This file is used when we met discontinuities in the part of data crawling, to restart the crawling process from the last checkpoint.

In [ ]:

```
# Run this document if you raised internet error when crawling the data.
# Note that if no temporary data has been saved, then no need to run this document.

import pandas as pd
import akshare as ak
import re
import datetime
import time
```

In [ ]:

```
startDate, endDate = "19950101", "20211231"

if int(startDate) > int(endDate) or int(endDate) > datetime.date.today().year * 10000 + \
        datetime.date.today().month * 100 + datetime.date.today().day:
    print("Invalid Time Interval")
    quit()
```

In [ ]:

```
def ParseDate(date) -> datetime.date:
    date = list(map(int, re.findall(pattern="[0-9]+", string=str(date))))
    return datetime.date(date[0], date[1], date[2])
```

In [ ]:

```
close_df = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\close_temp.csv",
                       encoding="gbk", index_col="trade_date")
return_df = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\return_temp.csv",
                        encoding="gbk", index_col="trade_date")
BM_df = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\BM_temp.csv",
                    encoding="gbk", index_col="trade_date")
MV_df = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\MV_temp.csv",
                    encoding="gbk", index_col="trade_date")

close_df.index = pd.Series(map(ParseDate, close_df.index), name="trade_date")
return_df.index = pd.Series(map(ParseDate, return_df.index), name="trade_date")
BM_df.index = pd.Series(map(ParseDate, BM_df.index), name="trade_date")
MV_df.index = pd.Series(map(ParseDate, MV_df.index), name="trade_date")
```

In [ ]:

```
calender = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\calender.csv")
calender = pd.Series(map(lambda x: datetime.date(int(x[0:4]), int(x[5:7]), int(x[8:])),
                         calender["trade_date"].values), index=list(calender.iloc[:, 0]), name=
```

In [ ]:

```
exceptionLst = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\exceptionLst.cs
exceptionLst = list(map(lambda x: (int(re.findall("([0-9]+),", x)[0]), re.findall("\'([0-9]+)\
                        exceptionLst.iloc[:, 0]))
```

In [ ]:

```
def GetCodeLst(fromWhat: str) -> list:
    if fromWhat == "HS300":
        hs300_Stocks = pd.read_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\hs300_stocks
                                   encoding="gbk").set_index("code")
        return list(map(lambda x: re.search(pattern="[0-9]+", string=x).group(),
                        list(hs300_Stocks.index)))

    elif fromWhat == "A":
        return list(ak.stock_info_sh_name_code(indicator="主板A股")["代码"]) + \
               list(ak.stock_info_sh_name_code(indicator="科创板")["代码"]) + \
               list(ak.stock_info_sz_name_code(indicator="A股列表")["A股代码"])
```

In [ ]:

```
codeLst = GetCodeLst(fromWhat="A")
```

In [ ]:

```
def ParseDate(date: str) -> datetime.date:
    date = list(map(int, re.findall(pattern="[0-9]+", string=str(date))))
    return datetime.date(date[0], date[1], date[2])
```

Restart from the last checkpoint.

Noted that a manual input of checkpoint (value of what_now) is required.

In [ ]:

```
# what_now should be the value of last checkpoint + 1
what_now, failure, maximum_failure_allowed, length = 1201, 0, 3, len(codeLst)
print(f"Data of {length} stocks in total need to be collected, waiting......")
while what_now < length:
    code = codeLst[what_now]

    try:
        this_stock_hist_daily = ak.stock_zh_a_hist(symbol=code, period="daily",
                                                   start_date=startDate, end_date=endDate,
                                                   adjust="hfq")[["日期", "收盘", "涨跌幅"]].se

        this_stock_BMMV_daily = \
            ak.stock_a_lg_indicator(symbol=code)[["trade_date", "pb", "total_mv"]].set_index("t

        this_stock_hist_daily.index = map(ParseDate, list(this_stock_hist_daily.index))
        this_stock_BMMV_daily.index = map(ParseDate, list(this_stock_BMMV_daily.index))

        failure = 0

        try:
            close_df[code] = this_stock_hist_daily["收盘"]
            return_df[code] = this_stock_hist_daily["涨跌幅"]
            BM_df[code] = 1 / this_stock_BMMV_daily["pb"]
            MV_df[code] = this_stock_BMMV_daily["total_mv"]

            print(f"{what_now}/{length}. Data collected and merged for code: {code}")

        except:
            print(f"{what_now}/{length}. Met an unknown error when merging data of code: {code}
            exceptionLst.append((what_now, code))

        if what_now % 100 == 0:
            close_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\close_temp.csv
                            index=True, header=True)
            return_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\return_temp.c
                             index=True, header=True)
            BM_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\BM_temp.csv", ind
                         header=True)
            MV_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\MV_temp.csv", ind
                         header=True)

            if len(exceptionLst) != 0:
                pd.Series(exceptionLst).to_csv(
                    "C:\\Users\\tianj\\Project 1\\data\\HS300_temp_data\\calender.csv",
                    index=False, header=True)

            print(f"Temporary file is saved at: {code}. Position is: {what_now}")

        if what_now % 30 == 0:
            print("Resuming in 45 seconds......")
            time.sleep(45)

        what_now += 1

    except:
        failure += 1
        print(f"{what_now}/{length}. Problem encountered at code: {code}. Failure = {failure}")

        if failure > maximum_failure_allowed:
```

```
            break
        else:
            print(f"Retrying in {60 * failure} seconds......")
            time.sleep(60 * failure)
            continue


else:
    close_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\close.csv", index=True, hea
    return_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\return.csv", index=True, h
    BM_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\BM.csv", index=True, header=T
    MV_df.to_csv("C:\\Users\\tianj\\Project 1\\data\\HS300_data\\MV.csv", index=True, header=T

    if len(exceptionLst) == 0:
        print("All data are collected and merged successfully")
    else:
        print("exceptionLst is not empty: failed to merge some data")

```