```
from pylab import *
class Oscillator:
    def __init__(self,B=1):
        #drive strength
        self.y=1
        #drive frequency
        self.w=2*pi
        #oscillator natural frequency
        self.w_0=5*self.w
        #damping constant
        self.B=B
    def dataWithTrans(self):
        '''
        Function to get the position of the pendulum for a range of times using the closed form solution
        Usage: t,x=Osc.dataWithTrans()
        Input: None
        Output: two arrays consisting of time and position values
        '''
        t=linspace(0.1,25,10000)
        x=exp(t*(-1*sqrt(abs(self.B**2-self.w_0**2))-self.B))+exp(t*(sqrt(abs(self.B**2-self.w_0**2))-self.B))+2*self.B*self.y*self.w*self.w_0**2*sin(self.w*t)/
        return t,x
    def deriv(self,t,x):
        '''
        Function to compute the derivative using the FDA for the first point, the BDA for the last point and the CDA for all other points
        Usage: v=Osc.deriv(t,x)
        Input: two arrays consisting of time and position values
        Output: an array consisting of the values of the derivative of position with respect to time
        '''
        v=[0]*len(t)
        v[0]=(x[1]-x[0])/(t[1]-t[0])
        v[-1]=(x[-1]-x[-2])/(t[-1]-t[-2])
        for i in range(1,len(t)-1,1):
            v[i]=(x[i+1]-x[i-1])/(t[i+1]-t[i-1])
        return v
    def graph(self,x,y,xmin=False,xmax=False,ymin=False,ymax=False,titl=False,xlab=False,ylab=False):
        '''
        Function to plot two arrays with optional x and y limits, x and y axis labels, and title
        Usage: Osc.graph(x,y)
        Input: two arrays to be plotted
        Output: None
        '''
        plot(x,y,'b-')
        if xmin != False and xmax != False:
            xlim(xmin,xmax)
        if ymin != False and ymax != False:
            ylim(ymin,ymax)
        if title != False:
            title(titl)
        if xlabel != False:
            xlabel(xlab)
        if ylabel != False:
            ylabel(ylab)
        show()
B=logspace(1,3,5)*pi
for i in B:
    Osc=Oscillator(i)
    t,x=Osc.dataWithTrans()
    v=Osc.deriv(t,x)
    Osc.graph(t,x,xmin=0.1,xmax=15,titl='$\phi$ vs. t for $\\beta=\pi10^{%3.1f}$'%(log(i/pi)/log(10)),xlab='t(s)',ylab='$\phi$(rad)')
    Osc.graph(x,v,titl='$\\frac{d\phi}{dt}$ vs. $\phi$ for $\\beta=\pi10^{%3.1f}$'%(log(i/pi)/log(10)),xlab='$\phi$(rad)',ylab='$\\frac{d\phi}{dt}$(rad/s)')
```