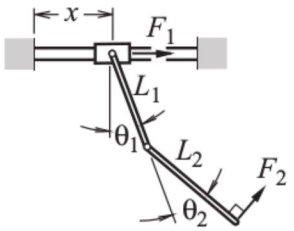
In [ ]:
```
import numpy as np
from numpy import sin,cos,pi
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

Homework #1

Example: plotting kinematic solutions

EXERCISE 1.10 The collar slides to the left as the interconnected bars swing in the vertical plane. The position of the collar is $x = 20\sin(50t)$ mm, and the angles are $\theta_1 = 0.2\pi\cos(50t)$, $\theta_2 = 0.2\pi\sin(50t - \pi/3)$ rad. Determine the velocity of the lower end by differentiating its position in terms of horizontal and vertical components.



Exercise 1.10

define kinematic known values

First, we can define the independent variable, `t`, to complete one full cycle. Then, we can use the inputs for $x,\ \theta_1,\ and\ \theta_2$.

In [ ]:
```
t=np.linspace(0,2*pi/50,100); # create time varying from 0-0.126 s (or one period)
x=20*sin(50*t);     # define x in terms of time
dx=20*50*cos(50*t); # define dx/dt in terms of time (note dx=dx/dt)
t1=0.2*pi*cos(50*t); # define theta1 (t1)
dt1=-10*pi*sin(50*t); # define dtheta1/dt (dt1)
t2=0.2*pi*sin(50*t-pi/3); # define theta2 (t2) ;
dt2=10*pi*sin(50*t-pi/3); # define dtheta2/dt (dt2);
L1=1;L2=1.5; # set lengths for L1 and L2 (none were given in problem so 1 and 1.5 m
             # chosen arbitrarily
```

define the other dependent kinematic values

In the lecture, you derived the positions of the link connections using the given values. Here, we apply those definitions to get the global positions of each link endpoint.

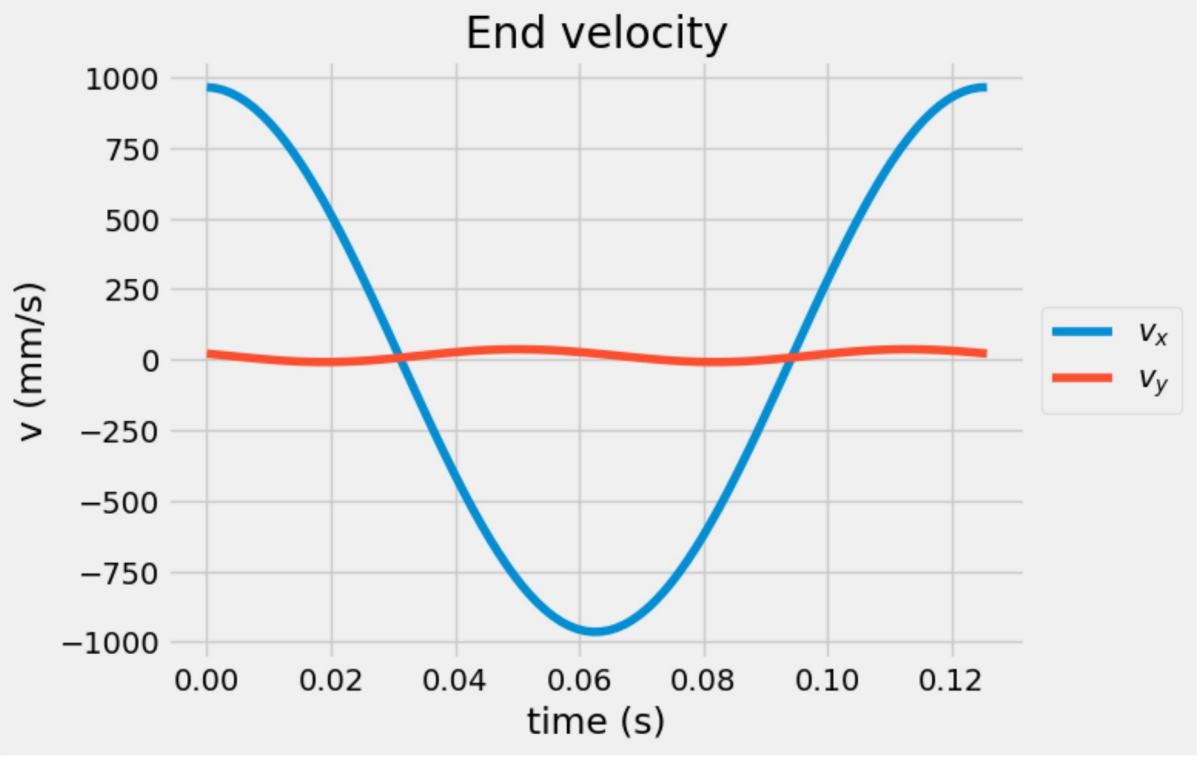
In [ ]:
```
rcc=np.array([x+L1*sin(t1),-L1*cos(t1)]) # position of connection between links
rco=np.array([x+L1*sin(t1)+L2*sin(t2),-(L1*cos(t1)+L2*cos(t2))])    # create a row
                                                                    # x-component a
                                                                    # pendulum posi

vco=np.array([dx+L1*cos(t1)*dt1+L2*cos(t2)*dt2,
     (L1*sin(t1)*dt1+L2*sin(t2)*dt2)]) # create row
                                       # vectors of
                                       # the x- and
                                       # y-component
                                       # velocity of
                                       # point C
```

Plot results

Here, you plot the velocity components of the end of the lower link.

In [ ]:
```
plt.plot(t,vco[0,:],label=r'$v_x$')
plt.plot(t,vco[1,:],label=r'$v_y$')
plt.xlabel('time (s)')
plt.ylabel('v (mm/s)')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.title('End velocity')
```

Out[ ]:
```
Text(0.5, 1.0, 'End velocity')
```



Animate the motion

Here, you create an HTML animation for the 2-bar linkage

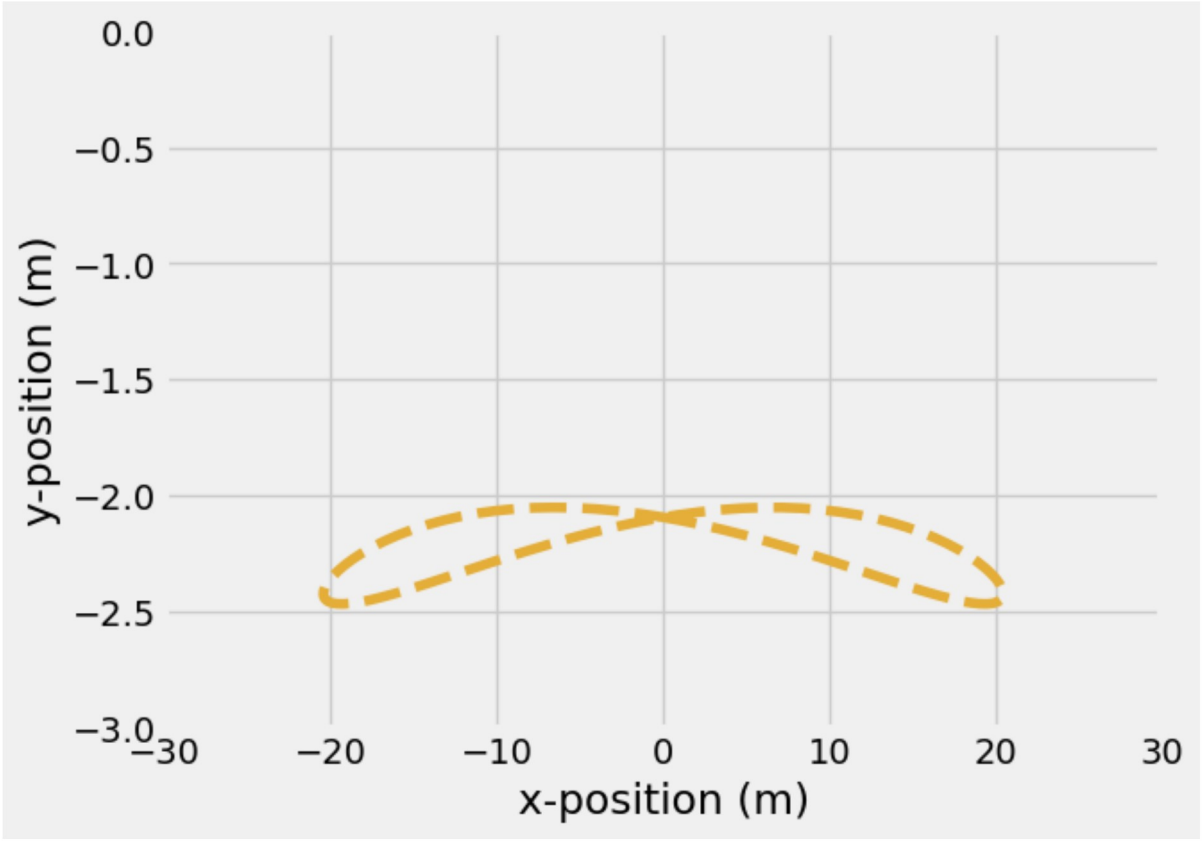
1. Create function that plots the links at timesteps
2. matplotlib's animation in HTML5

In [ ]:
```
from matplotlib import animation
from IPython.display import HTML
```

1. Create a figure to display the animation and add fixed background *the dashed line is added to show the path the end point takes*

In [ ]:
```
fig, ax = plt.subplots()

ax.set_xlim(( -30, 30))
ax.set_ylim((-3, 0))
ax.set_xlabel('x-position (m)')
ax.set_ylabel('y-position (m)')

line, = ax.plot([], [])
marker, = ax.plot([], [], 'o', markersize=10)
ax.plot(rco[0,:],rco[1,:],'--')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x2a786402310>]
```



1. Create an initializing (`init`) function that clears the previous line and marker

In [ ]:
```
def init():
    line.set_data([], [])
    marker.set_data([], [])
    return (line,marker,)
```

1. Create an animating (`animate`) function that updates the line

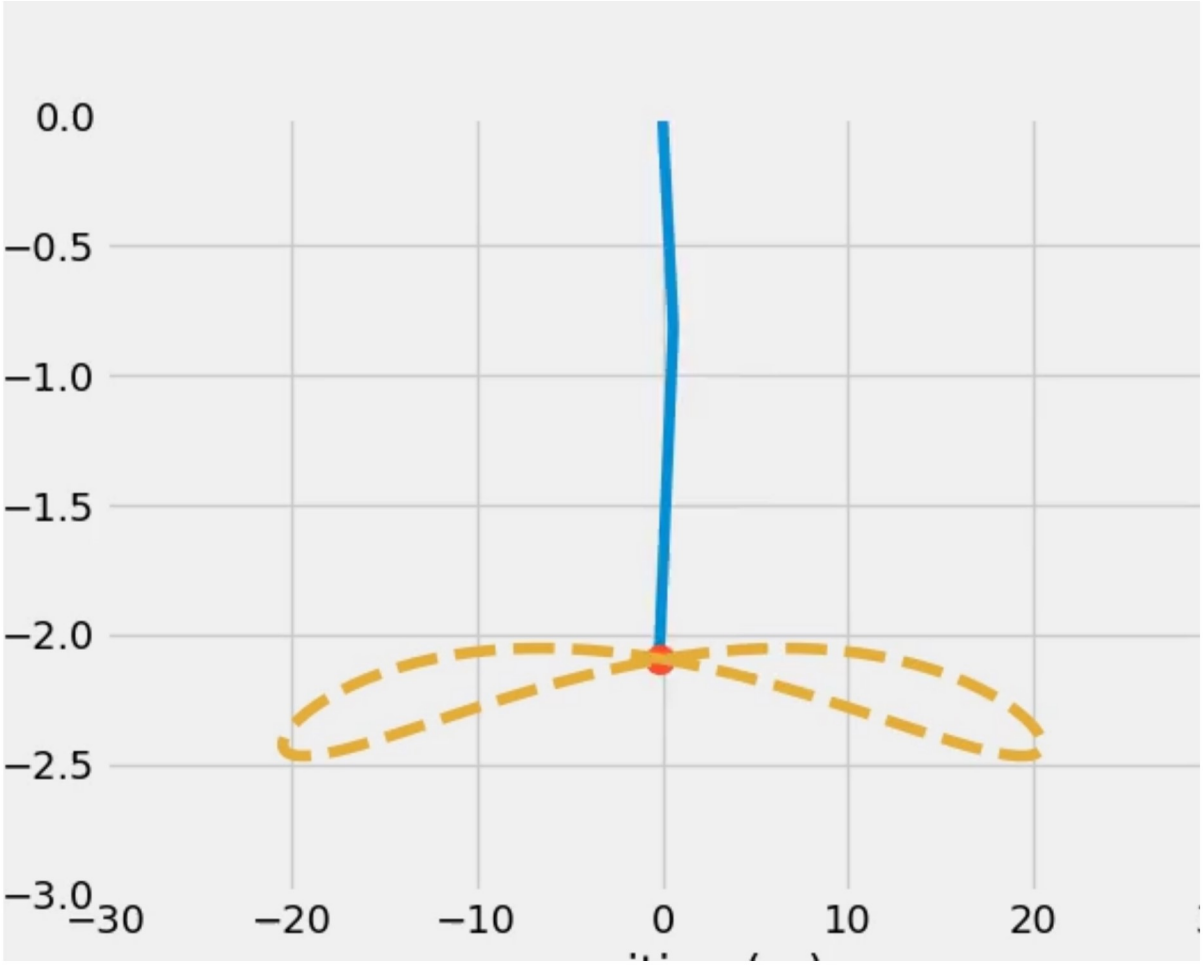
In [ ]:
```
def animate(i):
    '''function that updates the line and marker data
    arguments:
    ----------
    i: index of timestep
    outputs:
    --------
    line: the line object plotted in the above ax.plot(...)
    marker: the marker for the end of the 2-bar linkage plotted above with ax.plot(
    line.set_data([x[i],rcc[0,i],rco[0,i]],[0,rcc[1,i],rco[1,i]])
    marker.set_data([rco[0, i], rco[1,i]])
    return (line, marker, )
```

1. Create an animation (`anim`) variable using the `animation.FuncAnimation`

In [ ]:
```
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=range(0,len(t)), interval=100,
                               blit=True)
```

In [ ]:
```
HTML(anim.to_html5_video())
```

Out[ ]:



Problem 1

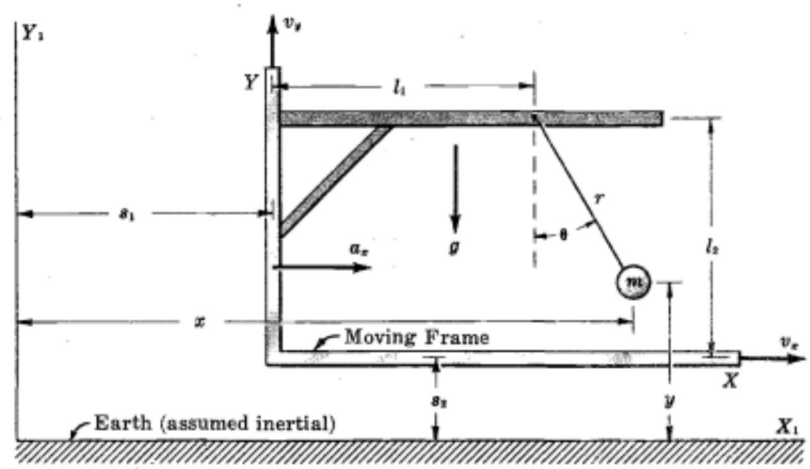


Fig. 1-5

The axis shown in Fig 1-5 has the following constants

$v_x = 1$ m/s.

$m = 1$ kg

$r = 500$ mm

$l_2 = 1000$ mm

$l_1 = 1200$ mm

$s_2 = 10$ mm

$v_y = 0$ m/s

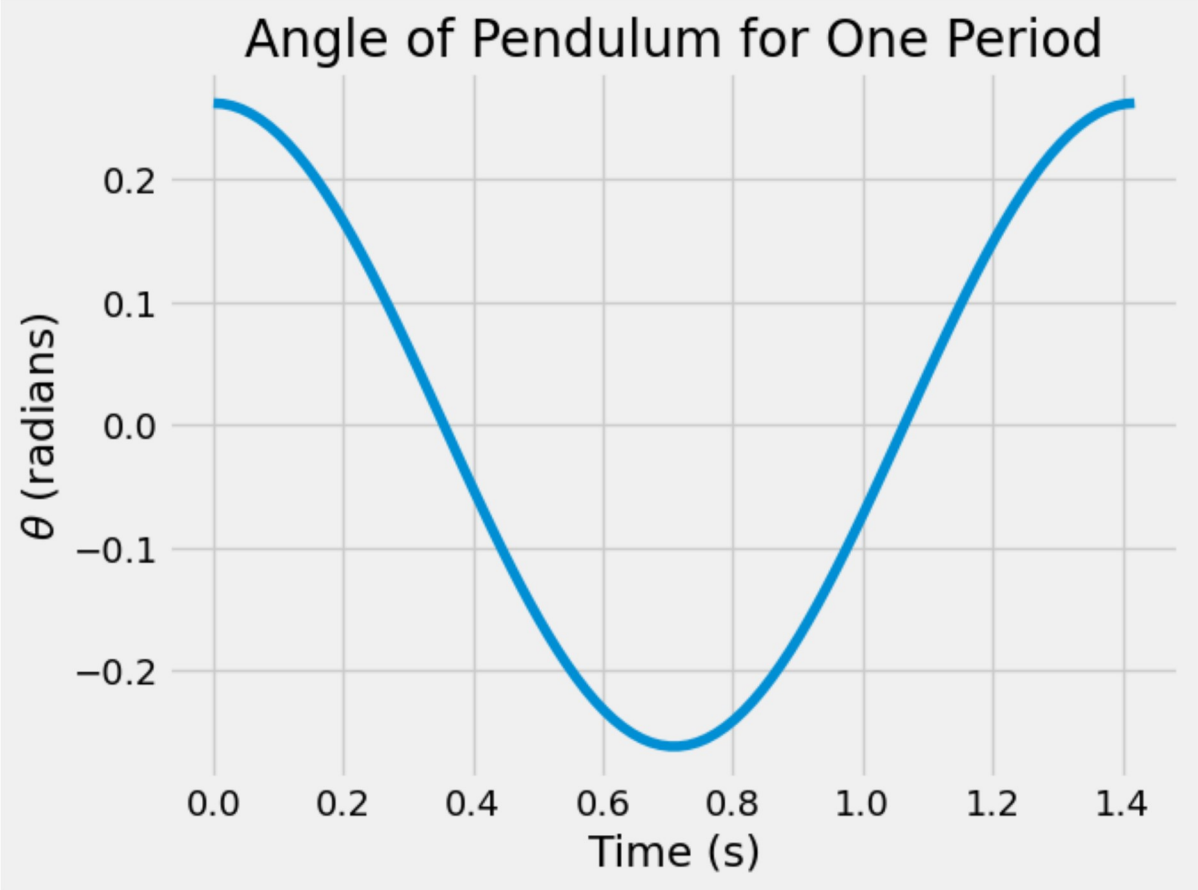
$a_x = 0$ m/s ^{2}

(a) Plot the angle of the pendulum for one full period of oscillation given $\theta(0) = \pi/12$ and $\dot{\theta}(0) = 0$

In [ ]:
```
v_x = 1 #m/2
m = 1 #kg
r = .5 #m
l_2 = 1 #m
l_1 = 1.2 #m
s_2 = 0.01 #m
v_y = 0 #m/s
a_x = 0 #m/s^2
g = 9.81 #m/s^2
```

In [ ]:
```
T = 2*pi*np.sqrt(r/g)
t = np.linspace(0,T,100)
theta_0 = pi/12
omega = np.sqrt(g/r)

theta = theta_0*np.cos(omega*t)

plt.plot(t,theta)
plt.title("Angle of Pendulum for One Period")
plt.xlabel("Time (s)")
plt.ylabel(r"$\theta$ (radians)")
```

Out[ ]:
```
Text(0, 0.5, '$\\theta$ (radians)')
```

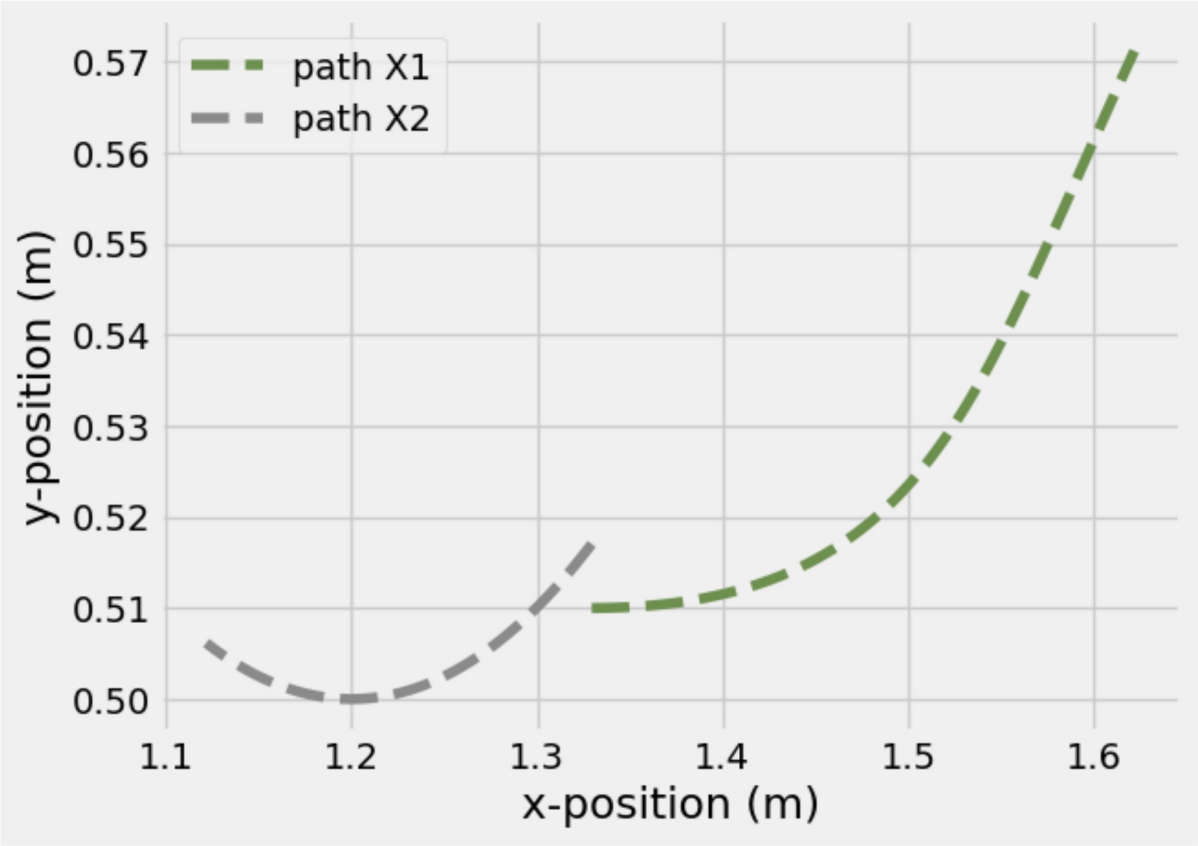


(b) make an animation of the position of the mass, m, in terms of X_1 and X_2 for $t = 0 - 0.5$ s

In [ ]:
```
# convert theta -> X1 and X2 positions
t_interest = np.linspace(0,.5,100)
theta_interest = theta_0*np.cos(omega*t_interest)

x_1 = r*np.sin(theta_interest) + v_x*t_interest + l_1
y_1 =  -r*np.cos(t_interest ) + s_2 + l_2
X1 = np.array([x_1,y_1]) #Inertial frame

x_2 = r*np.sin(theta_interest) + l_1
y_2 = -r*np.cos(theta_interest) + l_2
X2 = np.array([x_2, y_2]) #Moving frame
```

In [ ]:
```
fig2, ax2 = plt.subplots()

# ax2.set_xlim((0, 2))
# ax2.set_ylim((0, 1.2))
ax2.set_xlabel('x-position (m)')
ax2.set_ylabel('y-position (m)')

line, = ax2.plot([], [],'o-') # add marker at end points
line2, = ax2.plot([],[],'-o')
marker, = ax2.plot([], [], 'o', markersize=10)
ax2.plot(X1[0],X1[1],'--', label = 'path X1')
ax2.plot(X2[0],X2[1],'--', label = 'path X2')
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2a7864dc790>
```



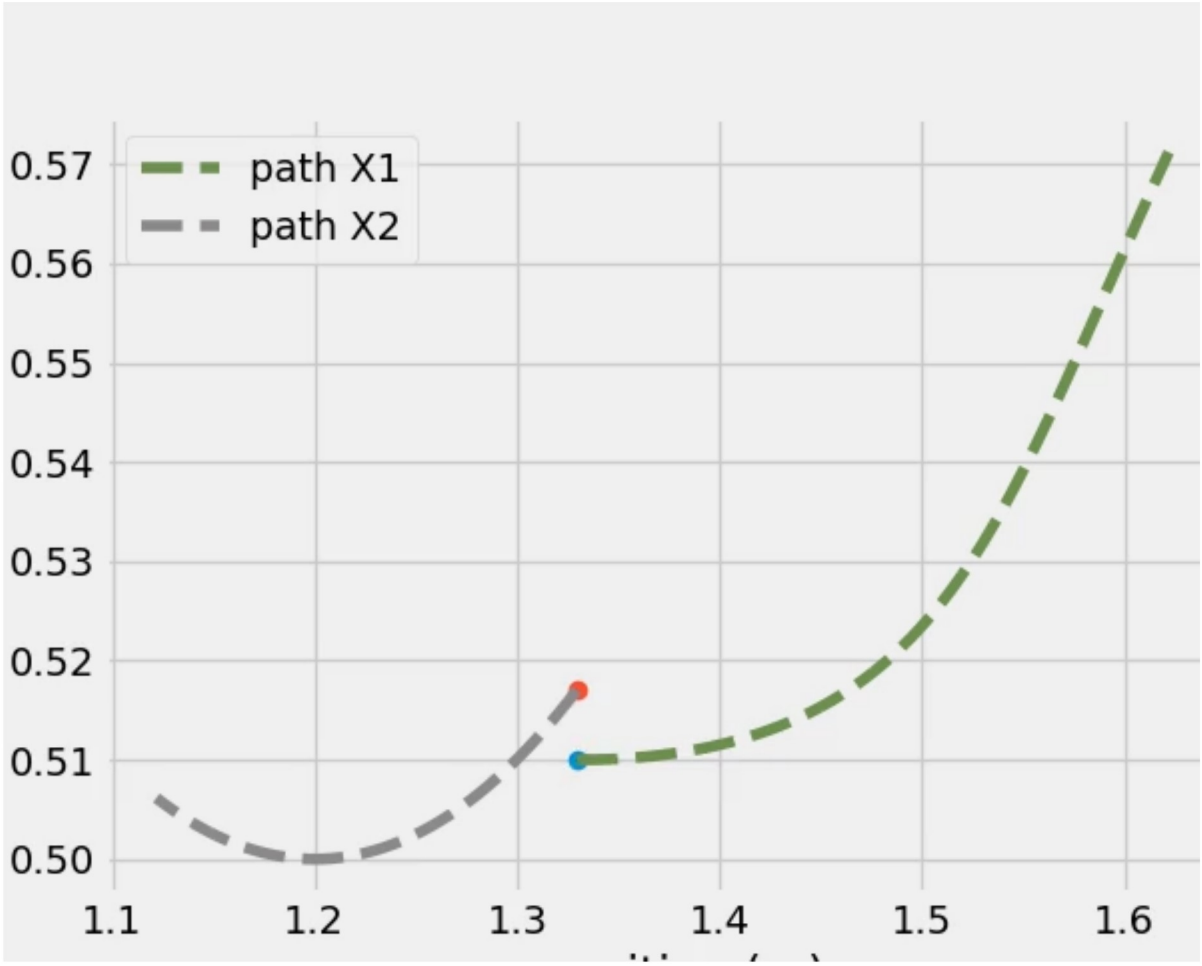
In [ ]:
```
def animate2(i):
    '''function that updates the line data for the pendulum
    arguments:
    ----------
    i: index of timestep
    outputs:
    --------
    line: the line object plotted in the above ax.plot(...)
    '''
    line.set_data([X1[0, i]], [X1[1, i]])
    line2.set_data([X2[0, i]], [X2[1, i]])
    return (line,line2,)
```

In [ ]:
```
def init2():
    line.set_data([], [])
    line2.set_data([], [])
    return (line,line2,)
```

In [ ]:
```
anim2 = animation.FuncAnimation(fig2, animate2, init_func=init2,
                                frames=range(0,len(t_interest)), interval=100,
                                blit=True)
```

In [ ]:
```
HTML(anim2.to_html5_video())
```

Out[ ]:



(c) Plot the angle θ for one period of oscillation for 2 conditions, label your two lines on one graph:

1. $a_x = 0$ m/s ^{2}
2. $a_x = 4$ m/s ^{2}

In [ ]:
```
#part c1

T_1 = 2*pi*np.sqrt(r/g)
t_1 = np.linspace(0,T_1,100)
theta_0 = pi/12
omega_1 = np.sqrt(g/r)

theta_1 = theta_0*np.cos(omega_1*t_1)

#part c2
a_x2 = 4 #m/s^2
g_eff = np.sqrt(g**2 + a_x2**2)
omega_2 = np.sqrt(g_eff/r)
T_2 = 2*pi*np.sqrt(r/g_eff)
t_2 = np.linspace(0,T_2,100)

theta_2 = theta_0*np.cos(omega_2*t_2)

plt.plot(t_1,theta_1,label = r'$a_x = 0 m/s^2$')
plt.plot(t_2,theta_2,label = r'$a_x = 4 m/s^2$')
plt.title("Angle of Pendulum for One Period")
plt.xlabel("Time (s)")
plt.ylabel(r"$\theta$ (radians)")
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x2a78694d280>
```