```python
# -*- coding: utf-8 -*-
"""
created on: 2024-05-20
@author:    Jasper Heuer
use:        1) convert NetCDF files to GeoTIFF
            2) adjust GDAL affine matrix
            3) crop to study area and export
"""

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

import os
import glob
import time
import shutil
import rasterio
import numpy as np
import pandas as pd
import xarray as xr

from osgeo import gdal
from datetime import datetime

# need to have rasterio and rioxarray installed

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

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

start_time = time.time() # set start time

# create new directories:
path_geotiff = "./MAR/daily_5km_tiff/"
path_cropped = "./MAR/daily_5km_cropped/"
path_resampled = "./MAR/daily_5km_resampled/"

if not os.path.exists(path_geotiff):
    os.makedirs(path_geotiff)
if not os.path.exists(path_cropped):
    os.makedirs(path_cropped)
if not os.path.exists(path_resampled):
    os.makedirs(path_resampled)

# create file list:
file_list = glob.glob("./MAR/daily_5km_raw/" + "*.nc", recursive=True)

# create list of dates:
date_list = []

for i in range(0, np.size(file_list)):
    date_list.append(file_list[i].split("ERA5-")[1][0:4])

# convert data to GeoTIFF format ===============================================

for i in range(0, np.size(file_list)):
    # read data:
    ds = xr.open_dataset(file_list[i], decode_coords="all")

    # create list of days:
    start_day = datetime.strptime(str(date_list[i]) + "-01-01", "%Y-%m-%d")
    print(start_day)
    year_length = len(ds.coords["TIME"])
    day_list = pd.date_range(start_day, periods=year_length)

    # read and export surface mass balance data:
    for j in range(0, len(ds.coords["TIME"])):
        smb = ds["SMB"][j]

        # set spatial extent and CRS:
        smb = smb.rio.set_spatial_dims(x_dim="x", y_dim="y")
```

```
        smb.rio.write_crs("epsg:3413", inplace=True) # CSR of MAR data

        # export as GeoTIFF:
        smb.rio.to_raster(path_geotiff + "MAR_SMB_" + str(day_list[j])[0:10] + ".tif")

    print("Exported: " + str(file_list[i].split("\\")[1][0:-3]))

# adjust GDAL affine matrix ============================================================

file_list2 = glob.glob(path_geotiff + "*.tif", recursive=True)

for i in range(0, np.size(file_list2)):
    # read data:
    data = gdal.Open(file_list2[i])
    geotrans = data.GetGeoTransform()

    # calculate new transform matrix:
    new_geotransform = [geotrans[0] * 1000, geotrans[1]* 1000, 0.0,
                        geotrans[3] * 1000, 0.0, geotrans[5] * 1000]

    # set new transform:
    data.SetGeoTransform(new_geotransform)

    # reproject to common grid:
    data_resample = gdal.Warp(path_resampled + str(file_list2[i].split("\\")[1][0:-4]) + "_res.tif",
                              data, dstSRS="EPSG:32624", xRes=30, yRes=-30,
                              cutlineDSName="./Masks/MAR_mask_UTM-24N.shp", # cut by extent of mask
                              cropToCutline=True,
                              outputType=gdal.GDT_Float32,  # comment this one out if UInt16 is wanted
                              dstNodata=np.nan)

    # set data to none:
    data = None
    data_resample = None

    # read resampled data:
    data = gdal.Open(path_resampled + str(file_list2[i].split("\\")[1][0:-4]) + "_res.tif")

    # crop data to extent of study area:
    data_cropped = gdal.Warp(path_cropped + str(file_list2[i].split("\\")[1][0:-4]) + "_crop.tif",
                              data, dstSRS="EPSG:32624", xRes=30, yRes=-30,
                              cutlineDSName="./Masks/mittivakkat_outline.shp", # cut by extend of mask
                              cropToCutline=True,
                              outputType=gdal.GDT_Float32,  # comment this one out if UInt16 is wanted
                              dstNodata=np.nan)

    # set data to none:
    data = None
    data_cropped = None

    print("Adjusted geotransform and cropped: " + str(file_list2[i].split("\\")[1][0:-4]))

# create SMB time series ===============================================================

# create file list:
file_list3 = glob.glob("./MAR/daily_5km_cropped/" + "*.tif", recursive=True)

year_list = []
month_list = []
day_list = []
date_list = []

# extract date loop:
for i in range(0, np.size(file_list3)):
    year_list.append(file_list3[i].split("SMB_")[1][0:4])
    month_list.append(file_list3[i].split("SMB_")[1][5:7])
    day_list.append(file_list3[i].split("SMB_")[1][8:10])
    date_list.append(file_list3[i].split("SMB_")[1][0:10])

# extract daily SMB:
smb_list = []

for i in range(0, np.size(file_list3)):
```

```
    data = rasterio.open(file_list3[i])
    raster = data.read()

    smb = np.nanmean(raster)
    smb_list.append(smb)

array = np.array((year_list, month_list, day_list, date_list, smb_list)).T

df = pd.DataFrame(array, columns=["Year", "Month", "Day", "Date", "SMB"])
df["Date"] = pd.to_datetime(df["Date"])
df["Date"] = df["Date"].dt.strftime("%Y%m%d")

df = df.sort_values(by=["Year", "Month", "Day"], axis=0, ascending=True) # sort dataframe by date

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

# clean up drive ═══════════════════════════════════════════════════════════════════════════════

shutil.rmtree("./MAR/daily_5km_resampled/")
shutil.rmtree("./MAR/daily_5km_tiff/")

# print duration:
print(f"Duration: {time.time() - start_time} seconds")

```