```
from pylab import *
from pyqg import qg_model
import time
import math
import multiprocessing

#last updated: Oct 11, 2015, Lei Wang
#some parameters
hour=3600.
day=24*hour
year=360*day
beta=1.5e-11                # gradient of coriolis parameter

# -------------Please specify model setup ----------------------------------------
nx=256;
dt=5.*60.                   # numerical timstep
tmax=60*year         # total time of integration
tsnapstart=10*year   # starting point

# -------------Please specify your parameter -------------------------------------

L=                       # domain size (m)
rd=                  # deformation radius (m)
delta=                   # layer thickness ratio (H1/H2)
U1=                      # upper layer flow (m/s)
U2=                       # lower layer flow (m/s)

# timestepping parameters
# write out snapshot
tsnapint=5*day

#long-term average
tavestart=tsnapstart    # start time for averaging
taveint=day             # time interval used for summation in longterm average in sec
onds

#don't change
tcfl=tsnapint/dt             # interval for cfl writeout (in timesteps)

#other parameters
ny=nx;
tsnapints = np.ceil(tsnapint/dt)
nt = np.ceil(np.floor((tmax-tsnapstart)/dt+1)/tsnapints)

# ----------------------------RUN THE MODEL-------------------------------------
def mymodel(tau):

    # set up the model parameter for each individual run
    rek=1./float(tau)/86400           # linear drag in lower layer
    ph1_snap=zeros((ny,nx,nt),dtype=complex_)
    ph2_snap=zeros((ny,nx,nt),dtype=complex_)
    t_snap=zeros((1,1,nt))
    tind=0

    # dynamical core of this model
    tic=time.time()
    m = qg_model.QGModel(nx=nx,L=L, beta=beta,rek=rek,rd=rd,delta=delta,U1=U1,U2=U2,
        dt=dt,tcfl=tcfl,tmax=tmax,tavestart=tavestart, taveint=taveint)

    for snapshot in m.run_with_snapshots(tsnapstart=tsnapstart, tsnapint=tsnapint):
        ph1_snap[:,:,tind]=m.ph1
        ph2_snap[:,:,tind]=m.ph2
        t_snap[0,0,tind]=m.t
        tind=tind+1

    # get some diagnostic
    #KE
    KE1spec = m.get_diagnostic('KE1spec')
    KE2spec = m.get_diagnostic('KE2spec')
    EKE1 = m.get_diagnostic('EKE1')
    EKE2 = m.get_diagnostic('EKE2')
    #balance terms
    EKEdisspec = -2*m.rek*m.del2*m.get_diagnostic('KE2spec')
```

```
    EKEdiss = m.get_diagnostic('EKEdiss')
    APEgenspec = m.get_diagnostic('APEgenspec')
    APEgen = m.get_diagnostic('APEgen')
    APEflux = m.get_diagnostic('APEflux')

    # save all snapshots
    filename = "/scratch/leiw/pyqg/fricnew/fricnew_%03d_u004_d025_nx256.npz" % tau
    savez(filename,ph1_snap=ph1_snap,ph2_snap=ph2_snap,t_snap=t_snap,rd=m.rd,
        beta=m.beta,delta=m.delta,U1=m.U1,U2=m.U2,
        beta1=m.beta1,beta2=m.beta2,F1=m.F1,F2=m.F2,L=m.L,W=m.W,dx=m.dx,
        dy=m.dy,del1=m.del1,del2=m.del2,wv2=m.wv2,kk=m.kk,ll=m.ll,
        KE1spec=KE1spec,KE2spec=KE2spec,EKE1=EKE1,EKE2=EKE2,
        EKEdisspec=EKEdisspec,EKEdiss=EKEdiss,APEgenspec=APEgenspec,APEgen=APEgen,AP
Eflux=APEflux)

    # now write out some statistics
    filename2 = "/scratch/leiw/pyqg/fricnew/fricnew_%03d_u004_d025_nx256.txt" % tau
    myfile = open(filename2, "w")
    myfile.write("-----------------------------------------------------\n")
    myfile.write("Finish Running the model!, with friction: {0}\n".format(tau))
    myfile.write("{0} s for parallel computation. \n".format( time.time() - tic))
    myfile.write("Finished write data! All is done!\n")
    myfile.write("nx=ny={0}\n".format(nx))
    myfile.write("sqrt EKE1/EKE2={0}\n".format( math.sqrt(EKE1/EKE2)))
    myfile.write("lost APEgen={0}\n".format( 100.*(APEgen-EKEdiss)/APEgen ))
    myfile.write("-----------------------------------------------------\n")
    myfile.close()

# ----------------------------RUN THE MODEL---------------------------
tic0=time.time()
taulist = [10,20,30,40,50,60,70,80] #this is the list of frictional strength (in day
s) that you want to explore.
if __name__ == "__main__":
    pool = multiprocessing.Pool(processes=8)
    pool.map(mymodel, taulist)
    pool.close()
    pool.join()

print "%f s for all computation." % (time.time() - tic0)
```