Github : https://github.com/JihongJeong/AI_Crypto, Team : 정지홍, 윤준호

```
import time
from datetime import datetime
import requests
import pandas as pd
import argparse
from requests.sessions import Session
from requests.adapters import Retry, HTTPAdapter
from urllib3.exceptions import InsecureRequestWarning
from urllib3 import disable_warnings

#Add new orderbook data to Orderbook.csv file
def write_csv(df):
    global timestamp
    new_timestamp = df.loc[0, "timestamp"].split(' ')[0]
    if new_timestamp != timestamp:
        df.to_csv("book-"+new_timestamp+"-exchange-market.csv", header = True, index = False, mode = 'w')
        timestamp = new_timestamp
    else:
        df.to_csv("book-"+timestamp+"-exchange-market.csv", header = False, index = False, mode = 'a')

#Get orderbook from DataPath, return orderbook data and status code.
#If response fail, return None
def get_response(url):
    try:
        response = session.get(DataPath_order, verify = False, timeout = 1, allow_redirects = True)
        response_data = response.json()["data"]
        response_status = response.status_code
    except:
        return None, "Response Error"

    return response_data, response_status

def get_order(res_time):
    data_order, status_order = get_response(DataPath_order)
    if data_order is None:
        return status_order
    df_bid = pd.DataFrame(data_order["bids"]).sort_values(by = "price", ascending = False)
    df_bid['type'] = 0
    df_ask = pd.DataFrame(data_order["asks"]).sort_values(by = "price", ascending = True)
    df_ask['type'] = 1
    df = pd.concat([df_bid, df_ask])
    df['timestamp'] = res_time
    df = df.reset_index().drop('index', axis = 1)
    write_csv(df)
    return status_order

def get_orderbook():
    time_start = datetime.now()
    time_last = time_start
    time_now = time_start

    #Collect orderbook data while 1 day(=86400sec)
    while (time_now - time_start).total_seconds() <= 86400:
        #Get orderbook in 1 sec interval
        time_now = datetime.now()
        if (time_now - time_last).total_seconds() < 1.0:
            continue
        time_last = time_now

        response_time = time_now.strftime('%Y-%m-%d %H:%M:%S.%f')
        status = get_order(response_time)
        if status == "Response Error":
            print(status)
            continue
        else:
            print("Orderbook : ", ((time_now - time_start).total_seconds()/86400)*100, "% is done.")
            print("Response status is " + str(status) + ", Response time is " + response_time)

#Decide what currency and how many lines to get orderbook
def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("--currency", help = "what crypto currency to get", choices = ['BTC', 'ETH'], dest = "currency", action = "store")
    parser.add_argument("--count", help = "how many orderbook lines to get", choices = ['5', '10'], dest = "count", action = "store")

    return parser.parse_args()

#Make request session
def init_session():
    session = requests.Session()
    #Update header
    my_header = {'User-Agent' : 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36'}
    session.headers.update(my_header)

    connect = 1
    backoff_factor = 0.1
    retry_status = ()

    #Set retry when connection fail
    retry = Retry(total = (connect+1), backoff_factor = backoff_factor, status_forcelist = retry_status)
    adaptor = HTTPAdapter(max_retries = retry)

    #Use retry while url starts with 'http://' or 'https://'
    session.mount("http://", adaptor)
    session.mount("https://", adaptor)
```

```
    return session

timestamp = ''
curency = ''
count = ''
DataPath_order = ''
session = init_session()

def main():
    disable_warnings(InsecureRequestWarning)

    global currency
    global count
    global DataPath_order

    args = parse_args()
    currency = args.currency
    count = args.count
    DataPath_order = 'https://api.bithumb.com/public/orderbook/' + currency + '_KRW/?count=' + count

    get_orderbook()

    session.close()

if __name__ == '__main__':
    main()
```