中国银河证券星耀数智
AmazingData
开发手册

目录

1. 版本说明

1.1 文档管理信息表

主题	中国银河证券星耀数智 AmazingData 开发手册
文档版本	V1.0.24
Python SDK 版本	V1.0.24
创建时间	2025 年 7 月 10 日
最新发布日期	2025 年 12 月 16 日

2. 功能介绍

本文档是 tgw 的 SDK 开发指南，包含了对 API 接口的说明以及示例，用于指引开发人员通过 tgw 金融数据功能接口进行数据接收和查询的开发，如需参考或使用本项目，需要提前联系官方获取权限。

2.1 金融数据服务

金融数据功能，是指用户使用 C++、Python 以及其他本功能可支持的程序设计语言或用户端页面，获取公司通过对证券交易所等渠道的公开信息加工而成的行情数据、金融资讯数据等金融数据的功能。

2.2 数据详情

1) 行情数据

品种	数据类型	数据起点	说明	是否支持实时订阅
股票	Level-1 快照 、K 线数据	2013 年至今	上交所、深交所、北交所	是
指数	Level-1 快照 、K 线数据		上交所、深交所、北交所	是
债券	Level-1 快照 、K 线数据		上交所、深交所	是
场内基金	Level-1 快照 、K 线数据		上交所、深交所	是
期权	Level-1 快照 、K 线数据	2015 年至今	深交所 ETF 期权、上交所 ETF 期权	是
港股通	港股通行情快照	2023 年至今	上交所、深交所	是
期货	Level-1 快照 、K 线数据	2010 年 4 月至今	中金所	是

2) 基础数据

每日最新证券信息，交易日早上 9 点前更新

复权因子

每日最新代码表，交易日早上 9 点前更新

历史代码表

交易日历

3) 财务数据

资产负债表

现金流量表

利润表

业绩快报

业绩预告

4) 股东股本数据

十大股东数据

股东户数

股本结构

股权冻结/质押

限售股解禁

5) 股东权益数据

分红数据

配股数据

6) 融资融券数据

融资融券成交汇总

融资融券交易明细

7) 交易异动数据

龙虎榜

大宗交易

3. python 开发指南

3.1 SDK 版本与下载

3.1.1 wheel 文件版本

wheel 文件名	操作系统	Python 版本
tgw-1.*.*-py3-none-any.whl	Linux/ Windows	Python 3.8 Python 3.9 Python 3.10 Python 3.11 Python 3.12 Python 3.13
AmazingData-1.*.*-cp38-none-any.whl	Linux/ Windows	Python 3.8 Python 3.9 Python 3.10 Python 3.11 Python 3.12 Python 3.13

3.1.2 wheel 文件下载路径

1. 银河网盘
https://cloud.chinastock.com.cn/p/DSG36jYQx2IY_Y8CIAA
2. 公众号“中国银河证券星耀数智”
路径：“业务介绍”——“安装包下载”

3.2 SDK 运行环境

3.2.1 Linux 推荐运行环境配置

类型	最低配置	推荐配置
处理器	2.10GHz,4 核	2.10GHz,8 核
内存	DDR4 4GB	DDR4 4GB
硬盘	200G 机械硬盘/SSD	480G 机械硬盘/SSD
网卡	普通网卡	普通万兆网卡
操作系统	REDHAT 7.2/7.4/7.6	REDHAT 7.2/7.4/7.6

3.2.2 Windows 推荐运行环境配置

类型	最低配置	推荐配置
处理器	2.60GHz，4 核	2.60GHz，8 核
内存	DDR4 4GB	DDR4 4GB
硬盘	200G 机械硬盘/SSD	480G 机械硬盘/SSD
网卡	普通网卡	普通万兆网卡
操作系统	Windows 10(64 位)	Windows 10(64 位)

3.3 SDK 安装

3.3.1 tgw 安装

pip install tgw-1.7.1-py3-none-any.whl

3.3.2 AmazingData 安装

选择对应的 python 版本
pip install AmazingData-1.0.0-cp312-none-any.whl

3.4 Python 开发步骤

登录 AmazingData 之后，实现数据获取。

3.4.1 登录 AmazingData

（1）所有数据接口调用前，必须登录
（2）import AmazingData 库，填写账号、密码、ip/port 等信息，调用登录 api。

```
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
```

3.4.2 调用数据接口

3.4.2.1 查询接口调用

（1）登录 api；
（2）实例化对应的数据查询类；

（3）调用查询数据接口，获取数据；

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
# 第二步 实例化对应的数据查询类
base_data_object = ad.BaseData()
# 第三步，调用查询数据接口，获取数据
code_list = base_data_object.get_code_list(security_type='EXTRA_ETF')
```

3.4.2.2 订阅接口调用

（1）登录 api；
（2）实例化对应的数据查询类；
（3）实例化数据订阅类；
（4）用装饰器装饰回调函数，接收订阅数据；
（5）订阅数据执行；

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
# 第二步 输入标的代码列表
base_data_object = ad.BaseData()
etf_code_list = base_data_object.get_code_list(security_type='EXTRA_ETF')
# 第三步 实例化数据订阅类
sub_data = ad.SubscribeData()
# 第四步  用装饰器装饰回调函数，接收订阅数据
@sub_data.register(code_list=etf_code_list, period=ad.constant.Period.snapshot.value)
def onSnapshot(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
    print(period, data)
# 第五步 订阅数据执行
sub_data.run()
```

3.5 数据接口详细

3.5.1 基础接口

3.5.1.1 登录

调用任何数据接口之前，必须先调用登录接口。

SDK 的账号、密码、ip 和端口号需联系您的开户营业部申请开通权限之后获取。

函数接口：login

功能描述：api 登陆

输入参数：

参数	数据类型	必选	解释
username	str	是	账号
password	str	是	密码
ip	str	是	服务器 ip
host	int	是	服务器端口号

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
```

3.5.1.2 登出

函数接口：logout

功能描述：api 退出登录链接，必须在登录状态下，才可使用；正常使用情况下，无需使用此接口

名称	类型	说明
username	str	用户名

3.5.1.3 更新密码

函数接口：update_password

功能描述：更新密码接口，必须先登录才能修改密码

名称	类型	说明
username	str	用户名
old_password	str	旧密码
new_password	str	新密码

3.5.2 基础数据

3.5.2.1 每日最新证券信息

函数接口：get_code_info

功能描述：获取每日最新证券信息，交易日早上 9 点前更新当日最新

输入：

参数	数据类型	必选	解释
security_type	str	否	代码类型 security_type（见附录），

			默认为 EXTRA_STOCK_A（上交所 A 股、深交所 A 股和北交所的股票列表）

输出：

参数	数据类型	解释
code_info	dataframe	index 为股票代码 column 为 symbol (证券简称) security_status（产品状态标志） pre_close (昨收价) high_limited (涨停价) low_limited (跌停价) price_tick (最小价格变动单位)

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_info = base_data_object.get_code_info(security_type='EXTRA_ETF')
```

3.5.2.2 每日最新代码表（沪深北）

交易日早上 9 点前更新

函数接口： get_code_list

功能描述： 获取代码表（每日最新），此接口无法获取历史代码表

输入：

参数	数据类型	必选	解释
security_type	str	否	代码类型 security_type（见附录），默认为 EXTRA_STOCK_A（上交所 A 股、深交所 A 股和北交所的股票列表）

输出参数：

返回值	数据类型	解释
code_list	list	证券代码

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A')
```

3.5.2.3 每日最新代码表（期货交易所）

交易日早上 9 点前更新

函数接口：get_future_code_list

功能描述：获取代码表（每日最新），此接口无法获取历史代码表

输入：

参数	数据类型	必选	解释
security_type	str	是	代码类型 security_type(期货交易所)（见附录），默认为 ZJ_FUTURE（期货，中金所）

输出参数：

返回值	数据类型	解释
code_list	list	证券代码

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_future_code_list(security_type='EXTRA_FUTURE')
```

3.5.2.4 每日最新代码表（期权）

交易日早上 9 点前更新

函数接口：get_option_code_list

功能描述：获取代码表（每日最新），此接口无法获取历史代码表

输入：

参数	数据类型	必选	解释
security_type	str	是	代码类型 security_type 期权)（见附录），默认为 EXTRA_ETF_OP（ETF 期权，包含上交所和深交所）

输出参数：

返回值	数据类型	解释
code_list	list	证券代码

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_option_code_list(security_type='EXTRA_ETF_OP')
```

3.5.2.5 复权因子（后复权因子）

函数接口：BaseData.get_backward_factor
功能描述：获取复权因子数据并本地存储，复权因子为根据交易所行情数据计算得出的后复权因子；
输入参数：

参数	数据类型	必选	解释
code_list	lis[str]	是	代码列表，支持股票、ETF
local_path	str	是	本地存储复权因子数据的文件夹地址
is_local	Bool	是	是否使用本地存储的数据，默认为 True

注：
（1）local_path
类似'D://AmazingData_local_data//'，只写文件夹的绝对路径即可

（2）is_local
True:
本地 local_path 有数据的情况下，从本地取数据，但无法从服务端获取最新的数据
本地 local_path 无数据的情况下，从互联网取数据，并更新本地 local_path 的数据
False:从互联网取数据，并更新本地 local_path 的数据
输出：

参数	数据类型	解释
backward_factor	dataframe	index 为交易日期 column 为股票代码

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A')
backward_factor = base_data_object.get_backward_factor(code_list, local_path='D://AmazingData_local_data//',
is_local=False)
```

3.5.2.6 复权因子（单次复权因子）

函数接口：BaseData.get_adj_factor
功能描述：获取复权因子数据并本地存储，复权因子为根据交易所行情数据计算得出的单次复权因子；
输入参数：

参数	数据类型	必选	解释
code_list	lis[str]	是	代码列表，支持股票、ETF
local_path	str	是	本地存储复权因子数据的文件夹地址

is_local	Bool	是	是否使用本地存储的数据，默认为 True

注：
（1）local_path
类似'D://AmazingData_local_data//'，只写文件夹的绝对路径即可

（2）is_local
True:
本地 local_path 有数据的情况下，从本地取数据，但有可能无法获取最新的数据
本地 local_path 无数据的情况下，从互联网取数据，并更新本地 local_path 的数据
False:从互联网取数据，并更新本地 local_path 的数据
输出：

参数	数据类型	解释
adj_factor	dataframe	index 为交易日期 column 为股票代码

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A')
adj_factor = base_data_object.get_adj_factor(code_list, local_path='D://AmazingData_local_data//',
is_local=False)
```

3.5.2.7 历史代码表

函数接口：BaseData 的 get_hist_code_list
功能描述：获取历史代码表，先检查本地数据，再从服务端补充，最后返回数据输入参数：
输入参数：

参数	数据类型	必选	解释
security_type	str	是	默认为 "EXTRA_STOCK_A_SH_SZ" 沪深 A 股，支持附录 security_type(沪深北)和 security_type(期货交易所)，
start_date	int	是	开始时间，闭区间
end_date	int	是	结束时间，闭区间
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”

输出参数：

返回值	数据类型	解释
code_list	List[str]	证券代码

```
# 第一步 登录 api
```

```
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ',start_date=20240101,
end_date=20240701, local_path=local_path)
```

3.5.2.8 交易日历

函数接口：get_calendar
功能描述：获取交易所的交易日历

输入参数：

参数	数据类型	必选	解释
data_type	str	否	选择返回数据的类型，默认为 str ，可选 datetime 或 str
market	str	否	选择市场 market（见附录），默认为 SH（上海）

输出参数：

返回值	数据类型	解释
calendar	List[int]	日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
```

3.5.2.9 证券基础信息

函数接口：get_stock_basic
功能描述：获取指定股票列表的上市公司的证券基础数据，包含沪深北三个交易所，所有股票（包含已退市标的）的中英文名称、上市日期、退市日期、上市板块等信息
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深北三个交易所的代码列表，可见示例

输出参数：

返回值	数据类型	解释
stock_basic	dataframe	column 为 stock_basic 的字段 index 为序号（无意义）

stock_basic 的字段说明：

参数	数据类型	必选	解释
MARKET_CODE	string	证券代码	
SECURITY_NAME	string	证券简称	
COMP_NAME	string	证券中文名称	
PINYIN	string	中文拼音简称	
COMP_NAME_ENG	string	证券英文名称	
LISTDATE	int	上市日期	
DELISTDATE	int	退市日期	
LISTPLATE_NAME	string	上市板块名称	
COMP_SNAME_ENG	string	英文名称缩写	
IS_LISTED	int	上市状态	1：上市交易 3：终止上市

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A_SH_SZ')
info_data_object = ad.InfoData()
stock_basic = info_data_object.get_stock_basic (code_list)
```

3.5.2.10 历史证券信息

函数接口：get_history_stock_status
功能描述：获取指定股票列表的上市公司的历史证券数据，以日度为频率，包含历史的涨跌停、st、除权除息等信息
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“D://AmazingData_local_data//”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日，本地数据缓存方案
end_date	int	否	交易日，本地数据缓存方案

输出参数：

返回值	数据类型	解释
history_stock_status	dataframe	column 为 history_stock_status 的字段 index 为序号（无意义）

history_stock_status 的字段说明：

参数	数据类型	必选	解释
MARKET_CODE	string	证券代码	

TRADE_DATE	string	日期	
PRECLOSE	float	前收价	
HIGH_LIMITED	float	涨停价	
LOW_LIMITED	float	跌停价	
PRICE_HIGH_LMT_RATE	float	涨停价上限	
PRICE_LOW_LMT_RATE	float	跌停价下限	
IS_ST_SEC	string	是否 ST	1 表示是，0 表示否
IS_SUSP_SEC	string	是否停牌	1 表示是，0 表示否
IS_WD_SEC	string	是否除息	1 表示是，0 表示否
IS_XR_SEC	string	是否除权	1 表示是，0 表示否

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101, end_date=today)
history_stock_status = info_data_object.get_history_stock_status(all_code_list)
```

3.5.2.11 北交所新旧代码对照表

函数接口：get_bj_code_mapping
功能描述：获取北交所的存量上市公司股票新旧代码对照表
输入参数：

参数	数据类型	必选	解释
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，首选从本地读取，读取失败再从服务器取数据 False，以本地数据为基础，增量从服务器取数据

输出参数：

参数	数据类型	解释
bj_code_mapping	dataframe	column 为 bj_code_mapping 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
bj_code_mapping = info_data_object.get_bj_code_mapping()
```

bj_code_mapping 的字段说明：

字段名称	类型	字段说明
OLD_CODE	string	旧代码
NEW_CODE	string	新代码
SECURITY_NAME	string	证券简称
LISTING_DATE	int	上市日期

3.5.3 实时行情数据

实时行情订阅接口使用步骤

（1） 实例化 AmazingData 的 SubscribeData

（2） 回调函数的装饰器传入 code_list(代码表)和 period(数据周期)两个参数

（3） 回调函数中获取数据

3.5.3.1 指数实时快照

函数接口： onSnapshotindex
功能描述： 交易所指数快照数据的实时订阅回调函数
输入参数： 入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持北交所、上交所、深交所的指数
period	Period	是	Period.snapshot.value

输出参数： 入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	指数为 SnapshotIndex（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
```

```
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type=' EXTRA_INDEX_A')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshotindex(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
    print(period, data)
sub_data.run()
```

3.5.3.2 股票实时快照

函数接口： onSnapshot
功能描述： level-1 快照数据的实时订阅回调函数
输入参数： 入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持北交所、上交所、深交所的股票
period	Period	是	Period.snapshot.value

输出参数： 入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	股票为 Snapshot（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshot(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
    print(period, data)
sub_data.run()
```

3.5.3.3 逆回购实时快照

函数接口： onSnapshotglra
功能描述： level-1 快照数据的实时订阅回调函数

输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持上交所、深交所的逆回购代码
period	Period	是	Period.snapshot.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	为 Snapshot（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_GLRA')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshotglra(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
        print(period, data)
sub_data.run()
```

3.5.3.4 期货实时快照

函数接口：onSnapshotfuture
功能描述：level-1 快照数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持中金所
period	Period	是	Period.snapshotfuture.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	期货为 SnapshotFuture（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
```

```
code_list = base_data_object.get_code_list(security_type='EXTRA_FUTURE')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshotfuture.value)
def onSnapshotfuture (data: Union[ad.constant.SnapshotFuture], period):
        print(period, data)
sub_data.run()
```

3.5.3.5 ETF 实时快照

函数接口：onSnapshotetf
功能描述：level-1 快照数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持上交所、深交所的 ETF
period	Period	是	Period.snapshot.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	ETF 为 Snapshot（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_ETF')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshotetf(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
        print(period, data)
sub_data.run()
```

3.5.3.6 可转债实时快照

函数接口：onSnapshotkzz
功能描述：level-1 快照数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持上交所、深交所的可转债
period	Period	是	Period.snapshot.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	可转债为 Snapshot（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_KZZ')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshotkzz(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
        print(period, data)
sub_data.run()
```

3.5.3.7 港股通实时快照

函数接口：onSnapshothkt
功能描述：港股通快照数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持上交所、深交所的可转债
period	Period	是	Period.snapshotHKT.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	港股通为 SnapshotHKT（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_HKT')
# 实时订阅
```

```
sub_data = ad.SubscribeData()
@sub_data.register(code_list=code_list, period=ad.constant.Period.snapshot.value)
def onSnapshothkt(data: Union[ad.constant.Snapshot, ad.constant.SnapshotIndex], period):
        print(period, data)
sub_data.run()
```

3.5.3.8 ETF 期权实时快照

函数接口：onSnapshotoption
功能描述：港股通快照数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持上交所、深交所的 ETF 期权
period	Period	是	Period.snapshotoption.value

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	ETF 期权为 SnapshotOption（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
option_code_list = base_data_object.get_option_code_list(security_type='EXTRA_ETF_OP')
# 实时订阅
sub_data = ad.SubscribeData()
@sub_data.register(code_list=option_code_list, period=ad.constant.Period.snapshotoption.value)
def onSnapshotoption(data: Union[ad.constant.SnapshotOption], period):
    print('onSnapshotoption: ', data)
sub_data.run()
```

3.5.3.9 实时 K 线

函数接口：OnKLine
功能描述：K 线数据的实时订阅回调函数
输入参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持北交所、上交所、深交所的可转债、股票、指数、ETF 等品种

			支持期货（中金所）
period	Period	是	Period（见附录）

输出参数：入参需传入装饰器中 SubscribeData.register

参数	数据类型	解释
data	Object	Kline（见附录）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A ')
# 实时订阅
sub_data = ad.SubscribeData()
# K 线
@sub_data.register(code_list=code_list, period=ad.constant.Period.min1.value)
def OnKLine(data: Union[ad.constant.Kline], period):
    print('OnKLine: ', data)
sub_data.run()
```

3.5.4 历史行情数据

（1） 实例化 AmazingData 的 MarketData，入参需交易日历
（2） 调用 MarketData 的方法获取数据

3.5.4.1 历史快照

函数接口：query_snapshot
功能描述：快照数据的历史数据查询接口
输入参数：

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持北交所、上交所、深交所的可转债、股票、指数、ETF、港股通等、ETF 期权等品种
begin_date	int	是	日期，填写 8 位的整型格式的日期，比如 20240101
end_date	int	是	日期，填写 8 位的整型格式的日期，比如 20240201
begin_time	int	否	时分秒毫秒的时间戳，填写 8 位或 9 位的整型格式的日期，时占一位或两位，分占两位，秒占两位，毫秒占三位，例如 9 点

			整 为 90000000，17 点 25 分为 172500000
end_time	int	否	时分秒毫秒的时间戳，填写 8 位或 9 位的整型格式的日期，时占一位或两位，分占两位，秒占两位，毫秒占三位，例如 9 点整 为 90000000，17 点 25 分为 172500000

输出参数：

参数	数据类型	解释
snapshot_dict	dict	指字典的 key：代码 字典的 value：dataframe， column 为快照数据（指数为 SnapshotIndex（见附录）， 股票、ETF 和可转债为 Snapshot（见附录）， 港股通为 SnapshotHKT（见附录））， ETF 期权为 SnapshotOption（见附录））， index 为日期（datetime）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A ')
calendar = base_data_object.get_calendar()
market_data_object=ad.MarketData(calendar)
snapshot_dict = market_data_object.query_snapshot(code_list, begin_date=20240530, end_date=20240530)
```

3.5.4.2 历史 K 线

函数接口：query_kline
功能描述：K 线数据的实时订阅回调函数 ，支持全部周期的 K 线数据查询
输入参数：

参数	数据类型	必选	解释
code_list	list:[str]	是	可传入列表，支持北交所、上交所、深交所的可转债、股票、指数、ETF 等品种，上交所、深交所的 ETF 期权； 支持期货（中金所）
begin_date	int	是	日期，填写 8 位的整型格式的日期，比如 20240101
end_date	int	是	日期，填写 8 位的整型格式的日期，比如 20240201
period	Period	是	数据周期 Period（见附录）

begin_time	int	否	时分的时间戳，填写 3 位或 4 位的整型格式的日期，时占一位或两位，分占两位，，例如 9 点整 为 900，17 点 25 分为 1725
end_time	int	否	时分的时间戳，填写 3 位或 4 位的整型格式的日期，时占一位或两位，分占两位，，例如 9 点整 为 900，17 点 25 分为 1725

输出参数：

参数	数据类型	解释
kline_dict	dict	字典的 key：代码 字典的 value：dataframe， column 为 K 线数据 Kline（见附录）， index 为日期（datetime）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_STOCK_A')
calendar = base_data_object.get_calendar()
market_data_object=ad.MarketData(calendar)
kline_dict = market_data_object.query_kline (code_list, begin_date=20240530, end_date=20240530)
```

3.5.5 财务数据

3.5.5.1 资产负债表

函数接口：get_balance_sheet
功能描述：获取指定股票列表的上市公司的资产负债表数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	报告期，本地数据缓存方案
end_date	int	否	报告期，本地数据缓存方案

输出参数：

参数	数据类型	解释
balance_sheet	dict	key：code value:dataframe column 为 balance_sheet 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
balance_sheet = info_data_object.get_balance_sheet(all_code_list)
```

balance_sheet 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	str	证券代码	
SECURITY_NAME	str	证券简称	
STATEMENT_TYPE	str	报表类型	参看报表类型代码表
REPORT_TYPE	str	报告期名称	参看报告期名称
REPORTING_PERIOD	str	报告期	
ANN_DATE	str	公告日期	
ACTUAL_ANN_DATE	str	实际公告日期	
ACC_PAYABLE	float	应付票据及应付账款	
ACC_RECEIVABLE	float	应收票据及应收账款	
ACC_RECEIVABLES	float	应收款项	
ACCRUED_EXP	float	预提费用	
ACCT_PAYABLE	float	应付账款	
ACCT_RECEIVABLE	float	应收账款	

ACT_TRADING_SEC	float	代理买卖证券款	
ACT_UW_SEC	float	代理承销证券款	
ADV_PREM	float	预收保费	
ADV_RECEIPT	float	预收款项	
AGENCY_ASSETS	float	代理业务资产	
AGENCY_BUSINESS_LIAB	float	代理业务负债	
ANTICIPATION_LIAB	float	预计负债	
ASSET_DEP_FUNDS_OTH_FIN_INST	float	存放同业和其它金融机构款项	
BONDS_PAYABLE	float	应付债券	
CAP_RESV	float	资本公积金	
CAP_STOCK	float	股本	金额（元），公布值
CASH_CENTRAL_BANK_DEPOSITS	float	现金及存放中央银行款项	
CED_INSUR_CONT_RESERVES_RCV	float	应收分保合同准备金	
CLAIMS_PAYABLE	float	应付赔付款	
CLIENTS_FUND_DEPOSIT	float	客户资金存款	
CLIENTS_RESERVES	float	客户备付金	
CNVD_DIFF_FOREIGN_CURR_STAT	float	外币报表折算差额	
COMP_TYPE_CODE	int	公司类型代码	1：非金融类 2：银行 3：保险 4：证券
CONST_IN_PROC	float	在建工程	
CONST_IN_PROC_TOTAL	float	在建工程(合计)(元)	
CONSUMP_BIO_ASSETS	float	消耗性生物资产	
CONT_ASSETS	float	合同资产	单位（元）

CONT_LIABILITIES	float	合同负债	单位（元）
CURRENCY_CAP	float	货币资金	
CURRENCY_CODE	float	货币代码	
DEBT_INV	float	债权投资(元)	
DEFERRED_INC_NONCUR_LIAB	float	递延收益-非流动负债	
DEFERRED_INCOME	float	递延收益	
DEFERRED_TAX_ASSETS	float	递延所得税资产	
DEFERRED_TAX_LIAB	float	递延所得税负债	
DEP_RECEIVED_IB_DEP	float	吸收存款及同业存放	
DEPOSIT_CAP_RECOG	float	存出资本保证金	
DEPOSIT_TAKING	float	吸收存款	
DEPOSITS_RECEIVED	float	存入保证金	
DER_FIN_ASSETS	float	衍生金融资产	
DERI_FIN_LIAB	float	衍生金融负债	
DEVELOP_EXP	float	开发支出	
DISPOSAL_FIX_ASSETS	float	固定资产清理	
DIV_PAYABLE	float	应付股利	
DIV_RECEIVABLE	float	应收股利	
EMPL_PAY_PAYABLE	float	应付职工薪酬	
ENGIN_MAT	float	工程物资	
FIN_ASSETS_AVA_FOR_SALE	float	可供出售金融资产	
FIN_ASSETS_COST_SHARING	float	以摊余成本计量的金融资产	
FIN_ASSETS_FAIR_VALUE	float	以公允价值计量且其变动计入其他综合收益的金融资产	

FIXED_ASSETS	float	固定资产	
FIXED_ASSETS_TOTAL	float	固定资产(合计)(元)	
FIXED_TERM_DEPOSITS	float	定期存款	
GOODWILL	float	商誉	
GUA_DEPOSITS_PAID	float	存出保证金	
GUA_PLEDGE_LOANS	float	保户质押贷款	
HOLD_ASSETS_FOR_SALE	float	持有待售的资产	
HOLD_TO_MTY_INV	float	持有至到期投资	
INC_PLEDGE_LOAN	float	其中:质押借款	
INCL_TRADING_SEAT_FEES	float	其中:交易席位费	
IND_ACCT_ASSETS	float	独立账户资产	
IND_ACCT_LIAB	float	独立账户负债	
INSURED_DEPOSIT_INV	float	保户储金及投资款	
INSURED_DIV_PAYABLE	float	应付保单红利	
INT_RECEIVABLE	float	应收利息	
INTANGIBLE_ASSETS	float	无形资产	
INTEREST_PAYABLE	float	应付利息	
INV	float	存货	
INV_REALESTATE	float	投资性房地产	
LEASE_LIABILITY	float	租赁负债	
LEND_FUNDS	float	融出资金	
LENDING_FUNDS	float	拆出资金	
LESS_TREASURY_STK	float	减:库存股	
LIA_HFS	float	持有待售的负债	
LIAB_DEP_FUNDS_OTH_FIN_INST	float	同业和其它金融机构存放款项	

LIFE_INSUR_RESV	float	寿险责任准备金	
LOAN_CENTRAL_BANK	float	向中央银行借款	
LOANS_AND_ADVANCES	float	发放贷款及垫款	
LOANS_FROM_OTH_BANKS	float	拆入资金	
LT_DEFERRED_EXP	float	长期待摊费用	
LT_EMP_COMP_PAY	float	长期应付职工薪酬	
LT_EQUITY_INV	float	长期股权投资	
LT_HEALTH_INSUR_RESV	float	长期健康险责任准备金	
LT_LOAN	float	长期借款	
LT_PAYABLE	float	长期应付款	
LT_PAYABLE_TOTAL	float	长期应付款(合计)(元)	
LT_RECEIVABLES	float	长期应收款	
MINORITY_EQUITY	float	少数股东权益	
NOM_RISKS_PREP	float	一般风险准备	
NONCUR_ASSETS_DUE_WITHIN_1Y	float	一年内到期的非流动资产	
NONCUR_LIAB_DUE_WITHIN_1Y	float	一年内到期的非流动负债	
NOTES_PAYABLE	float	应付票据	
NOTES_RECEIVABLE	float	应收票据	
OIL_AND_GAS_ASSETS	float	油气资产	
OTH_COMP_INCOME	float	其他综合收益	
OTH_EQUITY_TOOLS	float	其他权益工具	
OTH_EQUITY_TOOLS_PRE_SHR	float	其他权益工具:优先股	
OTH_NONCUR_ASSETS	float	其他非流动资产	

OTHER_ASSETS	float	其他资产	
OTHER_CUR_ASSETS	float	其他流动资产	
OTHER_CUR_LIAB	float	其他流动负债	
OTHER_DEBT_INV	float	其他债权投资(元)	
OTHER_EQUITY_INV	float	其他权益工具投资(元)	
OTHER_LIAB	float	其他负债	
OTHER_NONCUR_FIN_ASSETS	float	其他非流动金融资产(元)	
OTHER_NONCUR_LIAB	float	其他非流动负债	
OTHER_PAYABLE	float	其他应付款	
OTHER_PAYABLE_TOTAL	float	其他应付款(合计)(元)	
OTHER_RCV_TOTAL	float	其他应收款(合计)（元）	
OTHER_RECEIVABLE	float	其他应收款	
OTHER_SUSTAIN_BOND	float	其他权益工具:永续债(元)	
OUT_LOSS_RESV	float	未决赔款准备金	
PAYABLE	float	应付款项	
PAYABLE_FOR_REINSURER	float	应付分保账款	
PRECIOUS_METAL	float	贵金属	
PREPAYMENT	float	预付款项	
PROD_BIO_ASSETS	float	生产性生物资产	
RCV_CED_CLAIM_RESV	float	应收分保未决赔款准备金	
RCV_CED_LIFE_INSUR_RESV	float	应收分保寿险责任准备金	
RCV_CED_LT_HEALTH_INSUR_RESV	float	应收分保长期健康险责任准备金	
RCV_CED_UNEARNED	float	应收分保未到期责	

_PREM_RESV		任准备金	
RCV_FINANCING	float	应收款项融资	
RCV_INV	float	应收款项类投资	
RECEIVABLE_PREM	float	应收保费	
RED_MON_CAP_FOR_SALE	float	买入返售金融资产	
REINSURANCE_ACC_RCV	float	应收分保账款	
RSRV_FUND_INSUR_CONT	float	保险合同准备金	
SELL_REPO_FIN_ASSETS	float	卖出回购金融资产款	
SERVICE_CHARGE_COMM_PAYABLE	float	应付手续费及佣金	
SETTLE_FUNDS	float	结算备付金	
SPE_ASSETS_BAL_DIFF	float	资产差额(特殊报表科目)	
SPE_CUR_ASSETS_DIFF	float	流动资产差额(特殊报表科目)	
SPE_CUR_LIAB_DIFF	float	流动负债差额(特殊报表科目)	
SPE_LIAB_BAL_DIFF	float	负债差额(特殊报表科目)	
SPE_LIAB_EQUITY_BAL_DIFF	float	负债及股东权益差额(特殊报表项目)	
SPE_NONCUR_ASSETS_DIFF	float	非流动资产差额(特殊报表科目)	
SPE_NONCUR_LIAB_DIFF	float	非流动负债差额(特殊报表科目)	
SPE_SHARE_EQUITY_BAL_DIFF	float	股东权益差额(特殊报表科目)	
SPECIAL_PAYABLE	float	专项应付款	
SPECIAL_RESV	float	专项储备	

ST_BONDS_PAYABLE	float	应付短期债券	
ST_BORROWING	float	短期借款	
ST_FIN_PAYABLE	float	应付短期融资款	
SUBR_RCV	float	应收代位追偿款	
SURPLUS_RESV	float	盈余公积金	
TAX_PAYABLE	float	应交税费	
TOT_ASSETS_BAL_DIFF	float	资产差额(合计平衡项目)	
TOT_CUR_ASSETS_DIFF	float	流动资产差额(合计平衡项目)	
TOT_CUR_LIAB_DIFF	float	流动负债差额(合计平衡项目)	
TOT_LIAB_BAL_DIFF	float	负债差额(合计平衡项目)	
TOT_LIAB_EQUITY_BAL_DIFF	float	负债及股东权益差额(合计平衡项目)	
TOT_NONCUR_ASSETS	float	非流动资产合计	
TOT_NONCUR_ASSETS_DIFF	float	非流动资产差额(合计平衡项目)	
TOT_NONCUR_LIAB_DIFF	float	非流动负债差额(合计平衡项目)	
TOT_SHARE	float	期末总股本	单位（股）
TOT_SHARE_EQUITY_BAL_DIFF	float	股东权益差额(合计平衡项目)	
TOT_SHARE_EQUITY_EXCL_MIN_INT	float	股东权益合计(不含少数股东权益)	
TOT_SHARE_EQUITY_INCL_MIN_INT	float	股东权益合计(含少数股东权益)	
TOTAL_ASSETS	float	资产总计	
TOTAL_CUR_ASSETS	float	流动资产合计	
TOTAL_CUR_LIAB	float	流动负债合计	
TOTAL_LIAB	float	负债合计	

TOTAL_LIAB_SHARE_EQUITY	float	负债及股东权益总计	
TOTAL_NONCUR_LIAB	float	非流动负债合计	
TRADING_FIN_LIAB	float	交易性金融负债	
TRADING_FINASSETS	float	交易性金融资产	
UNAMORTIZED_EXP	float	待摊费用	
UNCONFIRMED_INV_LOSS	float	未确认的投资损失	
UNDISTRIBUTED_PRO	float	未分配利润	
UNEARNED_PREM_RESV	float	未到期责任准备金	
USE_RIGHT_ASSETS	float	使用权资产	

3.5.5.2 现金流量表

函数接口：get_cash_flow
功能描述：获取指定股票列表的上市公司的现金流量表数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	报告期，本地数据缓存方案
end_date	int	否	报告期，本地数据缓存方案

输出参数：

参数	数据类型	解释
cash_flow	dict	key：code value:dataframe column 为 cash_flow 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
```

```
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
cash_flow = info_data_object.get_cash_flow (all_code_list)
```

cash_flow 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	str	证券代码	
SECURITY_NAME	str	证券简称	
STATEMENT_TYPE	str	报表类型	参看报表类型代码表
REPORT_TYPE	str	报告期名称	参看报告期名称
REPORTING_PERIOD	str	报告期	
ANN_DATE	str	公告日期	
ACTUAL_ANN_DATE	str	实际公告日期	
ABSORB_CASH_RECP_INV	double	吸收投资收到的现金	
AMORT_INTAN_ASSETS	double	无形资产摊销	
AMORT_LT_DEFERRED_EXP	double	长期待摊费用摊销	
BEG_BAL_CASH_CASH_EQU	double	期初现金及现金等价物余额	
CASH_END_BAL	double	现金的期末余额	
CASH_FOR_CHARGE	double	支付手续费的现金	

CASH_PAID_INSUR_POLICY	double	支付保单红利的现金	
CASH_PAID_INV	double	投资支付的现金	
CASH_PAID_PUR_CONST_FIOLTA	double	购建固定资产、无形资产和其他长期资产支付的现金	
CASH_PAY_CLAIMS_OIC	double	支付原保险合同赔付款项的现金	
CASH_PAY_DIST_DIV_PRO_INT	double	分配股利、利润或偿付利息支付的现金	
CASH_PAY_EMPLOYEE	double	支付给职工以及为职工支付的现金	
CASH_PAY_FOR_DEBT	double	偿还债务支付的现金	
CASH_PAY_GOODS_SERVICES	double	购买商品、接受劳务支付的现金	
CASH_RECE_BORROW	double	取得借款收到的现金	
CASH_RECE_ISSUE_BONDS	double	发行债券收到的现金	
CASH_RECP_INV_INCOME	double	取得投资收益收到的现金	
CASH_RECP_PREM_OIC	double	收到原保险合同保费取得的现金	
CASH_RECP_RECOV_INV	double	收回投资收到的现金	
CASH_RECP_SG_AND_RS	double	销售商品、提供劳务收到的现金	
COMP_TYPE_CODE	str	公司类型代码	1：非金融类 2：银行 3：保险 4：证券

CONV_CORP_BONDS_DUE_WITHIN_1Y	double	一年内到期的可转换公司债券	
CONV_DEBT_INTO_CAP	double	债务转为资本	
CREDIT_IMPAIR_LOSS	double	信用减值损失	
CURRENCY_CODE	str	货币代码	
DECR_DEFE_INC_TAX_ASSETS	double	递延所得税资产减少	
DECR_DEFERRED_EXPENSE	double	待摊费用减少	
DECR_INVENTORY	double	存货的减少	
DECR_OPERA_RECEIVABLE	double	经营性应收项目的减少	
DEPRE_FA_OGA_PBA	double	固定资产折旧、油气资产折耗、生产性生物资产折旧	
EFF_FX_FLUC_CASH	double	汇率变动对现金的影响	
END_BAL_CASH_CASH_EQU	double	期末现金及现金等价物余额	
FINANCIAL_EXP	double	财务费用	
FIXED_ASSETS_FIN_LEASE	double	融资租入固定资产	
FREE_CASH_FLOW	double	企业自由现金流量	
INCL_CASH_RECP_SAIMS	double	其中:子公司吸收少数股东投资收到的现金	

INCL_DIV_PRO_PAID_SMS	double	其中:子公司支付给少数股东的股利、利润	
INCR_ACCRUED_EXP	double	预提费用增加	
INCR_DEFE_INC_TAX_LIAB	double	递延所得税负债增加	
INCR_OPERA_PAYABLE	double	经营性应付项目的增加	
IND_NET_CASH_FLOWS_OPERA_ACT	double	间接法-经营活动产生的现金流量净额	
IND_NET_INCR_CASH_AND_EQU	double	间接法-现金及现金等价物净增加额	
INV_LOSS	double	投资损失	
IS_CALCULATION	int	是否计算报表	
LESS_OPEN_BAL_CASH	double	减:现金的期初余额	
LESS_OPEN_BAL_CASH_EQU	double	减:现金等价物的期初余额	
LOSS_DISP_FIOLTA	double	处置固定、无形资产和其他长期资产的损失	
LOSS_FAIRVALUE_CHG	double	公允价值变动损失	
LOSS_FIXED_ASSETS	double	固定资产报废损失	
NET_CASH_FLOWS_FIN_ACT	double	筹资活动产生的现金流量净额	
NET_CASH_FLOWS_INV_ACT	double	投资活动产生的现金流量净额	

NET_CASH_FLOWS_OPERA_ACT	double	经营活动产生的现金流量净额	
NET_CASH_PAID_SOBU	double	取得子公司及其他营业单位支付的现金净额	
NET_CASH_REC_SEC	double	代理买卖证券收到的现金净额	
NET_CASH_RECP_DISP_FIOLTA	double	处置固定资产、无形资产和其他长期资产收回的现金净额	
NET_CASH_RECP_DISP_SOBU	double	处置子公司及其他营业单位收到的现金净额	
NET_CASH_RECP_REINSU_BUS	double	收到再保业务现金净额	
NET_INCR_BORR_FUND	double	拆入资金净增加额	
NET_INCR_BORR_OFI	double	向其他金融机构拆入资金净增加额	
NET_INCR_CASH_AND_CASH_EQU	double	现金及现金等价物净增加额	
NET_INCR_CUS_LOAN_ADV	double	客户贷款及垫款净增加额	
NET_INCR_DEP_CB_IB	double	存放央行和同业款项净增加额	
NET_INCR_DEP_CUS_AND_IB	double	客户存款和同业存放款项净增加额	
NET_INCR_DISMANTLE_CAP	double	拆出资金净增加额	

NET_INCR_DISP_FAAS	double	处置可供出售金融资产净增加额	
NET_INCR_DISP_TFA	double	处置交易性金融资产净增加额	
NET_INCR_INSURED_SAVE	double	保户储金净增加额	
NET_INCR_INT_AND_CHARGE	double	收取利息和手续费净增加额	
NET_INCR_LOANS_CENTRAL_BANK	double	向中央银行借款净增加额	
NET_INCR_PLEDGE_LOAN	double	质押贷款净增加额	
NET_INCR_REPU_BUS_FUND	double	回购业务资金净增加额	
NET_PROFIT	double	净利润	
OTH_CASH_PAY_INV_ACT	double	支付其他与投资活动有关的现金	
OTH_CASH_PAY_OPERA_ACT	double	支付其他与经营活动有关的现金	
OTH_CASH_RECP_INV_ACT	double	收到其他与投资活动有关的现金	
OTHER_ASSETS_IMPAIR_LOSS	double	其他资产减值损失	
OTHER_CASH_PAY_FIN_ACT	double	支付其他与筹资活动有关的现金	
OTHER_CASH_RECP_FIN_ACT	double	收到其他与筹资活动有关的现金	
OTHER_CASH_RECP_OPER_ACT	double	收到其他与经营活动有关的现金	

OTHERS	double	其他（废弃）	
PAY_ALL_TAX	double	支付的各项税费	
PLUS_ASSETS_DEPRE_PREP	double	加:资产减值准备	
PLUS_END_BAL_CASH_EQU	double	加:现金等价物的期末余额	
RECP_TAX_REFUND	double	收到的税费返还	
SPE_BAL_CASH_INFLOW_FIN_ACT	double	筹资活动现金流入差额	
SPE_BAL_CASH_INFLOW_INV_ACT	double	投资活动现金流入差额	
SPE_BAL_CASH_INFLOW_OPERA_ACT	double	经营活动现金流入差额	
SPE_BAL_CASH_OUTFLOW_FIN	double	筹资活动现金流出差额	
SPE_BAL_CASH_OUTFLOW_INV	double	投资活动现金流出差额	
SPE_BAL_CASH_OUTFLOW_OPERA	double	经营活动现金流出差额	
SPE_BAL_NETCASH_INC_DIFF_IND	double	间接法-现金净增加额差额	
SPE_BAL_NETCASH_INCR_DIFF	double	现金净增加额差额	
SPE_BAL_NETCASH_OPERA_IND	double	间接法-经营活动现金流量净额差额	
TOT_BAL_CASH_INFLOW_FIN_ACT	double	筹资活动现金流入差额	

TOT_BAL_CASH_INFLOW_INV_ACT	double	投资活动现金流入差额	
TOT_BAL_CASH_INFLOW_OPERA_ACT	double	经营活动现金流入差额	
TOT_BAL_CASH_OUTFLOW_FIN	double	筹资活动现金流出差额	
TOT_BAL_CASH_OUTFLOW_INV	double	投资活动现金流出差额	
TOT_BAL_CASH_OUTFLOW_OPERA	double	经营活动现金流出差额	
TOT_BAL_NETCASH_FLOW_FIN	double	筹资活动产生的现金流量净额差额	
TOT_BAL_NETCASH_FLOW_INV	double	投资活动产生的现金流量净额差额	
TOT_BAL_NETCASH_FLOW_OPERA	double	经营活动产生的现金流量净额差额	
TOT_BAL_NETCASH_INC_DIFF_IND	double	间接法-现金净增加额差额	
TOT_BAL_NETCASH_INCR_DIFF	double	现金净增加额差额	
TOT_BAL_NETCASH_OPERA_IND	double	间接法-经营活动现金流量净额差额	
TOT_CASH_INFLOW_FIN_ACT	double	筹资活动现金流入小计	
TOT_CASH_INFLOW_INV_ACT	double	投资活动现金流入小计	
TOT_CASH_INFLOW_OPER_ACT	double	经营活动现金流入小计	

TOT_CASH_OUTFLOW_FIN_ACT	double	筹资活动现金流出小计	
TOT_CASH_OUTFLOW_INV_ACT	double	投资活动现金流出小计	
TOT_CASH_OUTFLOW_OPERA_ACT	double	经营活动现金流出小计	
UNCONFIRMED_INV_LOSS	double	未确认投资损失	
USE_RIGHT_ASSET_DEP	double	使用权资产折旧	

3.5.5.3 利润表

函数接口：get_income
功能描述：获取指定股票列表的上市公司的利润表数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	报告期，本地数据缓存方案
end_date	int	否	报告期，本地数据缓存方案

输出参数：

参数	数据类型	解释
income	dict	key：code value:dataframe column 为 income 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
```

```
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
income = info_data_object.get_income (all_code_list)
```

income 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	str	证券代码	
SECURITY_NAME	str	证券简称	
STATEMENT_TYPE	str	报表类型	参看报表类型代码表
REPORT_TYPE	str	报告期名称	参看报告期名称
REPORTING_PERIOD	str	报告期	
ANN_DATE	str	公告日期	
ACTUAL_ANN_DATE	str	实际公告日期	
AMORT_COST_FIN_ASSETS_EAR	float	以摊余成本计量的金融资产终止确认收益	
ANN_DATE	str	公告日期	
BASIC_EPS	float	基本每股收益	
BEG_UNDISTRIBUTED_PRO	float	年初未分配利润	
CAPITALIZED_COM_STOCK_DIV	float	转作股本的普通股股利	
COMMENTS	str	备注	
COMMON_STOCK_DIV_PAYABLE	float	应付普通股股利	
COMP_TYPE_CODE	str	公司类型代码	1：非金融类 2：银行 3：保险 4：证券
CONTINUED_NET_OPERA_PRO	float	持续经营净利润	

CREDIT_IMPAIR_LOSS	float	信用减值损失	
CURRENCY_CODE	str	货币代码	
DILUTED_EPS	float	稀释每股收益	
DISTRIBUTIVE_PRO	float	可分配利润	
DISTRIBUTIVE_PRO_SHAREHOLDER	float	可供股东分配的利润	
DIV_EXP_INSUR	float	保户红利支出	
EBIT	float	息税前利润	正向法
EBITDA	float	息税折旧摊销前利润	
EMPLOYEE_WELFARE	float	职工奖金福利	
END_NET_OPERA_PRO	float	终止经营净利润	
EXT_INSUR_CONT_RSRV	float	提取保险责任准备金	
EXT_UNEARNED_PREM_RES	float	提取未到期责任准备金	
FIN_EXP_INT_EXP	float	财务费用:利息费用	
FIN_EXP_INT_INC	float	财务费用:利息收入	
GAIN_DISPOSAL_ASSETS	float	资产处置收益	
HANDLING_CHRG_COMM_FEE	float	手续费及佣金收入	
INCL_INC_INV_JV_ENTP	float	其中:对联营企业和合营企业的投资收益	
INCL_LESS_LOSS_DISP_NCUR_ASSET	float	其中:减:非流动资产处置净损失	

INCL_REINSUR_PREM_INC	float	其中:分保费收入	
INCOME_TAX	float	所得税	
INSUR_EXP	float	保险业务支出	
INSUR_PREM	float	已赚保费	
INTEREST_INC	float	利息收入	
IS_CALCULATION	float	是否计算报表	
LESS_ADMIN_EXP	float	减:管理费用	
LESS_AMORT_COMPEN_EXP	float	减:摊回赔付支出	
LESS_AMORT_INSUR_CONT_RSRV	float	减:摊回保险责任准备金	
LESS_AMORT_REINSUR_EXP	float	减:摊回分保费用	
LESS_ASSETS_IMPAIR_LOSS	float	减:资产减值损失	
LESS_BUS_TAX_SURCHARGE	float	减:营业税金及附加	
LESS_FIN_EXP	float	减:财务费用	
LESS_HANDLING_CHRG_COMM_FEE	float	减:手续费及佣金支出	
LESS_INTEREST_EXP	float	减:利息支出	
LESS_NON_OPERA_EXP	float	减:营业外支出	
LESS_OPERA_COST	float	减:营业成本	
LESS_REINSUR_PREM	float	减:分出保费	
LESS_SELLING_EXP	float	减:销售费用	
MARKET_CODE	str	证券代码	

MIN_INT_INC	float	少数股东损益	
NET_EXPOSURE_HEDGING_GAIN	float	净敞口套期收益	
NET_HANDLING_CHRG_COMM_FEE	float	手续费及佣金净收入	
NET_INC_EC_ASSET_MGMT_BUS	float	受托客户资产管理业务净收入	
NET_INC_SEC_BROK_BUS	float	代理买卖证券业务净收入	
NET_INC_SEC_UW_BUS	float	证券承销业务净收入	
NET_INTEREST_INC	float	利息净收入	
NET_PRO_AFTER_DED_NR_GL	float	扣除非经常性损益后净利润（扣除少数股东损益）	
NET_PRO_AFTER_DED_NR_GL_COR	float	扣除非经常性损益后的净利润(财务重要指标(更正前))	
NET_PRO_EXCL_MIN_INT_INC	float	净利润(不含少数股东损益)	
NET_PRO_INCL_MIN_INT_INC	float	净利润(含少数股东损益)	
NET_PRO_UNDER_INT_ACC_STA	float	国际会计准则净利润	
OPERA_EXP	float	营业支出	
OPERA_PROFIT	float	营业利润	
OPERA_REV	float	营业收入	
OTH_ASSETS_IMPAIR_LOSS	float	其他资产减值损失	
OTH_BUS_COST	float	其他业务成本	
OTH_BUS_INC	float	其他业务收入	
OTH_COMPRE_IN	float	其他综合收益	

C			
OTH_INCOME	float	其他收益	
OTH_NET_OPERA_INC	float	其他经营净收益	
PLUS_NET_FX_INC	float	加:汇兑净收益	
PLUS_NET_GAIN_CHG_FV	float	加:公允价值变动净收益	
PLUS_NET_INV_INC	float	加:投资净收益	
PLUS_NON_OPERA_REV	float	加:营业外收入	
PLUS_OTH_NET_BUS_INC	float	加:其他业务净收益	
PREFERRED_SHARE_DIV_PAYABLE	float	应付优先股股利	
PREM_BUS_INC	float	保费业务收入	
RD_EXP	float	研发费用	
REINSURANCE_EXP	float	分保费用	
REPORTING_PERIOD	str	报告期	
SECURITY_NAME	str	证券简称	
SPE_BAL_NET_PRO_MARG	float	净利润差额(特殊报表科目)	
SPE_BAL_OPERA_PRO_MARG	float	营业利润差额(特殊报表科目)	
SPE_BAL_TOT_OPERA_COST_DIF	float	营业总成本差额(特殊报表科目)	
SPE_BAL_TOT_OPERA_INC_DIF	float	营业总收入差额(特殊报表科目)	
SPE_BAL_TOT_PRO_MARG	float	利润总额差额(特殊报表科目)	

SPE_TOT_OPERA_COST_DIF_STATE	str	营业总成本差额说明(特殊报表科目)	
SPE_TOT_OPERA_INC_DIF_STATE	str	营业总收入差额说明(特殊报表科目)	
SURR_VALUE	float	退保金	
TOT_BAL_NET_PRO_MARG	float	净利润差额(合计平衡项目)	
TOT_BAL_OPERA_PRO_MARG	float	营业利润差额(合计平衡项目)	
TOT_BAL_TOT_PRO_MARG	float	利润总额差额(合计平衡项目)	
TOT_COMPEN_EXP	float	赔付总支出	
TOT_COMPRE_INC	float	综合收益总额	
TOT_COMPRE_INC_MIN_SHARE	float	综合收益总额(少数股东)	
TOT_COMPRE_INC_PARENT_COMP	float	综合收益总额(母公司)	
TOT_OPERA_COST	float	营业总成本	
TOT_OPERA_COST2	float	营业总成本 2	
TOT_OPERA_REV	float	营业总收入	
TOTAL_PROFIT	float	利润总额	
TRANSFER_HOUSING_REVO_FUNDS	float	住房周转金转入	
TRANSFER_OTHERS	float	其他转入	
TRANSFER_SURPLUS_RESERVE	float	盈余公积转入	
UNCONFIRMED_INV_LOSS	float	未确认投资损失	
WITHDRAW_ANY	float	提取任意盈余公积	

_SURPLUS_RESV		金	
WITHDRAW_ENT_DEVELOP_FUND	float	提取企业发展基金	
WITHDRAW_LEG_PUB_WEL_FUND	float	提取法定公益金	
WITHDRAW_LEG_SURPLUS	float	提取法定盈余公积	
WITHDRAW_RESV_FUND	float	提取储备基金	

3.5.5.4 业绩快报

函数接口： get_profit_express
功能描述： 获取指定股票列表的上市公司的业绩快报数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	报告期，本地数据缓存方案
end_date	int	否	报告期，本地数据缓存方案

输出参数：

参数	数据类型	解释
profit_express	dataframe	column 为 profit_express 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
```

```
                    end_date=today)
profit_express = info_data_object.get_profit_express (all_code_list)
```

profit_express 的字段说明：

参数	数据类型	字段说明	备注
MARKET_CODE	str	证券代码	
REPORTING_PERIOD	str	报告期	报告内容记录的截止时间点，报告成果的时期
ANN_DATE	str	公告日期	公告发布当天的日期；有多个阶段的事件，首次披露该事件的日期
ACTUAL_ANN_DATE	str	实际公告日期	实际数据来源公告的日期；更正发生公告的日期
TOTAL_ASSETS	float64	总资产(元)	指经济实体拥有或控制的能带来经济利益的全部资产
NET_PRO_EXCL_MIN_INT_INC	float64	净利润(元)	企业合并净利润中归属于母公司股东所有的那部分利润
TOT_OPERA_REV	float64	营业总收入(元)	企业从事销售商品、提供劳务和让渡资产使用权等日常业务过程形成的经济利益的总流入
TOTAL_PROFIT	float64	利润总额(元)	企业一定时期内的纯收入扣除应交纳后的余额
OPERA_PROFIT	float64	营业利润(元)	企业在其全部销售业务中实现的利润
EPS_BASIC	float64	每股收益-基本(元)	企业按照属于普通股股东的当期净利润，除以发行在外普通股的加权平均数计算得到的每股收益
TOT_SHARE_EQU_EXCL_MIN_INT	float64	股东权益合计(不含少数股东权益)(元)	公司集团的所有者权益中归属于母公司所有者权益的部分
IS_AUDIT	float64	是否审计	1:是 0：否
ROE_WEIGHTED	float64	净资产收益率-加权(%)	经营期间净资产赚取利润的结果的一个动态指标，反应企业净资产创造利润的能力
LAST_YEAR_REVISED_NET_PRO	float64	去年同期修正后净利润	元

PERFORMANCE_SUMMARY	str	业绩简要说明	针对业绩快报的简单说明
NET_ASSET_PS	float64	每股净资产	元
MEMO	str	备注	附加的注解说明
YOY_GR_GROSS_PRO	float64	同比增长率:营业利润	%
YOY_GR_GROSS_REV	float64	同比增长率:营业总收入	%
YOY_GR_NET_PROFIT_PARENT	float64	同比增长率:归属母公司股东的净利润	%
YOY_GR_TOT_PRO	float64	同比增长率:利润总额	%
YOY_ID_WAROE	float64	同比增减:加权平均净资产收益率	%
YOY_GR_EPS_BASIC	float64	同比增长率:基本每股收益	%
GROWTH_RATE_EQUITY	float64	比年初增长率:归属母公司的股东权益	%
GROWTH_RATE_ASSETS	float64	比年初增长率:总资产	%
GROWTH_RATE_NAPS	float64	比年初增长率:归属于母公司股东的每股净资产	%
LAST_YEAR_TOT_OPERA_REV	float64	去年同期营业总收入	元
LAST_YEAR_TOTAL_PROFIT	float64	去年同期利润总额	元
LAST_YEAR_OPERA_PRO	float64	去年同期营业利润	元

LAST_YEAR_EPS_DILUTED	float64	去年同期每股收益	元
LAST_YEAR_NET_PROFIT	float64	去年同期净利润	元
INITIAL_NET_ASSET_PS	float64	期初每股净资产	元
INITIAL_NET_ASSETS	float64	期初净资产	元

3.5.5.5 业绩预告

函数接口：get_profit_notice

功能描述：获取指定股票列表的上市公司的业绩预告数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	报告期，本地数据缓存方案
end_date	int	否	报告期，本地数据缓存方案

输出参数：

参数	数据类型	解释
profit_notice	dataframe	column 为 profit_notice 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
profit_notice = info_data_object.get_profit_notice (all_code_list)
```

profit_notice 的字段说明：

参数	数据类型	字段说明	备注
MARKET_CODE	str	证券代码	
SECURITY_NAME	str	证券简称	
P_TYPECODE	str	业绩预告类型代码	1：不确定 2：略减 3：略增 4：扭亏 5：其他 6：首亏 7：续亏 8：续盈 9：预减 10：预增 11：持平
REPORTING_PERIOD	str	报告期	分为年度、半年度、季度
ANN_DATE	str	公告日期	公告发布当天的日期
P_CHANGE_MAX	float64	预告净利润变动幅度上限（%）	对于净利润金额同比变动幅度预计的最高值
P_CHANGE_MIN	float64	预告净利润变动幅度下限（%）	对于净利润金额同比变动幅度预计的最低值
NET_PROFIT_MAX	float64	预告净利润上限（万元）	对于净利润金额预计的最高值
NET_PROFIT_MIN	float64	预告净利润下限（万元）	对于净利润金额预计的最低值
FIRST_ANN_DATE	str	首次公告日	首次披露本报告期业绩预告内容的公告日期
P_NUMBER	float64	公布次数	同一报告期的业绩预告公告的披露次数
P_REASON	str	业绩变动原因	
P_SUMMARY	str	业绩预告摘要	
P_NET_PARENT_FIRM	float64	上年同期归母净利润	业绩预告中直接公布的上年同期归母净利润
REPORT_TYPE	str	报告期名称	参看报告期名称

3.5.6 股东股本数据

3.5.6.1 十大股东数据

函数接口：get_share_holder
功能描述：获取指定股票列表的上市公司的十大股东数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	到期日期，本地数据缓存方案
end_date	int	否	到期日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
share_holder	dataframe	column 为 share_holder 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
share_holder = info_data_object.get_share_holder (all_code_list)
```

share_holder 的字段说明：

参数	数据类型	字段说明	备注
ANN_DATE	str	公告日期,	
MARKET_CODE	str	证券代码	
HOLDER_ENDDATE	str	到期日期	
HOLDER_TYPE	int	股东类别	10:十大股东

			20:流通股前十大股东
QTY_NUM	int	持股量序号	
HOLDER_NAME	str	股东名称	
HOLDER_HOLDER_CATEGORY	int	股东性质	1：个人 2：公司
HOLDER_QUANTITY,	float	持股数（股）	
HOLDER_PCT	float	持股比例（%），	
HOLDER_SHARECATEGORYNAME	str	股份类型	当 HOLDER_TYPE 为 20:流通股前十大股东时，全部为‘A Float Holder’
FLOAT_QTY	float	流通股数量	

3.5.6.2 股东户数

函数接口： get_holder_num
功能描述： 获取指定股票列表的上市公司的股东户数数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	股东户数统计的截止日期，本地数据缓存方案
end_date	int	否	股东户数统计的截止日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
holder_num	dataframe	column 为 holder_num 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
```

```
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
holder_num = info_data_object.get_holder_num (all_code_list)
```

holder_num 的字段说明：

参数	数据类型	字段说明
MARKET_CODE	string	证券代码
ANN_DT	string	公告日期
HOLDER_ENDDATE	string	股东户数统计的截止日期
HOLDER_TOTAL_NUM	float	A 股、B 股、H 股、境外股的总户数
HOLDER_NUM	float	A 股股东户数

3.5.6.3 股本结构

函数接口：get_equity_structure

功能描述：获取指定股票列表的上市公司的股本结构数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	变动日期，本地数据缓存方案
end_date	int	否	变动日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
equity_structure	dataframe	column 为 equity_structuree 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
```

```
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
equity_structure = info_data_object.get_equity_structure (all_code_list)
```

equity_structure 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	string	证券代码	
ANN_DATE	string	公告日期	
CHANGE_DATE	string	变动日期	注：股票分红送转股时的红股上市日;股票增发时的新股上市日
SHARE_CHANGE_REASON_STR	string	股本变动原因描述	
EX_CHANGE_DATE	string	除权日期	股票分红送转股时的除权日;股票增发时的登记日
CURRENT_SIGN	int	最新标志	1:是 0:否
IS_VALID	int	是否有效	用来区分除权日相同时，是否为公司公告公布的最新股份数 1:是 0:否
TOT_SHARE	float	总股本(万股)	
FLOAT_SHARE	float	流通股(万股)	
FLOAT_A_SHARE	float	流通 A 股(万股)	
FLOAT_B_SHARE	float	流通 B 股(万股)	
FLOAT_HK_SHARE	float	香港流通股(万股)	
FLOAT_OS_SHARE	float	海外流通股(万股)	
TOT_TRADABLE_SHARE	float	流通股合计	
RTD_A_SHARE_INST	float	限售 A 股(其他内资持股:机构配售股)	
RTD_A_SHARE_DOMESNP	float	限售 A 股(其他内资持股:境内自然人持股)	
RTD_SHARE_SENIOR	float	限售股份(高管持股)(万股)	
RTD_A_SHARE_FOREIGN	float	限售 A 股(外资持股)	
RTD_A_SHARE_FORJUR	float	限售 A 股(境外法人持股)	
RTD_A_SHARE_FORNP	float	限售 A 股(境外自然人持股)	
RESTRICTED_B_SHAR	float	限售 B 股(万股)	

E			
OTHER_RTD_SHARE	float	其他限售股	
NON_TRADABLE_SHARE	float	非流通股	
NTRD_SHARE_STATE_PCT	float	非流通股(国有股)	
NTRD_SHARE_STATE	float	非流通股(国家股)	
NTRD_SHARE_STATEJUR	float	非流通股(国有法人股)	
NTRD_SHARE_DOMESJUR	float	非流通股(境内法人股)	
NTRD_SHARE_DOMES_INITIATOR	float	非流通股(境内法人股:境内发起人股)	
NTRD_SHARE_IPOJURIS	float	非流通股(境内法人股:募集法人股)	
NTRD_SHARE_GENJURIS	float	非流通股(境内法人股:一般法人股)	
NTRD_SHARE_STRA_INVESTOR	float	非流通股(境内法人股:战略投资者持股)	
NTRD_SHARE_FUND	float	非流通股(境内法人股:基金持股)	
NTRD_SHARE_NAT	float	非流通股(自然人股)	
TRAN_SHARE	float	转配股(万股)	
FLOAT_SHARE_SENIOR	float	流通股(高管持股)	
SHARE_INEMP	float	内部职工股(万股)	
PREFERRED_SHARE	float	优先股(万股)	
NTRD_SHARE_NLIST_FRGN	float	非流通股(非上市外资股)	
STAQ_SHARE	float	STAQ 股(万股)	
NET_SHARE	float	NET 股(万股)	
SHARE_CHANGE_REASON	string	股本变动原因	
TOT_A_SHARE	float	A 股合计	
TOT_B_SHARE	float	B 股合计	
OTCA_SHARE	float	三板 A 股	
OTCB_SHARE	float	三板 B 股	
TOT_OTC_SHARE	float	三板合计	
SHARE_HK	float	香港上市股	
PRE_NON_TRADABLE	float	股改前非流通股	

_SHARE			
RESTRICTED_A_SHARE	float	限售 A 股(万股)	
RTD_A_SHARE_STATE	float	限售 A 股(国家持股)	
RTD_A_SHARE_STATEJUR	float	限售 A 股(国有法人持股)	
RTD_A_SHARE_OTHER_DOMES	float	限售 A 股(其他内资持股)	
RTD_A_SHARE_OTHER_DOMESJUR	float	限售 A 股(其他内资持股:境内法人持股)	
TOT_RESTRICTED_SHARE	float	限售股合计	

3.5.6.4 股权冻结/质押

函数接口： get_equity_pledge_freeze

功能描述： 获取指定股票列表的上市公司的股权冻结/质押数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	公告日期，本地数据缓存方案
end_date	int	否	公告日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
equity_pledge_freeze	dict	key：code value:dataframe column 为 equity_pledge_freeze 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
```

```
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
equity_pledge_freeze = info_data_object.get_equity_pledge_freeze (all_code_list)
```

equity_pledge_freeze 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	string	证券代码	
ANN_DATE	string	公告日期	
HOLDER_NAME	string	股东名称	
HOLDER_TYPE_CODE	int	股东类型代码	2:公司 3:个人
TOTAL_HOLDING_SHR"	float	持股总数（万股）	
TOTAL_HOLDING_SHR_RATIO	float	持股总数占公司总股本比例	
FRO_SHARES	float	本次冻结/质押股数	
FRO_SHR_TO_TOTAL_HOLDING_RATIO	float	本次冻结/质押占所持股比例	
FRO_SHR_TO_TOTAL_RATIO	float	本次冻结/质押占总股本比例	
TOTAL_PLEDGE_SHR	float	累计冻结/质押股数	
IS_EQUITY_PLEDGE_REPO	int	是否股权质押回购	1:是 0:否
BEGIN_DATE	string	冻结/质押起始日	
END_DATE	string	解冻/解押日期	
IS_DISFROZEN	int	是否质押或解冻	1:是 0:否
FROZEN_INSTITUTION	string	执行冻结机构/质权方	
DISFROZEN_TIME	string	解压或解冻日期	
SHR_CATEGORY_	int	股份性质类别代码	1:法人股 2:个人股 3:国有

CODE			股 4:国有股,法人股 5:流通股 6:流通股,限售流通股 7:外资股 8:限售流通股 9:优先股
FREEZE_TYPE	int	冻结/质押类型	1:质押 2:司法 3:质押式回购

3.5.6.5 限售股解禁

函数接口： get_equity_restricted

功能描述： 获取指定股票列表的上市公司的限售股解禁数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	解禁日期，本地数据缓存方案
end_date	int	否	解禁日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
equity_restricted	dict	key：code value:dataframe column 为 equity_restricted 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
equity_restricted = info_data_object.get_equity_restricted (all_code_list)
```

equity_restricted 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	string	证券代码	
LIST_DATE	string	解禁日期	
SHARE_RATIO	float	解禁股占总股本比(%)	
SHARE_LST_TYPE_NAME	string	解禁股份类型名称	
SHARE_LST	int	解禁数量（股）	
SHARE_LST_IS_ANN	int	上市数量是否公布值	0：否，为预测值 1: 是，为实际公布值
CLOSE_PRICE	float	前日收盘价（元）	
SHARE_LST_MARKET_VALUE	float	解禁市值（元）	SHARE_LST* CLOSE_PRICE

3.5.7 股东权益数据

3.5.7.1 分红数据

函数接口： get_dividend
功能描述： 获取指定股票列表的上市公司的分红数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	公告日期，本地数据缓存方案
end_date	int	否	公告日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
dividend	dataframe	column 为 dividend 的字段 index 为序号（无意义）

```
# 第一步 登录 api
```

```
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
dividend = info_data_object.get_dividend(all_code_list)
```

dividend 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	string	证券代码	
DIV_PROGRESS	string	方案进度	参看股票分红进度代码表
DVD_PER_SHARE_STK	float	每股送转	
DVD_PER_SHARE_PRE_TAX_CASH	float	每股派息(税前)(元)	
DVD_PER_SHARE_AFTER_TAX_CASH	float	每股派息(税后)(元)	
DATE_EQY_RECORD	string	股权登记日	
DATE_EX	string	除权除息日	
DATE_DVD_PAYOUT	string	派息日	
LISTINGDATE_OF_DVD_SHR	string	红股上市日	
DIV_PRELANDATE	string	预案公告日	董事会预案公告日期
DIV_SMTGDATE	string	股东大会公告日	
DATE_DVD_ANN	string	分红实施公告日	
DIV_BASEDATE	string	基准日期	
DIV_BASESHARE	float	基准股本(万股)	
CURRENCY_CODE	string	货币代码	
ANN_DATE	string	公告日期	
IS_CHANGED	int	方案是否变更	1：有变更过 0：未变更
REPORT_PERIOD	string	分红年度	
DIV_CHANGE	string	方案变更说明	
DIV_BONUSRATE	float	每股送股比例	
DIV_CONVERSEDRATE	float	每股转增比例	
REMARK	string	备注	
DIV_PREANN_DATE	string	预案预披露公告日	股东提议的公告日期
DIV_TARGET	string	分红对象	

3.5.7.2 配股数据

函数接口：get_right_issue
功能描述：获取指定股票列表的上市公司的配股数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	公告日期，本地数据缓存方案
end_date	int	否	公告日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
right_issue	dataframe	column 为 right_issue 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
right_issue = info_data_object.get_right_issue(all_code_list)
```

right_issue 的字段说明：

字段名称	类型	字段说明	备注
MARKET_CODE	string	证券代码	
PROGRESS	int	方案进度	参看股票配股进度代码表
PRICE	double	配股价格(元)	
RATIO	double	配股比例	
AMT_PLAN	double	配股计划数量(万股)	
AMT_REAL	double	配股实际数量(万股)	
COLLECTION_FUND	double	募集资金(元)	
SHAREB_REG_DATE	string	股权登记日	
EX_DIVIDEND_DATE	string	除权日	
LISTED_DATE	string	配股上市日	

PAY_START_DATE	string	缴款起始日	
PAY_END_DATE	string	缴款终止日	
PREPLAN_DATE	string	预案公告日	
SMTG_ANN_DATE	string	股东大会公告日	
PASS_DATE	string	发审委通过公告日	
APPROVED_DATE	string	证监会核准公告日	
EXECUTE_DATE	string	配股实施公告日	
RESULT_DATE	string	配股结果公告日	
LIST_ANN_DATE	string	上市公告日	
GUARANTOR	string	基准年度	
GUARTYPE	double	基准股本(万股)	
RIGHTSISSUE_CODE	string	配售代码	
ANN_DATE	string	公告日期	
RIGHTSISSUE_YEAR	string	配股年度	
RIGHTSISSUE_DESC	string	配股说明	
RIGHTSISSUE_NAME	string	配股简称	
RATIO_DENOMINATOR	double	配股比例分母	
RATIO_MOLECULAR	double	配股比例分子	
SUBS_METHOD	string	认购方式	
EXPECTED_FUND_RAISING	double	预计募集资金(元)	

3.5.8 融资融券数据

3.5.8.1 融资融券成交汇总

函数接口： get_margin_summary
功能描述： 获取指定日期的上市公司的融资融券成交汇总数据
输入参数：

参数	数据类型	必选	解释
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日，本地数据缓存方案
end_date	int	否	交易日，本地数据缓存方案

输出参数：

参数	数据类型	解释
margin_summary	dataframe	column 为 margin_summary 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
margin_summary = info_data_object.get_margin_summary()
```

margin_summary 的字段说明：

字段名称	类型	字段说明
TRADE_DATE	string	交易日期
SUM_BORROW_MONEY_BALANCE	float	融资余额(元)
SUM_PURCH_WITH_BORROW_MONEY	float	融资买入额(元)
SUM_REPAYMENT_OF_BORROW_MONEY	float	融资偿还额(元)
SUM_SEC_LENDING_BALANCE	float	融券余额(元)
SUM_SALES_OF_BORROWED_SEC	int	融券卖出量(股,份,手)
SUM_MARGIN_TRADE_BALANCE	float	融资融券余额(元)

3.5.8.2 融资融券交易明细

函数接口： get_margin_detail
功能描述： 获取指定股票列表的上市公司的融资融券交易明细数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日，本地数据缓存方案
end_date	int	否	交易日，本地数据缓存方案

输出参数：

参数	数据类型	解释
margin_detail	dict	key：code

		value:dataframe column 为 margin_detail 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
margin_detail = info_data_object.get_margin_detail(all_code_list)
```

margin_detail 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	证券代码
SECURITY_NAME	string	证券简称
TRADE_DATE	string	交易日期
BORROW_MONEY_BALANCE"	float	融资余额(元)
PURCH_WITH_BORROW_MONEY	float	融资买入额(元)
REPAYMENT_OF_BORROW_MONEY	float	融资偿还额(元)
SEC_LENDING_BALANCE	float	融券余额(元)
SALES_OF_BORROWED_SEC	int	融券卖出量(股,份,手)
REPAYMENT_OF_BORROW_SEC	int	融券偿还量(股,份,手)
SEC_LENDING_BALANCE_VOL	int	融券余量(股,份,手)
MARGIN_TRADE_BALANCE	float	融资融券余额(元)

3.5.9 交易异动数据

3.5.9.1 龙虎榜

函数接口：get_long_hu_bang
功能描述：获取指定股票列表的上市公司的龙虎榜数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日，本地数据缓存方案
end_date	int	否	交易日，本地数据缓存方案

输出参数：

参数	数据类型	解释
long_hu_bang	dataframe	column 为 long_hu_bang 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
long_hu_bang = info_data_object.get_long_hu_bang(all_code_list)
```

long_hu_bang 的字段说明：

参数	数据类型	字段说明	备注
MARKET_CODE	string	证券代码	
TRADE_DATE	string	交易日期	
SECURITY_NAME	string	证券名称	
REASON_TYPE	string	上榜原因类型	

REASON_TYPE_NAME	string	上榜原因	
CHANGE_RANGE	float	涨跌幅（%）	
TRADER_NAME	string	营业部名称	
BUY_AMOUNT	float	买入金额（元）	
SELL_AMOUNT	float	卖出金额（元）	
FLOW_MARK	int	买卖表示	1 表示买入，2 表示卖出
TOTAL_AMOUNT	float	实际交易金额（元）	
TOTAL_VOLUME	float	实际交易量（万股）	

3.5.9.2 大宗交易

函数接口：get_block_trading
功能描述：获取指定股票列表的大宗交易数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 A 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日，本地数据缓存方案
end_date	int	否	交易日，本地数据缓存方案

输出参数：

参数	数据类型	解释
block_trading	dataframe	column 为 block_trading 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
```

```
today = calendar[-1]
all_code_list = base_data_object.get_hist_code_list(security_type='EXTRA_STOCK_A_SH_SZ', start_date=20130101,
                        end_date=today)
block_trading = info_data_object. block_trading (all_code_list)
```

block_trading 的字段说明：

参数	数据类型	字段说明
MARKET_CODE	string	证券代码
TRADE_DATE	string	交易日期
B_SHARE_PRICE	float	成交价（元）
B_SHARE_VOLUME	float	成交量（万股）
B_FREQUENCY	int	笔数
BLOCK_AVG_VOLUME	float	每笔成交数量（万股份）
B_SHARE_AMOUNT	float	成交金额（万元）
B_BUYER_NAME	string	买方营业部名称
B_SELLER_NAME	string	卖方营业部名称

3.5.10期权数据

3.5.10.1 期权基本资料

函数接口：get_option_basic_info

功能描述：获取指定期权的基本资料（沪深交易所的 ETF 期权）

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 期权的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'””
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
option_basic_info	dataframe	column 为 option_basic_info 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
code_list = base_data_object.get_option_code_list(security_type='EXTRA_ETF_OP')
hist_code_list =
base_data_object.get_hist_code_list(security_type='EXTRA_ETF_OP", start_date=20130101, end
_date=today)
option_basic_info =info_data_object.get_option_basic_info(code_list, is_local=False)
```

option_basic_info 的字段说明：

参数	数据类型	字段说明	备注
CONTRACT_FULL_NAME	string	合约全称	
CONTRACT_TYPE	string	合约类别	C 表示认购 P 表示认沽
DELIVERY_MONTH	string	交割月份	
EXPIRY_DATE	string	到期日	
EXERCISE_PRICE	float	行权价格	
EXERCISE_END_DATE	string	最后行权日	
START_TRADE_DATE	string	开始交易日	
LISTING_REF_PRICE	float	挂牌基准价	
LAST_TRADE_DATE	string	最后交易日	
EXCHANGE_CODE	string	合约交易所代码	
DELIVERY_DATE	string	最后交割日	
CONTRACT_UNIT	Int	合约单位	
IS_TRADE	string	是否交易	
EXCHANGE_SHORT_NAME	string	合约交易所简称	
CONTRACT_ADJUST_FLAG	string	合约调整标志	
MARKET_CODE	string	合约代码	

3.5.10.2 期权标准合约属性

函数接口：get_option_std_ctr_specs
功能描述：获取指定期权标准合约属性（沪深交易所的 ETF 期权）
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 的的代码列表，目前包含 159919.SZ 159915.SZ 159922.SZ 159901.SZ 510300.SH 588000.SH 588080.SH 510050.SH 510500.SH
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
option_std_ctr_specs	dataframe	column 为 option_std_ctr_specs 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
option_std_ctr_specs =info_data_object.get_option_std_ctr_specs(['510050.SH'], is_local=False)
```

option_std_ctr_specs 的字段说明：

参数	数据类型	字段说明	备注
EXERCISE_DATE	string	期权行权日	
CONTRACT_UNIT	int	合约单位	
POSITION_DECLARE_MIN	string	头寸申报下限	
QUOTE_CURRENCY_UNIT	string	报价货币单位	
LAST_TRADING_DATE	string	最后交易日	

POSITION_LIMIT	string	头寸限制	
DELIST_DATE	string	退市日期	
NOTIONAL_VALUE	string	立约价值	
EXERCISE_METHOD	string	行权方式	
DELIVERY_METHOD	string	交割方式	
SETTLEMENT_MONTH	string	合约结算月份	
TRADING_FEE	string	交易费用	
EXCHANGE_NAME	string	交易所名称	
OPTION_EN_NAME	string	期权英文名称	
CONTRACT_VALUE	float	合约价值	
IS_SIMULATION	int	是否仿真合约	0 否 1 是
CONTRACT_UNIT_DIMENSION	string	合约单位量纲	
OPTION_STRIKE_PRICE	string	期权行权价	
IS_SIMULATION_TRADE	string	是否仿真交易 0 否 1 是	
LISTED_DATE	string	上市日期	
OPTION_NAME	string	期权名称	
PREMIUM	string	期权金	
OPTION_TYPE	string	期权类型	ETF 期权等
TRADING_HOURS_DESC	string	交易时间说明	
FINAL_SETTLEMENT_DATE	string	最后结算日	
FINAL_SETTLEMENT_PRICE	string	最后结算价	
MIN_PRICE_UNIT	string	最小报价单位	
MARKET_CODE	string	市场代码	
CONTRACT_MULTIPLIER	int	合约乘数	

3.5.10.3 期权月合约属性变动

函数接口：get_option_mon_ctr_specs

功能描述：获取指定期权月合约属性变动（沪深交易所的 ETF 期权）

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 期权的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
block_trading	dataframe	column 为 block_trading 的字段 index 为序号（无意义）

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
today = calendar[-1]
code_list = base_data_object.get_option_code_list(security_type='EXTRA_ETF_OP')
hist_code_list =
base_data_object.get_hist_code_list(security_type='EXTRA_ETF_OP", start_date=20130101, end
_date=today)
option_mon_ctr_specs =info_data_object.get_option_mon_ctr_specs(code_list, is_local=False)
```

option_mon_ctr_specs 的字段说明：

参数	数据类型	字段说明
CODE_OLD	string	原交易代码
CHANGE_DATE	string	调整日期
MARKET_CODE	string	市场代码
NAME_NEW	string	新合约简称
EXERCISE_PRICE_NEW	float	新行权价(元)
NAME_OLD	string	原合约简称
CODE_NEW	string	新交易代码
EXERCISE_PRICE_OLD	float	原行权价(元)
UNIT_OLD	float	原合约单位(股)

UNIT_NEW	float	新合约单位(股)
CHANGE_REASON	string	调整原因

3.5.11ETF 数据

3.5.11.1 ETF 每日最新申赎数据

函数接口： get_etf_pcf

功能描述：获取指定 ETF 的申赎和成分股数据（沪深交易所的 ETF）

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 的的代码列表，可见示例

输出参数：

参数	数据类型	解释
etf_pcf_info	dataframe	column 为 etf_pcf_info 的字段 index 为 ETF 代码
etf_pcf_constituent	dict	字典的 key：ETF 代码 字典的 value：dataframe， column 为 etf_pcf_constituent 的字段， index 为序号

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
base_data_object = ad.BaseData()
code_list = base_data_object.get_hist_code_list(security_type='EXTRA_ETF')
etf_pcf_info, etf_pcf_constituent = base_data_object.get_etf_pcf(code_list)
```

etf_pcf_info 的字段说明：

参数	数据类型	字段说明	备注
creation_redemption_unit	int	每个篮子对应的 ETF 份数	
max_cash_ratio	string	最大现金替代比例	
publish	string	是否发布 IOPV	Y=是,N=否
creation	string	是否允许申购	Y=是,N=否(仅深圳有效)
redemption	string	是否允许赎回	Y=是,N=否(仅深圳

			有效)
creation_redemption_switch	string	申购赎回切换	(仅上海有效,0-不允许申购/赎回,1-申购和赎回皆允许,2-仅允许申购,3-仅允许赎回)
record_num	int	深市成份证券数目	
total_record_num	int	所有成份证券数量	
estimate_cash_component	int	预估现金差额	
trading_day	int	当前交易日	(格式:YYYYMMDD)
pre_trading_day	int	前一交易日	(格式:YYYYMMDD)
cash_component	int	前一日现金差额	
nav_per_cu	int	前一日最小申赎单位净值	
nav	int	前一日基金份额净值	
symbol	string	基金名称	仅深圳有效
fund_management_company	string	基金公司名称	仅深圳有效
underlying_security_id	string	拟合指数代码	仅深圳有效
underlying_security_id_source	string	拟合指数市场	参考 Market，仅深圳有效
dividend_per_cu	int	红利金额	
creation_limit	int	累计申购总额限制	为 0 表示没有限制(仅深圳有效)
redemption_limit	int	累计赎回总额限制	0 表示没有限制(仅深圳有效)
creation_limit_per_user	int	单个账户累计申购总额限制	0 表示没有限制(仅深圳有效)
redemption_limit_per_user	int	单个账户累计赎回总额限制	0 表示没有限制(仅深圳有效)

net_creation_limit	int	净申购总额限制	0 表示没有限制(仅深圳有效)
net_redemption_limit	int	净赎回总额限制	0 表示没有限制(仅深圳有效)
net_creation_limit_per_user	int	单个账户净申购总额限制	0 表示没有限制(仅深圳有效)
net_redemption_limit_per_user	int	单个账户净赎回总额限制	0 表示没有限制(仅深圳有效)

etf_pcf_constituent 的字段说明：

参数	数据类型	字段说明	备注
underlying_symbol	string	成份证券简称	
component_share	int	成份证券数量	
substitute_flag	string	现金替代标志	//**深圳现金替代标志* //0=禁止现金替代(必须有证券),1=可以进行现金替代(先用证券,证券不足时差额部分用现金替代),2=必须用现金替代 //**上海现金替代标志* //ETF 公告文件 1.0 版格式 //0 –沪市不可被替代, 1 – 沪市可以被替代, 2 – 沪市必须被替代, 3 – 深市退补现金替代, 4 – 深市必须现金替代 //5 – 非沪深市场成份证券退补现金替代(不适用于跨沪深港 ETF 产品), 6 – 非沪深市场成份证券必须现

			金替代(不适用于跨沪深港 ETF 产品) //ETF 公告文件 2.1 版格式 //0 –沪市不可被替代, 1 – 沪市可以被替代, 2 – 沪市必须被替代, 3 – 深市退补现金替代, 4 – 深市必须现金替代 //5 – 非沪深市场成份证券退补现金替代(不适用于跨沪深港 ETF 产品), 6 – 非沪深市场成份证券必须现金替代(不适用于跨沪深港 ETF 产品) //7 – 港市退补现金替代(仅适用于跨沪深港 ETF 产品), //8 – 港市必须现金替代(仅适用于跨沪深港 ETF 产品)
premium_ratio	int	溢价比例	
discount_ratio	int	折价比例	
creation_cash_substitute	int	申购替代金额	仅深圳有效
redemption_cash_substitute	int	赎回替代金额	仅深圳有效
substitution_cash_amount	int	替代总金额	仅上海有效
underlying_security_id	string	成份证券所属市场 ID	仅对跨市场债券(银行间)ETF 启用

3.5.11.2 ETF 基金份额

函数接口：get_fund_share
功能描述：获取指定 ETF 列表的基金份额数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	变动日期，本地数据缓存方案
end_date	int	否	变动日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
fund_share	dict	key：code value:dataframe column 为 fund_share 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
etf_code_list = base_data_object.get_code_list(security_type='EXTRA_ETF')
# ETF 份额
fund_share = info_data_object.get_fund_share(etf_code_list, is_local=False)
```

fund_share 的字段说明：

字段名称	类型	字段说明	备注
FUND_SHARE	float	基金份额(万份)	
CHANGE_REASON	string	份额变动原因	
IS_CONSOLIDATED_DATA	int	是否合并数据	0：非合并数据 1：合并数据 2：合并数据，但该基金代码属于不实际交易基金

MARKET_CODE	string	市场代码	
ANN_DATE	string	公告日期	
TOTAL_SHARE	float	基金总份额(万份)	
CHANGE_DATE	string	变动日期	
FLOAT_SHARE	float	流通份额(万份)	

3.5.11.3 ETF 每日收盘 iopv

函数接口：get_fund_iopv
功能描述：获取指定 ETF 列表的基金份额数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深 ETF 的的代码列表，可见示例
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	变动日期，本地数据缓存方案
end_date	int	否	变动日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
fund_iopv	dict	key：code value:dataframe column 为 fund_iopv 的字段 index 为序号，无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
etf_code_list = base_data_object.get_code_list(security_type='EXTRA_ETF')
# ETF 份额
fund_iopv = info_data_object.get_fund_iopv(etf_code_list, is_local=False)
```

fund_iopv 的字段说明：

字段名称	类型	字段说明

MARKET_CODE	string	市场代码
PRICE_DATE	string	日期
IOPV_NAV	float	IOPV 收盘净值

3.5.12交易所指数数据

3.5.12.1 交易所指数成分股

函数接口：get_index_constituent
功能描述：获取指定交易所指数列表的成分股数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持沪深指数的的代码列表，可见示例，仅支持常用指数，约 600 多只，无返回数据则不支持。
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，仅从本地获取，不从服务器获取数据； False ，仅从服务器获取，不从本地获取数据； 因为原始数据的剔除日期会根据最新数据修改，所以第一次运行 is_local 需要设置成 False 才会从服务器获取数据。

输出参数：

参数	数据类型	解释
index_constituent	dict	key：code value:dataframe column 为 index_constituent 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list(security_type='EXTRA_INDEX_A')
index_constituent = info_data_object.get_index_constituent(code_list, is_local=False)
```

index_constituent 的字段说明：

字段名称	类型	字段说明	备注
INDEX_CODE	string	指数代码	
CON_CODE	string	成份股代码	
INDATE	string	纳入日期	
OUTDATE	string	剔除日期	未剔除时为 nan
INDEX_NAME	string	指数名称	

3.5.12.2 交易所指数成分股日权重

函数接口：get_index_weight

功能描述：获取指定交易所指数列表的成分股日权重数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持指数列表； 指数代码：支持以下 5 个指数 上证 50： 000016.SH 沪深 300： 000300.SH 中证 500： 000905.SH 中证 800： 000906.SH 中证 1000： 000852.SH
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	变动日期，本地数据缓存方案
end_date	int	否	变动日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
index_weight	dict	key：code value:dataframe column 为 index_weight 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
```

```
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
index_weight = info_data_object.get_index_weight(['000016.SH', '000300.SH', '000905.SH','000906.SH','000852.SH'], is_local=False)
```

index_weight 的字段说明：

字段名称	类型	字段说明
INDEX_CODE	string	指数代码
CON_CODE	string	标的代码
TRADE_DATE	string	生效日期
TOTAL_SHARE	float	总股本（股）
FREE_SHARE_RATIO	float	自由流通比例（%）（归档后）
CALC_SHARE	float	计算用股本（股）
WEIGHT_FACTOR	float	权重因子
WEIGHT	float	权重（%）
CLOSE	float	收盘价

3.5.13行业指数数据

3.5.13.1 行业指数基本信息

函数接口：get_industry_base_info
功能描述：获取行业指数的基本信息数据
输入参数：

参数	数据类型	必选	解释
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，仅从本地获取，不从服务器获取数据； False ，仅从服务器获取，不从本地获取数据； 因为原始数据的剔除日期会根据最新数据修改，所以第一次运行 is_local 需要设置成 False 才会从服务器获取数据。

输出参数：

参数	数据类型	解释
industry_base_info	dict	key：code value:dataframe column 为 industry_base_info 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
industry_base_info = info_data_object.get_industry_base_info()
```

industry_base_info 的字段说明：

字段名称	类型	字段说明	备注
INDEX_CODE	string	指数代码	
INDUSTRY_CODE	string	行业代码	
LEVEL_TYPE	int	指数类别 1：一级行业 2：二级行业 3：三级行业	
LEVEL1_NAME	string	一级行业	
LEVEL2_NAME	string	二级行业	
LEVEL3_NAME	string	三级行业	
IS_PUB	int	是否发布	1：已发布； 2：未发布
CHANGE_REASON	string	变动原因	

3.5.13.2 行业指数成分股

函数接口：get_industry_constituent

功能描述：获取指定行业指数列表的成分股数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持行业指数的的代码列表，可见示例，仅从 get_industry_base_info 取到的指数代码。

local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，仅从本地获取，不从服务器获取数据； False ，仅从服务器获取，不从本地获取数据； 因为原始数据的剔除日期会根据最新数据修改，所以第一次运行 is_local 需要设置成 False 才会从服务器获取数据。

输出参数：

参数	数据类型	解释
industry_constituent	dict	key：code value:dataframe column 为 industry_constituent 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
industry_base_info = info_data_object.get_industry_base_info()
industry_base_list = list(industry_base_info['INDEX_CODE'])
# 行业指数成分股
industry_constituent = info_data_object.get_industry_constituent(industry_base_list, is_local=False)
```

industry_constituent 的字段说明：

字段名称	类型	字段说明	备注
INDEX_CODE	string	指数代码	
CON_CODE	string	成份股代码	
INDATE	string	纳入日期	
OUTDATE	string	剔除日期	未剔除时为 nan
INDEX_NAME	string	指数名称	

3.5.13.3 行业指数成分股日权重

函数接口：get_industry_weight

功能描述：获取指定行业指数列表的成分股日权重数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持行业指数的的代码列表，可见示例，仅从 get_industry_base_info 取到的指数代码。
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日期，本地数据缓存方案
end_date	int	否	交易日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
industry_weight	dict	key：code value:dataframe column 为 industry_weight 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
industry_base_info = info_data_object.get_industry_base_info()
industry_base_list = list(industry_base_info['INDEX_CODE'])
# 行业指数日权重
industry_weight = info_data_object.get_industry_weight(industry_base_list)
```

industry_weight 的字段说明：

字段名称	类型	字段说明
WEIGHT	float	权重
CON_CODE	string	成份股代码
TRADE_DATE	string	交易日期
INDEX_CODE	string	指数代码

3.5.13.4 行业指数日行情

函数接口：get_industry_daily
功能描述：获取指定行业指数列表的日行情数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持行业指数的的代码列表，可见示例，仅从 get_industry_base_info 取到的指数代码。
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	交易日期，本地数据缓存方案
end_date	int	否	交易日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
industry_daily	dict	key：code value:dataframe column 为 industry_daily 的字段 index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
industry_base_info = info_data_object.get_industry_base_info()
industry_base_list = list(industry_base_info['INDEX_CODE'])
# 行业指数日行情
industry_daily = info_data_object.get_industry_daily(industry_base_list, is_local=False)
```

industry_daily 的字段说明：

字段名称	类型	字段说明
OPEN	float	开盘价
HIGH	float	最高价
CLOSE	float	收盘价
LOW	float	最低价
AMOUNT	float	成交金额(元)

VOLUME	float	成交量(股)
PB	float	指数市净率
PE	float	指数市盈率
TOTAL_CAP	float	总市值(万元)
A_FLOAT_CAP	float	A 股流通市值(万元)
INDEX_CODE	string	指数代码
PRE_CLOSE	float	昨收盘价
TRADE_DATE	string	交易日期

3.5.14 可转债数据

3.5.14.1 可转债发行

函数接口： get_kzz_issuance
功能描述： 获取指定可转债列表的可转债发行数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_issuance	dict	dataframe column 为 kzz_issuance 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_issuance = info_data_object.get_kzz_issuance(code_list, is_local=False)
```

kzz_issuance 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
STOCK_CODE	string	正股代码
CRNCY_CODE	string	货币代码
ANN_DT	string	公告日期
PRE_PLAN_DATE	string	预案公告日
SMTG_ANN_DATE	string	股东大会公告日
LISTED_ANN_DATE	string	上市公告日
LISTED_DATE	string	上市日期
PLAN_SCHEDULE	string	方案进度 1: 董事会预案 2: 股东大会通过 3: 实施 4: 未通过 5: 证监会通过 6: 达成转让意向 7: 签署转让协议 8: 国资委批准 9: 商务部批准 10: 过户 11: 延期实施 12: 停止实施 13: 分红方案待定
IS_SEPARATION	int	是否分离交易可转债
RECOMMENDER	string	上市推荐人
CLAUSE_IS_INT_CHA_DEPO_RATE	int	利率是否随存款利率调整
CLAUSE_IS_COM_INT	int	是否有利息补偿条款
CLAUSE_COM_INT_RATE	float	补偿利率（%）
CLAUSE_COM_INT_DESC	string	补偿利率说明
CLAUSE_INIT_CONV_PRICE_ITEM	string	初始转股价条款
CLAUSE_CONV_ADJ_ITEM	string	转股价格调整条款

CLAUSE_CONV_PERIOD_ITEM	string	转换期条款
CLAUSE_INI_CONV_PRICE	float	初始转换价格
CLAUSE_INI_CONV_PREMIUM_RATIO	float	初始转股价溢价比例（%）
CLAUSE_PUT_ITEM	string	回售条款
CLAUSE_CALL_ITEM	string	赎回条款
CLAUSE_SPEC_DOWN_ADJ	string	特别向下修正条款
CLAUSE_ORIG_RATION_ARR_ITEM	string	向原股东配售安排条款
LIST_PASS_DATE	string	发审通过公告日
LIST_PERMIT_DATE	string	证监会核准公告日
LIST_ANN_DATE	string	发行公告日
LIST_RESULT_ANN_DATE	string	发行结果公告日
LIST_TYPE	string	发行方式
LIST_FEE	float	发行费用
LIST_RATION_DATE	string	老股东配售日期
LIST_RATION_REG_DATE	string	老股东配售股权登记日
LIST_RATION_PAYMT_DATE	string	老股东配售缴款日
LIST_RATION_CODE	string	老股东配售代码
LIST_RATION_NAME	string	老股东配售简称
LIST_RATION_PRICE	float	老股东配售价格
LIST_RATION_RATIO_DE	float	老股东配售比例分母
LIST_RATION_RATIO_MO	float	老股东配售比例分子
LIST_RATION_VOL	float	向老股东配售数量(张)）
LIST_HOUSEHOLD	float	老股东配售户数
LIST_ONL_DATE	string	上网发行日期

LIST_PCHASE_CODE_ONL	string	上网发行申购代码
LIST_PCH_NAME_ONL	string	上网发行申购名称
LIST_PCH_PRICE_ONL	float	上网发行申购价格
LIST_ISSUE_VOL_ONL	float	上网发行数量(不含优先配售)(张)
LIST_CODE_ONL	float	上网发行配号总数
LIST_EXCESS_PCH_ONL	float	上网发行超额认购倍数(不含优先配售)
RESULT_EF_SUBSCR_P_OFF	float	网上有效申购户数(不含优先配售)
RESULT_SUC_RATE_OFF	float	网上有效申购手数(不含优先配售)
LIST_DATE_INST_OFF	string	网下向机构投资者发行日期
LIST_VOL_INST_OFF	float	网下向机构投资者发行数量(不含优先配售)(张)
RESULT_SUC_RATE_ON	float	网上中签率(不含优先配售)(%)
LIST_EFFECT_PC_HVOL_OFF	float	网下有效申购手数(不含优先配售)
LIST_EFF_PC_H_OF	float	网下有效申购户数(不含优先配售)
LIST_SUC_RATE_OFF	float	网下中签率(不含优先配售)(%)
PRE_RATION_VOL	float	网下优先配售数量(张)
LIST_ISSUE_SIZE	float	发行规模(万元)
LIST_ISSUE_QUANTITY	float	发行数量(万张)
MIN_OFF_INST_SUBSCR_QTY	float	网下最小申购数量(机构)
OFF_INST_DEP_RATIO	string	网下定金比例(机构)
MAX_OFF_INST_SUBSCR_QTY	float	网下最大申购数量(机构)

OFF_SUBSCR_UNIT_INC_DESC	string	网下申购累进单位说明
IS_CONV_BONDS	int	是否可转债
MIN_UNLINE_PUBLIC	float	网下最小申购数量(公众)(元)
MAX_UNLINE_PUBLIC	float	网上最大申购数量(公众)(元)
TERM_YEAR	float	借款期限(年)
INTEREST_TYPE	string	利率类型
COUPON_RATE	float	利率(%)
INTEREST_FRE_QUENCY	string	付息频率
RESULT_SUC_RATE_ON2	float	网上中签率(不含优先配售)(%)
COUPON_TXT	string	利率说明
RATIO_ANNCE_DATE	string	网上中签率公告日
RATIO_DATE	string	网上中签结果公告日

3.5.14.2 可转债份额

函数接口：get_kzz_share

功能描述：获取指定可转债列表的可转债份额数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_share	dict	dataframe column 为 kzz_share 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_share = info_data_object.get_kzz_share(code_list, is_local=False)
```

kzz_share 的字段说明：

字段名称	类型	字段说明
CHANGE_DATE	string	变动日期
ANN_DATE	string	公告日期
MARKET_CODE	string	市场代码
BOND_SHARE	float	债券份额（万元）
CONV_SHARE	float	已转成股份数
CHANGE_REASON	string	变动原因代码,目前包含的枚举类型: ZZG 转债转股 SH 赎回 KZZS 可转债上市 HS 回售 DQ 到期 QLXQ 权利行权 TQDF 本金提前兑付 GH 购回 HSZG 回售转股 HGZG 回购转股

3.5.14.3 可转债转股数据

函数接口：get_kzz_conv

功能描述：获取指定可转债列表的可转债转股数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式

			类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_conv	dict	dataframe column 为 kzz_conv 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_conv = info_data_object.get_kzz_conv(code_list, is_local=False)
```

kzz_conv 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
ANN_DATE	string	公告日期
CONV_CODE	string	转股申报代码
CONV_NAME	string	转股简称
CONV_PRICE	float	股转价格
CURRENCY_CODE	string	股转申报代码
CONV_START_DATE	string	自愿转换期起始日
CONV_END_DATE	string	自愿转换期截止日
TRADE_DATE_LAST	string	可转换债停止交易日
FORCED_CONV_DATE	string	强制转换日
FORCED_CONV_PRICE	float	强制转换价格
REL_CONV_MONTH	float	相对转换期(月)
IS_FORCED	float	是否强制转股
FORCED_CONV_REASON	string	强制转换原因

3.5.14.4 可转债转股变动数据

函数接口：get_kzz_conv_change

功能描述：获取指定可转债列表的可转债转股变动数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_conv_change	dict	dataframe column 为 kzz_conv_change 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_conv_change = info_data_object.get_kzz_conv_change(code_list, is_local=False)
```

kzz_conv_change 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
CHANGE_DATE	string	变动日期
ANN_DATE	string	公告日期
CONV_PRICE	float	转股价格
CHANGE_REASON	string	变动原因， 变动原因 | 变动原因名称 1 | 发行 2 | 换股吸收合并

		3 派息 4 配股 5 上市 6 送股 7 送转股 8 送转股,派息 9 修正 10 增发 11 转增,派息 12 送股,派息 13 公司选择不行使赎回权 14 回购注销 15 回购注销,派息 16 增发,回购注销 17 增发,回购注销,派息 18 增发,派息 19 换股 20 派息,转增 21 派息,转增,增发 22 派息,送转股 24 调整 25 转增 26 除息

3.5.14.5 可转债修正数据

函数接口：get_kzz_corr

功能描述：获取指定可转债列表的可转债修正数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“

			'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_corr	dict	dataframe column 为 kzz_corr 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_corr = info_data_object.get_kzz_corr(code_list, is_local=False)
```

kzz_corr 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
START_DATE	string	特别修正起始时间
END_DATE	string	特别修正结束时间
CORR_TRIG_CALC_MAX_PERIOD	float	修正触发计算最大时间区间（天）
CORR_TRIG_CALC_PERIOD	float	修正触发计算时间区间（天）
SPEC_CORR_TRIG_RATIO	float	特别修正触发比例（%）
CORR_CONV_PRICE_FLOOR_DESC	string	修正后转股价格底线说明
REF_PRICE_IS_AVG_PRICE	int	参考价格是否为算术平均价
CORR_TIMES_LIMIT	string	修正次数限制
IS_TIMEPOINT_CORR_CLAUSE_FLAG	int	是否有时点修正条款
TIMEPOINT_COUNT	float	时点数
TIMEPOINT_CORR_TEXT_CLAUSE	string	时点修正文字条款

SPEC_CORR_RANGE	float	特别修正幅度
IS_SPEC_DOWN_CORR_CLAUSE_FLAG	int	是否有特别向下修正条款

3.5.14.6 可转债赎回数据

函数接口： get_kzz_call

功能描述： 获取指定可转债列表的可转债赎回数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_call	dict	dataframe column 为 kzz_call 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_call = info_data_object.get_kzz_call(code_list, is_local=False)
```

kzz_call 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
CALL_PRICE	float	赎回价
BEGIN_DATE	string	起始日期
END_DATE	string	截止日期
TRI_RATIO	float	触发比例（%）

3.5.14.7 可转债回售数据

函数接口：get_kzz_put
功能描述：获取指定可转债列表的可转债回售数据
输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_put	dict	dataframe column 为 kzz_put 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_put = info_data_object.get_kzz_put(code_list, is_local=False)
```

kzz_put 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
PUT_PRICE	float	回售价
BEGIN_DATE	string	起始日期
END_DATE	string	截止日期
TRI_RATIO	float	触发比例（%）

3.5.14.8 可转债回售赎回条款

函数接口：get_kzz_put_call_item
功能描述：获取指定可转债列表的可转债回售赎回条款数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_put_call_item	dict	dataframe column 为 kzz_put_call_item 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_put_call_item = info_data_object.get_kzz_put_call_item(code_list, is_local=False)
```

kzz_put_call_item 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
MAND_PUT_PERIOD	string	无条件回售期
MAND_PUT_PRICE	float	无条件回售价
MAND_PUT_START_DATE	string	无条件回售开始日期
MAND_PUT_END_DATE	string	无条件回售结束日期
MAND_PUT_TEXT	string	无条件回售文字条款
IS_MAND_PUT_CONTAIN_CURRENT	int	无条件回售是否含当期利息
CON_PUT_START_DATE	string	有条件回售起始日期
CON_PUT_END_DATE	string	有条件回售结束日期
MAX_PUT_TRI_PER	float	回售触发计算最大时间区间
PUT_TRI_PERIOD	float	回售触发计算时间区间

ADD_PUT_CON	string	附加回售条件
ADD_PUT_PRICE_INS	string	股价回售价格说明
PUT_NUM_INS	string	回售次数说明
PUT_PRO_PERIOD	float	相对回售期（月）
PUT_NO_PERY	float	每年回售次数
IS_PUT_ITEM	int	是否有回售条款
IS_TERM_PUT_ITEM	int	是否有到期回售条款
IS_MAND_PUT_ITEM	int	是否有无条件回售条款
IS_TIME_PUT_ITEM	int	是否有时点回售条款
TIME_PUT_NO	float	时点回售数
TIME_PUT_ITEM	string	时点回售文字条款
TERM_PUT_PRICE	float	到期回售价
CON_CALL_START_DATE	string	有条件赎回起始日期
CON_CALL_END_DATE	string	有条件赎回结束日期
CALL_TRI_CON_INS	string	赎回触发条件说明
MAX_CALL_TRI_PER	float	赎回触发计算最大时间区间
CALL_TRI_PER	float	赎回触发计算时间区间
CALL_NUM_BER_INS	string	赎回次数说明
IS_CALL_ITEM	int	是否有赎回条款
CALL_PRO_PERIOD	float	相对赎回期（月）
CALL_NO_PERY	float	每年赎回次数
IS_TIME_CALL_ITEM	int	是否有时点赎回条款
TIME_CALL_NO	float	时点赎回数
TIME_CALL_TEXT	string	时点赎回文字条款
EXPIRED_REDEMPTION_PRICE	float	到期赎回价
PUT_TRI_CON_DESC	string	回售触发条件说明

3.5.14.9 可转债回售条款执行说明

函数接口：get_kzz_put_explanation

功能描述：获取指定可转债列表的可转债回售条款执行说明数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_put_explanation	dict	dataframe column 为 kzz_put_explanation 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_put_explanation = info_data_object.get_kzz_put_explanation(code_list, is_local=False)
```

kzz_put_explanation 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
PUT_FUND_ARRIVAL_DATE	string	回售资金到账日
PUT_PRICE	float	每百元面值回收价格（元）
PUT_ANNOUNCEMENT_DATE	string	回售公告日
PUT_EX_DATE	string	回售履行结果公告日
PUT_AMOUNT	float	回售总面额（亿元）
PUT_OUTSTANDING	float	继续托管总面额（亿元）

REPURCHASE_START_DATE	string	回售行使开始日
REPURCHASE_END_DATE	string	回售行使截止日
RESALE_START_DATE	string	转售开始日
FUND_END_DATE	string	回售日
REPURCHASE_CODE	string	回售代码
RESALE_AMOUNT	float	转售总面额（亿元）
RESALE_IMP_AMOUNT	float	实施转售总面额（亿元）
RESALE_END_DATE	string	转售截止日

3.5.14.10 可转债赎回条款执行说明

函数接口：get_kzz_call_explanation

功能描述：获取指定可转债列表的可转债赎回条款执行说明数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“'D://AmazingData_local_data//'””
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_call_explanation	dict	dataframe column 为 kzz_call_explanation 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_call_explanation = info_data_object.get_kzz_call_explanation(code_list, is_local=False)
```

kzz_call_explanation 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
CALL_DATE	string	赎回日
CALL_PRICE	float	每百元面值赎回价格(元)
CALL_ANNOUNCEMENT_DATE	string	赎回公告日
CALL_FUL_RES_ANN_DATE	string	赎回履行结果公告日
CALL_AMOUNT	float	赎回总面额(亿元)
CALL_OUTSTANDING_AMOUNT	float	继续托管总面额（亿元）
CALL_DATE_PUB	string	赎回日（公布）
CALL_FUND_ARRIVAL_DATE	string	赎回资金到账日
CALL_RECORD_DAY	string	赎回登记日
CALL_REASON	string	赎回原因

3.5.14.11 可转债停复牌信息

函数接口：get_kzz_suspend

功能描述：获取指定可转债列表的可转债停复牌信息数据

输入参数：

参数	数据类型	必选	解释
code_list	list[str]	是	支持可转债的的代码列表
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案

输出参数：

参数	数据类型	解释
kzz_suspend	dict	dataframe column 为 kzz_suspend 的字段 index 无意义

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
```

```
base_data_object = ad.BaseData()
code_list = base_data_object.get_code_list('EXTRA_KZZ')
kzz_suspend = info_data_object.get_kzz_suspend(code_list, is_local=False)
```

kzz_suspend 的字段说明：

字段名称	类型	字段说明
MARKET_CODE	string	市场代码
SUSPEND_DATE	string	停牌日期
SUSPEND_TYPE	int	停牌类型代码 001-上午停牌 002-下午停牌 003-今起停牌 004-盘中停牌 007-停牌 1 小时 016-停牌 1 天
RESUMP_DATE	string	复牌日期
CHANGE_REASON	string	停牌原因
CHANGE_REASON_CODE	int	停牌原因代码
RESUMP_TIME	string	停复牌时间

3.5.15 国债收益率数据

3.5.15.1 国债收益率

函数接口： get_treasury_yield
功能描述： 获取指定期限的国债收益率数据
输入参数：

参数	数据类型	必选	解释
term_list	list[str]	是	支持不同期限的国债收益率 'm3'：3 个月, 'm6'：6 个月, 'y1'：1 年, 'y2'：2 年, 'y3'：3 年, 'y5'：5 年, 'y7'：7 年, 'y10'：20 年,

			'y30'：30 年
local_path	str	是	本地存储数据的路径，需绝对路径，格式类似“ 'D://AmazingData_local_data//' ”
is_local	bool	否	默认为 True，本地数据缓存方案
begin_date	int	否	变动日期，本地数据缓存方案
end_date	int	否	变动日期，本地数据缓存方案

输出参数：

参数	数据类型	解释
treasury_yield	dict	字典的 key：期限 字典的 value：dataframe， column 为 YIELD，国债收益率数据， index 为日期

```
# 第一步 登录 api
import AmazingData as ad
ad.login(username='username', password='password',host='***.***.***.***',port=****)
info_data_object = ad.InfoData()
treasury_yield = info_data_object.get_treasury_yield(['m3', 'm6', 'y1', 'y2', 'y3', 'y5', 'y7', 'y10', 'y30'])
```

3.6 金融算子详细

3.6.1 数学函数

数学函数用于基本的数学运算，包括三角函数、对数、取整等操作。所有函数返回 pd.Series 类型。

3.6.1.1 函数列表

序号	函数名称	函数用法
1	ABS	ABS(X)为 X 的平方根
2	ACOS	ACOS(X)返回 X 的反余弦值
3	ASIN	ASIN(X)返回 X 的反正弦值
4	ATAN	ATAN(X)返回 X 的反正切值
5	BETWEEN	BETWEEN(A,B,C)表示 A 处于 B 和 C 之间时返回 1(B<=A<=C 或 C<=A<=B),否则返回 0
6	CEILING	CEILING(x)沿 A 数值增大方向最接近的整数
7	COS	COS(X)返回 X 的余弦值

8	EXP	EXP(X)返回 e 的 X 次幂
9	FLOOR	FLOOR(x)沿 A 数值减小方向最接近的整数
10	FRACPART	FRACPART(X),返回 X 的小数部分
11	IF	IF(X,A,B) 若 X 不为 0 则返回 A,否则返回 B
12	INTPART	INTPART(x)沿 A 绝对值减小方向最接近的整数
13	LN	LN(X)以 e 为底的对数
14	LOG	LOG(X)以 10 为底的对数
15	MAX	MAX(A,B,C,D,E,F,...)返回较大值
16	MIN	MIN(A,B,C,D,E,F,...)返回较大值
17	MOD	MOD(M,N),返回 M 关于 N 的模(M 除以 N 的余数)
18	POW	POW(A,B)返回 A 的 B 次幂
19	RAND	RAND(a,b),返回一个范围在[a, b]的随机整数
20	ROUND	ROUND(X,N),返回 X 四舍五入到 N 位小数的数值
21	SIGN	SIGN(X),返回 X 的符号.当 X>0,X=0,X<0 分别返回 1,0,-1
22	SIN	SIN(X)返回 X 的正弦值
23	SQRT	SQRT(X)为 X 的平方根
24	TAN	TAN(X)返回 X 的正切值

3.6.1.2 函数说明

(1) ABS(x: Series) 绝对值

用法: ABS(X)为 X 的平方根

(2) ACOS(x: Series) 反余弦值

用法: ACOS(X)返回 X 的反余弦值

(3) ASIN(x: Series) 反正弦值

用法: ASIN(X)返回 X 的反正弦值

(4) ATAN(x: Series) 反正切值

用法: ATAN(X)返回 X 的反正切值

(5) BETWEEN(a: Series, b: Series, c: Series) 介于

用法: BETWEEN(A,B,C)表示 A 处于 B 和 C 之间时返回 1(B<=A<=C 或 C<=A<=B),否则返回 0

(6) CEILING(x: Series) 向上舍入

用法: CEILING(x)沿 A 数值增大方向最接近的整数

(7) COS(x: Series) 余弦值

用法: COS(X)返回 X 的余弦值

(8) EXP(x: Series) X 次幂

用法：EXP(X)返回 e 的 X 次幂

（9）FLOOR(x: Series) 向下舍入
用法： FLOOR(x)沿 A 数值减小方向最接近的整数

（10）FRACPART(x: Series) 小数部分
用法： FRACPART(X),返回 X 的小数部分

（11）IF(cond: Series, a: Series, b: Series) 条件选择函数
用法：IF(X,A,B) 若 X 不为 0 则返回 A,否则返回 B

（12）INTPART(x: Series) 取整
用法： INTPART(x)沿 A 绝对值减小方向最接近的整数

（13）LN(x: Series) 自然对数
用法：LN(X)以 e 为底的对数

（14）LOG(x: Series) 10 为底的对数.
用法：LOG(X)以 10 为底的对数

（15）MAX(args: Series) 求 n 个参数中的最大值
用法：MAX(A,B,C,D,E,F,...)返回较大值

（16）MIN(args: Series) 求 n 个参数中的最小值
用法：MIN(A,B,C,D,E,F,...)返回较大值

（17）MOD(x: Series, n: int) 取模
用法： MOD(M,N),返回 M 关于 N 的模(M 除以 N 的余数)

（18）POW(a: Series, b: Series) 乘幂
用法： POW(A,B)返回 A 的 B 次幂

（19）RAND(a: int, b: int) 取随机数
用法： RAND(a,b),返回一个范围在[a, b]的随机整数

（20）ROUND(x: Series, n: int) 四舍五入
用法： ROUND(X,N),返回 X 四舍五入到 N 位小数的数值

（21）SIGN(x: Series) 取符号
用法： SIGN(X),返回 X 的符号.当 X>0,X=0,X<0 分别返回 1,0,-1

（22）SIN(x: Series) 正弦值
用法：SIN(X)返回 X 的正弦值

（23）SQRT(x: Series) 开平方

用法： SQRT(X)为 X 的平方根

（24）TAN(x: Series) 正切值

用法：TAN(X)返回 X 的正切值

3.6.1.3 API 案例

```
import AmazingData as ad
from AmazingData.operator.math_function import MathFunction

ad.login(username='username', password='password',host='***.***.***.***',port=****)

# 获取数据
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
market_data_object = ad.MarketData(calendar)
code = '00000*.SH'
kline_day = market_data_object.query_kline([code], begin_date=20130101, end_date=20250722,
                                           period=ad.constant.Period.day.value)
df = kline_day[code]

# MAX - 求 n 个参数中的最大值
result = MathFunction.MAX(df['close'], df['open'])

# MIN - 求 n 个参数中的最小值
result = MathFunction.MIN(df['close'], df['open'])

# ACOS - 反余弦值
result = MathFunction.ACOS(df['close'] / df['close'].max())

# ASIN - 反正弦值
result = MathFunction.ASIN(df['close'] / df['close'].max())

# ATAN - 反正切值
result = MathFunction.ATAN(df['close'])

# COS - 余弦值
result = MathFunction.COS(df['close'])

# SIN - 正弦值
```

```
result = MathFunction.SIN(df['close'])

# TAN - 正切值
result = MathFunction.TAN(df['close'])

# EXP - e 的 X 次幂
result = MathFunction.EXP(df['close'] / 1000)

# LN - 自然对数
result = MathFunction.LN(df['close'])

# LOG - 10 为底的对数
result = MathFunction.LOG(df['close'])

# SQRT - 开平方
result = MathFunction.SQRT(df['close'])

# ABS - 绝对值
result = MathFunction.ABS(df['close'] - df['open'])

# POW - 乘幂
result = MathFunction.POW(df['close'] / 1000, df['open'] / df['open'])

# CEILING - 向上舍入
result = MathFunction.CEILING(df['close'])

# FLOOR - 向下舍入
result = MathFunction.FLOOR(df['close'])

# INTPART - 取整
result = MathFunction.INTPART(df['close'])

# BETWEEN - 介于
result = MathFunction.BETWEEN(df['close'], df['low'], df['high'])

# FRACPART - 小数部分
result = MathFunction.FRACPART(df['close'])

# ROUND - 四舍五入
result = MathFunction.ROUND(df['close'], 2)

# SIGN - 取符号
result = MathFunction.SIGN(df['close'] - df['open'])
```

```
# MOD - 取模
result = MathFunction.MOD(df['close'], 100)

# IF - 条件选择函数
result = MathFunction.IF(df['close'] > df['open'], df['high'], df['low'])

# RAND - 取随机数
result = MathFunction.RAND(1, 100)
```

3.6.2 统计函数

统计函数用于计算时序数据的统计指标，如标准差、方差、相关系数等。

3.6.2.1 函数列表

序号	函数名称	函数用法
1	AVEDEV	AVEDEV(X,N) 返回 X 在 N 周期内的平均绝对偏差
2	BETA	BETA(X,BENCHMARK,N) 返回当前证券 N 周期收益与对应大盘指数收益相比的贝塔系数,N 支持变量
3	BETAEX	BETAEX(X,Y,N) 返回 X 与 Y 的 N 周期的相关放大系数,N 支持变量
4	COVAR	COVAR(X,Y,N) 返回 X 和 Y 的 N 周期的协方差,N 支持变量
5	DEVSQ	DEVSQ(X,N) 返回 X 在 N 周期内的数据偏差平方和
6	FORCAST	FORCAST(X,N) 返回 X 在 N 周期内的线性回归预测值,N 支持变量
7	KURTOSIS	KURTOSIS(X,N) 计算指标在过去 N 个交易日的峰度
8	MEAN	MEAN(X,N) 计算指标在过去 N 个交易日的平均值
9	MEDIAN	MEDIAN(X,N) 计算指标在过去 N 个交易日的中位数
10	QUANTILE	QUANTILE(X,N,M) 计算指标在过去 N 个交易日排名 M 百分点对应的值
11	RELATE	RELATE(X,Y,N) 返回 X 和 Y 的 N 周期的相关系数,N 支持变量
12	SKEW	SKEW(X,N) 计算指标在过去 N 个交易日的偏度
13	SLOPE	SLOPE(X,N) 返回 X 在 N 周期内的线性回归斜率,N 支持变量
14	STD	STD(X,N) 返回 X 在 N 周期内的估算标准差,N 支持变量
15	STDDEV	STDDEV(X,N) 返回 X 在 N 周期内的标准偏差
16	STDP	STDP(X,N) 返回 X 在 N 周期内的总体标准差,N 支持变量
17	VAR	VAR(X,N) 返回 X 在 N 周期内的估算样本方差,N 支持变量
18	VARP	VARP(X,N) 返回 X 在 N 周期内的总体样本方差,N 支持变量

3.6.2.2 函数说明

（1）AVEDEV(x: Series, n: int) 平均绝对偏差
用法：AVEDEV(X,N) 返回 X 在 N 周期内的平均绝对偏差

（2）BETA(x: Series, benchmark: Series, n: int) 贝塔系数
用法：BETA(X,BENCHMARK,N) 返回当前证券 N 周期收益与对应大盘指数收益相比的贝塔系数,N 支持变量

（3）BETAEX(x: Series, y: Series, n: int) 相关放大系数
用法：BETAEX(X,Y,N) 返回 X 与 Y 的 N 周期的相关放大系数,N 支持变量

（4）COVAR(x: Series, y: Series, n: int) 协方差
用法：COVAR(X,Y,N) 返回 X 和 Y 的 N 周期的协方差,N 支持变量

（5）DEVSQ(x: Series, n: int) 数据偏差平方和
用法：DEVSQ(X,N) 返回 X 在 N 周期内的数据偏差平方和

（6）FORCAST(x: Series, n: int) 线性回归预测值
用法：FORCAST(X,N) 返回 X 在 N 周期内的线性回归预测值,N 支持变量

（7）KURTOSIS(x: Series, n: int) 峰度
用法：KURTOSIS(X,N) 计算指标在过去 N 个交易日的峰度

（8）MEAN(x: Series, n: int) 平均值
用法：MEAN(X,N) 计算指标在过去 N 个交易日的平均值

（9）MEDIAN(x: Series, n: int) 中位数
用法：MEDIAN(X,N) 计算指标在过去 N 个交易日的中位数

（10）QUANTILE(x: Series, n: int, m: float) 分位数
用法：QUANTILE(X,N,M) 计算指标在过去 N 个交易日排名 M 百分点对应的值

（11）RELATE(x: Series, y: Series, n: int) 相关系数
用法：RELATE(X,Y,N) 返回 X 和 Y 的 N 周期的相关系数,N 支持变量

（12）SKEW(x: Series, n: int) 偏度
用法：SKEW(X,N) 计算指标在过去 N 个交易日的偏度

（13）SLOPE(x: Series, n: int) 线性回归斜率
用法：SLOPE(X,N) 返回 X 在 N 周期内的线性回归斜率,N 支持变量

（14）STD(x: Series, n: int) 估算标准差(样本标准差)

用法：STD(X,N) 返回 X 在 N 周期内的估算标准差,N 支持变量

（15）STDDEV(x: Series, n: int) 标准偏差
用法：STDDEV(X,N) 返回 X 在 N 周期内的标准偏差

（16）STDP(x: Series, n: int) 总体标准差
用法：STDP(X,N) 返回 X 在 N 周期内的总体标准差,N 支持变量

（17）VAR(x: Series, n: int) 估算样本方差
用法：VAR(X,N) 返回 X 在 N 周期内的估算样本方差,N 支持变量

（18）VARP(x: Series, n: int) 总体样本方差
用法：VARP(X,N) 返回 X 在 N 周期内的总体样本方差,N 支持变量

3.6.2.3 API 案例

```
import AmazingData as ad
from AmazingData.operator.math_function import StatisticsFunction

ad.login(username='username', password='password',host='***.***.***.***',port=****)

# 获取数据
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
market_data_object = ad.MarketData(calendar)
code = '00000*.SH'
kline_day = market_data_object.query_kline([code], begin_date=20130101, end_date=20250722,
                                           period=ad.constant.Period.day.value)
df = kline_day[code]
# AVEDEV - 平均绝对偏差
result = StatisticsFunction.AVEDEV(df['close'], 20)

# DEVSQ - 数据偏差平方和
result = StatisticsFunction.DEVSQ(df['close'], 20)

# FORCAST - 线性回归预测值
result = StatisticsFunction.FORCAST(df['close'], 20)

# SLOPE - 线性回归斜率
result = StatisticsFunction.SLOPE(df['close'], 20)

# STD - 估算标准差(样本标准差)
```

```
result = StatisticsFunction.STD(df['close'], 20)

# STDP - 总体标准差
result = StatisticsFunction.STDP(df['close'], 20)

# STDDEV - 标准偏差
result = StatisticsFunction.STDDEV(df['close'], 20)

# VAR - 估算样本方差
result = StatisticsFunction.VAR(df['close'], 20)

# VARP - 总体样本方差
result = StatisticsFunction.VARP(df['close'], 20)

# COVAR - 协方差
result = StatisticsFunction.COVAR(df['close'], df['open'], 20)

# RELATE - 相关系数
result = StatisticsFunction.RELATE(df['close'], df['open'], 20)

# BETA - 贝塔系数
result = StatisticsFunction.BETA(df['close'], df['open'], 20)

# BETAEX - 相关放大系数
result = StatisticsFunction.BETAEX(df['close'], df['open'], 20)

# KURTOSIS - 峰度
result = StatisticsFunction.KURTOSIS(df['close'], 20)

# SKEW - 偏度
result = StatisticsFunction.SKEW(df['close'], 20)

# MEAN - 平均值
result = StatisticsFunction.MEAN(df['close'], 20)

# MEDIAN - 中位数
result = StatisticsFunction.MEDIAN(df['close'], 20)

# QUANTILE - 分位数
result = StatisticsFunction.QUANTILE(df['close'], 20, 0.75)
```

3.6.3 时序函数

时序函数用于时间序列数据的处理，包括引用、移动平均、条件统计等。

3.6.3.1 函数列表

序号	函数名称	函数用法
1	AMA	AMA(X,A) A 为自适应系数,必须小于 1
2	BARSLAST	BARSLAST(X) 上一次 X 不为 False 到现在的周期数
3	BARSLASTCOUNT	BARSLASTCOUNT(X) 统计连续满足 X 条件的周期数
4	BARSLASTS	BARSLASTS(X,N) X 倒数第 N 满足到现在的周期数,N 支持变量
5	BARSNEXT	BARSNEXT(X) 下一次 X 不为 0 到现在的周期数
6	BARSSINCE	BARSSINCE(X) 第一次 X 不为 0 到现在的周期数
7	BARSSINCEN	BARSSINCEN(X,N) N 周期内第一次 X 不为 0 到现在的周期数,N 为常量
8	BARSTATUS	BARSTATUS(X) 结果 1 表示第一根 K 线,2 表示最后一根 K 线,0 表示在中间位置
9	COUNT	COUNT(X,N) 统计 N 周期中满足 X 条件的周期数,若 N<=0 则从第一个有效值开始
10	CROSS	CROSS(A,B) 表示当 A 从下方向上穿过 B 时返回 1,否则返回 0
11	CUMSUM	CUMSUM(X) 从第一个有效值开始对 X 累计求和
12	CURRBARSCOUNT	CURRBARSCOUNT(X) 从最新一根 K 线倒数编号,从 1 开始计数
13	DMA	DMA(X,A) 求 X 的动态移动平均
14	DOWNNDAY	DOWNNDAY(CLOSE,M) 表示连跌 M 个周期,M 为常量
15	EMA	EMA(X,N) X 的 N 日指数移动平均，算法:Y=(X*2+Y'*(N-1))/(N+1)
16	EVERY	EVERY(X,N) 表示 N 日内一直满足条件 X (N 应大于 0,小于总周期数,N 支持变量)
17	EXIST	EXIST(X,N) 表示 N 日内是否存在满足条件 X
18	EXISTR	EXISTR(X,A,B) 表示从前 A 日内到前 B 日内是否存在满足条件 X
19	EXPMEMA	EXPMEMA(X,N) X 的 N 日指数平滑移动平均,N 不支持变量
20	FILTER	FILTER(X,N) X 满足条件后,将其后 N 周期内的数据置为 0,N 为常量
21	FILTERX	FILTERX(X,N) X 满足条件后,将其前 N 周期内的数据置为 0,N 为常量
22	HHV	HHV(X,N) 求 N 周期内 X 最高值,N=0 则从第一个有效值开始
23	HHVBARS	HHVBARS(X,N) 求 N 周期内 X 最高值到当前周期数,N=0 表示从第一个有效值开始统计
24	HHVLLV	HHVLLV(X,T,N1,N2) 表示前 N1 日到前 N2 日时段内的 X 最高值(T=0 时)或最低值(T=1 时)
25	HOD	HOD(X,N) 求当前 X 数据是 N 周期内的第几个高值,N=0 则从第一

		个有效值开始
26	LAST	LAST(X,A,B) 表示从前 A 日到前 B 日内一直满足条件 X
27	LLV	LLV(X,N) 求 N 周期内 X 最低值,N=0 则从第一个有效值开始
28	LLVBARS	LLVBARS(X,N) 求 N 周期内 X 最低值到当前周期数,N=0 表示从第一个有效值开始统计
29	LOD	LOD(X,N) 求当前 X 数据是 N 周期内的第几个低值,N=0 则从第一个有效值开始
30	LONGCROSS	LONGCROSS(A,B,N) 表示 A 在 N 周期内都小于 B,本周期从下方向上穿过 B 时返回 1,否则返回 0
31	MA	MA(X,N) X 的 N 日简单移动平均,算法(X1+X2+X3+...+Xn)/N,N 支持变量
32	MEMA	MEMA(X,N) X 的 N 日平滑移动平均,如 Y=(X+Y'*(N-1))/N
33	MULAR	MULAR(X,N) 统计 N 周期中 X 的乘积,N=0 则从第一个有效值开始
34	NDAY	NDAY(CLOSE,OPEN,3) 表示连续 3 日收阳线
35	RANGE	RANGE(A,B,C) A 在 B 和 C 范围之间,B<A<C
36	REF	REF(X,A) 引用 A 周期前的 X 值,A 可以是变量
37	REFV	REFV(X,A) 引用 A 周期前的 X 值,A 可以是变量
38	REFX	REFX(X,A) 引用 A 周期后的 X 值,A 可以是变量
39	REFXV	REFXV(X,A) 引用 A 周期后的 X 值,A 可以是变量
40	REVERSE	REVERSE(X) 返回-X
41	SAR	SAR(HIGH,LOW,CLOSE,N,STEP,MAXAF)
42	SHIFT	SHIFT(A,N) 获取 A 的 N 个交易日前的值
43	SMA	SMA(X,N,M) X 的 N 日移动平均,M 为权重,如 Y=(X*M+Y'*(N-M))/N
44	SUM	SUM(X,N) 统计 N 周期中 X 的总和,N=0 则从第一个有效值开始
45	SUMBARS	SUMBARS(X,A) 将 X 向前累加直到大于等于 A,返回这个区间的周期数
46	SUMBARSX	SUMBARSX(X,A) 将 X 向前累加直到大于等于 A,返回这个区间的周期数
47	TMA	TMA(X,A,B) A 和 B 必须小于 1,算法 Y=(A*Y'+B*X),其中 Y'表示上一周期 Y 值,初值为 X
48	TOTALBARSCOUNT	TOTALBARSCOUNT(X) 从第一根 K 线开始编号,从 1 开始递增计数
49	TR	TR(HIGH,LOW,CLOSE) 求真实波幅
50	UPNDAY	UPNDAY(CLOSE,M) 表示连涨 M 个周期,M 为常量
51	WMA	WMA(X,N) X 的 N 日加权移动平均,算法:Yn=(1*X1+2*X2+...+n*Xn)/(1+2+...+n)

3.6.3.2 函数说明

(1) AMA(x: Series, a: Series) 自适应均线值

用法: AMA(X,A) A 为自适应系数,必须小于 1

（2）BARSLAST(x: Series) 上一次条件成立到当前的周期数
用法：BARSLAST(X) 上一次 X 不为 False 到现在的周期数

（3）BARSLASTCOUNT(x: Series) 统计连续满足条件的周期数
用法：BARSLASTCOUNT(X) 统计连续满足 X 条件的周期数

（4）BARSLASTS(x: Series, n: int) 倒数第 N 次成立时距今的周期数
用法：BARSLASTS(X,N) X 倒数第 N 满足到现在的周期数,N 支持变量

（5）BARSNEXT(x: Series) 下一次条件成立到当前的周期数(未来函数)
用法：BARSNEXT(X) 下一次 X 不为 0 到现在的周期数

（6）BARSSINCE(x: Series) 第一个条件成立到当前的周期数
用法：BARSSINCE(X) 第一次 X 不为 0 到现在的周期数

（7）BARSSINCEN(x: Series, n: int) N 周期内第一个条件成立到当前的周期数
用法：BARSSINCEN(X,N) N 周期内第一次 X 不为 0 到现在的周期数,N 为常量

（8）BARSTATUS(x: Series) 返回数据的位置信息
用法：BARSTATUS(X) 结果 1 表示第一根 K 线,2 表示最后一根 K 线,0 表示在中间位置

（9）COUNT(x: Series, n: int) 统计满足条件的周期数
用法：COUNT(X,N) 统计 N 周期中满足 X 条件的周期数,若 N<=0 则从第一个有效值开始

（10）CROSS(a: Series, b: Series) 两条线交叉
用法：CROSS(A,B) 表示当 A 从下方向上穿过 B 时返回 1,否则返回 0

（11）CUMSUM(x: Series) 累计求和
用法：CUMSUM(X) 从第一个有效值开始对 X 累计求和

（12）CURRBARSCOUNT(x: Series) 求到最后 K 线的周期数
用法：CURRBARSCOUNT(X) 从最新一根 K 线倒数编号,从 1 开始计数

（13）DMA(x: Series, a: Series) 动态移动平均
用法：DMA(X,A) 求 X 的动态移动平均

（14）DOWNNDAY(x: Series, n: int) 返回周期数内是否连跌
用法：DOWNNDAY(CLOSE,M) 表示连跌 M 个周期,M 为常量

（15）EMA(x: Series, n: int) 指数移动平均
用法：EMA(X,N) X 的 N 日指数移动平均,算法:Y=(X*2+Y'*(N-1))/(N+1)

（16）EVERY(x: Series, n: int) 一直存在
用法：EVERY(X,N) 表示 N 日内一直满足条件 X (N 应大于 0,小于总周期数,N 支持变量)

（17）EXIST(x: Series, n: int) 是否存在

用法：EXIST(X,N) 表示 N 日内是否存在满足条件 X

（18）EXISTR(x: Series, a: int, b: int) 是否存在(前几日到前几日间)

用法：EXISTR(X,A,B) 表示从前 A 日内到前 B 日内是否存在满足条件 X

（19）EXPMEMA(x: Series, n: int) 指数平滑移动平均

用法：EXPMEMA(X,N) X 的 N 日指数平滑移动平均,N 不支持变量

（20）FILTER(x: Series, n: int) 过滤连续出现的信号

用法：FILTER(X,N) X 满足条件后,将其后 N 周期内的数据置为 0,N 为常量

（21）FILTERX(x: Series, n: int) 反向过滤连续出现的信号

用法：FILTERX(X,N) X 满足条件后,将其前 N 周期内的数据置为 0,N 为常量

（22）HHV(x: Series, n: int) 求 N 周期内最高值

用法：HHV(X,N) 求 N 周期内 X 最高值,N=0 则从第一个有效值开始

（23）HHVBARS(x: Series, n: int) 求上一高点到当前的周期数

用法：HHVBARS(X,N) 求 N 周期内 X 最高值到当前周期数,N=0 表示从第一个有效值开始统计

（24）HHVLLV(x: Series, t: int, n1: int, n2: int) 阶段最高最低值

用法：HHVLLV(X,T,N1,N2) 表示前 N1 日到前 N2 日时段内的 X 最高值(T=0 时)或最低值(T=1 时)

（25）HOD(x: Series, n: int) 求高值名次

用法：HOD(X,N) 求当前 X 数据是 N 周期内的第几个高值,N=0 则从第一个有效值开始

（26）LAST(x: Series, a: int, b: int) 持续存在

用法：LAST(X,A,B) 表示从前 A 日到前 B 日内一直满足条件 X

（27）LLV(x: Series, n: int) 求 N 周期内最低值

用法：LLV(X,N) 求 N 周期内 X 最低值,N=0 则从第一个有效值开始

（28）LLVBARS(x: Series, n: int) 求上一低点到当前的周期数

用法：LLVBARS(X,N) 求 N 周期内 X 最低值到当前周期数,N=0 表示从第一个有效值开始统计

（29）LOD(x: Series, n: int) 求低值名次

用法：LOD(X,N) 求当前 X 数据是 N 周期内的第几个低值,N=0 则从第一个有效值开始

（30）LONGCROSS(a: Series, b: Series, n: int) 两条线维持一定周期后交叉

用法：LONGCROSS(A,B,N) 表示 A 在 N 周期内都小于 B,本周期从下方向上穿过 B 时返回 1,否则返回 0

（31）MA(x: Series, n: int) 简单移动平均

用法：MA(X,N) X 的 N 日简单移动平均,算法(X1+X2+X3+...+Xn)/N,N 支持变量

（32）MEMA(x: Series, n: int) 平滑移动平均

用法：MEMA(X,N) X 的 N 日平滑移动平均,如 Y=(X+Y'*(N-1))/N

（33）MULAR(x: Series, n: int) 求累乘

用法：MULAR(X,N) 统计 N 周期中 X 的乘积,N=0 则从第一个有效值开始

（34）NDAY(x: Series, y: Series, n: int) 返回是否持续存在 X>Y

用法：NDAY(CLOSE,OPEN,3) 表示连续 3 日收阳线

（35）RANGE(a: Series, b: Series, c: Series) 范围判断

用法：RANGE(A,B,C) A 在 B 和 C 范围之间,B<A<C

（36）REF(x: Series, n: int) 引用若干周期前的数据

用法：REF(X,A) 引用 A 周期前的 X 值,A 可以是变量

（37）REFV(x: Series, n: int) 引用若干周期前的数据(平滑处理)

用法：REFV(X,A) 引用 A 周期前的 X 值,A 可以是变量

（38）REFX(x: Series, n: int) 引用若干周期后的数据(未来函数)

用法：REFX(X,A) 引用 A 周期后的 X 值,A 可以是变量

（39）REFXV(x: Series, n: int) 引用若干周期后的数据(平滑处理)(未来函数)

用法：REFXV(X,A) 引用 A 周期后的 X 值,A 可以是变量

（40）REVERSE(x: Series) 求相反数

用法：REVERSE(X) 返回-X

（41）SAR(high: Series, low: Series, close: Series, n: int, step: float, max_af: float) 抛物线转向指标

用法：SAR(HIGH,LOW,CLOSE,N,STEP,MAXAF)

（42）SHIFT(x: Series, n: int) 获取 N 个交易日前的值

用法：SHIFT(A,N) 获取 A 的 N 个交易日前的值

（43）SMA(x: Series, n: int, m: int) 移动平均

用法：SMA(X,N,M) X 的 N 日移动平均,M 为权重,如 Y=(X*M+Y'*(N-M))/N

（44）SUM(x: Series, n: int) 求总和

用法：SUM(X,N) 统计 N 周期中 X 的总和,N=0 则从第一个有效值开始

（45）SUMBARS(x: Series, a: float) 向前累加到指定值到现在的周期数
用法：SUMBARS(X,A) 将 X 向前累加直到大于等于 A,返回这个区间的周期数

（46）SUMBARSX(x: Series, a: float) 向前累加到指定值到现在的周期数
用法：SUMBARSX(X,A) 将 X 向前累加直到大于等于 A,返回这个区间的周期数

（47）TMA(x: Series, a: float, b: float) 移动平均
用法：TMA(X,A,B) A 和 B 必须小于 1,算法 Y=(A*Y'+B*X),其中 Y'表示上一周期 Y 值,初值为 X

（48）TOTALBARSCOUNT(x: Series) 求到当前的周期数
用法：TOTALBARSCOUNT(X) 从第一根 K 线开始编号,从 1 开始递增计数

（49）TR(high: Series, low: Series, close: Series) 求真实波幅
用法：TR(HIGH,LOW,CLOSE) 求真实波幅

（50）UPNDAY(x: Series, n: int) 返回周期数内是否连涨
用法：UPNDAY(CLOSE,M) 表示连涨 M 个周期,M 为常量

（51）WMA(x: Series, n: int) 加权移动平均
用法：WMA(X,N) X 的 N 日加权移动平均,算法:Yn=(1*X1+2*X2+...+n*Xn)/(1+2+...+n)

3.6.3.3 API 案例

```
import AmazingData as ad
from AmazingData.operator.time_series_function import TimeSeriesFunction
ad.login(username='username', password='password',host='***.***.***.***',port=****)

# 获取数据
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
market_data_object = ad.MarketData(calendar)
code = '00000*.SH'
kline_day = market_data_object.query_kline([code], begin_date=20130101, end_date=20250722,
                                           period=ad.constant.Period.day.value)
df = kline_day[code]
# ========== 位置信息函数 ==========
# BARSTATUS - 返回数据的位置信息
result = TimeSeriesFunction.BARSTATUS(df['close'])

# CURRBARSCOUNT - 求到最后 K 线的周期数
```

```
result = TimeSeriesFunction.CURRBARSCOUNT(df['close'])

# TOTALBARSCOUNT - 求到当前的周期数(从 1 开始递增)
result = TimeSeriesFunction.TOTALBARSCOUNT(df['close'])

# ========== 条件周期统计函数 ==========
# BARSLAST - 上一次条件成立到当前的周期数
condition = df['close'] > df['open']
result = TimeSeriesFunction.BARSLAST(condition)

# BARSLASTS - 倒数第 N 次成立时距今的周期数
result = TimeSeriesFunction.BARSLASTS(condition, 3)

# BARSNEXT - 下一次条件成立到当前的周期数(未来函数)
result = TimeSeriesFunction.BARSNEXT(condition)

# BARSSINCEN - N 周期内第一个条件成立到当前的周期数
result = TimeSeriesFunction.BARSSINCEN(condition, 10)

# BARSSINCE - 第一个条件成立到当前的周期数
result = TimeSeriesFunction.BARSSINCE(condition)

# COUNT - 统计满足条件的周期数
result = TimeSeriesFunction.COUNT(condition, 20)

# BARSLASTCOUNT - 统计连续满足条件的周期数
result = TimeSeriesFunction.BARSLASTCOUNT(condition)

# ========== 最值函数 ==========
# HHV - 求 N 周期内最高值
result = TimeSeriesFunction.HHV(df['high'], 20)

# HHVBARS - 求上一高点到当前的周期数
result = TimeSeriesFunction.HHVBARS(df['high'], 20)

# HOD - 求高值名次
result = TimeSeriesFunction.HOD(df['high'], 20)

# LLV - 求 N 周期内最低值
result = TimeSeriesFunction.LLV(df['low'], 20)

# LLVBARS - 求上一低点到当前的周期数
result = TimeSeriesFunction.LLVBARS(df['low'], 20)
```

```
# LOD - 求低值名次
result = TimeSeriesFunction.LOD(df['low'], 20)

# HHVLLV - 阶段最高最低值
result = TimeSeriesFunction.HHVLLV(df['high'], 0, 20, 5)

# ========== 引用函数 ==========
# REVERSE - 求相反数
result = TimeSeriesFunction.REVERSE(df['close'])

# REF - 引用若干周期前的数据
result = TimeSeriesFunction.REF(df['close'], 1)

# REFX - 引用若干周期后的数据(未来函数)
result = TimeSeriesFunction.REFX(df['close'], 1)

# REFV - 引用若干周期前的数据(平滑处理)
result = TimeSeriesFunction.REFV(df['close'], 1)

# REFXV - 引用若干周期后的数据(平滑处理)(未来函数)
result = TimeSeriesFunction.REFXV(df['close'], 1)

# SHIFT - 获取 N 个交易日前的值
result = TimeSeriesFunction.SHIFT(df['close'], 5)

# ========== 累计函数 ==========
# SUM - 求总和
result = TimeSeriesFunction.SUM(df['volume'], 20)

# MULAR - 求累乘
result = TimeSeriesFunction.MULAR(df['close'] / TimeSeriesFunction.REF(df['close'], 1), 5)

# SUMBARS - 向前累加到指定值到现在的周期数
result = TimeSeriesFunction.SUMBARS(df['volume'], 1000000000)

# SUMBARSX - 向前累加到指定值到现在的周期数(未达到返回 nan)
result = TimeSeriesFunction.SUMBARSX(df['volume'], 1000000000)

# CUMSUM - 累计求和
result = TimeSeriesFunction.CUMSUM(df['volume'])

# ========== 移动平均函数 ==========
# MA - 简单移动平均
result = TimeSeriesFunction.MA(df['close'], 20)
```

```
# SMA - 移动平均
result = TimeSeriesFunction.SMA(df['close'], 20, 1)

# TMA - 移动平均
result = TimeSeriesFunction.TMA(df['close'], 0.9, 0.1)

# MEMA - 平滑移动平均
result = TimeSeriesFunction.MEMA(df['close'], 20)

# EMA - 指数移动平均
result = TimeSeriesFunction.EMA(df['close'], 20)

# EXPMEMA - 指数平滑移动平均
result = TimeSeriesFunction.EXPMEMA(df['close'], 20)

# WMA - 加权移动平均
result = TimeSeriesFunction.WMA(df['close'], 20)

# DMA - 动态移动平均
alpha = df['volume'] / df['volume'].rolling(20).sum()
result = TimeSeriesFunction.DMA(df['close'], alpha)

# AMA - 自适应均线值
result = TimeSeriesFunction.AMA(df['close'], alpha)

# ========== 信号过滤函数 ==========
# FILTER - 过滤连续出现的信号
result = TimeSeriesFunction.FILTER(condition, 5)

# FILTERX - 反向过滤连续出现的信号
result = TimeSeriesFunction.FILTERX(condition, 5)

# ========== 条件判断函数 ==========
# TR - 求真实波幅
result = TimeSeriesFunction.TR(df['high'], df['low'], df['close'])

# RANGE - 范围判断
result = TimeSeriesFunction.RANGE(df['close'], df['low'], df['high'])

# CROSS - 两条线交叉
ma5 = TimeSeriesFunction.MA(df['close'], 5)
ma10 = TimeSeriesFunction.MA(df['close'], 10)
result = TimeSeriesFunction.CROSS(ma5, ma10)
```

```
# LONGCROSS - 两条线维持一定周期后交叉
result = TimeSeriesFunction.LONGCROSS(ma5, ma10, 5)

# UPNDAY - 返回周期数内是否连涨
result = TimeSeriesFunction.UPNDAY(df['close'], 3)

# DOWNNDAY - 返回周期数内是否连跌
result = TimeSeriesFunction.DOWNNDAY(df['close'], 3)

# NDAY - 返回是否持续存在 X>Y
result = TimeSeriesFunction.NDAY(df['close'], df['open'], 3)

# EXIST - 是否存在
result = TimeSeriesFunction.EXIST(condition, 10)

# EXISTR - 是否存在(前几日到前几日间)
result = TimeSeriesFunction.EXISTR(condition, 10, 5)

# EVERY - 一直存在
result = TimeSeriesFunction.EVERY(condition, 5)

# LAST - 持续存在
result = TimeSeriesFunction.LAST(condition, 10, 5)

# ========== 技术指标函数 ==========
# SAR - 抛物线转向指标
result = TimeSeriesFunction.SAR(df['high'], df['low'], df['close'], n=4, step=0.02, max_af=0.2)
```

3.6.4 截面函数

截面函数用于计算同一交易日内多个标的之间的统计指标。输入数据为 DataFrame，行为日期，列为标的代码。

3.6.4.1 函数列表

序号	函数名称	函数用法
1	CSCORR	CSCORR(X,Y) 返回每交易日两个指标的当日相关度
2	CSCOUNT	CSCOUNT(X) 统计交易日截面的标的个数
3	CSCOV	CSCOV(X,Y) 返回每交易日两个指标(X,Y)的当日协方差
4	CSDEMEAN	CSDEMEAN(X) 对每个交易日的截面数据减去均值
5	CSMAX	CSMAX(X) 计算交易日截面指标的最大值
6	CSMEAN	CSMEAN(X) 计算交易日截面指标的平均值
7	CSMEDIAN	CSMEDIAN(X) 计算交易日截面指标的中位数
8	CSMIN	CSMIN(X) 计算交易日截面指标的最小值
9	CSNORMALIZE	CSNORMALIZE(X) 对每个交易日的截面数据进行归一化到[0,1]
10	CSPCTRANK	CSPCTRANK(X) 计算交易日截面指标的百分位排名
11	CSQUANTILE	CSQUANTILE(X,N) 计算交易日截面指标的分位数 N
12	CSRANK	CSRANK(X,B) 计算交易日截面指标的排名
13	CSSTD	CSSTD(X) 计算交易日截面指标的标准差
14	CSSUM	CSSUM(X) 计算交易日截面指标的求和
15	CSVAR	CSVAR(X) 计算交易日截面指标的方差
16	CSZSCORE	CSZSCORE(X) 对每个交易日的截面数据进行 Z-score 标准化

3.6.4.2 函数说明

（1）CSCORR(x: DataFrame, y: DataFrame) 截面相关度

用法：CSCORR(X,Y) 返回每交易日两个指标的当日相关度

（2）CSCOUNT(x: DataFrame) 截面标的个数

用法：CSCOUNT(X) 统计交易日截面的标的个数

（3）CSCOV(x: DataFrame, y: DataFrame) 截面协方差

用法：CSCOV(X,Y) 返回每交易日两个指标(X,Y)的当日协方差

（4）CSDEMEAN(x: DataFrame) 截面去均值

用法：CSDEMEAN(X) 对每个交易日的截面数据减去均值

（5）CSMAX(x: DataFrame) 截面最大值

用法：CSMAX(X) 计算交易日截面指标的最大值

（6）CSMEAN(x: DataFrame) 截面平均值
用法：CSMEAN(X) 计算交易日截面指标的平均值

（7）CSMEDIAN(x: DataFrame) 截面中位数
用法：CSMEDIAN(X) 计算交易日截面指标的中位数

（8）CSMIN(x: DataFrame) 截面最小值
用法：CSMIN(X) 计算交易日截面指标的最小值

（9）CSNORMALIZE(x: DataFrame) 截面归一化(Min-Max)
用法：CSNORMALIZE(X) 对每个交易日的截面数据进行归一化到[0,1]

（10）CSPCTRANK(x: DataFrame) 截面百分位排名
用法：CSPCTRANK(X) 计算交易日截面指标的百分位排名

（11）CSQUANTILE(x: DataFrame, n: float) 截面分位数
用法：CSQUANTILE(X,N) 计算交易日截面指标的分位数 N

（12）CSRANK(x: DataFrame, ascending: bool) 截面排名
用法：CSRANK(X,B) 计算交易日截面指标的排名

（13）CSSTD(x: DataFrame) 截面标准差
用法：CSSTD(X) 计算交易日截面指标的标准差

（14）CSSUM(x: DataFrame) 截面求和
用法：CSSUM(X) 计算交易日截面指标的求和

（15）CSVAR(x: DataFrame) 截面方差
用法：CSVAR(X) 计算交易日截面指标的方差

（16）CSZSCORE(x: DataFrame) 截面 Z-score 标准化
用法：CSZSCORE(X) 对每个交易日的截面数据进行 Z-score 标准化

3.6.4.3 API 案例

```
import AmazingData as ad
from AmazingData.operator.time_series_function import TimeSeriesFunction
ad.login(username='username', password='password',host='***.***.***.***',port=****)


# 获取数据
```

```
base_data_object = ad.BaseData()
calendar = base_data_object.get_calendar()
market_data_object = ad.MarketData(calendar)

# 多只股票数据 (用于截面函数)
codes = ['00000*.SZ', '00000*.SZ', '00000*.SZ', '00000*.SZ', '00000*.SZ']
kline_multi = market_data_object.query_kline(codes, begin_date=20240101, end_date=20250101,
                                             period=ad.constant.Period.day.value)
# 构建截面数据 DataFrame (行:日期, 列:标的)
close_df = pd.DataFrame({c: kline_multi[c]['close'] for c in codes if c in kline_multi})
open_df = pd.DataFrame({c: kline_multi[c]['open'] for c in codes if c in kline_multi})

# CSCOV - 截面协方差
result = CrossSectionFunction.CSCOV(close_df, open_df)

# CSCOUNT - 截面标的个数
result = CrossSectionFunction.CSCOUNT(close_df)

# CSQUANTILE - 截面分位数
result = CrossSectionFunction.CSQUANTILE(close_df, 0.5)

# CSRANK - 截面排名
result = CrossSectionFunction.CSRANK(close_df, ascending=True)

# CSSTD - 截面标准差
result = CrossSectionFunction.CSSTD(close_df)

# CSSUM - 截面求和
result = CrossSectionFunction.CSSUM(close_df)

# CSVAR - 截面方差
result = CrossSectionFunction.CSVAR(close_df)

# CSPCTRANK - 截面百分位排名
result = CrossSectionFunction.CSPCTRANK(close_df)

# CSMEAN - 截面平均值
result = CrossSectionFunction.CSMEAN(close_df)

# CSMAX - 截面最大值
result = CrossSectionFunction.CSMAX(close_df)

# CSCORR - 截面相关度
result = CrossSectionFunction.CSCORR(close_df, open_df)
```

```
# CSMIN - 截面最小值
result = CrossSectionFunction.CSMIN(close_df)

# CSMEDIAN - 截面中位数
result = CrossSectionFunction.CSMEDIAN(close_df)

# CSZSCORE - 截面 Z-score 标准化
result = CrossSectionFunction.CSZSCORE(close_df)

# CSNORMALIZE - 截面归一化(Min-Max)
result = CrossSectionFunction.CSNORMALIZE(close_df)

# CSDEMEAN - 截面去均值
result = CrossSectionFunction.CSDEMEAN(close_df)
```

4. 附录

4.1 字段取值说明

4.1.1 代码类型 security_type(沪深北)

数据类型	枚举值	说明
str	EXTRA_STOCK_A	上交所 A 股、深交所 A 股和北交所的股票列表
str	SH_A	上交所 A 股的股票列表
str	SZ_A	深交所 A 股的股票列表
str	BJ_A	北交所的股票列表
str	EXTRA_STOCK_A_SH_SZ	上交所 A 股和深交所 A 股的股票列表
str	EXTRA_INDEX_A_SH_SZ	上交所和深交所指数列表
str	EXTRA_INDEX_A	上交所、深交所和北交所的指数列表
str	SH_INDEX	上交所指数列表
str	SZ_INDEX	深交所指数列表
str	BJ_INDEX	北交所的指数列表
str	SH_ETF	上交所的 ETF 列表
str	SZ_ETF	深交所的 ETF 列表
str	EXTRA_ETF	上交所、深交所的 ETF 列表
str	SH_KZZ	上交所的可转债列表
str	SZ_KZZ	深交所的可转债列表

str	EXTRA_KZZ	上交所、深交所的可转债列表
str	SH_HKT	沪港通
str	SZ_HKT	深港通
str	EXTRA_HKT	沪深港通
str	SH_GLRA	上交所逆回购
str	SZ_GLRA	深交所逆回购
str	EXTRA_ GLRA	沪深逆回购

4.1.2 代码类型 security_type(期货交易所)

数据类型	枚举值	说明
str	ZJ_FUTURE	期货，包含中金所

4.1.3 代码类型 security_type(期权)

数据类型	枚举值	说明
str	EXTRA_ETF_OP	ETF 期权，上交所/深交所
str	SH_OPTION	ETF 期货，包含上交所
str	SZ_OPTION	ETF 期货，包含深交所

4.1.4 市场类型 market

数据类型	枚举值	说明
str	SH	上交所
str	SZ	深交所
str	BJ	北交所
str	CFE	中金所
str	SHN	沪港通
str	SZN	深港通
str	HK	港交所

4.1.5 交易阶段代码 trading_phase_code

（1） 上海现货快照交易状态

该字段为 8 位字符数组,左起每位表示特定的含义,无定义则填空格。

第 0 位: ‘S’表示启动(开市前)时段,‘C’表示开盘集合竞价时段,‘T’表示连续交易时段,‘E’表示闭市时段,‘P’表示产品停牌。

第 1 位: ‘0’表示此产品不可正常交易,‘1’表示此产品可正常交易。

第 2 位: ‘0’表示未上市,‘1’表示已上市。

第 3 位: ‘0’表示此产品在当前时段不接受进行新订单申报,‘1’ 表示此产品在当前时段可接受

进行新订单申报。

（2） 深圳现货快照交易状态
第 0 位: ‘S’= 启动(开市前)‘O’= 开盘集合竞价‘T’= 连续竞价‘B’= 休市‘C’= 收盘集合竞价‘E’= 已闭市‘H’= 临时停牌‘A’= 盘后交易‘V’=波动性中断。
第 1 位: ‘0’= 正常状态 ‘1’= 全天停牌。交易阶段代码

（3） 港股股票行情交易状态
‘1’表示正常交易，‘2’表示停牌，‘3’表示复牌
（4） 上海期权快照交易状态
第 1 位：‘S’表示启动（开市前）时段，‘C’表示集合竞价时段，‘T’表示连续交易时段，‘B’表示休市时段，‘E’表示闭市时段，‘V’表示波动性中断，‘P’表示临时停牌、‘U’表示收盘集合竞价。‘M’表示可恢复交易的熔断（盘中集合竞价）,‘N’表示不可恢复交易的熔断（暂停交易至闭市）；
第 2 位：‘0’表示未连续停牌，‘1’表示连续停牌。（预留，暂填空格）；
第 3 位：‘0’表示不限制开仓，‘1’表示限制备兑开仓，‘2’表示卖出开仓，‘3’表示限制卖出开仓、备兑开仓，‘4’表示限制买入开仓，‘5’表示限制买入开仓、备兑开仓，‘6’表示限制买入开仓、卖出开仓，‘7’表示限制买入开仓、卖出开仓、备兑开仓；
第 4 位：‘0’表示此产品在当前时段不接受进行新订单申报，‘1’ 表示此产品在当前时段可接受进行新订单申报。

4.1.6 产品状态标志 security_status

状态	标志	说明
停牌	1	深交所、北交所
除权	2	上交所、深交所、北交所
除息	3	上交所、深交所、北交所
风险警示	4	上交所、深交所、北交所
退市整理期	5	上交所、深交所、北交所
上市首日	6	上交所、深交所、北交所
公司再融资	7	深交所
恢复上市首日	8	深交所、北交所
网络投票	9	深交所
增发股份上市	10	深交所
合约调整	11	深交所
暂停上市后协议转让	12	深交所
实施双转单调整	13	深交所
特定债券转让	14	深交所、北交所
上市初期	15	深圳有效
退市整理期首日	16	深交所、北交所
新增股份	57	北交所
是否可作为融资融券可充抵保证金证券	62	北交所
是否为融资标的	63	北交所

是否为融券标的	64	北交所
是否可质押入库	65	北交所
是否跨市场	66	北交所
是否处于转股回售期	67	北交所

4.1.7 数据周期 Period

数据类型	枚举值	说明
int	Period.min1.value	1 分钟线
int	Period.min3.value	3 分钟线
int	Period.min5.value	5 分钟线
int	Period.min10.value	10 分钟线
int	Period.min15.value	15 分钟线
int	Period.min30.value	30 分钟线
int	Period.min60.value	60 分钟线
int	Period.min120.value	120 分钟线
int	Period.day.value	日线
int	Period.week.value	周线
int	Period.month.value	月线
int	Period.season.value	季度线
int	Period.year.value	年线

4.1.8 报告期名称 REPORT_TYPE

报告期类型代码	报告期月份
1	3 月
2	6 月
3	9 月
4	12 月

4.1.9 报表类型代码表 STATEMENT_TYPE

报表类型代码	报表类型	备注
1	合并报表	涵盖母公司的财务报表数据，为最新报表
2	合并报表(单季度)	合并报表(单季度)=合并报表(本期)-合并报表(上一季)
3	合并报表(单季度调整)	合并报表(单季度调整)=合并报表(本期调整)-合并报表(上一季调整)
4	合并报表(调整)	本年度公布上年同期的财务报表数据，报告期为上年度

5	合并报表(更正前)	即出更正公告后，把合并报表的记录修改为合并报表(更正前)；复制原来的记录，更正后报表类型改为合并报表
6	母公司报表	该公司母公司的财务报表数据
7	母公司报表(单季度)	母公司报表(单季度)=母公司报表(本期)-母公司报表(上一季)
8	母公司报表(单季度调整)	母公司报表(单季度调整)=母公司报表(本期调整)-母公司报表(上一季调整)
9	母公司报表(调整)	该公司母公司的本年度公布上年同期的财务报表数据
10	母公司报表(更正前)	之前上市公司已披露财务报表数据，但是由于某些特定原因导致出错，未调整之前的原始财务报表数据。
11	合并报表(未公开)	未在公开信息源披露的财报且加工为合并报表口径
12	合并报表(调整未公开)	未在公开信息源披露的财报且加工为合并报表调整口径
13	合并报表(单季度未公开)	未在公开信息源披露的财报且加工为合并报表单季度口径
14	合并报表(单季度调整未公开)	未在公开信息源披露的财报且加工为母公司报表口径
15	母公司报表(未公开)	未在公开信息源披露的财报且加工为母公司报表口径
16	母公司报表(调整未公开)	未在公开信息源披露的财报且加工为母公司报表调整口径
17	母公司报表(单季度未公开)	未在公开信息源披露的财报且加工或计算为母公司报表单季度口径
18	母公司报表(单季度调整未公开)	未在公开信息源披露的财报且加工或计算为母公司报表单季度调整口径
19	合并报表(调整借壳前)	借壳前的合并报表(调整)
20	合并调整	对合并前各公司的财务报表进行调整，以确保合并财务报表的准确性和可比性
21	合并报表(单季度借壳前)	借壳前的合并报表(单季度)
22	合并报表(单季度调整借壳前)	借壳前的合并报表(单季度调整)
23	母公司报表(借壳前)	借壳前的母公司报表
24	母公司报表(调整借壳前)	借壳前的母公司报表(调整)
25	母公司报表(单季度借壳前)	借壳前的母公司报表(单季度)
26	母公司报表(单季度调整借壳	借壳前的母公司报表(单季度调整)

	前)	
27	合并报表(第一次更正)	有多次更正时，合并报表的第一次更正
28	合并报表(第二次更正)	有多次更正时，合并报表的第二次更正
29	合并调整(第一次更正)	有多次更正时，合并调整的第一次更正
30	合并报表(单月度)	根据披露的券商月报公告加工为合并报表口径
31	合并调整(第二次更正)	有多次更正时，合并调整的第二次更正
32	母公司调整(第二次更正)	有多次更正时，母公司调整的第二次更正
33	母公司调整(第一次更正)	有多次更正时，母公司调整的第一次更正
34	母公司报表(第二次更正)	有多次更正时，母公司报表的第二次更正
35	母公司报表(第一次更正)	有多次更正时，母公司报表的第一次更正
36	合并报表(第三次更正)	有多次更正时，合并报表的第三次更正
37	合并调整(第三次更正)	有多次更正时，合并调整的第三次更正
38	母公司报表(第三次更正)	有多次更正时，母公司报表的第三次更正
39	母公司调整(第三次更正)	有多次更正时，母公司调整的第三次更正
40	母公司报表(单月度)	根据披露的券商月报公告加工为母公司报表口径的数据
41	合并报表(业绩快报)	加工业绩快报中的财务数据（海外数据专用）
42	合并调整(第一次)	第一次合并调整数据
43	合并调整(第二次)	第二次合并调整数据
44	合并调整(第三次)	第三次合并调整数据
45	合并报表(第四次更正)	有多次更正时，合并报表的第四次更正
46	合并调整(第四次更正)	有多次更正时，合并调整的第四次更正
47	母公司报表(第四次更正)	有多次更正时，母公司报表的第四次更正

48	母公司调整(第四次更正)	有多次更正时，母公司调整的第四次更正
50	合并调整(更正前)	即出更正公告后，把合并报表（调整）的记录修改为合并调整(更正前)；复制原来的记录，更正后报表类型改为合并报表(调整)
51	合并报表(下半年报)	合并下半年度的报表
60	母公司调整(更正前)	该公司母公司的本年度公布上年同期的财务报表数据，但是由于某些特定原因导致出错，未调整之前的原始财务报表数据。
70	合并报表(借壳前)	公司主体在借壳上市前披露或者计算的为合并报表口径的报表类型
80	合并报表(预测)	REITS 基金的定期报告中披露的预测的合并报表数据
81	合并报表(公司预测)	
90	项目资产报表	由项目资产管理人编制的一种财务报表，用于反映项目资产的财务状况和经营情况
91	合并报表(日历年)	

4.1.10 股票分红进度代码表 DIV_PROGRESS

分红进度描述	进度代码
董事会预案	1
股东大会通过	2
实施	3
未通过	4
停止实施	12
股东提议	17
董事会预案预披露	19

分红实施进程：股东提议--董事会预案--股东大会--实施

4.1.11 股票配股进度代码表 PROGRESS

配股进度描述	进度代码
董事会预案	1
股东大会通过	2
实施	3
未通过	4
证监会核准	5
达成转让意向	6

签署转让协议	7
国资委批准	8
商务部批准	9
过户	10
延期实施	11
停止实施	12
分红方案待定	13
传闻	14
证监会受理	15
传闻被否认	16
股东提议	17
保监会批复	18
董事会预案预披露	19
发审委通过	20
发审委未通过	21
股东大会未通过	22
银监会批准	23
证监会恢复审核	24
预发行	25
提交注册	26

4.2 数据结构说明

4.2.1 Level-1 快照 Snapshot

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	trade_time	交易所行情数据时间
float	pre_close	昨收价
float	last	最新价
float	open	开盘价
float	high	最高价
float	low	最低价
float	close	收盘价
float	volume	成交总量
float	amount	成交总金额
float	num_trades	成交笔数
float	high_limited	涨停价
float	low_limited	跌停价
float	ask_price1	卖 1 档价格
float	ask_price2	卖 2 档价格

float	ask_price3	卖 3 档价格
float	ask_price4	卖 4 档价格
float	ask_price5	卖 5 档价格
int	ask _volume1	卖 1 档量
int	ask _volume2	卖 2 档量
int	ask _volume3	卖 3 档量
int	ask _volume4	卖 4 档量
int	ask _volume5	卖 5 档量
float	bid_price1	买 1 档价格
float	bid_price2	买 2 档价格
float	bid_price3	买 3 档价格
float	bid_price4	买 4 档价格
float	bid_price5	买 5 档价格
int	bid _volume1	买 1 档量
int	bid _volume2	买 2 档量
int	bid _volume3	买 3 档量
int	bid _volume4	买 4 档量
int	bid _volume5	买 5 档量
float	iopv	净值估产（仅基金品种有效）
str	trading_phase_code	交易阶段代码

4.2.2 ETF 期权快照 SnapshotOption

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	trade_time	交易所行情数据时间
str	trading_phase_code	交易阶段代码
int	total_long_position	总持仓量
float	volume	成交总量
float	amount	成交总金额
float	pre_close	昨收价
float	pre_settle:	上次结算价
float	auction_price	动态参考价（波动性中断参考价，仅上海有效），
int	auction_volume	虚拟匹配数量（仅上海有效）
float	last	最新价
float	open	开盘价
float	high	最高价
float	low	最低价
float	close	收盘价
float	settle	本次结算价
float	high_limited	涨停价
float	low_limited	跌停价

float	ask_price1	卖 1 档价格
float	ask_price2	卖 2 档价格
float	ask_price3	卖 3 档价格
float	ask_price4	卖 4 档价格
float	ask_price5	卖 5 档价格
int	ask _volume1	卖 1 档量
int	ask _volume2	卖 2 档量
int	ask _volume3	卖 3 档量
int	ask _volume4	卖 4 档量
int	ask _volume5	卖 5 档量
float	bid_price1	买 1 档价格
float	bid_price2	买 2 档价格
float	bid_price3	买 3 档价格
float	bid_price4	买 4 档价格
float	bid_price5	买 5 档价格
int	bid _volume1	买 1 档量
int	bid _volume2	买 2 档量
int	bid _volume3	买 3 档量
int	bid _volume4	买 4 档量
int	bid _volume5	买 5 档量
str	contract_type	合约类别
int	expire_date	到期日
str	underlying_security_cod	标的代码
float	exercise_price	行权价

4.2.3 期货快照 SnapshotFuture

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	trade_time	交易所行情数据时间
str	action_day	业务日期
str	trading_day	交易日期
float	pre_close	昨收价
float	pre_settle:	上次结算价
int	pre_open_interest	昨持仓量
int	open_interest	持仓量
float	last	最新价
float	open	开盘价
float	high	最高价
float	low	最低价
float	close	收盘价

float	volume	成交总量
float	amount	成交总金额
float	high_limited	涨停价
float	low_limited	跌停价
float	ask_price1	卖 1 档价格
float	ask_price2	卖 2 档价格
float	ask_price3	卖 3 档价格
float	ask_price4	卖 4 档价格
float	ask_price5	卖 5 档价格
int	ask _volume1	卖 1 档量
int	ask _volume2	卖 2 档量
int	ask _volume3	卖 3 档量
int	ask _volume4	卖 4 档量
int	ask _volume5	卖 5 档量
float	bid_price1	买 1 档价格
float	bid_price2	买 2 档价格
float	bid_price3	买 3 档价格
float	bid_price4	买 4 档价格
float	bid_price5	买 5 档价格
int	bid _volume1	买 1 档量
int	bid _volume2	买 2 档量
int	bid _volume3	买 3 档量
int	bid _volume4	买 4 档量
int	bid _volume5	买 5 档量
float	average_price	当日均价
float	settle	本次结算价

4.2.4 指数快照 SnapshotIndex

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	trade_time	交易所行情数据时间
float	last	最新价
float	pre_close	前收盘价
float	open	今开盘价
float	high	最高价
float	low	最低价
float	close	收盘价（仅上海有效）
int	volume	成交总量（上交所:手，深交所:张）
float	amount	成交总金额

4.2.5 港股通快照 SnapshotHKT

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	trade_time	交易所行情数据时间
float	pre_close	昨收价
float	last	最新价
float	high	最高价
float	low	最低价
float	volume	成交总量
float	amount	成交总金额
float	nominal_price	暗盘价
float	ref_price	参考价
float	bid_price_limit_up	买盘上限价
float	bid_price_limit_down	买盘下限价
float	offer_price_limit_up	卖盘上限价
float	offer_price_limit_down	卖盘下限价
float	high_limited	冷静期价格上限
float	low_limited	冷静期价格下限
float	ask_price1	卖 1 档价格
float	ask_price2	卖 2 档价格
float	ask_price3	卖 3 档价格
float	ask_price4	卖 4 档价格
float	ask_price5	卖 5 档价格
int	ask _volume1	卖 1 档量
int	ask _volume2	卖 2 档量
int	ask _volume3	卖 3 档量
int	ask _volume4	卖 4 档量
int	ask _volume5	卖 5 档量
float	bid_price1	买 1 档价格
float	bid_price2	买 2 档价格
float	bid_price3	买 3 档价格
float	bid_price4	买 4 档价格
float	bid_price5	买 5 档价格
int	bid _volume1	买 1 档量
int	bid _volume2	买 2 档量
int	bid _volume3	买 3 档量
int	bid _volume4	买 4 档量
int	bid _volume5	买 5 档量
str	trading_phase_code	交易阶段代码

4.2.6 K 线 Kline

数据类型	字段名称	说明
str	code	证券代码+市场
datetime	kline_time	交易所行情数据时间
float	open	今开盘价
float	high	最高价
float	low	最低价
float	close	收盘价
int	volume	成交总量
float	amount	成交总金额

4.3 相关算法说明

4.3.1 K 线算法说明

（1） 集合竞价的处理

对于分钟 K 线，开盘集合竞价数据的成交量包含在当日第一根 K 线，收盘集合竞价数据的成交量包含在当日最后一根 K 线。

（2） 前推算法

9:30 的 1 分钟 K 线，计算的是 9:30:00.000~9:30:59.999 期间的 K 线。

9:35 的 5 分钟 K 线，计算的是 9:35:00.000~9:39:59.999 期间的 K 线。

4.4 本地数据缓存方案说明

应用场景：

（1） 接口取**全量历史时间区间**的数据

查询接口包含 local_path 和 is_local 两个参数的接口，这两个参数必须同时配对使用，支持此本地缓存方案，本地保存全量历史数据，且每次调用接口默认增量更新本地数据，从而加速接口读取速度；

（2） 接口取**指定时间区间**的数据

查询接口包含 begin_date 和 end_date 两个参数的接口，这两个参数必须同时配对使用，仅从服务器获取数据，不本地缓存数据，速度较慢，且无增量更新机制。

4.4.1 函数入参说明

local_path 和 is_local 为参数组 1，begin_date 和 end_date 为参数组 2；

一个参数组内的参数必须同时使用；

两个参数组需独立使用，即使用参数组 1 时，参数组 2 无效；使用参数组 2 时，参数组 1 无效。

（1）local_path
类似'D://AmazingData_local_data//'，只写文件夹的绝对路径即可

（2）is_local
True:
本地 local_path 有数据的情况下，从本地取数据，但无法从服务端获取最新的数据
本地 local_path 无数据的情况下，从互联网取数据，并更新本地 local_path 的数据
False:从互联网取数据，并更新本地 local_path 的数据

（3） begin_date, end_date
开始日期、结束日期，在不同的接口中代表交易日、公告期等不同含义，具体见接口说明；即按照日期从服务端取数据，不从本地取数据（即 local_path 和 is_local 两个参数无效）。

4.4.2 本地存储文件说明

文件格式为 hdf5 格式

4.4.3 本地存储空间说明

本地存储空间，不同的数据类型和标的范围，所需空间不同。
建议本地存储空间在 500GB 以上。

5. 免责声明

为了使客户更好地了解使用中国银河证券股份有限公司(以下简称“本公司”)星耀数智服务平台（以下简称“本平台”)的相关风险，根据相关法律、行政法规、部门规章、自律组织规则和监管规定，特提供风险揭示书，请客户务必详细阅读并充分理解以下风险：

（1） 本公司使用外购或者自有的数据源作为基础数据进行数据加工、计算和分析，但并不能保证数据的及时性、准确性、真实性和完整性。

（2） 由于计算机故障以及互联网数据传输等原因，数据传输可能会出现中断、停顿、延迟、数据错误等情况；因特网和移动通讯网络遭到黑客恶意攻击、您的网络终端设备及软件系统收到非法攻击或病毒感染、您的网络终端设备及软件系统与本不兼容、因电脑的故障或互联网故障引起的中断和错误等，都可能会造成数据传输故障，由此导致的损失由您自行承担；

（3） 本平台所提供的信息数据等全部内容仅供参考，投资者须自行确认自己具备理解相关信息数据内容的专业能力，保持自身的独立判断，任何情况下本平台提

供的内容不构成对投资者的投资建议，据此操作的一切风险和损失由投资者自行承担，本公司不对任何人因参考上述内容造成的直接或间接损失或与此有关的其他损失承担任何责任。

（4） 您使用本平台过程中，凡使用您本人的用户名和密码，针对平台账号进行的操作均视为您亲自办理，由此所产生的一切后果由您承担。本公司提醒您加强账号、密码等信息的保护工作，不得出借他人使用，并建议您定期修改密码、增强密码强度、防止密码泄露、及时查询交易记录、防止用于网上交易的计算机或手机终端感染木马、病毒等。如果本公司发现同一使用账号和验证码在同一时间内被多人同时登录使用，本公司有权停止向您提供本平台相关服务，且不承担任何责任。

（5） 由于地震、水灾、火灾等不可抗力因素或者无法控制和不可预测的系统故障、设备故障、通讯故障、电力故障、网络故障及其它因素，可能本平台非正常运行甚至瘫痪，出现信息异常或信息传递异常等情况，由此产生的损失将由您承担。

（6） 本公司可能不时更新或升级本平台，您应按照本公司的技术要求在规定的时间内配合做好更新或升级工作；因您未按本公司通知要求进行变更、升级的，由此发生的任何损失由您自行承担。

（7） 如果本公司依据自身判断认为您违反本平台相关的国家法律法规、规范性文件，以及证券交易所、行业协会等自律组织的规则和要求(以下合称“法律法规”)，且不按法律法规或乙方要求及时纠正的，或影响本公司信息系统安全运行的，或监管机构、交易所、行业自律组织对本平台提出监管要求或相关业务规则发生变化，可能导致本平台的服务形式发生变化或本公司决定完全停止提供该项服务的，本公司有权立即停止您使用本平台，并且不承担任何责任，由此产生的任何损失由您承担。

（8） 本公司在遵守国家相关法律、法规、规章及自律组织规则、监管政策前提下，尽力为客户提供高速、完整、准确的金融数据服务，但因受制于数据来源、技术能力等多种因素影响，本公司不保证数据源的及时性、准确性或者完整性，因数据源的遗漏、错误、丢失、延迟、中断而可能造成的损失将由您承担，本公司不承担任何责任。

（9） 本平台的相关用户文档仅供您操作参考，如您对于本平台的使用不熟悉，可能因操作不当造成本平台出现非正常现象，上述风险可能导致发生的损失应由您自身承担，本公司不承担任何责任。

（10） 您申请使用本平台前应如实填写相关信息和资料，使用过程中信息资料发生变更应及时告知本公司，因您未及时、准确、完整地提供或变更相关信息和资料，导致本公司不能及时、有效地为您提供服务，或导致本公司依据不准确、不完整的信息提供服务，由此可能造成的损失由您自行承担。

（11） 对于客户未及时更新信息，或者不再符合本平台使用条件，或本平台权限期限到期，或存在重大风险隐患，公司认为不适合使用星耀数智服务平台时，公司可关闭客户的系统相关权限，由此导致的损失由您自行承担。

（12） 本公司开发的本平台及本平台提供的相关数据知识产权归本公司所有。本公司为您开通本平台账号后，仅供您个人使用，如您把本平台提供的全部或部分资料和数据以任何形式转移、出售和公开给任何第三人，或因您未采取必要和合适的措施保护本平台提供的资料和数据的知识产权而造成数据资料信息泄露给任何第三人，本公司有权暂停或终止您使用本平台，由此导致的损失由您自行承担。

（13） 本免责声明无法揭示您使用本平台及通过本平台从事投资交易的所有风险，故您在使用本平台之前，应全面了解相关法律法规及有关规定，对您自身的经济承受能力、风险承受能力、投资目标、风险控制能力等综合考虑，作出客观判断，对投资交易作仔细的研究。