```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
These tools are for downloading and processing the tide data
Author: Alexandra Christensen
Created: Wed Dec 23 10:26:26 2020
Updated: 12/01/2022
"""

from __future__ import print_function

import sys
import os
import getopt
import datetime
import numpy as np
import matplotlib.pyplot as plt

import pyTMD.time
from pyTMD.calc_delta_time import calc_delta_time
from pyTMD.infer_minor_corrections import infer_minor_corrections
from pyTMD.predict_tidal_ts import predict_tidal_ts
from pyTMD.read_tide_model import extract_tidal_constants
from pyTMD.read_netcdf_model import extract_netcdf_constants
from pyTMD.read_GOT_model import extract_GOT_constants
from pyTMD.read_FES_model import extract_FES_constants


def maketides(LAT,LON,startdate,enddate,freq):
    #LAT,LON = 28.,-91.
    print('\n Getting Tides from pyTMD at %s,%s ' %(LAT,LON))
    #-- verify longitudes
    LON %= 360
    #-- convert from calendar date to days relative to Jan 1, 1992 (48622
MJD)
    YMD = datetime.datetime.strptime(startdate, '%Y%m%d')
    YMD2 = datetime.datetime.strptime(enddate, '%Y%m%d')
    duration = (YMD2-YMD).days
    #YMD = datepick.value
    #-- calculate a weeks forecast every minute
    TIME = pyTMD.time.convert_calendar_dates(YMD.year, YMD.month,
        YMD.day, hour=np.arange(duration*24))
    TIDE_MODEL = 'TPXO9-atlas-v3'
    #-- select between tide models

    model_format = 'netcdf'


    model_directory =
os.path.join("/Users/Alchrist/Documents/Tools/TPXO9_atlas_nc/")
    grid_file = model_directory + 'grid_tpxo9_atlas_30_v3.nc.gz'
    model_files =
['h_q1_tpxo9_atlas_30_v3.nc.gz','h_o1_tpxo9_atlas_30_v3.nc.gz',
        'h_p1_tpxo9_atlas_30_v3.nc.gz','h_k1_tpxo9_atlas_30_v3.nc.gz',
```

```
        'h_n2_tpxo9_atlas_30_v3.nc.gz','h_m2_tpxo9_atlas_30_v3.nc.gz',
        'h_s2_tpxo9_atlas_30_v3.nc.gz','h_k2_tpxo9_atlas_30_v3.nc.gz',
        'h_m4_tpxo9_atlas_30_v3.nc.gz','h_ms4_tpxo9_atlas_30_v3.nc.gz',
        'h_mn4_tpxo9_atlas_30_v3.nc.gz','h_2n2_tpxo9_atlas_30_v3.nc.gz']
    model_files = [model_directory+s for s in model_files]
    TYPE = 'z'
    SCALE = 1.0/1000.0
    METHOD = 'spline'

    #amp,ph,D,c = extract_netcdf_constants(np.array([LON]),
np.array([LAT]),
    #    model_directory, grid_file, model_files, TYPE=TYPE,
METHOD='spline',
    #    SCALE=SCALE) OLD VERSION includes model_directory as input
variable
    ##old version (5 variables)
    # amp,ph,D,c = extract_netcdf_constants(np.array([LON]),
np.array([LAT]),
    #     model_directory,grid_file,model_files, TYPE=TYPE,
GZIP=True,METHOD=METHOD,
    #     SCALE=SCALE)

    amp,ph,D,c = extract_netcdf_constants(np.array([LON]),
np.array([LAT]),
        grid_file,model_files,
        TYPE=TYPE, GZIP=True,METHOD=METHOD,
        SCALE=SCALE)
    deltat = np.zeros_like(TIME)


    #-- calculate complex phase in radians for Euler's
    cph = -1j*ph*np.pi/180.0
    #-- calculate constituent oscillation
    hc = amp*np.exp(cph)

    #-- convert time from MJD to days relative to Jan 1, 1992 (48622 MJD)
    #-- predict tidal elevations at time 1 and infer minor corrections
    TIDE = predict_tidal_ts(TIME, hc, c,
        DELTAT=deltat, CORRECTIONS=model_format)
    MINOR = infer_minor_corrections(TIME, hc, c,
        DELTAT=deltat, CORRECTIONS=model_format)
    TIDE.data[:] += MINOR.data[:]
    #-- convert to centimeters
    TIDE.data[:] *= 100.0

    #-- differentiate to calculate high and low tides
    diff = np.zeros_like(TIME, dtype=np.float)
    #-- forward differentiation for starting point
    diff[0] = TIDE.data[1] - TIDE.data[0]
    #-- backward differentiation for end point
    diff[-1] = TIDE.data[-1] - TIDE.data[-2]
    #-- centered differentiation for all others
    diff[1:-1] = (TIDE.data[2:] - TIDE.data[0:-2])/2.0
    #-- indices of high and low tides
```

```
    htindex, = np.nonzero((np.sign(diff[0:-1]) >= 0) & (np.sign(diff[1:])
< 0))
    ltindex, = np.nonzero((np.sign(diff[0:-1]) <= 0) & (np.sign(diff[1:])
> 0))

    #abstime = np.linspace(0,duration*60*60*24-
(60*freq),int(duration*24*(60/freq)))
    abstime = np.linspace(0,duration*60*60*24-(3600),int(duration*24))
    tides = TIDE.data/100
    return abstime, tides
```