Bitbank 自動取引

関藤大凱

2021 年 2 月 17 日

目 次

1 準備

1.1 API の発行

itbank のアカウント画面から「API」をクリックし，API を発行

1.2 python_bitbankcc のインストール

コマンドプロンプトで以下をインストールする

```
pip install git+https://github.com/bitbankinc/python-bitbankcc.git
```

1.3 API について

Bitbank の API は大きく 3 つに分類される．

- public API
  公開しているテイカー情報，板情報等
- rest API(private API)
  自身の注文履歴，資産残高の確認や，新規注文をする
- public stream API
  データをリアルタイムで受信・送信する

2 パブリック API

Table 1: パブリック API

関数	説明	引数
get_ticker	市場価格を取得	pair
get_depth	板情報を取得	pair
	最新の全約定履歴を取得	pair
get_transactions	または、	または
	指定日の全約定履歴を取得	pair, yyyymmdd=None
get_candlestick	指定日のロウソク足データを取得	pair, candle_type, yyyymmdd

2.1 テイカー情報の取得

```
#ティッカー情報の取得

#python_bitbankcc のパッケージをインポート
import python_bitbankcc

# public API class のオブジェクトを取得
pub = python_bitbankcc.public()

# ティッカー情報を取得
value = pub.get_ticker( 'btc_jpy' )

#現在の売り注文の最安値
print('sell:' + value['sell'])
#現在の買い注文の最安値
print('buy:' + value['buy'])
#過去 24時間の最高値取引価格
print('high:' + value['high'])
#過去 24時間の最安値取引価格
print('low:' + value['low'])
#最新取引価格
print('last:' + value['last'])
#過去 24時間の出来高
print('vol:' + value['vol'])

"""
通貨ペア
    btc_jpy, xrp_jpy, xrp_btc, ltc_jpy, ltc_btc, eth_jpy, eth_btc, mona_jpy,
        mona_btc, bcc_jpy, bcc_btc, xlm_jpy, xlm_btc, qtum_jpy, qtum_btc
"""
```

sell: 現在の売り注文の最安値
buy: 現在の買い注文の最安値

high: 過去 24 時間の最高値取引価格
low: 過去 24 時間の最安値取引価格
last: 最新取引価格
vol: 過去 24 時間の出来高

2.2 板情報の取得

get_transactions を用いることで，約定履歴が取得可能になる．

```
#板情報の取得

#python_bitbankcc のパッケージをインポート
import python_bitbankcc

# public API class のオブジェクトを取得
pub = python_bitbankcc.public()

# 板情報を取得
value = pub.get_depth( 'xrp_jpy' )

#売り板
print('売り板 [価格，数量]:' + ',').join(map(str,value['asks'])))
#買い板
print('買い板 [価格，数量]:' + ',').join(map(str,value['bids'])))
```

get_depth を使用することで板情報が取得可能．ここで，value には dict 型で「asks:list 型」「bids:list 型」が格納されているので，join と map を用いて文字列に変換している．

2.3 全約定履歴を取得

```
#全約定履歴の取得

#python_bitbankcc と datetime のパッケージをインポート
import python_bitbankcc
import datetime

# public API class のオブジェクトを取得
pub = python_bitbankcc.public()

# 全約定履歴を取得
value = pub.get_transactions( 'btc_jpy' )

#約定履歴の一例
trans_ex = value['transactions'][0]
print('取引ID:' + str(trans_ex['transaction_id']))
print('売買情報:' + trans_ex['side'])
print('約定価格:' + trans_ex['price'])
print('数量:' + trans_ex['amount'])
print('約定日時:' + str(datetime.datetime.fromtimestamp(trans_ex['executed_at
    ']/1000)))
```

「transcation_id」と「execued_at」はint型なのでstr型に変換している．また,「execued_at」はUnix-Timeのミリ表示なので, 日付に変更する．そこで, datetimeをインポートし, datetime.datetime.fromtimestampを用いている．日時を指定するとその日の全約定履歴も取得可能.

Src. 1: bitbank3.5

```
指定した日時の全約定履歴の取得
value = get_transactions( 'ペア' )
を変更する
value = get_transactions( 'ペア', 'YYYYMMDD 型の日付' )
YYYYMMDD 型の日付例⇒ 20180504等
```

2.4 ロウソク足データの取得

get_candlestick を用いることで，ローソク足データが取得可能となる．

```
#ロウソク足データの取得

#python_bitbankcc と datetime のパッケージをインポート
import python_bitbankcc
import datetime

# public API class のオブジェクトを取得
pub = python_bitbankcc.public()

# ロウソク足データを取得
value = pub.get_candlestick( 'xrp_jpy', '1hour', '20180504' )

#ローソク足情報の一例 (2018/5/4 午前 9時の 1時間足)
candle = value['candlestick'][0]
print('candle type:' + candle['type'])
print('始値: ' + candle['ohlcv'][0][0])
print('高値: ' + candle['ohlcv'][0][1])
print('安値: ' + candle['ohlcv'][0][2])
print('終値: ' + candle['ohlcv'][0][3])
print('出来高: ' + candle['ohlcv'][0][4])
print('対象時刻: ' + str(datetime.datetime.fromtimestamp(candle['ohlcv
    '][0][5]/1000)))
```

3 プライベート API

```
#AIP キー
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
#シークレット
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'
```

Table 2: プライベート API

関数	説明	引数
get_asset	資産の一覧を取得	
get_orders	オーダー情報を取得	pair, order_id
get_active_orders	アクティブなオーダー情報を取得	pair, options=None
order	オーダーを入れる	pair, price, amount, side, order_type
cancel_order	オーダーをキャンセルする	pair, order_id
cancel_orders	複数のオーダーをキャンセルする	pair, order_ids
get_orders_info	複数のオーダー情報を取得	pair, order_ids
get_trade_history	約定履歴を取得	pair, order_count
get_withdraw_account	出金アカウントを取得	asset
request_withdraw	出金をリクエスト	asset, uuid, amount, token

3.1 アセット情報の取得

get.asset を用いることでアセット情報が取得可能になる.

```
#アセット情報の取得

#python_bitbankcc のパッケージをインポート
import python_bitbankcc

#API キー，シークレットの設定
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'

#privte API class のオブジェクトを取得
prv = python_bitbankcc.private(API_KEY, API_SECRET)

#アセット一覧の取得
value = prv.get_asset()

#アセット情報の一例
asset_ex = value['assets'][1]
print('通貨名: ' + asset_ex['asset'])
print('保有量: ' + asset_ex['onhand_amount'])
print('ロックされている量: ' + asset_ex['locked_amount'])
print('利用可能な量: ' + asset_ex['free_amount'])
print('引き出し手数料: ' + asset_ex['withdrawal_fee'])
```

value の値を変更することで取得する情報を変更できる.

3.2 注文 ID を指定して注文情報を取得

get_order を用いることで，指定した ID の注文情報が取得可能となる.

```
#注文ID を指定して注文情報を取得

#python_bitbankcc と datetime のパッケージをインポート
import python_bitbankcc
import datetime

#API キー，シークレットの設定
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'

#privte API class のオブジェクトを取得
prv = python_bitbankcc.private(API_KEY, API_SECRET)

#注文情報の取得
value = prv.get_order( 'btc_jpy', '11702854576' )#'ペア', '注文ID'

#注文情報の一例
print('取引ID：' + str(value['order_id']))
print('通貨ペア:' + value['pair'])
print('売買情報:' + value['side'])
print('注文タイプ:' + value['type'])
print('注文時の数量:' + value['start_amount'])
print('未約定の数量:' + value['remaining_amount'])
print('約定済の数量:' + value['executed_amount'])
print('注文価格:' + value['price'])
print('平均約定価格:' + value['average_price'])
print('注文状態:' + value['status'])
print('注文日時:' + str(datetime.datetime.fromtimestamp(value['ordered_at
    ']/1000)))
```

prv.get_orders_info を用いることで，複数の注文 ID を指定することが可能となる.

単数：prv.get_order('ペア', '注文 ID')
複数：prv.get_orders_info('ペア', ['注文 ID', '注文 ID', ...])

「注文状態」には,「UNFILLED(注文中)」「PARTIALLY_FILLED(注文中 (一部約定))」「FULLY_FILLED(約定済み)」「CANCELED_UNFILLED(取消済)」「CANCELED_PARTIALLY_FILLED(取消済 (一部約定))」の 5 種類存在する.

3.3 アクティブな注文情報を取得

get_active_orders を用いることで，アクティブな注文情報が取得可能となる.

```
#アクティブな注文情報を取得

#python_bitbankcc と datetime のパッケージをインポート
import python_bitbankcc
import datetime

#API キー，シークレットの設定
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'

#privte API class のオブジェクトを取得
prv = python_bitbankcc.private(API_KEY, API_SECRET)

#アクティブな注文一覧の取得
value = prv.get_active_orders( 'btc_jpy' )

#注文一覧の一例
order_ex = value['orders'][0]
print('取引ID：' + str(order_ex['order_id']))
print('通貨ペア:' + order_ex['pair'])
print('売買情報:' + order_ex['side'])
print('注文タイプ:' + order_ex['type'])
print('注文時の数量:' + order_ex['start_amount'])
print('未約定の数量:' + order_ex['remaining_amount'])
print('約定済の数量:' + order_ex['executed_amount'])
print('注文価格:' + order_ex['price'])
print('平均約定価格:' + order_ex['average_price'])
print('注文状態:' + order_ex['status'])
print('注文日時:' + str(datetime.datetime.fromtimestamp(order_ex['ordered_at
    ']/1000)))
```

3.4 新規注文を行う

本当に注文が通るので，自己責任で行う

order により，新規注文が可能となる．一方で，cancel_order により，注文をキャンセル可能となる (複数キャンセルも可能).

```
#新規注文を行う

#python_bitbankcc と datetime のパッケージをインポート
import python_bitbankcc
import datetime

#API キー，シークレットの設定
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'

#privte API class のオブジェクトを取得
prv = python_bitbankcc.private(API_KEY, API_SECRET)

#新規注文を行う: 'ペア', '価格', '注文枚数', '売or 買', '指値or 成行'
value = prv.order( 'bcc_jpy', '200000', '0.07', 'sell', 'limit' )

#新規注文内容
print('取引ID：' + str(value['order_id']))
print('通貨ペア:' + value['pair'])
print('売買情報:' + value['side'])
print('注文タイプ:' + value['type'])
print('注文時の数量:' + value['start_amount'])
print('未約定の数量:' + value['remaining_amount'])
print('約定済の数量:' + value['executed_amount'])
print('注文価格:' + value['price'])
print('平均約定価格:' + value['average_price'])
print('注文状態:' + value['status'])
print('注文日時:' + str(datetime.datetime.fromtimestamp(value['ordered_at
    ']/1000)))

#注文のキャンセル (ここでは,上記の注文を即時にキャンセルしている)
prv.cancel_order( value['pair'], str(value['order_id']) )  #'ペア', '注文ID'
```

単数：cancel_order('ペア', '注文 ID')
複数：cancel_order('ペア', ['注文 ID', '注文 ID', ...])

ただし，以下の様に「取引」を選択した API キーを発行・使用する必要があることに注意.

Fig. 1: API キーの確認

3.5 約定履歴を取得する

故障中?

trade_history を使用することで約定履歴を取得できる.

```
#約定履歴を取得する

#python_bitbankcc のパッケージをインポート
import python_bitbankcc
import datetime

#API キー，シークレットの設定
API_KEY = '5f2d8797-5d8f-4ccb-ac3a-d2bd05d4e5ea'
API_SECRET = '216c999292e8ee068359f440cb58d5e8c895450356203cd2946f1882491a7394'

#privte API class のオブジェクトを取得
prv = python_bitbankcc.private(API_KEY, API_SECRET)

#約定履歴を取得する： 'ペア', '取得する約定数'
value = prv.get_trade_history('btc_jpy', '10')


print('取引ID：' + str(value['trade_id']))
print('通貨ペア:' + value['pair'])
print('売買情報:' + value['side'])
print('注文タイプ:' + value['type'])
print('注文時の数量:' + value['start_amount'])
print('未約定の数量:' + value['remaining_amount'])
print('約定済の数量:' + value['executed_amount'])
print('注文価格:' + value['price'])
print('平均約定価格:' + value['average_price'])
print('注文状態:' + value['status'])
print('注文日時:' + str(datetime.datetime.fromtimestamp(value['ordered_at
    ']/1000)))
print('約定日時:' + str(datetime.datetime.fromtimestamp(value['executed_at
    ']/1000)))
```