```python
# -*- coding: utf-8 -*-
"""
Created on: 2024-05-20
@author:    Jasper Heuer
use:        merge all data into final analysis table
"""

# import packages ==============================================================

import os
import numpy as np
import pandas as pd
from datetime import datetime

# import data ==================================================================

basepath = "C:/Jasper/Master/Thesis/Data/"
os.chdir(basepath)

# read own data:
smb = pd.read_csv("./CSV/SMB_table_latest.csv", sep=",")
aar = pd.read_csv("./CSV/ELA_AAR_analysis_table_latest.csv")
aar = aar.loc[aar.groupby("Year").AAR.idxmin()].reset_index(drop=True) # get lowest AAR per year

# read WGMS ELA and AAR data:
wgms = pd.read_csv("./Other/DOI-WGMS-FoG-2024-01/data/mass_balance_overview.csv", sep=",")
wgms = wgms[wgms["NAME"] == "MITTIVAKKAT"]
wgms = wgms[["YEAR", "ELA", "AAR"]]

# adjust scaling issue:
wgms["ELA"] = wgms["ELA"]
wgms["AAR"] = wgms["AAR"] / 100 # missing decimal point and expressed in percent

# read WGMS mass balance data:
wgms_mass = pd.read_csv("./Other/DOI-WGMS-FoG-2024-01/data/mass_balance.csv", sep=",")

# lower bound = 9999 singles out rows with value for the entire glacier:
wgms_mass = wgms_mass[(wgms_mass["NAME"] == "MITTIVAKKAT") & (wgms_mass["LOWER_BOUND"] == 9999)]
wgms_mass = wgms_mass[["YEAR", "ANNUAL_BALANCE"]]
wgms_mass["ANNUAL_BALANCE"] = wgms_mass["ANNUAL_BALANCE"]

wgms = wgms_mass.merge(wgms, on="YEAR")
wgms = wgms.rename(columns={"YEAR": "Year"})

# get annual SMB ===============================================================

date_list = []
index_list = [623] # initalize with index of 15th of September 1984
smb_list = []
hydro_list = []

for i in range(0, len(aar)):
    date_list.append(aar["Date"][i])

for i in range(0, len(date_list)):
    index = smb[(smb["Date"] == date_list[i])].index[0]
    index_list.append(index)

# insert 15th of September as season end for no data years:
index_list.insert(10, 4275)
index_list.insert(24, 9389)
index_list.insert(26, 10119)

# calculate annual SMB:
for i in range(0, len(index_list)-1):
    section = smb[(smb.index > index_list[i]) & (smb.index <= index_list[i+1])]
    total_smb = sum(section["SMB"])
    smb_list.append(total_smb)

# calculate length of melt season:
for i in range(0, len(index_list)-1):
```

```
    hydro_year_length = index_list[i+1] - index_list[i]

    # manage unrealistically long melt seasons, due to missing data in year before:
    # (not really needed anymore)
    if hydro_year_length > 500:
        hydro_year_length_length = np.nan

    hydro_list.append(hydro_year_length)

# create dataframe ═══════════════════════════════════════════════════════════════════════

aar = aar.drop("Unnamed: 0", axis=1)

# create list with all years:
year_list = pd.DataFrame(np.arange(1985, 2024, 1), columns=["Year"])

df = year_list.merge(aar, on="Year", how="outer")
df = df.merge(wgms, on="Year", how="outer")
df = df.merge(sens_df, on="Year", how="outer")

df["SMB"] = smb_list
df["Hydro_year"] = hydro_list

df = df.rename(columns={"ELA_x": "ELA", "AAR_x": "AAR", "ANNUAL_BALANCE": "WGMS_SMB",
                        "ELA_y": "WGMS_ELA", "AAR_y": "WGMS_AAR"})

# export to disk:
df.to_csv("./CSV/complete_table_" + datetime.now().strftime("%Y%m%d_%H%M%S") + ".csv", sep=",")
df.to_csv("./CSV/complete_table_latest.csv", sep=",")
df.to_excel("./CSV/complete_table_latest.xlsx")

```