```
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 massSpringAnimation:

    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.

        self.mass ={'w':0.5,                   # Box width, m
                    'h':0.2}                   # Box heigh, m

        plt.axis([-2*self.mass['w'],10*self.mass['w'],
                -0.1, 5*self.mass['h']]) # Change the x,y axis limits
        plt.plot([10*self.mass['w'],0],[0,0],'b--')    # Draw a base line
        plt.plot([0.0,0.0],[0.0,2*self.mass['h']])     # Draw left wall

        # Draw Mass Spring is the main function that will call the functions:
        # drawMass, and drawSpring to create the animation.
    def drawMassSpring(self, u):
        # Process inputs to function
        z= u[0]        # Horizontal position of mass, m

        self.drawMass(z)
        self.drawSpring(z)
        # 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 drawMass(self,z):
        x = z          # x coordinate
        y = 0          # y coordinate
        xy = (x,y)     # Bottom left corner of rectangle

        # When the class is initialized, a Rectangle patch object will be
        # created and added to the axes. After initialization, the Rectangle
        # patch object will only be updated.
        if self.flagInit == True:
            # Create the Rectangle patch and append its handle
            # to the handle list
            self.handle.append(mpatches.Rectangle(xy,
                self.mass['w'],self.mass['h'], fc = 'orange', ec = 'black'))
            self.ax.add_patch(self.handle[0]) # Add the patch to the axes
        else:
            self.handle[0].set_xy(xy)         # Update patch

    def drawSpring(self,z):
        num = 10

        # Create the Springs
        h = self.mass['h']/2.0
        h2 = self.mass['h']/4.0*np.exp(-z/0.5)
        y = np.ones((num-2))*h
        for i in range(len(y)):
            if (i+2) % 2 == 0:
                y[i] = y[i]+h2
            else:
                y[i] = y[i]-h2


        X = [0] + np.linspace(z/6.0,5.0*z/6.0,num).tolist()+ [z]   # X data points
        Y = [h,h] +y.tolist() + [h,h]                              # 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 = 1, c = 'black')
            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 = massSpringAnimation()    # Create Animate object
    z = 0.5                                # Position of cart, m
    simAnimation.drawMassSpring([z])        # Draw the pendulum
    plt.show()
```