```
import matplotlib.pyplot as plt
import sys
import matplotlib.patches as mpatches
import numpy as np
sys.path.append('hw_a/')
import param as P

class ballBeamAnimation:

    def __init__(self):
        self.flagInit = True                  # Used to indicate initialization
        self.fig, self.ax = plt.subplots()    # Initializes a figure and axes object
        self.handle = []                      # Initializes a list object that will
                                              # be used to contain handles to the
                                              # patches and line objects.
        plt.axis([-P.ell,2*P.ell, -2*P.ell, 2*P.ell]) # Change the x,y axis limits
        plt.plot([0,P.ell],[0,0],'b--')    # Draw a base line

        self.radius = 0.05                     # Radius of the ball, m

        # Draw Ball Beam is the main function that will call the functions:
        # drawBall, and drawBeam to create the animation.
    def drawBallBeam(self, u):
        # Process inputs to function
        z= u[0]        # Horizontal position of ball, m
        theta = u[1]   # Angle of beam, rads

        self.drawBall(z,theta)
        self.drawBeam(theta)
        # self.ax.axis('equal') # This will cause the image to not distort

        # After each function has been called, initialization is over.
        if self.flagInit == True:
            self.flagInit = False


    def drawBall(self,z,theta):
        x = z*np.cos(theta)-self.radius*np.sin(theta) # x coordinate
        y = z*np.sin(theta)+self.radius*np.cos(theta) # y coordinate
        xy = (x,y)                                    # Center of circle

        # When the class is initialized, a CirclePolygon patch object will
        # be created and added to the axes. After initialization, the
        # CirclePolygon patch object will only be updated.
        if self.flagInit == True:
            # Create the CirclePolygon patch and append its handle
            # to the handle list
            self.handle.append(mpatches.CirclePolygon(xy,
                radius = self.radius, resolution = 15,
                fc = 'blue', ec = 'black'))
            self.ax.add_patch(self.handle[0])  # Add the patch to the axes
        else:
            self.handle[0]._xy=xy

    def drawBeam(self,theta):
        X = [0,P.ell*np.cos(theta)] # X data points
        Y = [0,P.ell*np.sin(theta)] # Y data points

        # When the class is initialized, a line object will be
        # created and added to the axes. After initialization, the
        # line object will only be updated.
        if self.flagInit == True:
            # Create the line object and append its handle
```

```
            # to the handle list.
            line, =self.ax.plot(X,Y,lw = 3, c = 'gray')
            self.handle.append(line)
        else:
            self.handle[1].set_xdata(X)               # Update the line
            self.handle[1].set_ydata(Y)



# Used see the animation.
if __name__ == "__main__":

    simAnimation = ballBeamAnimation()     # Create Animate object
    z = 0.5                                # Position of cart, m
    theta = 30.0*np.pi/180                 # Angle of pendulum, rads
    simAnimation.drawBallBeam([z,theta])   # Draw the pendulum
    plt.show()
```