```python
"""
Registration: xxxx;
Description: Predictor-Corrector Method dx/dt = f(t,x) with IVP (t0,x0), tn and dt.
Author: AKB
"""

import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")

# Choose first solve=1 to generate files and then plot=1 to plot them all
solve = 0;
plot  = 1;

# Enter initial conditions
#lam, x, t, tn, dt= input('Enter rate constant and initial value x, t, tn, dt: ')
lam = 1.0; x = 5.0; t = 0.0; tn = 10.0; dt = 0.5;
x0 = x;
step = int(tn/dt)

if(solve):

   def f(t,x,lam): return -lam*x   # Radioactive decay xdot = -lam*x; exact = x0*exp(-
lam*t)

   # Open a file (in C, fp is file pointer)
   if  (dt==0.5):  fp = open("data/eulerPC_ode1_dt0.5.dat","w")
   elif(dt==0.1):  fp = open("data/eulerPC_ode1_dt0.1.dat","w")
   elif(dt==0.05): fp = open("data/eulerPC_ode1_dt0.05.dat","w")
   elif(dt==0.01): fp = open("data/eulerPC_ode1_dt0.01.dat","w")

   # Predictor-Corrector iteration step
   for i in range(step):

       # Predict x using Euler method
       tp = t + dt
       xp = x + dt*f(t,x,lam)

       # Correct using Modified Euler
       xc = x + 0.5*dt*(f(t,x,lam)+f(tp,xp,lam))
       t = tp
       x = xc

       # Print it using file pointer
       print >> fp,t,x

   # Close the file
   fp.close()
   print 'Final value at x = ',t,' is y = ', x

if(plot):

   # Read the datafiles
   fp1 = np.loadtxt('data/eulerPC_ode1_dt0.5.dat');  T1 = fp1[:,0]; X1 = fp1[:,1]
   fp2 = np.loadtxt('data/eulerPC_ode1_dt0.1.dat');  T2 = fp2[:,0]; X2 = fp2[:,1]
   fp3 = np.loadtxt('data/eulerPC_ode1_dt0.05.dat'); T3 = fp3[:,0]; X3 = fp3[:,1]
   fp4 = np.loadtxt('data/eulerPC_ode1_dt0.01.dat'); T4 = fp4[:,0]; X4 = fp4[:,1]

   # Solve with odeint (note the order)
   def f(x,t): return -lam*x
   sol = odeint(f, x0, T4)

   # Plot the data
   plt.figure(1)
   plt.subplot(2,1,1)
   plt.semilogy(T1, X1, 'd', lw=2, ms=8,  c='olive', label=r'$dt=5\times 10^{-1}$')
   plt.semilogy(T2, X2, 'x', lw=2, ms=8,  c='b',     label=r'$dt=10^{-1}$')
   plt.semilogy(T3, X3, '+', lw=2, ms=16, c='m',     label=r'$dt=5\times 10^{-2}$')
   plt.semilogy(T4, X4, '<', lw=2, ms=8,  c='c',     label=r'$dt=10^{-2}$')
   plt.semilogy(T4, x*np.exp(-lam*T4),    c='k',     label="Analytic Result")
   plt.semilogy(T4, sol, '^',lw=2, ms=6,  c='pink',label='ODEINT')
   plt.legend(loc='best', prop={'size':24})
   plt.title(r'Heun''s Method : $\dot{x} = -\lambda x$',size=30)
   #plt.xlabel('t', size = 26)
   plt.xticks(size = 20)
   plt.ylabel('Concentration', size = 26)
   plt.yticks(size = 20)
   plt.grid()

   plt.subplot(2,1,2)
   plt.plot(T1, x*np.exp(-lam*T1)-X1, 'd', lw=2, ms=8, c='olive')
   plt.plot(T2, x*np.exp(-lam*T2)-X2, 'x', lw=2, ms=8, c='b')
   plt.plot(T3, x*np.exp(-lam*T3)-X3, '+', lw=2, ms=16, c='m')
   plt.plot(T4, x*np.exp(-lam*T4)-X4, '<', lw=2, ms=8, c='c')
   plt.xlabel('Time', size = 26)
   plt.xticks(size = 20)
   plt.ylabel(r'$L_1$ norm', size = 26)
   plt.yticks(size = 20)
   plt.grid()

   #plt.savefig('plot/09_eulerPC.pdf')
   plt.show()
```