```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
#*************************************************************************
#*     Example 14.1                                                      *
#*     filename: ch14pr01.m                                              *
#*     program listing number: 14.1                                      *
#*                                                                       *
#*     This program solves 2-dimensional Laplace equation using Jacobi   *
#*     method.  (Too slow for Python)                                    *
#*                                                                       *
#*     Programed by Ryoichi Kawai for Computational Physics Course.      *
#*     Last modification:  04/16/2017.                                   *
#*************************************************************************
"""
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

# parameters
a=1.0
b=1.0
V=1.0

# spacial domain
Nx=201 # number of grids
Ny=101
dx=0.1 # spacial step
dy=0.1
x=np.linspace(-b,b,Nx)
y=np.linspace(0,a,Ny)

# time step
h=1./4.

#tolerence
tol=1.e-9

# sampling interval
M=10

phi0=None
phi1=None

# initial profile
phi0=np.zeros((Ny,Nx))
phi1=np.zeros((Ny,Nx))
phi1[:,0]=phi0[:,0]=V
phi1[:,-1]=phi0[:,-1]=V
phi1[0,:]=phi0[0,:]=0.0
phi1[0,:]=phi0[-1,:]=0.0

plt.close('all')
fig, ax =plt.subplots()
k=0
diff=tol+1.
while diff>tol:
    k=k+1
    for i in range(1,Ny-1):
        for j in range(1,Nx-1):
            phi1[i,j]=h*(phi0[i-1,j]+phi0[i+1,j]+phi0[i,j-1]+phi0[i,j+1])

    if np.mod(k,M)==0:  # record the results
        diff=np.sum((phi1[:,:]-phi0[:,:])**2)
```

```
        print('{0:d} : diff={1:14.6e}'.format(k,diff))
        cax = ax.imshow(phi1,extent=(-b,b,0.0,a))
        plt.pause(0.0001)

    phi0[:,:]=phi1[:,:]

c_min=phi1.min()
c_max=phi1.max()
print(c_max,c_min)

cbar=fig.colorbar(cax, ticks=[0.0,0.2,0.4,0.6,0.8,1.0])

"""
# Plot time evolution as cuntour
figure(2)
contour(x,y,phi1)
hold on
[X,Y]=meshgrid(x(6:10:Nx-1),y(6:10:Ny-1))
[GX,GY]=gradient(phi1)
G=sqrt(GY.^2+GY.^2)
GX=GX./GGY=GY./G
quiver(X,Y,GX(6:10:Ny-1,6:10:Nx-1),GY(6:10:Ny-1,6:10:Nx-1),2)
hold off
axis equal tight
xlabel('x')
ylabel('y')
"""
```