest_assoc_jv_entp'	#联营企业和合营企业的投资收益
'oper_profit'	#营业利润
'plus_non_oper_rev'	#营业外收入
'less_non_oper_exp'	#营业外支出
'tot_profit'	#利润总额
'net_profit_incl_min_int_inc'	#净利润
'net_profit_incl_min_int_inc_after'	#净利润(扣除非经常性损益后)
'minority_int_inc'	#少数股东损益
's_fa_eps_basic'	#基本每股收益
's_fa_eps_diluted'	#稀释每股收益
'total_income'	#综合收益总额
'total_income_minority'	#归属于少数股东的综合收益总额
'other_compreh_inc'	#其他收益

CashFlow - 现金流量表

'm_anntime'	#披露日期
'm_timetag'	#截止日期
'cash_received_ori_ins_contract_pre'	#收到原保险合同保费取得的现金
'net_cash_received_rei_ope'	#收到再保险业务现金净额
'net_increase_insured_funds'	#保户储金及投资款净增加额
'Net'	#处置交易性金融资产净增加额
increase_in_disposal	
'cash_for_interest'	#收取利息、手续费及佣金的现金
'net_increase_in_repurchase_funds'	#回购业务资金净增加额
'cash_for_payment_original_insurance'	#支付原保险合同赔付款项的现金
'cash_payment_policy_dividends'	#支付保单红利的现金
'disposal_other_business_units'	#处置子公司及其他收到的现金
'cash_received_from_pledges'	#减少质押和定期存款所收到的现金
'cash_paid_for_investments'	#投资所支付的现金
'net_increase_in_pledged_loans'	#质押贷款净增加额
'cash_paid_by_subsidiaries'	#取得子公司及其他营业单位支付的现金净额
'increase_in_cash_paid'	#增加质押和定期存款所支付的现金
'cass_received_sub_abs'	#其中子公司吸收现金
'cass_received_sub_investments'	#其中:子公司支付给少数股东的股利、利润
'minority_shareholder_profit_loss'	#少数股东损益
'unrecognized_investment_losses'	#未确认的投资损失
'ncrease_deferred_income'	#递延收益增加(减:减少)
'projected_liability'	#预计负债
'increase_operational_payables'	#经营性应付项目的增加
'reduction_outstanding_amounts_less'	#已完工尚未结算款的减少(减:增加)
'reduction_outstanding_amounts_more'	#已结算尚未完工款的增加(减:减少)
'goods_sale_and_service_render_cash'	#销售商品、提供劳务收到的现金
'net_incr_dep_cob'	#客户存款和同业存放款项净增加额
'net_incr_loans_central_bank'	#向中央银行借款净增加额(万元)
'net_incr_fund_borr_ofi'	#向其他金融机构拆入资金净增加额
'net_incr_fund_borr_ofi'	#拆入资金净增加额
'tax_levy_refund'	#收到的税费与返还
'cash_paid_invest'	#投资支付的现金

'other_cash_recip_ral_oper_act'	#收到的其他与经营活动有关的现金
'stot_cash_inflows_oper_act'	#经营活动现金流入小计
'goods_and_services_cash_paid'	#购买商品、接受劳务支付的现金
'net_incr_clients_loan_adv'	#客户贷款及垫款净增加额
'net_incr_dep_cbob'	#存放中央银行和同业款项净增加额
'handling_chrg_paid'	#支付利息、手续费及佣金的现金
'cash_pay_beh_empl'	#支付给职工以及为职工支付的现金
'pay_all_typ_tax'	#支付的各项税费
'other_cash_pay_ral_oper_act'	#支付其他与经营活动有关的现金
'stot_cash_outflows_oper_act'	#经营活动现金流出小计
'net_cash_flows_oper_act'	#经营活动产生的现金流量净额
'cash_recip_disp_withdrwl_invest'	#收回投资所收到的现金
'cash_recip_return_invest'	#取得投资收益所收到的现金
'net_cash_recip_disp_fiolta'	#处置固定资产、无形资产和其他长期投资收到的
现金	
'other_cash_recip_ral_inv_act'	#收到的其他与投资活动有关的现金
'stot_cash_inflows_inv_act'	#投资活动现金流入小计
'cash_pay_acq_const_fiolta'	#购建固定资产、无形资产和其他长期投资支付的
现金	
'other_cash_pay_ral_oper_act'	#支付其他与投资的现金
'stot_cash_outflows_inv_act'	#投资活动现金流出小计
'net_cash_flows_inv_act'	#投资活动产生的现金流量净额
'cash_recip_cap_contrib'	#吸收投资收到的现金
'cash_recip_borrow'	#取得借款收到的现金
'proc_issue_bonds'	#发行债券收到的现金
'other_cash_recip_ral_fnc_act'	#收到其他与筹资活动有关的现金
'stot_cash_inflows_fnc_act'	#筹资活动现金流入小计
'cash_prepay_amt_borr'	#偿还债务支付现金
'cash_pay_dist_dpcp_int_exp'	#分配股利、利润或偿付利息支付的现金
'other_cash_pay_ral_fnc_act'	#支付其他与筹资的现金
'stot_cash_outflows_fnc_act'	#筹资活动现金流出小计
'net_cash_flows_fnc_act'	#筹资活动产生的现金流量净额
'eff_fx_flu_cash'	#汇率变动对现金的影响
'net_incr_cash_cash_equ'	#现金及现金等价物净增加额
'cash_cash_equ_beg_period'	#期初现金及现金等价物余额
'cash_cash_equ_end_period'	#期末现金及现金等价物余额
'net_profit'	#净利润
'plus_prov_depr_assets'	#资产减值准备
'depr_fa_coga_dpba'	#固定资产折旧、油气资产折耗、生产性物资折旧
'amort_intang_assets'	#无形资产摊销
'amort_lt_deferred_exp'	#长期待摊费用摊销
'decr_deferred_exp'	#待摊费用的减少
'incr_acc_exp'	#预提费用的增加
'loss_disp_fiolta'	#处置固定资产、无形资产和其他长期资产的损失
'loss_scr_fa'	#固定资产报废损失
'loss_fv_chg'	#公允价值变动损失
'fin_exp'	#财务费用
'invest_loss'	#投资损失
'decr_deferred_inc_tax_assets'	#递延所得税资产减少
'incr_deferred_inc_tax_liab'	#递延所得税负债增加
'decr_inventories'	#存货的减少
'decr_oper_payable'	#经营性应收项目的减少
'others'	#其他
'im_net_cash_flows_oper_act'	#经营活动产生现金流量净额
'conv_debt_into_cap'	#债务转为资本
'conv_corp_bonds_due_within_1y'	#一年内到期的可转换公司债券
'fa_fnc_leases'	#融资租入固定资产
'end_bal_cash'	#现金的期末余额

'less_beg_bal_cash'	#现金的期初余额
'plus_end_bal_cash_equ'	#现金等价物的期末余额
'less_beg_bal_cash_equ'	#现金等价物的期初余额
'im_net_incr_cash_cash_equ'	#现金及现金等价物的净增加额
'tax_levy_refund'	#收到的税费返还

PershareIndex - 主要指标

's_fa_ocfps'	#每股经营活动现金流量
's_fa_bps'	#每股净资产
's_fa_eps_basic'	#基本每股收益
's_fa_eps_diluted'	#稀释每股收益
's_fa_undistributedps'	#每股未分配利润
's_fa_surpluscapitalps'	#每股资本公积金
'adjusted_earnings_per_share'	#扣非每股收益
'du_return_on_equity'	#净资产收益率
'sales_gross_profit'	#销售毛利率
'inc_revenue_rate'	#主营收入同比增长
'du_profit_rate'	#净利润同比增长
'inc_net_profit_rate'	#归属于母公司所有者的净利润同比增长
'adjusted_net_profit_rate'	#扣非净利润同比增长
'inc_total_revenue_annual'	#营业总收入滚动环比增长
'inc_net_profit_to_shareholders_annual'	#归属净利润滚动环比增长
'adjusted_profit_to_profit_annual'	#扣非净利润滚动环比增长
'equity_roe'	#加权净资产收益率
'net_roe'	#摊薄净资产收益率
'total_roe'	#摊薄总资产收益率
'gross_profit'	#毛利率
'net_profit'	#净利率
'actual_tax_rate'	#实际税率
'pre_pay_operate_income'	#预收款 / 营业收入
'sales_cash_flow'	#销售现金流 / 营业收入
'gear_ratio'	#资产负债比率
'inventory_turnover'	#存货周转率
'm_anntime'	#公告日
'm_timetag'	#报告截止日

CapitalStructure - 股本表

'total_capital'	#总股本
'circulating_capital'	#已上市流通A股
'restrict_circulating_capital'	#限售流通股份
'm_timetag'	#报告截止日
'm_anntime'	#公告日

代码示例

时间戳转换

```
import time
def conv_time(ct):
    '''
    conv_time(1476374400000) --> '20161014000000.000'
    '''
    local_time = time.localtime(ct / 1000)
    data_head = time.strftime('%Y%m%d%H%M%S', local_time)
    data_secs = (ct - int(ct)) * 1000
    time_stamp = '%s.%03d' % (data_head, data_secs)
    return time_stamp
```