```
"""
Author   : Abraham Flores
File     : p1037.py
Language : Python 3.6
Created  : 5/8/2018
Edited   : 5/8/2018

San Digeo State University
MTH 693b : Computational Partial Differential Equations

Strikwerda 10.3.7 : Periodic Boundary Conditions

One Way Wave Equation
    U_t + U_x = 0
    x = [-1,1]
    t = [0,1]

    h = 1/10 , 1/20 , 1/40 , 1/80
    lambda = 0.8

                 |   1  for |x| < 1/2
    A.  u_0(x) = |  1/2 for |x| = 1/2
                 |   0  for |x| > 1/2

    B. u_0(x) = cos(piX)


    C. U_0(x) = (cos(piX))^2 for |x| < 1/2 : Else U_0(x) = 0


"""
import os,glob
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

#Generates intial value function
def foo1(x):
    if abs(x) < .5:
        return 1
    elif abs(x) > .5:
        return 0

    return 0.5

def foo2(x):
    return np.cos(np.pi*x)

def foo3(x):
    if abs(x) <= .5:
        return np.cos(np.pi*x)**2
    return 0

def plot(x,U,err,time,title,fileLoc):
    sns.set(font_scale = 2.0)
    sns.set_style("darkgrid", {"axes.facecolor": ".9"})

    fig,ax = plt.subplots()
    fig.set_size_inches(14.4,9)
    plt.plot(x,U,linewidth=5.0,label="t = "+ str(round(time,3)))
    plt.plot(x,err,linewidth=5.0,label=r"|ERROR|")

    plt.xticks(rotation=45)
    plt.yticks(rotation=45)
    plt.axis([-1, 1, 0, 1.0])
    plt.xlabel('x (Spatial)')
    plt.ylabel('U(x,t)')
    plt.title(title)
    plt.legend(loc=1)
    plt.savefig(fileLoc+".png")
    plt.close()

def makeGif(gifName):
```

```
    os.chdir('Figures')
    #Create txt file for gif command
    fileList = glob.glob('*.png') #star grabs everything,
    fileList.sort()
    #writes txt file
    file = open('FileList.txt', 'w')
    for item in fileList:
        file.write("%s\n" % item)
    file.close()

    os.system('convert -delay 10 @FileList.txt ' + gifName + '.gif')
    os.system('del FileList.txt')
    os.system('del *.png')
    os.chdir('..')

#Lax Friedrichs Scheme
def LaxFriedrichs(h,Lamb,intial_foo,title):
    #generate array of intial values at t = 0
    x = np.arange(-1,1+h,h)
    next_ = np.array([intial_foo(dx) for dx in x])

    dt = lamb*h
    time = 0

    #plot
#    plt_title = "INITIAL: "+title + " -- h: " + str(h) + " LAMBDA: " +str(Lamb)
#    outFile = "Figures\Lax0000"
#    plot(x,next_,[0.0]*len(x),0.0,plt_title,outFile)

    iters = 1
    while time < 1.0:
        time = iters*dt
        #implement Scheme w/ Periodic Boundary Conditions
        next_ = .5*((1+Lamb)*np.roll(next_,1) + (1-Lamb)*np.roll(next_,-1))
        tmp = []
        for dx in x:
            x_new = dx - time
            if x_new < -1:
                x_new += 2.0
            tmp.append(intial_foo(x_new))
        exact = np.array(tmp)
        err = abs(exact-next_)
        inf = max(err)
        L2 = np.sqrt(sum(err*err))

        #plot
#        str_time = '0'*(4-len(str(iters)))+str(iters)
#        outFile = "Figures\Lax" + str_time
#        plot(x,next_,err,time,plt_title,outFile)
        #Step Forward
        iters += 1

    #makeGif
#    makeGif(title+"_h_"+str(h)+"_Lambda_"+str(Lamb))
    return inf,L2

def best_fit(X, Y):

    xbar = sum(X)/len(X)
    ybar = sum(Y)/len(Y)
    n = len(X) # or len(Y)

    numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
    denum = sum([xi**2 for xi in X]) - n * xbar**2

    b = numer / denum
    a = ybar - b * xbar

    return a, b

def plot_norm(h,inf_norm,L2_norm,title):
    sns.set(font_scale = 2.0)
    sns.set_style("darkgrid", {"axes.facecolor": ".9"})
```

```
    fig,ax = plt.subplots()
    fig.set_size_inches(14.4,9)
    plt.scatter(h,inf_norm,linewidth=3.0,color="r",label=r'$-Log_{10}$[INFINITY NORM
]')
    plt.scatter(h,L2_norm,linewidth=3.0,color="b",label=r'$-Log_{10}$[L2 NORM]')

    plt.xlabel(r'$-Log_{10}$[dx]')
    plt.ylabel(r'$-Log_{10}$|ERROR|')

    a_inf, b_inf = best_fit(h, inf_norm)
    yfit = [a_inf + b_inf * xi for xi in h]
    plt.plot(h, yfit,color="k",label="(INF) SLOPE: "+str(round(b_inf,5)))

    a_L2, b_L2 = best_fit(h, L2_norm)
    yfit = [a_L2 + b_L2 * xi for xi in h]
    plt.plot(h, yfit,color="k",label="(L2)  SLOPE: "+str(round(b_L2,5)))
    plt.legend()

    plt.savefig("Figures/Error/norm_err_"+title+".png")
    plt.close()

if __name__ == '__main__':
    grid_spacing = [1.0/10,1.0/20,1.0/40,1/80]
    lamb = 0.8
    infA = []
    infB = []
    infC = []
    L2A = []
    L2B = []
    L2C = []
    #Run all Cases
    for h in grid_spacing:
        i,l = LaxFriedrichs(h,lamb,foo1,"A")
        infA.append(i)
        L2A.append(l)

        i,l = LaxFriedrichs(h,lamb,foo2,"B")
        infB.append(i)
        L2B.append(l)

        i,l = LaxFriedrichs(h,lamb,foo3,"C")
        infC.append(i)
        L2C.append(l)

    plot_norm(-np.log10(grid_spacing),-np.log10(infA),-np.log10(L2A),"A")
    plot_norm(-np.log10(grid_spacing),-np.log10(infB),-np.log10(L2B),"B")
    plot_norm(-np.log10(grid_spacing),-np.log10(infC),-np.log10(L2C),"C")
"""
Report:

    We obtain convergence in both the L2 and inf norm for the error h decreases.
    This is easily seen in the error norm plots.

"""
```