On Boading to data scrapper for deribit

September 2022

В данном файле содержится краткая информация о том что и как работает. По большому счету для начала выкачки данных необходимы только (1 и 4)

1 Настойка окружения

Скрипт протестирован для следующего окружения:

```
pandas==1.5.0
requests==2.28.1
python-decouple==3.6
```

В случае выполнения скриптов, требующих наличие API_KEY необходимо выполнить следующее (Подробнее о получении тестового API KEY: 5):

```
touch .env
nano .env
Deribit_API=[Place key here]
```

2 Скрипты для выкачки данных

В данном блоке будут представлены пояснения относительно работы скриптов.

2.1 oldDeribitAPI.py

Основной класс подключения к Deribit. Использует старую версию API, которая позволяет выкачать исторические данные за последние 5 лет. В конструктор класса передается параметр api_key, однако для выкачивания данных он не нужен. Основным методом обращения к API (без ключа) является статистический метод call api without key

```
    @staticmethod
    def call_api_without_key(additional, query: dict):
        """
        Method to call deribit api.
        :param additional: endpoint for request
        :param query: query of parameters
        :return: response json.
        """
        response = requests.get(f"https://history.deribit.com/api/v2/public/{additional}",
            params=query)
        return response.json()
```

В качестве additional выступают доступные entrypoints.
https://history.deribit.com/api/v2/public/get_instrument - Info about instrument
https://history.deribit.com/api/v2/public/get_instruments - Info about instruments
https://history.deribit.com/api/v2/public/get_last_trades_by_currency - Get last trades by currency.
https://history.deribit.com/api/v2/public/get_last_trades_by_currency_and_time - Get the latest transactions (The number of transactions is counted from the transmitted finite time) by currency.
https://history.deribit.com/api/v2/public/get_last_trades_by_instrument - Same but for instrument
https://history.deribit.com/api/v2/public/get_last_trades_by_instrument_and_time - Same but for instrument

Для каждого запроса передаваемые параметры будут иметь свой вид. К примеру для получения последних сделок для валюты:

```
    def get_last_trades(self, currency: Currency, number_of_last_trades: int):
        """
        Get trades by instrument. For each request, we take the last N trades,
        :param currency: Currency Enum.
        :param number_of_last_trades: How many last N trades we want to get.
        :return:
        """
        if number_of_last_trades > 10_000:
            raise ValueError("Too much number_of_last_trades")
        query = {'currency': currency.currency, 'count': f'{number_of_last_trades}',
            'include_old': 'true'}
        additional = "get_last_trades_by_currency"
        pprint(self.call_api_without_key(additional=additional, query=query))
```

Метод для получения исторических данных по сделкам для инструмента:

```
    def get_instrument_last_prices(self, instrument: Instrument, number_of_last_trades:
        int, number_of_requests=10_00):
    """
    Get trades by instrument. For each request, we take the last N trades, we make such
        requests M pieces.
    The API method takes as parameters end_timestamp, after each request, the last
        collected time_stamp
    set as end_timestamp for the next request. Duplicates are cut using drop_duplicates.
    :param instrument: Get Info about Instrument ENUM
    :param number_of_last_trades: Dotes in one request N
    :param number_of_requests: Number of requests M
    :return: pd.DataFrame with information of all trades.
    """
    if number_of_last_trades > 10_000:
        raise ValueError("Too much number_of_last_trades")

    instrument_request_name = instrument.instrument
    query = {'instrument_name': instrument_request_name, 'count':
        f'{number_of_last_trades}',
             'include_old': 'true', 'end_timestamp': f'{int(round(time.time() *
                1000))}'}
    # 1590480022768
    additional = "get_last_trades_by_instrument_and_time"
    response = self.call_api_without_key(additional=additional, query=query)
    df = pd.DataFrame(response["result"]["trades"])

    for _pointer in tqdm(range(0, number_of_requests)):
        query = {'instrument_name': instrument_request_name, 'count':
            f'{number_of_last_trades}',
                 'include_old': 'true', 'end_timestamp': f"{df.iloc[-1].timestamp}"}
        additional = "get_last_trades_by_instrument_and_time"
        response = self.call_api_without_key(additional=additional, query=query)
        df = pd.concat([df,
            pd.DataFrame(response["result"]["trades"])]).drop_duplicates(
            subset=['trade_seq']).reset_index(drop=True)
```

Тут важно понимать что Deribit принимает end_timestamp в виде UNIX миллисекунд. Каждый новый end_timestamp создается как timestamp последней выкачанной строчки. Фактически мы M раз выкачиваем по N последних сделок идя в прошлое по времени.

2.2 AvailableCurrencies.py

Класс перечисление доступных валют. Самым лучшим способом для добавления новых является вызов additional /public/get_currencies и создание новых экземпляров по полученному ответу

2.3 AvailableInstruments.py

Класс перечисление доступных инструментов. Нужный нам BTC-PERPETUAL является инстурментом. Самым лучшим способом для добавления новых является вызов additional /public/get_instruments и создание новых экземпляров по полученному ответу. Важно то что в параметрах этого запроса будет также присутствовать базовая валюта. К примеру чтобы получить что существует BTC_PERPETUAL необходимо написать запрос:
Не ПРОВЕРЕНО!

```
query = {"currency": currency.currency, "kind" : "future", "expired" : False}
```

3 Обработка тиковых данных о сделках

TODO: Здесь точно что-то будет.
На данный момент тиковые данные ресемплятся только в минутные бары.

4 Как начать загружать данные

PipeLine загрузки представлен в файле processingData.ipynb. По умолчанию данные начнут выкачиваться с настоящего момента и вплоть до установленных параметров (Спрогнозировать как связаны параметры с тем, сколько данных скачается - не представляется возможным, по крайней мере легким способом). Чтобы выбрать собственную начальную точку необходимо изменить флаг FROM_NOW и ввести дату начала.

```
FROM_NOW = True
deribit = DeribitConnectionOld("")
# Way to create start loading data
if FROM_NOW:
    start_load = int(datetime.datetime.now().timestamp() * 1000)
else:
    start_load = int(datetime.datetime(year=2021, month=1, day=1, hour=1, minute=1,
        second=1).timestamp() * 1000)
# Sample to get information about BTC-PERPETUAL.
df = deribit.get_instrument_last_prices(Instrument.BTC_PERPETUAL, 10_00,
    number_of_requests=5, date_of_start_loading_data=start_load)
```

5 Получение API ключа

Для получения API ключа необходимо зарегистрироваться по адресу: https://test.deribit.com/. После регистрации необходимо зайти в профиль пользователя (1) и выбрать из выпадающего списка пункт API.
После этого необхоидмо нажать "ADD NEW KEY"и создать ключ (Во всех полях я выбирать read доступ). После этого необходимо нажать "Load Keys"и ваш ключ появится в поле "Client Secret".

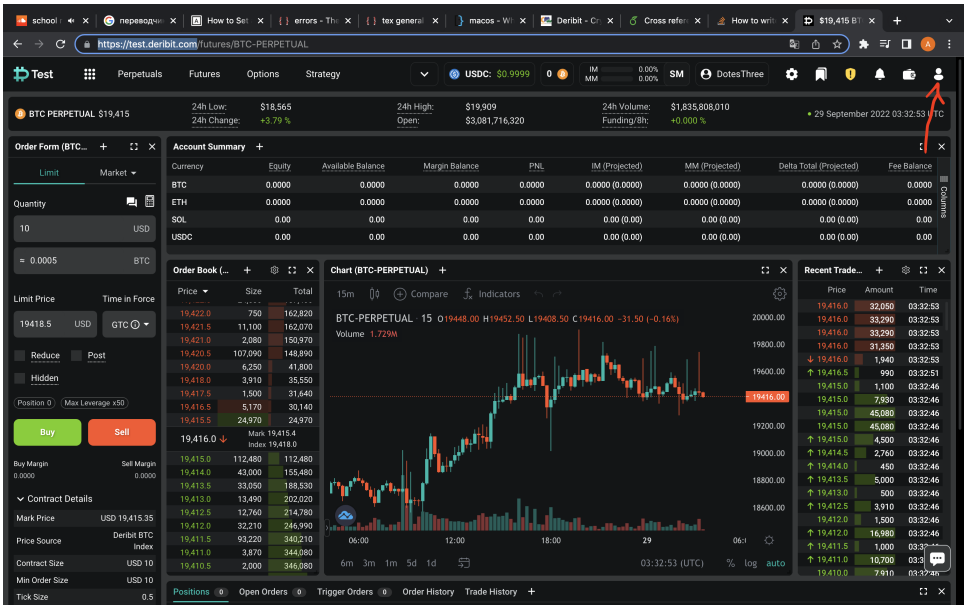

Рис. 1: User Profile