```
"""Script to run Octabuoy in Wasim"""
#!/usr/bin/env python
from WasimHandler import WasimJob, WasimJobList # Required for running fourier, harmonic2sif
etc...
from Rules import *
import sys, time
import numpy
from scipy.stats import norm

#=================================================================================================
#   W A S I M J O B S   B A S E   C A S E S
#=================================================================================================


run_0kn = RuleBasedJob(  linear, speed( 0.0,'kn'), mass(type='pnt',massfile=False) )
run_0kn_nl = RuleBasedJob(  nonlinear, speed( 0.0,'kn'), mass(type='pnt',massfile=False) )
"""Calculates Tz"""
def tz(tp,gamma):
    return tp*(0.6673+0.05037*gamma-0.006230*gamma**2 + 0.0003341*gamma**3)

periods     = [9.0, 9.8, 10.4, 13.0, 11.4, 15.4, 14.9, 15.8, 17.2]
wave_heights = [4.2, 3.9, 5.6, 10.0, 6.7, 15.8, 15.2, 17.0, 19.8]
titles = ['1 year 1', '1 year 2', '10 year 1', '10 year 2', '100 year 1', '100 year 2','100
year 3','100 year 4', '1000 year 1']
gamma = 2.2
def generate_irregular():



    for tp,h  in zip(periods,wave_heights):
        job = RuleBasedJob()
        a = {
        'var.wavegen.irregular.PM.Hs'      : h,
        'var.wavegen.irregular.PM.Tz'      : tz(tp,gamma),
        'var.wavegen.irregular.PM.gamma'   : gamma,    # 1.0: PM spectrum   <=7: JONSWAP
spectrum

         # Frequency spacing ( 1 and 2 are without signal repetition )
        'var.wavegen.irregular.PM.spacing'  : 2,    # 0: Constant  1: Faltinsen  2: Pastoor

        'var.wavegen.irregular.PM.shift.t'  :  0.0,
        'var.wavegen.irregular.PM.shift.x'  :  0.0,
        'var.wavegen.irregular.PM.shift.y'  :  0.0,

        'var.wavegen.irregular.PM.nregion'         :  1,
        'var.wavegen.irregular.PM.region(1).nfreq' :  400,
        'var.wavegen.irregular.PM.region(1).Tmin'  :  4.0,
        'var.wavegen.irregular.PM.region(1).Tmax'  :  40,#     ! math expressions inside {}

        'var.wavegen.irregular.PM.heading.n'         : 1,
        'var.wavegen.irregular.PM.heading.npowcos'   : 0,#      ! 2, 4, 6, 8, 10, ....
        'var.wavegen.irregular.PM.heading.beta_main' : 180.0,
        'var.wavegen.output.format'                     : '3',
        'var.wavegen.output.file.wave'                  : 'Hs%.3f_Tp%.3f_Beta180.0.sea'%(h,tp)
        }
        job.update(a)
        job.GenerateWaves()
```

```
def run_irregular_sea():

    jobTemplate = {
             'dir.project'       : dir_project,
             'exe.setup'         : os.path.join(dir_exe,'wsetup_20080523.exe'),
             'exe.solve'         : os.path.join(dir_exe,'wsolve_07052008.exe'),
             'exe.mesh'          : os.path.join(dir_exe,'WASIM_MESH.EXE'),
             'exe.dformd'        : os.path.join(dir_exe,'DFORMD.DLL'),
             'import.wasim.inp'  : os.path.join(dir_project,r'input/inp_template/template.inp'),
             'exe.wave_generator': os.path.join(dir_project,r'exe/wave_generator_061215.exe'),
             }


    bases = [run_0kn, run_0kn_nl]


    jobs = WasimJobList()
    for base in bases:
        for tp,h,title  in zip(periods,wave_heights,titles):
            title = title.replace(' ','-')
            job=WasimJob(base,jobTemplate)
            job.delete(['var.wasim.period',
                        'var.wasim.ampltd',
                        'var.wasim.phase'])
            job.update({'var.wasim.iwave_model' : 3,
                        'var.wasim.iwave_file'  :
os.path.join(os.getcwd(),'input','sea','Hs%.3f_Tp%.3f_Beta180.0.sea'%(h,tp)),
                        'var.wasim.nomega'      : 0,
                        'dir.case' : title
                       })
            jobs.append(job)
    jobs.install()
    jobs.enque()

""" test() preforms statistical operations to the time series"""
def test():
    from matplotlib import pyplot as plt
    import numpy as np
    jobTemplate = {
         'dir.project'       : dir_project,
         'exe.setup'         : os.path.join(dir_exe,'wsetup_20080523.exe'),
         'exe.solve'         : os.path.join(dir_exe,'wsolve_07052008.exe'),
         'exe.mesh'          : os.path.join(dir_exe,'WASIM_MESH.EXE'),
         'exe.dformd'        : os.path.join(dir_exe,'DFORMD.DLL'),
         'import.wasim.inp'  : os.path.join(dir_project,r'input/inp_template/template.inp'),
         'exe.wave_generator': os.path.join(dir_project,r'exe/wave_generator_061215.exe'),
         }


    bases = [run_0kn, run_0kn_nl]


    jobs = WasimJobList()
    for tp,h,title  in zip(periods,wave_heights,titles):
        print '-'*20
        heaves = []
        pitchs = []
        legends = []
        for base in bases:
```

```
            title = title.replace(' ','-')
            job=WasimJob(base,jobTemplate)
            job.delete(['var.wasim.period',
                        'var.wasim.ampltd',
                        'var.wasim.phase'])
            job.update({'var.wasim.iwave_model' : 3,
                        'var.wasim.iwave_file'  :
os.path.join(os.getcwd(),'input','sea','Hs%.3f_Tp%.3f_Beta180.0.sea'%(h,tp)),
                        'var.wasim.nomega'      : 0,
                        'dir.case' : title
                        })
            jobs.append(job)
            paths = job.paths()
            mot = paths['file.run.mot']
            t,surge,sway,heave,roll,pitch,yaw,wave = np.loadtxt(mot,skiprows=3,unpack=True)
            """Calculates MPL and STD"""
            print title, base['id'], 2*(2*np.std(heave)**2*np.log(10800/tz(tp,gamma)))**0.5,
2*(2*np.std(pitch)**2*np.log(10800/tz(tp,gamma)))**0.5
            print np.std(heave), np.std(pitch)
            plt.figure()
            plt.hist(heave,100)
            plt.savefig(os.path.join(os.path.split(mot)[0],'Heave.png'))

            plt.figure()
            plt.hist(pitch,100)
            plt.savefig(os.path.join(os.path.split(mot)[0],'Pitch.png'))

            heaves.append(heave)
            legends.append(base['id'].split()[0])
            pitchs.append(pitch)

        """Makes QQ plots"""
        heave1 = heaves[0]
        heave2 = heaves[1]

        heave1.sort()
        heave2.sort()

        plt.figure()
        plt.plot(heave1, heave2)
        plt.xlabel(legends[0])
        plt.ylabel(legends[1])
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_heave.png'))

        F_H = numpy.arange(len(heave1), dtype=float) / len(heave1)
        heave_norm = norm.ppf(F_H)

        plt.figure()
        plt.plot(heave_norm, heave1)
        plt.plot(heave_norm, heave2)
        plt.legend(legends, loc='best')
        plt.xlabel('Standard normal')
        plt.ylabel('Data')
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_norm_heave.png'))

        pitch1=pitchs[0]
        pitch2=pitchs[1]

        pitch1.sort()
        pitch2.sort()
```

```
        plt.figure()
        plt.plot(pitch1, pitch2)
        plt.xlabel(legends[0])
        plt.ylabel(legends[1])
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_pitch.png'))

        F_P = numpy.arange(len(pitch1), dtype=float) / len(pitch1)
        pitch_norm = norm.ppf(F_P)

        plt.figure()
        plt.plot(pitch_norm, pitch1)
        plt.plot(pitch_norm, pitch2)
        plt.legend(legends, loc='best')
        plt.xlabel('Standard normal')
        plt.ylabel('Data')
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_norm_pitch.png'))

        plt.figure()
        plt.plot(pitch_norm, pitch1)
        plt.xlabel('Standard normal')
        plt.ylabel('Data')
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_norm_pitch1.png'))
        plt.figure()
        plt.plot(pitch_norm, pitch2)
        plt.xlabel('Standard normal')
        plt.ylabel('Data')
        plt.savefig(os.path.join(os.path.split(mot)[0],'QQ_norm_pitch2.png'))

    for job in jobs:
        job.trend()


#=============================================================================================
=====

def main():
    tasks = sys.argv[1:]
    out = open('task.log','a')
    for task in tasks:
        if not '(' in task: task+='()'
        date = time.ctime()
        exec task
        out.write('%s # %s # %s\n' % (date, task, __file__))
    out.close()
    logScript(tasks,__file__)

if __name__ == "__main__":
    main()
```