# Pytdx - Python tdx数据接口

## 概述

Pytdx 是一款纯Python语言开发的类似TradeX的行情数据接口的实现。

## 特点

* 纯python实现，无须引入动态连接库.dll/.so文件
* 支持python2.7+/3.5+， 以及全平台Windows/MacOS/Linux
* 可以通过设置参数提供线程安全接口调用
* 实现心跳包机制,可以在长时间没有交互的情况下保持不断线
* (试验）支持多连接构成的连接池机制，和failover处理机制，保证稳定性。

pytdx将重点放在底层包的解析，对于更高层的封装，建议参考或者直接使用[Quantaxis](https://github.com/yutiansut/QUANTAXIS/blob/master/QUANTAXIS/QAFetch/QATdx.py)的实现..

## 安装

pip install pytdx

## 接口实现

### 标准行情 pytdx.hq

用于读取标准行情信息

### 扩展行情 pytdx.exhq

用于读取扩展行情（外盘，期权，期货等）

### 数据文件读取 pytdx.reader

用于读取导出的k线数据

### pytdx.pool (试验性质)

用于实现备用连接池以及failover支持的行情接口

## 命令行

我们提供了方便命令行调试和导出数据的命令行工具 hqget 以及hqreader 具体使用方法请参考这里。

## 缘起

因为之前TradeX的接口是使用Python扩展的方式调用C++代码实现的，功能上有诸多的限制，如只支持32位的Python， 不支持MacOS, Linux等， 无奈我自己使用的电脑是Mac系统, 服务器又是基于Linux的，所以只能自己重新实现一份。

## 声明

次代码用于个人对网络协议的研究和习作，不对外提供服务，任何人使用本代码遇到问题请自行解决，也可以在github提issue给我，但是我不保证能即时处理。 由于我们连接的是既有的行情软件兼容行情服务器，机构请不要使用次代码，对此造成的任何问题本人概不负责。

## 其它

欢迎对量化交易感兴趣的朋友互相交流，可以来我们的智矿社区看看 [http://zhikuang.org](http://zhikuang.org/)

# 安装pytdx

## 普通安装命令

> pip install pytdx

## 升级到最新版本

> pip install -U pytdx

## 安装github上的最新版本

> pip install git+https://github.com/rainx/pytdx

或

> git clone https://github.com/rainx/pytdx.git

> cd pytdx

> python setup.py install

## 说明

根据您的python和使用的操作系统版本的不同，上述命令里的 pip, python 可能需要变为pip3, python3, 如果需要全局安装，部分Linux,MacOS 系统需要使用 sudo命令。

# 接口API

pytdx 实现了读取行情和数据文件等不同的接口，您可以使用 ipython 通过交互的方式进行调用，也可以在您的python脚本程序中使用。

# 行情接口API

下面是如何在程序里面调用本接口

首先需要引入

from pytdx.hq import TdxHq\_API

然后，创建对象

api = TdxHq\_API()

之后，通常是如下的格式

if api.connect('119.147.212.81', 7709):

# ... same codes...

api.disconnect()

当然，我们也支持with 语法,可以省略disconnect()语句

with api.connect('119.147.212.81', 7709):

# some codes

我们的数据获取届接口一般返回list结构，如果需要转化为pandas Dataframe接口，可以使用 api.to\_df 进行转化 如：

data = api.get\_security\_bars(9, 0, '000001', 0, 10) #返回普通list

data = api.to\_df(api.get\_security\_bars(9, 0, '000001', 0, 10)) # 返回DataFrame

可以使用的api方法有下列的几个。

## api方法列表

### 参数一般性约定

一般来说，股票代码和文件名称使用字符串类型，其它参数都使用数值类型

### 1 : 获取股票行情

可以获取多只股票的行情信息

需要传入一个列表，每个列表由一个市场代码， 一个股票代码构成的元祖构成 [ (市场代码1， 股票代码1)，(市场代码2， 股票代码2) ... (市场代码n， 股票代码n) ]

如：

api.get\_security\_quotes([(0, '000001'), (1, '600300')])

### 2 : 获取k线

* category->

K线种类

0 5分钟K线 1 15分钟K线 2 30分钟K线 3 1小时K线 4 日K线

5 周K线

6 月K线

7 1分钟

8 1分钟K线 9 日K线

10 季K线

11 年K线

* market -> 市场代码 0:深圳，1:上海
* stockcode -> 证券代码;
* start -> 指定的范围开始位置;
* count -> 用户要请求的 K 线数目，最大值为 800

如：

api.get\_security\_bars(9,0, '000001', 4, 3)

### 3 : 获取市场股票数量

0 - 深圳， 1 - 上海

api.get\_security\_count(0)

### 4 : 获取股票列表

参数：市场代码, 起始位置 如： 0,0 或 1,100

api.get\_security\_list(1, 0)

### 5 : 获取指数k线

* category->

K线种类

0 5分钟K线 1 15分钟K线 2 30分钟K线 3 1小时K线 4 日K线

5 周K线

6 月K线

7 1分钟

8 1分钟K线 9 日K线

10 季K线

11 年K线

* market -> 市场代码 0:深圳，1:上海
* stockcode -> 证券代码;
* start -> 指定的范围开始位置;
* count -> 用户要请求的 K 线数目，最大值为 800

如：

api.get\_index\_bars(9,1, '000001', 1, 2)

### 6 : 查询分时行情

参数：市场代码， 股票代码， 如： 0,000001 或 1,600300

api.get\_minute\_time\_data(1, '600300')

### 7 : 查询历史分时行情

参数：市场代码， 股票代码，时间 如： 0,000001,20161209 或 1,600300,20161209

api.get\_history\_minute\_time\_data(TDXParams.MARKET\_SH, '600300', 20161209)

注意，在引入 TDXParams 之后， （from pytdx.params import TDXParams） 我们可以使用 TDXParams.MARKET\_SH , TDXParams.MARKET\_SZ 常量来代替 1 和 0 作为参数

### 8 : 查询分笔成交

参数：市场代码， 股票代码，起始位置， 数量 如： 0,000001,0,10

api.get\_transaction\_data(TDXParams.MARKET\_SZ, '000001', 0, 30)

### 9 : 查询历史分笔成交

参数：市场代码， 股票代码，起始位置，日期 数量 如： 0,000001,0,10,20170209

api.get\_history\_transaction\_data(TDXParams.MARKET\_SZ, '000001', 0, 10, 20170209)

### 10 : 查询公司信息目录

参数：市场代码， 股票代码， 如： 0,000001 或 1,600300

api.get\_company\_info\_category(TDXParams.MARKET\_SZ, '000001')

### 11 : 读取公司信息详情

参数：市场代码， 股票代码, 文件名, 起始位置， 数量, 如：0,000001,000001.txt,2054363,9221

api.get\_company\_info\_content(0, '000001', '000001.txt', 0, 100)

注意这里的 起始位置， 数量 参考上面接口的返回结果。

### 12 : 读取除权除息信息

参数：市场代码， 股票代码， 如： 0,000001 或 1,600300

api.get\_xdxr\_info(1, '600300')

### 13 : 读取财务信息

参数：市场代码， 股票代码， 如： 0,000001 或 1,600300

api.get\_finance\_info(0, '000001')

### 14 : 读取k线信息

参数：市场代码， 开始时间， 结束时间

get\_k\_data('000001','2017-07-03','2017-07-10')

参考 <https://github.com/rainx/pytdx/issues/5>

### 15 ：读取板块信息

参数： 板块文件名称，可以取的值限于

# 板块相关参数

BLOCK\_SZ = "block\_zs.dat"

BLOCK\_FG = "block\_fg.dat"

BLOCK\_GN = "block\_gn.dat"

BLOCK\_DEFAULT = "block.dat"

api.get\_and\_parse\_block\_info("block.dat")

# 或者用我们定义好的params

api.get\_and\_parse\_block\_info(TDXParams.BLOCK\_SZ)

## 多线程支持

由于Python的特性，一般情况下，不太建议使用多线程代码，如果需要并发访问，建议使用多进程来实现，如果要使用多线程版本，请在初始化时设置multithread参数为True

api = TdxHq\_API(multithread=True)

## 心跳包

由于长时间不与服务器交互，服务器将关闭连接，所以我们实现了心跳包的机制，可以通过

api = TdxHq\_API(heartbeat=True)

设置心跳包，程序会启动一个心跳包发送线程，在空闲状态下隔一段时间发送一个心跳包，注意，打开heartbeat=True选项的同时会自动打开multithread=True

## 抛出异常

我们的错误处理有两套机制，根据TdxHq\_API 构造函数里的 raise\_exception 参数来确定，如果

# 默认情况

api = TdxHq\_API(raise\_exception=False)

如果在调用connect 的时候，失败会返回false, 调用普通接口时候，如果出错的情况返回None

如果

api = TdxHq\_API(raise\_exception=True)

如果在调用connect 的时候，失败会抛出TdxConnectionError异常, 调用普通接口时候，如果出错的情况抛出TdxFunctionCallError异常

## 重连机制

在调用函数的时候，如果服务器连接断开或者其它的异常情况下，为了保证在偶发的连接断开下自动重连并重新请求数据。关于重试的周期和次数，我们通过一个自定义的类实现，你可以实现自己的重试策略

如果开启的话，需要

api = TdxHq\_API(auto\_retry=True)

下面是我们默认的重试策略

class DefaultRetryStrategy(RetryStrategy):

"""

默认的重试策略，您可以通过写自己的重试策略替代本策略, 改策略主要实现gen方法，该方法是一个生成器，

返回下次重试的间隔时间, 单位为秒，我们会使用 time.sleep在这里同步等待之后进行重新connect,然后再重新发起

源请求，直到gen结束。

"""

@classmethod

def gen(cls):

# 默认重试4次 ... 时间间隔如下

for time\_interval in [0.1, 0.5, 1, 2]:

yield time\_interval

你可以实现自己的重试机制并替换默认的，如永远重复, 间隔1秒一次（慎用）

class MyRetryStrategy(RetryStrategy):

@classmethod

def gen(cls):

while True:

yield 1

# 然后覆盖默认的

api.retry\_strategy = MyRetryStrategy()

## 调试模式

如果您需要调试本代码，监控传输过程中的数据包传输情况，可以使用调试模式，使用方法是设定环境变量 TDX\_DEBUG 为 1 如

> TDX\_DEBUG=1 hqget -f 1

## 行情服务器列表

为了方便连接服务器，我把一些常用的服务器列表整理到到 hosts.py 文件中. 在程序中可以通过

from pytdx.config.hosts import hq\_hosts

获取列表, 列表里的数据参考了 <https://github.com/rainx/pytdx/issues/3>

## 获取流量统计信息

In [12]: api.get\_traffic\_stats()

Out[12]:

{'first\_pkg\_send\_time': datetime.datetime(2017, 9, 13, 13, 42, 3, 596519),

'recv\_bytes\_per\_second': 116.0,

'recv\_pkg\_bytes': 2759,

'recv\_pkg\_num': 18,

'send\_bytes\_per\_second': 15.0,

'send\_pkg\_bytes': 368,

'send\_pkg\_num': 9,

'total\_seconds': 23.716146}

# 扩展行情接口API

首先需要引入

from pytdx.exhq import TdxExHq\_API

然后，创建对象

api = TdxExHq\_API()

之后，通常是如下的格式

if api.connect('61.152.107.141', 7727):

# ... same codes...

api.disconnect()

当然，我们也支持with 语法,可以省略disconnect()语句

with api.connect('61.152.107.141', 7727):

# some codes

## api方法列表

### 参数一般性约定

一般来说，股票代码和文件名称使用字符串类型，其它参数都使用数值类型

### 1: 获取市场代码

可以获取该api服务器可以使用的市场列表，类别等信息

api.get\_markets()

返回结果 api.to\_df(api.get\_markets()) 一般某个服务器返回的类型比较固定，该结果可以缓存到本地或者内存中。

2017-07-31 21:22:06,067 - PYTDX - INFO - 获取市场代码

market category name short\_name

0 1 1 临时股 TP

1 4 12 郑州商品期权 OZ

2 5 12 大连商品期权 OD

3 6 12 上海商品期权 OS

4 8 12 上海个股期权 QQ

5 27 5 香港指数 FH

6 28 3 郑州商品 QZ

7 29 3 大连商品 QD

8 30 3 上海期货 QS

9 31 2 香港主板 KH

10 32 2 香港权证 KR

11 33 8 开放式基金 FU

12 34 9 货币型基金 FB

13 35 8 招商理财产品 LC

14 36 9 招商货币产品 LB

15 37 11 国际指数 FW

16 38 10 国内宏观指标 HG

17 40 11 中国概念股 CH

18 41 11 美股知名公司 MG

19 43 1 B股转H股 HB

20 44 1 股份转让 SB

21 47 3 股指期货 CZ

22 48 2 香港创业板 KG

23 49 2 香港信托基金 KT

24 54 6 国债预发行 GY

25 60 3 主力期货合约 MA

26 62 5 中证指数 ZZ

27 71 2 港股通 GH

### 2: 查询代码列表

参数， 起始位置， 获取数量

api.get\_instrument\_info(0, 100)

Demo: 

### 3: 查询市场中商品数量

api.get\_instrument\_count()

### 4: 查询五档行情

参数 市场ID，证券代码

* 市场ID可以通过 get\_markets 获得

api.get\_instrument\_quote(47, "IF1709")

### 5: 查询分时行情

参数 市场ID，证券代码

* 市场ID可以通过 get\_markets 获得

api.get\_minute\_time\_data(47, "IF1709")

### 6: 查询历史分时行情

参数 市场ID，证券代码，日期

* 市场ID可以通过 get\_markets 获得
* 日期格式 YYYYMMDD 如 20170811

api.get\_history\_minute\_time\_data(31, "00020", 20170811)

### 7: 查询k线数据

参数： K线周期， 市场ID， 证券代码，起始位置， 数量

* K线周期参考 TDXParams
* 市场ID可以通过 get\_markets 获得

api.get\_instrument\_bars(TDXParams.KLINE\_TYPE\_DAILY, 8, "10000843", 0, 100)

### 8: 查询分笔成交

参数：市场ID，证券代码

* 市场ID可以通过 get\_markets 获得

api.get\_transaction\_data(31, "00020")

注意，这个接口最多返回1800条记录, 如果有超过1800条记录的请求，我们有一个start 参数作为便宜量，可以取出超过1800条记录

如期货的数据：这个接口可以取出1800条之前的记录，数量也是1800条

api.get\_history\_transaction\_data(47, "IFL0", 20170810, start=1800)

### 9: 查询历史分笔成交

参数：市场ID，证券代码, 日期

* 市场ID可以通过 get\_markets 获得
* 日期格式 YYYYMMDD 如 20170810

api.get\_history\_transaction\_data(31, "00020", 20170810)

## 多线程支持

由于Python的特性，一般情况下，不太建议使用多线程代码，如果需要并发访问，建议使用多进程来实现，如果要使用多线程版本，请在初始化时设置multithread参数为True

api = TdxExHq\_API(multithread=True)

## 心跳包

由于长时间不与服务器交互，服务器将关闭连接，所以我们实现了心跳包的机制，可以通过

api = TdxExHq\_API(heartbeat=True)

设置心跳包，程序会启动一个心跳包发送线程，在空闲状态下隔一段时间发送一个心跳包，注意，打开heartbeat=True选项的同时会自动打开multithread=True

## 抛出异常 和 重连机制

参考 [标准行情 pytdx.hq](https://rainx.gitbooks.io/pytdx/content/pytdx_hq.html) 对应的章节

## 获取流量统计信息

In [12]: api.get\_traffic\_stats()

Out[12]:

{'first\_pkg\_send\_time': datetime.datetime(2017, 9, 13, 13, 42, 3, 596519),

'recv\_bytes\_per\_second': 116.0,

'recv\_pkg\_bytes': 2759,

'recv\_pkg\_num': 18,

'send\_bytes\_per\_second': 15.0,

'send\_pkg\_bytes': 368,

'send\_pkg\_num': 9,

'total\_seconds': 23.716146}

# Reader接口

## 读取通达信的日K线

通过下面的接口，我们可以解析通达信的日K线文件，该文件可以通过读取通达信的软件本地目录导出的数据获取，也可以从通达信的官网上下载， 如果您安装了通达信的终端，可以在安装目录下找到 vipdoc 子目录。

比如我的通达信客户端安装在 c:\new\_tdx 下，

即

* c:\new\_tdx\vipdoc\sz\lday\ 下是深圳的日k线数据
* c:\new\_tdx\vipdoc\sh\lday\ 下是上海的日k线数据

该目录下每个股票为一个文件，如 sz000001.day 为深圳的日k行情，

读取行情的接口非常简单

from pytdx.reader import TdxDailyBarReader, TdxFileNotFoundException

reader = TdxDailyBarReader()

df = reader.get\_df("/Users/rainx/tmp/vipdoc/sz/lday/sz000001.day")

df 是pandas 的DateFrame格式， 输出为：

open high low close amount volume

date

1991-12-23 27.70 27.90 27.60 27.80 3.530600e+06 127000

1991-12-24 27.90 29.30 27.00 29.05 3.050250e+06 105000

1991-12-25 29.15 30.00 29.10 29.30 6.648170e+06 226900

1991-12-26 29.30 29.30 28.00 28.00 5.370400e+06 191800

1991-12-27 28.00 28.50 28.00 28.45 5.988725e+06 210500

... ... ... ... ... ... ...

2017-06-22 9.15 9.40 9.14 9.25 1.325211e+09 142695815

2017-06-23 9.23 9.27 9.16 9.25 5.383036e+08 58400441

2017-06-26 9.26 9.40 9.26 9.30 6.637629e+08 71076995

[6031 rows x 6 columns]

# 可以通过pandas将它保存为csv 等文件, 如:

df.to\_csv("/tmp/000001.csv")

## 读取扩展行情的日线（如期货，期权，现货等）

In [1]: from pytdx.reader import TdxExHqDailyBarReader

In [2]: reader = TdxExHqDailyBarReader()

In [3]: df = reader.get\_df("/Users/rainx/Downloads/lday/29#A1801.day")

In [4]: df

Out[4]:

open high low close amount volume jiesuan

date

2017-08-07 3830.0 3936.0 3826.0 3925.0 167038 224516 3881.0

2017-08-08 3926.0 3990.0 3921.0 3951.0 188460 256984 3958.0

2017-08-09 3951.0 3997.0 3951.0 3982.0 194150 157330 3976.0

2017-08-10 3978.0 4015.0 3970.0 3995.0 206944 174878 3993.0

2017-08-11 3997.0 4017.0 3927.0 3954.0 202010 258036 3971.0

## 读取通达信的分钟K线（目前支持1，5分钟k线）

分钟线有两种格式，第一种是.1 .5 为后缀的

from pytdx.reader import TdxMinBarReader, TdxFileNotFoundException

reader = TdxMinBarReader()

df = reader.get\_df("/Users/rainx/Downloads/sh000001.5")

In [2]: df

Out[2]:

open high low close amount \

date

2015-07-09 09:35:00 3432.45 3454.14 3374.32 3423.61 6.189348e+10

2015-07-09 09:40:00 3420.56 3424.16 3395.07 3396.33 2.341652e+10

volume

date

2015-07-09 09:35:00 618934736

2015-07-09 09:40:00 234165181

还有一种为 .lc1 .lc5 后缀的

from pytdx.reader import TdxLCMinBarReader, TdxFileNotFoundException

reader = TdxLCMinBarReader()

df = reader.get\_df("/Users/rainx/Downloads/sz000001.lc5")

print(df)

open high low close amount \

date

2017-07-26 09:35:00 10.920000 10.990000 10.860000 10.940000 118572536.0

2017-07-26 09:40:00 10.929999 10.990000 10.910000 10.969999 43107384.0

2017-07-26 09:45:00 10.969999 11.050000 10.969999 11.050000 40586544.0

2017-07-26 09:50:00 11.050000 11.130000 11.010000 11.120000 100486624.0

2017-07-26 09:55:00 11.110000 11.179999 11.099999 11.179999 78094816.0

....

....

...

## 读取板块信息文件

文件位置参考： <http://blog.sina.com.cn/s/blog_623d2d280102vt8y.html>

样例代码：

# 默认扁平格式

df = BlockReader().get\_df("/Users/rainx/tmp/block\_zs.dat")

print(df)

blockname block\_type code\_index code

0 沪深300 2 0 000001

1 沪深300 2 1 000002

2 沪深300 2 2 000008

3 沪深300 2 3 000009

# 分组格式

df2 = BlockReader().get\_df("/Users/rainx/tmp/block\_zs.dat", BlockReader\_TYPE\_GROUP)

print(df2)

blockname block\_type stock\_count \

0 重点沪指 2 0

1 沪深300 2 300

2 深证成指 2 40

3 中小板指 2 100

code\_list

0

1 000001,000002,000008,000009,000060,000063,0000...

2 000001,000002,000063,000069,000100,000157,0001...

3 002001,002004,002007,002008,002010,002013,0020...

## 读取通达信的自定义板块信息文件夹

在通达信客户端备份自定义板块数据，设置-->数据维护工具-->数据备份，备份后会生出类似TdxBak\_20171011/blocknew的文件夹，然后使用如下代码读取：

# 默认扁平格式

df = CustomerBlockReader().get\_df('C:/Users/fit/Desktop/TdxBak\_20171011/blocknew')

print(df)

blockname block\_type code\_index code

0 领袖 LX 1 1600516

1 领袖 LX 2 0300678

2 领袖 LX 3 0300675

3 领袖 LX 4 1600230

4 领袖 LX 5 0002497

5 领袖 LX 6 0002460

6 领袖 LX 7 0000807

7 领袖 LX 8 1600874

#分组格式

df = CustomerBlockReader().get\_df('C:/Users/fit/Desktop/TdxBak\_20171011/blocknew', BlockReader\_TYPE\_GROUP)

print(df)

blockname block\_type stock\_count \

0 领袖 LX 20

1 核心 HX 20

2 潜力 QL 11

code\_list

0 1600516,0300678,0300675,1600230,0002497,000246...

1 1603501,0300597,0002467,0300081,0002194,000086...

2 1600686,0300648,1600476,0300036,1603066,030062...

# 行情连接池 API （实验阶段）

连接池API只针对特定的场景下使用，它还在非常早期的开发阶段

# 描述

它即可以在普通行情api上使用，也可以在扩展行情api中使用，它的基础的逻辑为： 我们需要提供一组ip，他们被划分为3类角色

* 一个正在进行通讯的主连接 我们命名它为M
* 一个热备选连接，它通过心跳包和备选服务器连接, 它为 H
* 一组备选连接池，它们周期性的重拍顺序（可能是根据连接服务器的速度），始终准备替换上面两个连接。 我们命名它们为P

当主连接P的接口调用出现问题时，我们将：

* 热备选连接H 转化为主连接M，重发主连接的请求
* 从连接池P中选出最快的一个连接，重新成为热备选连接H, 并通过心跳包维持服务器的链接
* 连接M回到备选连接池P中。

如此，保证我们的API请求的可靠性

# 例子：

from pytdx.hq import TdxHq\_API

from pytdx.pool.ippool import AvailableIPPool

from pytdx.config.hosts import hq\_hosts

import random

import logging

import pprint

ips = [(v[1], v[2]) for v in hq\_hosts]

# 获取5个随机ip作为ip池

random.shuffle(ips)

ips5 = ips[:5]

## IP 池对象

ippool = AvailableIPPool(TdxHq\_API, ips5)

## 选出M, H

primary\_ip, hot\_backup\_ip = ippool.sync\_get\_top\_n(2)

print("make pool api")

## 生成hqpool对象，第一个参数为TdxHq\_API后者 TdxExHq\_API里的一个，第二个参数为ip池对象。

api = TdxHqPool\_API(TdxHq\_API, ippool)

## connect 函数的参数为M, H 两组 (ip, port) 元组

with api.connect(primary\_ip, hot\_backup\_ip):

## 这里的借口和对应TdxHq\_API 或者 TdxExHq\_API里的一样，我们通过反射调用正确的接口

ret = api.get\_xdxr\_info(0, '000001')

print("send api call done")

pprint.pprint(ret)

# 常见问题

# 同花顺的一个爬虫 可以获取前后复权因子

<https://github.com/rainx/pytdx/issues/20> via yutiansut

import requests

import pandas as pd

import numpy as np

# 00 不复权 01前复权 02后复权

def get\_k\_data\_year(code,year,if\_fq):

data\_=[]

url='http://d.10jqka.com.cn/v2/line/hs\_%s/%s/%s.js'%(str(code),str(if\_fq),str(year))

for item in requests.get(url).text.split('\"')[3].split(';'):

data\_.append(item.split(','))

return pd.DataFrame(data\_,index=list(np.asarray(data\_).T[0]),columns=['date','open','high','low','close','volume','amount','factor'])

if \_\_name\_\_=='\_\_main\_\_':

print(get\_k\_data\_year('000001','2016','01'))

print(get\_k\_data\_year(600010,2016,'01'))

date open high low close volume amount factor

20160104 20160104 9.72 9.74 9.07 9.16 56349787 660376150.00 0.573

20160105 20160105 9.11 9.36 9.01 9.22 66326995 755531350.00 0.674

20160106 20160106 9.23 9.35 9.21 9.32 51570644 591698520.00 0.524

20160107 20160107 9.22 9.22 8.81 8.83 17476110 194869490.00 0.178

20160108 20160108 9.06 9.12 8.80 8.98 74752758 831334540.00 0.760

20160111 20160111 8.88 8.95 8.62 8.68 73201399 800683640.00 0.744

20160112 20160112 8.74 8.81 8.58 8.72 56164230 605970830.00 0.571

20160113 20160113 8.79 8.83 8.63 8.64 39170948 424371690.00 0.398

20160114 20160114 8.54 8.72 8.45 8.69 66631454 708535000.00 0.677

20160115 20160115 8.60 8.72 8.40 8.43 44820214 474908140.00 0.456

20160118 20160118 8.33 8.52 8.30 8.39 42104088 439917820.00 0.428

20160119 20160119 8.42 8.70 8.39 8.64 50110908 532074700.00 0.509

20160120 20160120 8.63 8.72 8.42 8.50 60375250 640968950.00 0.614

20160121 20160121 8.45 8.67 8.32 8.32 60614511 638127880.00 0.616

20160122 20160122 8.38 8.42 8.23 8.38 46675214 482984450.00 0.475

20160125 20160125 8.38 8.42 8.32 8.36 37643172 390734880.00 0.383

20160126 20160126 8.32 8.32 7.93 7.94 64790114 653561590.00 0.659

20160127 20160127 7.99 8.03 7.72 7.95 56903705 558510640.00 0.579

20160128 20160128 7.90 7.96 7.76 7.79 30254078 296055320.00 0.308

20160129 20160129 7.83 8.12 7.79 8.05 54443576 540544460.00 0.554

20160201 20160201 8.03 8.06 7.83 7.88 41773214 412635640.00 0.425

20160202 20160202 7.88 8.07 7.87 8.01 36910416 367360500.00 0.375

20160203 20160203 7.92 7.96 7.86 7.92 27457217 269997840.00 0.279

20160204 20160204 7.96 8.05 7.95 8.01 37309947 370586190.00 0.379

20160205 20160205 8.02 8.02 7.97 7.98 27089334 269184380.00 0.275

20160215 20160215 7.77 7.92 7.76 7.87 27849946 271173370.00 0.283

20160216 20160216 7.92 8.07 7.91 8.06 42838638 427507770.00 0.436

20160217 20160217 8.07 8.23 8.04 8.17 58516706 590538920.00 0.595

20160218 20160218 8.20 8.23 8.12 8.12 40617824 412337570.00 0.413

20160219 20160219 8.12 8.17 8.04 8.08 31889825 320939650.00 0.324

... ... ... ... ... ... ... ... ...

20161121 20161121 9.02 9.12 9.01 9.08 85024326 785475680.00 0.581

20161122 20161122 9.08 9.20 9.07 9.20 118025694 1097408710.00 0.807

20161123 20161123 9.18 9.40 9.18 9.29 175286940 1660465700.00 1.198

20161124 20161124 9.28 9.36 9.26 9.31 77974884 738580570.00 0.533

20161125 20161125 9.32 9.46 9.30 9.46 101367499 968127770.00 0.693

20161128 20161128 9.53 9.62 9.44 9.47 127968924 1240686230.00 0.875

20161129 20161129 9.43 9.54 9.39 9.46 88777923 854308660.00 0.607

20161130 20161130 9.49 9.56 9.34 9.39 102596305 985758450.00 0.701

20161201 20161201 9.41 9.47 9.39 9.44 64600437 619633750.00 0.442

20161202 20161202 9.44 9.44 9.28 9.39 82968650 790044440.00 0.567

20161205 20161205 9.34 9.38 9.25 9.30 76436570 723222050.00 0.522

20161206 20161206 9.32 9.36 9.29 9.33 60290276 571585180.00 0.412

20161207 20161207 9.32 9.33 9.25 9.32 49340476 466085500.00 0.337

20161208 20161208 9.34 9.39 9.27 9.36 67145216 638058500.00 0.459

20161209 20161209 9.34 9.59 9.32 9.49 151419920 1460778300.00 1.035

20161212 20161212 9.49 9.61 9.28 9.34 125687400 1206662620.00 0.859

20161213 20161213 9.32 9.34 9.17 9.26 64577157 607769970.00 0.441

20161214 20161214 9.26 9.35 9.24 9.24 59770574 564787520.00 0.409

20161215 20161215 9.21 9.25 9.05 9.09 82761287 767751840.00 0.566

20161216 20161216 9.08 9.13 9.05 9.09 39681397 366925350.00 0.271

20161219 20161219 9.06 9.07 9.01 9.04 49401064 453899740.00 0.338

20161220 20161220 9.04 9.04 8.92 8.95 63663837 580118170.00 0.435

20161221 20161221 8.96 9.00 8.95 9.00 36992065 338201450.00 0.253

20161222 20161222 8.99 9.00 8.95 8.98 34134124 311546860.00 0.233

20161223 20161223 8.98 8.98 8.91 8.92 38291216 348140450.00 0.262

20161226 20161226 8.90 8.97 8.86 8.96 30205896 273934290.00 0.206

20161227 20161227 8.96 8.97 8.91 8.92 26884124 244788270.00 0.184

20161228 20161228 8.92 8.95 8.88 8.90 33605509 304898610.00 0.230

20161229 20161229 8.91 8.93 8.89 8.92 33875853 307183500.00 0.232

20161230 20161230 8.92 8.94 8.90 8.94 30260736 274882700.00 0.207

[244 rows x 8 columns]

date open high low close volume amount factor

20160104 20160104 2.59 2.59 2.32 2.33 69437299 242332010.00 0.618

20160105 20160105 2.28 2.44 2.23 2.39 114924349 378947390.00 1.022

20160106 20160106 2.41 2.61 2.37 2.59 183924160 648683940.00 1.636

20160107 20160107 2.51 2.54 2.33 2.34 51144318 173416790.00 0.455

20160108 20160108 2.41 2.57 2.34 2.54 269896420 947258110.00 2.400

20160111 20160111 2.49 2.49 2.34 2.36 196359560 661641860.00 1.746

20160112 20160112 2.36 2.39 2.24 2.31 98270625 317175040.00 0.874

20160113 20160113 2.34 2.46 2.29 2.31 125690384 417415510.00 1.118

20160114 20160114 2.23 2.39 2.21 2.35 98322435 316861960.00 0.874

20160115 20160115 2.32 2.33 2.23 2.24 93528953 297474680.00 0.832

20160118 20160118 2.23 2.27 2.20 2.24 64590691 202761380.00 0.574

20160119 20160119 2.26 2.31 2.24 2.29 96680129 307322480.00 0.860

20160120 20160120 2.26 2.34 2.24 2.26 101959509 324735300.00 0.907

20160121 20160121 2.22 2.27 2.18 2.19 87420957 272307120.00 0.778

20160122 20160122 2.22 2.26 2.16 2.24 80863272 251230110.00 0.719

20160125 20160125 2.26 2.28 2.23 2.24 63713723 201200220.00 0.567

20160126 20160126 2.22 2.24 2.02 2.09 113888133 344049080.00 1.013

20160127 20160127 2.11 2.13 2.03 2.11 103701991 302859880.00 0.922

20160128 20160128 2.08 2.09 1.90 1.92 97593135 272647830.00 0.868

20160129 20160129 1.94 2.11 1.91 2.11 197683570 572679180.00 1.758

20160201 20160201 2.09 2.10 1.96 2.01 131984720 375889720.00 1.174

20160202 20160202 1.99 2.17 1.99 2.06 108175246 315234680.00 0.962

20160203 20160203 2.04 2.06 1.99 2.02 91365909 258485030.00 0.813

20160204 20160204 2.04 2.09 2.02 2.06 83281507 240504290.00 0.741

20160205 20160205 2.08 2.09 2.02 2.03 52837897 151741470.00 0.470

20160215 20160215 1.96 2.06 1.94 2.04 58809771 164621270.00 0.523

20160216 20160216 2.03 2.10 2.01 2.09 93010401 271062480.00 0.827

20160217 20160217 2.09 2.16 2.07 2.11 115949211 342859230.00 1.031

20160218 20160218 2.13 2.15 2.09 2.11 99191116 294640880.00 0.882

20160219 20160219 2.09 2.12 2.08 2.10 58237732 170913540.00 0.518

... ... ... ... ... ... ... ... ...

20161121 20161121 2.11 2.14 2.10 2.11 114557751 339795860.00 1.019

20161122 20161122 2.11 2.15 2.11 2.14 167943830 502591470.00 1.494

20161123 20161123 2.14 2.16 2.11 2.12 101154491 301503020.00 0.900

20161124 20161124 2.11 2.14 2.11 2.12 91680268 272503710.00 0.815

20161125 20161125 2.14 2.14 2.09 2.14 172758960 512616630.00 1.536

20161128 20161128 2.14 2.16 2.12 2.14 149050060 446304570.00 1.326

20161129 20161129 2.14 2.19 2.13 2.16 285636000 869041780.00 2.540

20161130 20161130 2.15 2.16 2.11 2.13 118658206 354947140.00 1.055

20161201 20161201 2.13 2.19 2.12 2.19 187748290 569961340.00 1.670

20161202 20161202 2.19 2.21 2.16 2.16 152353480 464076130.00 1.355

20161205 20161205 2.13 2.16 2.11 2.11 117903892 351497450.00 1.049

20161206 20161206 2.12 2.13 2.10 2.11 58917294 174115600.00 0.524

20161207 20161207 2.11 2.16 2.11 2.16 156476360 469002600.00 1.392

20161208 20161208 2.16 2.16 2.11 2.11 117791919 351791910.00 1.048

20161209 20161209 2.11 2.15 2.11 2.12 115871897 343863510.00 1.031

20161212 20161212 2.14 2.16 2.05 2.06 206053250 609641440.00 1.833

20161213 20161213 2.05 2.08 2.04 2.07 83568115 241203240.00 0.743

20161214 20161214 2.07 2.07 2.01 2.03 101446171 290766760.00 0.902

20161215 20161215 2.01 2.04 2.01 2.04 55941526 158843460.00 0.498

20161216 20161216 2.03 2.04 2.01 2.04 53356473 151462500.00 0.475

20161219 20161219 2.04 2.05 2.02 2.02 55753284 158820550.00 0.496

20161220 20161220 2.03 2.03 2.00 2.01 60642763 170695500.00 0.539

20161221 20161221 2.01 2.03 2.01 2.03 70177026 198392800.00 0.624

20161222 20161222 2.02 2.03 2.01 2.01 52148443 146992360.00 0.464

20161223 20161223 2.01 2.01 1.99 2.01 51380195 144042030.00 0.457

20161226 20161226 2.00 2.01 1.98 2.01 54378359 151634810.00 0.484

20161227 20161227 2.00 2.01 1.99 2.00 29679659 83301045.00 0.264

20161228 20161228 2.01 2.01 1.99 1.99 31189281 87149698.00 0.277

20161229 20161229 1.99 2.01 1.98 1.99 31461786 87826414.00 0.280

20161230 20161230 1.99 2.00 1.98 1.99 36529644 101796632.00 0.325

[244 rows x 8 columns]

# 数据的批量完整下载方式代码

<https://github.com/rainx/pytdx/issues/21> via yutiansut

看到有issue提出不知道如果下载完整的数据,关于数据位置指针的使用,给一个示例代码:

from pytdx.hq import TdxHq\_API

api=TdxHq\_API()

with api.connect():

data=[]

for i in range(10):

data+=api.get\_security\_bars(9,0,'000001',(9-i)\*800,800)

print(api.to\_df(data))

| open | close | high | low | vol | amount | year | month | day | hour | minute | datetime |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 27.70 | 27.80 | 27.90 | 27.60 | 1270.0 | 3.530600e+06 | 1991 | 12 | 23 | 15 | 0 | 1991-12-23 15:00 |
| 27.90 | 29.05 | 29.30 | 27.00 | 1050.0 | 3.050250e+06 | 1991 | 12 | 24 | 15 | 0 | 1991-12-24 15:00 |
| 29.15 | 29.30 | 30.00 | 29.10 | 2269.0 | 6.648170e+06 | 1991 | 12 | 25 | 15 | 0 | 1991-12-25 15:00 |
| 29.30 | 28.00 | 29.30 | 28.00 | 1918.0 | 5.370400e+06 | 1991 | 12 | 26 | 15 | 0 | 1991-12-26 15:00 |
| 28.00 | 28.45 | 28.50 | 28.00 | 2105.0 | 5.988725e+06 | 1991 | 12 | 27 | 15 | 0 | 1991-12-27 15:00 |
| 28.40 | 29.25 | 29.30 | 28.40 | 1116.0 | 3.264300e+06 | 1991 | 12 | 28 | 15 | 0 | 1991-12-28 15:00 |
| 29.30 | 28.80 | 29.40 | 28.80 | 1059.0 | 3.049920e+06 | 1991 | 12 | 30 | 15 | 0 | 1991-12-30 15:00 |
| 29.15 | 29.15 | 29.40 | 29.00 | 1886.0 | 5.497690e+06 | 1992 | 1 | 2 | 15 | 0 | 1992-01-02 15:00 |
| 29.20 | 29.10 | 29.30 | 29.00 | 2212.0 | 6.436920e+06 | 1992 | 1 | 3 | 15 | 0 | 1992-01-03 15:00 |
| 29.10 | 29.70 | 29.80 | 29.05 | 2634.0 | 7.822980e+06 | 1992 | 1 | 6 | 15 | 0 | 1992-01-06 15:00 |
| 29.70 | 29.90 | 30.20 | 29.55 | 2931.0 | 8.763690e+06 | 1992 | 1 | 7 | 15 | 0 | 1992-01-07 15:00 |
| 29.80 | 29.65 | 30.20 | 29.55 | 1928.0 | 5.716520e+06 | 1992 | 1 | 8 | 15 | 0 | 1992-01-08 15:00 |
| 29.70 | 29.70 | 29.80 | 29.30 | 1535.0 | 4.558950e+06 | 1992 | 1 | 9 | 15 | 0 | 1992-01-09 15:00 |
| 29.65 | 29.80 | 29.80 | 29.50 | 1525.0 | 4.544500e+06 | 1992 | 1 | 10 | 15 | 0 | 1992-01-10 15:00 |
| 29.80 | 29.65 | 29.80 | 29.60 | 1353.0 | 4.011645e+06 | 1992 | 1 | 13 | 15 | 0 | 1992-01-13 15:00 |
| 29.65 | 29.20 | 29.65 | 29.00 | 1711.0 | 4.996120e+06 | 1992 | 1 | 14 | 15 | 0 | 1992-01-14 15:00 |
| 29.20 | 29.10 | 29.40 | 29.00 | 2205.0 | 6.416550e+06 | 1992 | 1 | 15 | 15 | 0 | 1992-01-15 15:00 |
| 29.00 | 28.90 | 29.00 | 28.55 | 1268.0 | 3.664520e+06 | 1992 | 1 | 16 | 15 | 0 | 1992-01-16 15:00 |
| 28.90 | 28.80 | 28.90 | 28.35 | 1143.0 | 3.291840e+06 | 1992 | 1 | 17 | 15 | 0 | 1992-01-17 15:00 |
| 28.80 | 29.45 | 29.50 | 28.80 | 1085.0 | 3.195325e+06 | 1992 | 1 | 20 | 15 | 0 | 1992-01-20 15:00 |
| 29.50 | 29.30 | 29.80 | 29.20 | 1576.0 | 4.617680e+06 | 1992 | 1 | 21 | 15 | 0 | 1992-01-21 15:00 |
| 29.25 | 28.95 | 29.30 | 28.80 | 1592.0 | 4.608840e+06 | 1992 | 1 | 22 | 15 | 0 | 1992-01-22 15:00 |
| 28.95 | 29.20 | 29.30 | 28.95 | 1477.0 | 4.312840e+06 | 1992 | 1 | 23 | 15 | 0 | 1992-01-23 15:00 |
| 29.25 | 29.05 | 29.70 | 29.05 | 2327.0 | 6.759935e+06 | 1992 | 1 | 24 | 15 | 0 | 1992-01-24 15:00 |
| 29.55 | 29.40 | 29.60 | 29.30 | 2571.0 | 7.558740e+06 | 1992 | 1 | 27 | 15 | 0 | 1992-01-27 15:00 |
| 29.40 | 29.50 | 29.50 | 29.35 | 2336.0 | 6.891200e+06 | 1992 | 1 | 28 | 15 | 0 | 1992-01-28 15:00 |
| 29.50 | 30.00 | 30.05 | 29.50 | 2127.0 | 6.381000e+06 | 1992 | 1 | 29 | 15 | 0 | 1992-01-29 15:00 |
| 29.95 | 30.20 | 30.35 | 29.95 | 1735.0 | 5.239700e+06 | 1992 | 1 | 30 | 15 | 0 | 1992-01-30 15:00 |
| 31.15 | 32.00 | 32.00 | 31.15 | 2228.0 | 7.129600e+06 | 1992 | 2 | 1 | 15 | 0 | 1992-02-01 15:00 |
| 32.00 | 32.32 | 32.70 | 31.90 | 1735.0 | 5.607520e+06 | 1992 | 2 | 2 | 15 | 0 | 1992-02-02 15:00 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 9.30 | 9.36 | 9.39 | 9.27 | 546016.0 | 5.091620e+08 | 2017 | 6 | 27 | 15 | 0 | 2017-06-27 15:00 |
| 9.35 | 9.43 | 9.49 | 9.33 | 1168796.0 | 1.102438e+09 | 2017 | 6 | 28 | 15 | 0 | 2017-06-28 15:00 |
| 9.43 | 9.43 | 9.45 | 9.37 | 488804.0 | 4.598104e+08 | 2017 | 6 | 29 | 15 | 0 | 2017-06-29 15:00 |
| 9.40 | 9.39 | 9.43 | 9.31 | 499633.0 | 4.680035e+08 | 2017 | 6 | 30 | 15 | 0 | 2017-06-30 15:00 |
| 9.40 | 9.40 | 9.43 | 9.34 | 388349.0 | 3.644659e+08 | 2017 | 7 | 3 | 15 | 0 | 2017-07-03 15:00 |
| 9.40 | 9.34 | 9.41 | 9.30 | 488362.0 | 4.565770e+08 | 2017 | 7 | 4 | 15 | 0 | 2017-07-04 15:00 |
| 9.29 | 9.37 | 9.38 | 9.27 | 567720.0 | 5.292941e+08 | 2017 | 7 | 5 | 15 | 0 | 2017-07-05 15:00 |
| 9.36 | 9.40 | 9.41 | 9.31 | 738911.0 | 6.913872e+08 | 2017 | 7 | 6 | 15 | 0 | 2017-07-06 15:00 |
| 9.37 | 9.47 | 9.48 | 9.34 | 760369.0 | 7.170844e+08 | 2017 | 7 | 7 | 15 | 0 | 2017-07-07 15:00 |
| 9.45 | 9.59 | 9.66 | 9.44 | 1360815.0 | 1.303090e+09 | 2017 | 7 | 10 | 15 | 0 | 2017-07-10 15:00 |
| 9.61 | 10.25 | 10.46 | 9.61 | 3812086.0 | 3.842010e+09 | 2017 | 7 | 11 | 15 | 0 | 2017-07-11 15:00 |
| 10.27 | 10.34 | 10.58 | 10.20 | 2998844.0 | 3.113681e+09 | 2017 | 7 | 12 | 15 | 0 | 2017-07-12 15:00 |
| 10.30 | 10.90 | 10.90 | 10.24 | 2994534.0 | 3.180145e+09 | 2017 | 7 | 13 | 15 | 0 | 2017-07-13 15:00 |
| 10.81 | 10.90 | 10.94 | 10.66 | 1722570.0 | 1.864449e+09 | 2017 | 7 | 14 | 15 | 0 | 2017-07-14 15:00 |
| 10.95 | 10.81 | 11.33 | 10.72 | 3273123.0 | 3.608692e+09 | 2017 | 7 | 17 | 15 | 0 | 2017-07-17 15:00 |
| 10.75 | 11.05 | 11.14 | 10.62 | 2349431.0 | 2.558434e+09 | 2017 | 7 | 18 | 15 | 0 | 2017-07-18 15:00 |
| 10.99 | 11.09 | 11.19 | 10.88 | 1933075.0 | 2.131336e+09 | 2017 | 7 | 19 | 15 | 0 | 2017-07-19 15:00 |
| 11.08 | 10.97 | 11.22 | 10.91 | 1537338.0 | 1.695061e+09 | 2017 | 7 | 20 | 15 | 0 | 2017-07-20 15:00 |
| 10.83 | 10.89 | 10.95 | 10.69 | 1501020.0 | 1.625416e+09 | 2017 | 7 | 21 | 15 | 0 | 2017-07-21 15:00 |
| 10.82 | 10.95 | 11.06 | 10.73 | 1692664.0 | 1.846887e+09 | 2017 | 7 | 24 | 15 | 0 | 2017-07-24 15:00 |
| 10.98 | 11.00 | 11.27 | 10.95 | 1954768.0 | 2.172115e+09 | 2017 | 7 | 25 | 15 | 0 | 2017-07-25 15:00 |
| 10.92 | 10.74 | 11.18 | 10.66 | 1697412.0 | 1.846282e+09 | 2017 | 7 | 26 | 15 | 0 | 2017-07-26 15:00 |
| 10.72 | 10.59 | 10.77 | 10.53 | 1194490.0 | 1.273889e+09 | 2017 | 7 | 27 | 15 | 0 | 2017-07-27 15:00 |
| 10.61 | 10.74 | 10.81 | 10.58 | 819195.0 | 8.777693e+08 | 2017 | 7 | 28 | 15 | 0 | 2017-07-28 15:00 |
| 10.80 | 10.67 | 10.82 | 10.45 | 1575864.0 | 1.671814e+09 | 2017 | 7 | 31 | 15 | 0 | 2017-07-31 15:00 |
| 10.64 | 11.04 | 11.08 | 10.60 | 2035709.0 | 2.222888e+09 | 2017 | 8 | 1 | 15 | 0 | 2017-08-01 15:00 |
| 11.05 | 11.15 | 11.34 | 10.96 | 2062069.0 | 2.307727e+09 | 2017 | 8 | 2 | 15 | 0 | 2017-08-02 15:00 |
| 11.14 | 11.01 | 11.22 | 10.97 | 984219.0 | 1.090954e+09 | 2017 | 8 | 3 | 15 | 0 | 2017-08-03 15:00 |
| 11.00 | 11.17 | 11.29 | 10.93 | 1353951.0 | 1.511390e+09 | 2017 | 8 | 4 | 15 | 0 | 2017-08-04 15:00 |
| 11.06 | 11.00 | 11.17 | 10.90 | 860644.0 | 9.469757e+08 | 2017 | 8 | 7 | 15 | 0 | 2017-08-07 15:00 |

这么做的原因很简单,改变指针的位置

分别是

0-799

800-1599

1600-2399

....

依次

然后需要注意的是 0代表的是今天的指针 ,所以 指针获取方式要翻过来写 首先是 7200-7999的数据

最后才是0-799的数据

基本上 日线级别 8000条足够覆盖了

分钟线,小时线要长一点

封装成函数

from pytdx.hq import TdxHq\_API

api=TdxHq\_API()

def get\_all\_day\_data():

with api.connect():

data=[]

for i in range(10):

data+=api.get\_security\_bars(9,0,'000001',(9-i)\*800,800)

print(api.to\_df(data))

# 基金价格问题

<https://github.com/rainx/pytdx/issues/16>

get\_minute\_time\_data，get\_transaction\_data，get\_security\_quotes这些函数如果传入基金代码，返回的价格都大了10倍。

这个是通达信的行情数据本身就给乘了10倍吗？

wopalm：

在证券的清单中，有"价格小数位数"的信息，可以看到，159922的小数位数是3位。pytdx证券清单的函数，舍去了这个信息，建议加上。返回的数据格式如下

在 get\_security\_list 接口中增加了 小数点位数 (decimal\_point)

如：

In [11]: api.to\_df(api.get\_security\_list(market=0, start=800))

Out[11]:

code volunit decimal\_point name pre\_close

0 000978 100 2 桂林旅游 312.022500

1 000979 100 2 中弘股份 1.960000

2 000980 100 2 众泰汽车 616.005000

3 000981 100 2 银亿股份 488.015000

4 000982 100 2 \*ST 中绒 4.070000

5 000983 100 2 西山煤电 632.052500

6 000985 100 2 大庆华科 24.670000

7 000987 100 2 越秀金控 920.027500

8 000988 100 2 华工科技 1864.020000

9 000989 100 2 九 芝 堂 19.299999

10 000990 100 2 诚志股份 17.270000

11 000993 100 2 闽东电力 536.027500

12 000995 100 2 皇台酒业 1608.030000

13 000996 100 2 中国中期 1304.027500

14 000997 100 2 新 大 陆 23.500000

15 000998 100 2 隆平高科 23.799999

16 000999 100 2 华润三九 27.820000

17 001696 100 2 宗申动力 7.340000

18 001896 100 2 豫能控股 6.690000

19 001979 100 2 招商蛇口 19.180000

20 002001 100 2 新 和 成 20.260000

21 002002 100 2 鸿达兴业 200.000000

22 002003 100 2 伟星股份 712.030000

23 002004 100 2 华邦健康 8.040000

24 002005 100 2 德豪润达 4.680000

25 002006 100 2 精功科技 7.910000

26 002007 100 2 华兰生物 29.000000

27 002008 100 2 大族激光 64.169998

28 002009 100 2 天奇股份 1224.050000

29 002010 100 2 传化智联 1896.005000

.. ... ... ... ... ...

970 101301 10 3 国债1301 103.000000

971 101303 10 3 国债1303 99.580002

972 101305 10 3 国债1305 100.000000

973 101308 10 3 国债1308 99.099998

974 101309 10 3 国债1309 100.000000

975 101310 10 3 国债1310 100.000000

976 101311 10 3 国债1311 98.626999

977 101313 10 3 国债1313 99.290001

978 101315 10 3 国债1315 99.011002

979 101316 10 3 国债1316 100.000000

980 101318 10 3 国债1318 100.000000

981 101319 10 3 国债1319 100.000000

982 101320 10 3 国债1320 100.500000

983 101323 10 3 国债1323 100.000000

984 101324 10 3 国债1324 100.000000

985 101325 10 3 国债1325 100.000000

986 101401 10 3 国债1401 100.000000

987 101403 10 3 国债1403 101.540001

988 101405 10 3 国债1405 100.000000

989 101406 10 3 国债1406 100.000000

990 101408 10 3 国债1408 100.000000

991 101409 10 3 国债1409 100.000000

992 101410 10 3 国债1410 100.000000

993 101412 10 3 国债1412 99.500000

994 101413 10 3 国债1413 101.690002

995 101416 10 3 国债1416 100.000000

996 101417 10 3 国债1417 100.000000

997 101420 10 3 国债1420 100.000000

998 101421 10 3 国债1421 100.000000

999 101424 10 3 国债1424 100.000000

# 股票数据下载的封装(已完成) [日线/分钟线/实时价格/股票列/指数/etf/历史分笔/除权除息/复权]

<https://github.com/rainx/pytdx/issues/78> by yutiansut

# coding:utf-8

#

# The MIT License (MIT)

#

# Copyright (c) 2016-2017 yutiansut/QUANTAXIS

#

# Permission is hereby granted, free of charge, to any person obtaining a copy

# of this software and associated documentation files (the "Software"), to deal

# in the Software without restriction, including without limitation the rights

# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

# copies of the Software, and to permit persons to whom the Software is

# furnished to do so, subject to the following conditions:

#

# The above copyright notice and this permission notice shall be included in all

# copies or substantial portions of the Software.

#

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,

# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE

# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER

# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,

# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

# SOFTWARE.

import datetime

import numpy as np

import pandas as pd

from pytdx.hq import TdxHq\_API

from pytdx.exhq import TdxExHq\_API

from QUANTAXIS.QAUtil import (QA\_util\_date\_stamp, QA\_util\_date\_str2int,

QA\_util\_date\_valid, QA\_util\_get\_real\_date,

QA\_util\_get\_real\_datelist, QA\_util\_log\_info,

QA\_util\_time\_stamp, QA\_util\_web\_ping,

trade\_date\_sse)

#from pypinyin import lazy\_pinyin

import tushare as ts

# 基于Pytdx的数据接口,好处是可以在linux/mac上联入通达信行情

# 具体参见rainx的pytdx(https://github.com/rainx/pytdx)

#

def ping(ip):

api = TdxHq\_API()

\_\_time1 = datetime.datetime.now()

try:

with api.connect(ip, 7709):

if len(api.get\_security\_list(0, 1)) > 800:

return datetime.datetime.now() - \_\_time1

except:

print('Bad REPSONSE %s' % ip)

return datetime.timedelta(9, 9, 0)

def select\_best\_ip():

QA\_util\_log\_info('Selecting the Best Server IP of TDX')

listx = ['218.75.126.9', '115.238.90.165',

'124.160.88.183', '60.12.136.250', '218.108.98.244', '218.108.47.69',

'14.17.75.71', '180.153.39.51']

data = [ping(x) for x in listx]

QA\_util\_log\_info('===The BEST SERVER is : %s ===' %

(listx[data.index(min(data))]))

return listx[data.index(min(data))]

best\_ip = select\_best\_ip()

# return 1 if sh, 0 if sz

def \_\_select\_market\_code(code):

code = str(code)

if code[0] in ['5', '6', '9'] or code[:3] in ["009", "126", "110", "201", "202", "203", "204"]:

return 1

return 0

def QA\_fetch\_get\_stock\_day(code, start\_date, end\_date, if\_fq='00', level='day', ip=best\_ip, port=7709):

api = TdxHq\_API()

with api.connect(ip, port):

if level in ['day', 'd', 'D', 'DAY', 'Day']:

level = 9

elif level in ['w', 'W', 'Week', 'week']:

level = 5

elif level in ['month', 'M', 'm', 'Month']:

level = 6

elif level in ['Q', 'Quarter', 'q']:

level = 10

elif level in ['y', 'Y', 'year', 'Year']:

level = 11

data = pd.concat([api.to\_df(api.get\_security\_bars(level, \_\_select\_market\_code(

code), code, (9 - i) \* 800, 800)) for i in range(10)], axis=0)

if if\_fq in ['00', 'bfq']:

data = data.assign(date=data['datetime'].apply(lambda x: str(x[0:10]))).assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)[start\_date:end\_date]

return data.assign(date=data['date'].apply(lambda x: str(x)[0:10]))

elif if\_fq in ['01', 'qfq']:

xdxr\_data = QA\_fetch\_get\_stock\_xdxr(code)

info = xdxr\_data[xdxr\_data['category'] == 1]

bfq\_data = data\

.assign(date=pd.to\_datetime(data['datetime'].apply(lambda x: x[0:10])))\

.assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)

bfq\_data['if\_trade'] = True

data = pd.concat([bfq\_data, info[['category']]

[bfq\_data.index[0]:]], axis=1)

data['date'] = data.index

data['if\_trade'].fillna(value=False, inplace=True)

data = data.fillna(method='ffill')

data = pd.concat([data, info[['fenhong', 'peigu', 'peigujia',

'songzhuangu']][bfq\_data.index[0]:]], axis=1)

data = data.fillna(0)

data['preclose'] = (data['close'].shift(1) \* 10 - data['fenhong'] + data['peigu']

\* data['peigujia']) / (10 + data['peigu'] + data['songzhuangu'])

data['adj'] = (data['preclose'].shift(-1) /

data['close']).fillna(1)[::-1].cumprod()

data['open'] = data['open'] \* data['adj']

data['high'] = data['high'] \* data['adj']

data['low'] = data['low'] \* data['adj']

data['close'] = data['close'] \* data['adj']

data['preclose'] = data['preclose'] \* data['adj']

data = data[data['if\_trade']]

return data.drop(['fenhong', 'peigu', 'peigujia', 'songzhuangu', 'if\_trade', 'category'], axis=1)[data['open'] != 0].assign(date=data['date'].apply(lambda x: str(x)[0:10]))[start\_date:end\_date]

elif if\_fq in ['03', 'ddqfq']:

xdxr\_data = QA\_fetch\_get\_stock\_xdxr(code)

info = xdxr\_data[xdxr\_data['category'] == 1]

bfq\_data = data\

.assign(date=pd.to\_datetime(data['datetime'].apply(lambda x: x[0:10])))\

.assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)

bfq\_data['if\_trade'] = True

data = pd.concat([bfq\_data, info[['category']]

[bfq\_data.index[0]:end\_date]], axis=1)

data['date'] = data.index

data['if\_trade'].fillna(value=False, inplace=True)

data = data.fillna(method='ffill')

data = pd.concat([data, info[['fenhong', 'peigu', 'peigujia',

'songzhuangu']][bfq\_data.index[0]:end\_date]], axis=1)

data = data.fillna(0)

data['preclose'] = (data['close'].shift(1) \* 10 - data['fenhong'] + data['peigu']

\* data['peigujia']) / (10 + data['peigu'] + data['songzhuangu'])

data['adj'] = (data['preclose'].shift(-1) /

data['close']).fillna(1)[::-1].cumprod()

data['open'] = data['open'] \* data['adj']

data['high'] = data['high'] \* data['adj']

data['low'] = data['low'] \* data['adj']

data['close'] = data['close'] \* data['adj']

data['preclose'] = data['preclose'] \* data['adj']

data = data[data['if\_trade']]

return data.drop(['fenhong', 'peigu', 'peigujia', 'songzhuangu', 'if\_trade', 'category'], axis=1)[data['open'] != 0].assign(date=data['date'].apply(lambda x: str(x)[0:10]))[start\_date:end\_date]

elif if\_fq in ['02', 'hfq']:

xdxr\_data = QA\_fetch\_get\_stock\_xdxr(code)

info = xdxr\_data[xdxr\_data['category'] == 1]

bfq\_data = data\

.assign(date=pd.to\_datetime(data['datetime'].apply(lambda x: x[0:10])))\

.assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)

bfq\_data['if\_trade'] = True

data = pd.concat([bfq\_data, info[['category']]

[bfq\_data.index[0]:]], axis=1)

data['date'] = data.index

data['if\_trade'].fillna(value=False, inplace=True)

data = data.fillna(method='ffill')

data = pd.concat([data, info[['fenhong', 'peigu', 'peigujia',

'songzhuangu']][bfq\_data.index[0]:]], axis=1)

data = data.fillna(0)

data['preclose'] = (data['close'].shift(1) \* 10 - data['fenhong'] + data['peigu']

\* data['peigujia']) / (10 + data['peigu'] + data['songzhuangu'])

data['adj'] = (data['preclose'].shift(-1) /

data['close']).fillna(1).cumprod()

data['open'] = data['open'] / data['adj']

data['high'] = data['high'] / data['adj']

data['low'] = data['low'] / data['adj']

data['close'] = data['close'] / data['adj']

data['preclose'] = data['preclose'] / data['adj']

data = data[data['if\_trade']]

return data.drop(['fenhong', 'peigu', 'peigujia', 'songzhuangu', 'if\_trade', 'category'], axis=1)[data['open'] != 0].assign(date=data['date'].apply(lambda x: str(x)[0:10]))[start\_date:end\_date]

elif if\_fq in ['04', 'ddhfq']:

xdxr\_data = QA\_fetch\_get\_stock\_xdxr(code)

info = xdxr\_data[xdxr\_data['category'] == 1]

bfq\_data = data\

.assign(date=pd.to\_datetime(data['datetime'].apply(lambda x: x[0:10])))\

.assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)

bfq\_data['if\_trade'] = True

data = pd.concat([bfq\_data, info[['category']]

[bfq\_data.index[0]:end\_date]], axis=1)

data['date'] = data.index

data['if\_trade'].fillna(value=False, inplace=True)

data = data.fillna(method='ffill')

data = pd.concat([data, info[['fenhong', 'peigu', 'peigujia',

'songzhuangu']][bfq\_data.index[0]:end\_date]], axis=1)

data = data.fillna(0)

data['preclose'] = (data['close'].shift(1) \* 10 - data['fenhong'] + data['peigu']

\* data['peigujia']) / (10 + data['peigu'] + data['songzhuangu'])

data['adj'] = (data['preclose'].shift(-1) /

data['close']).fillna(1).cumprod()

data['open'] = data['open'] / data['adj']

data['high'] = data['high'] / data['adj']

data['low'] = data['low'] / data['adj']

data['close'] = data['close'] / data['adj']

data['preclose'] = data['preclose'] / data['adj']

data = data[data['if\_trade']]

return data.drop(['fenhong', 'peigu', 'peigujia', 'songzhuangu', 'if\_trade', 'category'], axis=1)[data['open'] != 0].assign(date=data['date'].apply(lambda x: str(x)[0:10]))[start\_date:end\_date]

def QA\_fetch\_get\_stock\_min(code, start, end, level='1min', ip=best\_ip, port=7709):

api = TdxHq\_API()

type\_ = ''

if str(level) in ['5', '5m', '5min', 'five']:

level, type\_ = 0, '5min'

elif str(level) in ['1', '1m', '1min', 'one']:

level, type\_ = 8, '1min'

elif str(level) in ['15', '15m', '15min', 'fifteen']:

level, type\_ = 1, '15min'

elif str(level) in ['30', '30m', '30min', 'half']:

level, type\_ = 2, '30min'

elif str(level) in ['60', '60m', '60min', '1h']:

level, type\_ = 3, '60min'

with api.connect(ip, port):

data = pd.concat([api.to\_df(api.get\_security\_bars(level, \_\_select\_market\_code(

str(code)), str(code), (25 - i) \* 800, 800)) for i in range(26)], axis=0)

data = data\

.assign(datetime=pd.to\_datetime(data['datetime']), code=str(code))\

.drop(['year', 'month', 'day', 'hour', 'minute'], axis=1, inplace=False)\

.assign(date=data['datetime'].apply(lambda x: str(x)[0:10]))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(x)))\

.assign(time\_stamp=data['datetime'].apply(lambda x: QA\_util\_time\_stamp(x)))\

.assign(type=type\_).set\_index('datetime', drop=False, inplace=False)[start:end]

return data.assign(datetime=data['datetime'].apply(lambda x: str(x)))

def QA\_fetch\_get\_stock\_latest(code, ip=best\_ip, port=7709):

code = [code] if isinstance(code, str) else code

api = TdxHq\_API(multithread=True)

with api.connect(ip, port):

data = pd.concat([api.to\_df(api.get\_security\_bars(

9, \_\_select\_market\_code(item), item, 0, 1)).assign(code=item) for item in code], axis=0)

return data\

.assign(date=pd.to\_datetime(data['datetime']

.apply(lambda x: x[0:10])), date\_stamp=data['datetime']

.apply(lambda x: QA\_util\_date\_stamp(str(x[0:10]))))\

.set\_index('date', drop=False)\

.drop(['year', 'month', 'day', 'hour', 'minute', 'datetime'], axis=1)

def QA\_fetch\_get\_stock\_realtime(code=['000001', '000002'], ip=best\_ip, port=7709):

api = TdxHq\_API()

\_\_data = pd.DataFrame()

with api.connect(ip, port):

code = [code] if type(code) is str else code

for id\_ in range(int(len(code) / 80) + 1):

\_\_data = \_\_data.append(api.to\_df(api.get\_security\_quotes(

[(\_\_select\_market\_code(x), x) for x in code[80 \* id\_:80 \* (id\_ + 1)]])))

\_\_data['datetime'] = datetime.datetime.now()

data = \_\_data[['datetime', 'code', 'open', 'high', 'low', 'price']]

return data.set\_index('code', drop=False, inplace=False)

def QA\_fetch\_get\_stock\_list(type\_='stock', ip=best\_ip, port=7709):

api = TdxHq\_API()

with api.connect(ip, port):

data = pd.concat([pd.concat([api.to\_df(api.get\_security\_list(j, i \* 1000)).assign(sse='sz' if j == 0 else 'sh').set\_index(

['code', 'sse'], drop=False) for i in range(int(api.get\_security\_count(j) / 1000) + 1)], axis=0) for j in range(2)], axis=0)

if type\_ in ['stock', 'gp']:

return pd.concat([data[data['sse'] == 'sz'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 10000 <= 30][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 100000 != 2],

data[data['sse'] == 'sh'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 100000 == 6]]).assign(code=data['code'].apply(lambda x: str(x)))

#.assign(szm=data['name'].apply(lambda x: ''.join([y[0] for y in lazy\_pinyin(x)])))\

#.assign(quanpin=data['name'].apply(lambda x: ''.join(lazy\_pinyin(x))))

elif type\_ in ['index', 'zs']:

return pd.concat([data[data['sse'] == 'sz'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 1000 >= 399],

data[data['sse'] == 'sh'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 1000 == 0]]) \

.sort\_index()\

.assign(code=data['code'].apply(lambda x: str(x)))

#.assign(szm=data['name'].apply(lambda x: ''.join([y[0] for y in lazy\_pinyin(x)])))\

#.assign(quanpin=data['name'].apply(lambda x: ''.join(lazy\_pinyin(x))))

elif type\_ in ['etf', 'ETF']:

return pd.concat([data[data['sse'] == 'sz'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 10000 == 15],

data[data['sse'] == 'sh'][data.assign(code=data['code'].apply(lambda x: int(x)))['code'] // 10000 == 51]]).sort\_index().assign(code=data['code'].apply(lambda x: str(x)))\

#.assign(szm=data['name'].apply(lambda x: ''.join([y[0] for y in lazy\_pinyin(x)])))\

#.assign(quanpin=data['name'].apply(lambda x: ''.join(lazy\_pinyin(x))))

else:

return data.assign(code=data['code'].apply(lambda x: str(x)))

#.assign(szm=data['name'].apply(lambda x: ''.join([y[0] for y in lazy\_pinyin(x)])))\

# .assign(quanpin=data['name'].apply(lambda x: ''.join(lazy\_pinyin(x))))

def QA\_fetch\_get\_index\_day(code, start\_date, end\_date, level='day', ip=best\_ip, port=7709):

'指数日线'

api = TdxHq\_API()

if level in ['day', 'd', 'D', 'DAY', 'Day']:

level = 9

elif level in ['w', 'W', 'Week', 'week']:

level = 5

elif level in ['month', 'M', 'm', 'Month']:

level = 6

elif level in ['Q', 'Quarter', 'q']:

level = 10

elif level in ['y', 'Y', 'year', 'Year']:

level = 11

with api.connect(ip, port):

if str(code)[0] in ['5', '1']: # ETF

data = pd.concat([api.to\_df(api.get\_security\_bars(

level, 1 if str(code)[0] in ['0', '8', '9', '5'] else 0, code, (25 - i) \* 800, 800)) for i in range(26)], axis=0)

else:

data = pd.concat([api.to\_df(api.get\_index\_bars(

level, 1 if str(code)[0] in ['0', '8', '9', '5'] else 0, code, (25 - i) \* 800, 800)) for i in range(26)], axis=0)

data = data.assign(date=data['datetime'].apply(lambda x: str(x[0:10]))).assign(code=str(code))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(str(x)[0:10])))\

.set\_index('date', drop=False, inplace=False)\

.drop(['year', 'month', 'day', 'hour',

'minute', 'datetime'], axis=1)[start\_date:end\_date]

return data.assign(date=data['date'].apply(lambda x: str(x)[0:10]))

def QA\_fetch\_get\_index\_min(code, start, end, level='1min', ip=best\_ip, port=7709):

'指数分钟线'

api = TdxHq\_API()

type\_ = ''

if str(level) in ['5', '5m', '5min', 'five']:

level, type\_ = 0, '5min'

elif str(level) in ['1', '1m', '1min', 'one']:

level, type\_ = 8, '1min'

elif str(level) in ['15', '15m', '15min', 'fifteen']:

level, type\_ = 1, '15min'

elif str(level) in ['30', '30m', '30min', 'half']:

level, type\_ = 2, '30min'

elif str(level) in ['60', '60m', '60min', '1h']:

level, type\_ = 3, '60min'

with api.connect(ip, port):

if str(code)[0] in ['5', '1']: # ETF

data = pd.concat([api.to\_df(api.get\_security\_bars(

level, 1 if str(code)[0] in ['0', '8', '9', '5'] else 0, code, (25 - i) \* 800, 800)) for i in range(26)], axis=0)

else:

data = pd.concat([api.to\_df(api.get\_index\_bars(

level, 1 if str(code)[0] in ['0', '8', '9', '5'] else 0, code, (25 - i) \* 800, 800)) for i in range(26)], axis=0)

data = data\

.assign(datetime=pd.to\_datetime(data['datetime']), code=str(code))\

.drop(['year', 'month', 'day', 'hour', 'minute'], axis=1, inplace=False)\

.assign(date=data['datetime'].apply(lambda x: str(x)[0:10]))\

.assign(date\_stamp=data['datetime'].apply(lambda x: QA\_util\_date\_stamp(x)))\

.assign(time\_stamp=data['datetime'].apply(lambda x: QA\_util\_time\_stamp(x)))\

.assign(type=type\_).set\_index('datetime', drop=False, inplace=False)[start:end]

# data

return data.assign(datetime=data['datetime'].apply(lambda x: str(x)))

def \_\_QA\_fetch\_get\_stock\_transaction(code, day, retry, api):

data\_ = pd.concat([api.to\_df(api.get\_history\_transaction\_data(

\_\_select\_market\_code(str(code)), str(code), (20 - i) \* 800, 800, QA\_util\_date\_str2int(day))) for i in range(21)], axis=0)

for \_ in range(retry):

if len(data\_) < 2:

return \_\_QA\_fetch\_get\_stock\_transaction(code, day, 0, api)

else:

return data\_.assign(date=day).assign(datetime=pd.to\_datetime(data\_['time'].apply(lambda x: str(day) + ' ' + x)))\

.assign(code=str(code)).assign(order=range(len(data\_.index))).set\_index('datetime', drop=False, inplace=False)

def QA\_fetch\_get\_stock\_transaction(code, start, end, retry=2, ip=best\_ip, port=7709):

'逐笔成交'

api = TdxHq\_API()

real\_start, real\_end = QA\_util\_get\_real\_datelist(start, end)

real\_id\_range = []

with api.connect(ip, port):

data = pd.DataFrame()

for index\_ in range(trade\_date\_sse.index(real\_start), trade\_date\_sse.index(real\_end) + 1):

try:

data\_ = \_\_QA\_fetch\_get\_stock\_transaction(

code, trade\_date\_sse[index\_], retry, api)

if len(data\_) < 1:

return None

except:

QA\_util\_log\_info('Wrong in Getting %s history transaction data in day %s' % (

code, trade\_date\_sse[index\_]))

else:

QA\_util\_log\_info('Successfully Getting %s history transaction data in day %s' % (

code, trade\_date\_sse[index\_]))

data = data.append(data\_)

return data.assign(datetime=data['datetime'].apply(lambda x: str(x)[0:19]))

def QA\_fetch\_get\_stock\_xdxr(code, ip=best\_ip, port=7709):

'除权除息'

api = TdxHq\_API()

market\_code = \_\_select\_market\_code(code)

with api.connect(ip, port):

category = {

'1': '除权除息', '2': '送配股上市', '3': '非流通股上市', '4': '未知股本变动', '5': '股本变化',

'6': '增发新股', '7': '股份回购', '8': '增发新股上市', '9': '转配股上市', '10': '可转债上市',

'11': '扩缩股', '12': '非流通股缩股', '13': '送认购权证', '14': '送认沽权证'}

data = api.to\_df(api.get\_xdxr\_info(market\_code, code))

data = data\

.assign(date=pd.to\_datetime(data[['year', 'month', 'day']]))\

.drop(['year', 'month', 'day'], axis=1)\

.assign(category\_meaning=data['category'].apply(lambda x: category[str(x)]))\

.assign(code=str(code))\

.rename(index=str, columns={'panhouliutong': 'liquidity\_after',

'panqianliutong': 'liquidity\_before', 'houzongguben': 'shares\_after',

'qianzongguben': 'shares\_before'})\

.set\_index('date', drop=False, inplace=False)

return data.assign(date=data['date'].apply(lambda x: str(x)[0:10]))

def QA\_fetch\_get\_stock\_block(ip=best\_ip, port=7709):

'板块数据'

api = TdxHq\_API()

with api.connect(ip, port):

data = pd.concat([api.to\_df(api.get\_and\_parse\_block\_info("block\_gn.dat")),

api.to\_df(api.get\_and\_parse\_block\_info("block.dat")),

api.to\_df(api.get\_and\_parse\_block\_info(

"block\_zs.dat")),

api.to\_df(api.get\_and\_parse\_block\_info("block\_fg.dat"))])

if len(data) > 10:

return data.assign(source='tdx').set\_index('code', drop=False, inplace=False)

else:

QA\_util\_log\_info('Wrong with fetch block ')

def QA\_fetch\_get\_stock\_info(code, ip=best\_ip, port=7709):

'股票财务数据'

api = TdxHq\_API()

market\_code = \_\_select\_market\_code(code)

with api.connect(ip, port):

return api.to\_df(api.get\_finance\_info(market\_code, code))

if \_\_name\_\_ == '\_\_main\_\_':

# print(QA\_fetch\_get\_stock\_day('000001','2017-07-03','2017-07-10'))

# print(QA\_fetch\_get\_stock\_day('000001', '2013-07-01', '2013-07-09'))

# print(QA\_fetch\_get\_stock\_realtime('000001'))

print(QA\_fetch\_get\_index\_day('000001', '2017-01-01', '2017-07-01'))

# print(QA\_fetch\_get\_stock\_transaction('000001', '2017-07-03', '2017-07-10'))