```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Dec  1 14:01:18 2016

@author: andrewalferman
"""

import numpy as np
import matplotlib.pyplot as plt

problem = 1

def reynoldsfun(properties, l, v):
    """Function that finds the reynolds number given fluid properties,
    length, and velocity"""
    rho, mu = properties
    return rho * v * l / mu


def laminarbigcf(reynolds):
    return 1.328 / np.sqrt(reynolds)


def turbbigcf(reynolds):
    return 0.072 / (reynolds**0.2)


def combinedcf(reynolds):
    Recrit = 5.e5
    A = 1.328 * (Recrit**0.5) * (0.056 * (Recrit**0.3) - 1.0)
    return turbbigcf(reynolds) - (A / reynolds)


def dragforce(dragparams):
    cf, rho, u_inf, b, L = dragparams
    return cf * 0.5 * rho * u_inf**2 * b * L


def critlength(v, properties):
    rho, mu = properties
    return (5.e5) * mu / (rho * v)


def findvel(reynolds, parameters, L):
    rho, mu = parameters
    return mu * reynolds / (rho * L)


# Note that all units are in meters, seconds, kg, etc.

# Initialize basic geometry of the problem
```

```
surflength = 5
surfwidth = 8
surfarea = surflength * surfwidth

# Fluid properties assumed at 20 degrees C
rho_water = 1000
mu_water = 1.e-3

reproperties = rho_water, mu_water

roughnesses = [2000., 5000., 1.e4, 5.e4]

hoursperday = 8

secondsperyear = hoursperday * 60 * 60 * 365. * 10

# Obtain velocity values
if problem == 1:
    #velocity = [0.01, 0.05, 1., 5.]
    velocity = [x*0.01 for x in range(1, 1000)]
else:
    # Picking some nice and easy values for the subsequent problems
    reynoldsvals = [6.e5, 1.e6, 5.e8, 1.e9]
    velocity = []
    for i in reynoldsvals:
        velocity.append(findvel(i, reproperties, surflength))
    # These values manually picked from the provided graph
    # Each row corresponds to a velocity, each column a roughness value
    cds = [[0.0060, 0.0047, 0.0045, 0.0045], [0.0065, 0.0047, 0.004, 0.004]
           [0.007, 0.0058, 0.005, 0.0036], [0.007, 0.0058, 0.005, 0.0036]]

# Initialize lists for saving values
etas, relist, drags, powers = [], [], [], []

# Calculate roughness values, just in case they are needed
roughnessvalues = [2000., 5000., 1.e4, 5.e4]
for e in roughnessvalues:
    etas.append(surflength/e)

# Calculate drag force for each velocity
initdragmatrix = True
findpower = True
for v in range(len(velocity)):
    # Shorthand for the velocity in this for loop
    velv = velocity[v]
    if problem == 1:
        # Re across at the trailing edge of the plate
        rev = reynoldsfun(reproperties, surflength, velv)
        relist.append(reynoldsfun(reproperties, surflength, velv))
        critl = critlength(velv, reproperties)
        if rev < 5.e5:
            cf = laminarbigcf(rev)
```

```python
            else:
                cf = combinedcf(rev)
            dragparams = cf, rho_water, velv, surfwidth, surflength
            drags.append(dragforce(dragparams))
            powers.append(drags[v] * velv)
            if drags[v] * velv > (8775.77 * 4) and findpower == True:
                print('THE VELOCITY NEEDED IS:')
                print(velv)
                findpower = False
        else:
            if initdragmatrix == True:
                drags = np.zeros([len(velocity), len(roughnesses)])
                powers = np.zeros([len(velocity), len(roughnesses)])
                joulesperyear = np.zeros([len(velocity), len(roughnesses)])
                costperyear = np.zeros([len(velocity), len(roughnesses)])
                initdragmatrix = False
            for r in range(len(roughnesses)):
                cf = cds[v][r]
                dragparams = cf, rho_water, velv, surfwidth, surflength
                drags = np.array(drags)
                drags[r][v] = dragforce(dragparams)
                powers = np.array(powers)
                powers[r][v] = drags[r][v] * velv
                joulesperyear = np.array(joulesperyear)
                joulesperyear[r][v] = powers[r][v] * secondsperyear
                costperyear[r][v] = joulesperyear[r][v] * 2.168e-8

# Turn all the lists into arrays and format for nice printouts
relist = np.array(relist)
#etas = np.array(etas)
drags = np.array(drags)
powers = np.array(powers)
np.set_printoptions(formatter={'float': lambda x: format(x, '6.3E')})

#relist = ['%.2f' % elem for elem in relist]
#etas = ['%.2f' % elem for elem in etas]
#drags = ['%.2f' % elem for elem in drags]
#powers = ['%.2f' % elem for elem in powers]

"""
print('Velocities:')
print(velocity)
print('Reynolds Values:')
if problem == 1:
    print(relist)
else:
    print(reynoldsvals)
print('Drag Forces:')
print(drags)
print('Power Required:')
print(powers)
"""
```

```
if problem == 1:
    fig = plt.figure()
    ax = plt.gca()
    ax.set_xscale('log')
    ax.set_yscale('log')
    fig.suptitle('Velocity vs. Power', fontsize = 14)
    plt.xlabel('Velocity (meters/second)')
    plt.ylabel('Power Needed (Watts)')
    plt.plot(velocity, powers)
    plt.grid(b=True, which='both')
    plt.show()
else:
    print('Cost per Year:')
    print(costperyear)
    fig = plt.figure()
    ax = plt.gca()
    ax.set_xscale('log')
    ax.set_yscale('log')
    fig.suptitle('Velocity vs. Power', fontsize = 14)
    plt.xlabel('Velocity (meters/second)')
    plt.ylabel('Power Needed (Watts)')
    for r in range(len(roughnesses)):
        plt.plot(velocity, powers[r],
                 label='Roughness: {}'.format(roughnesses[r]))
    plt.grid(b=True, which='both')
    plt.legend(bbox_to_anchor=(1, 1), loc=2)
    plt.show()
```