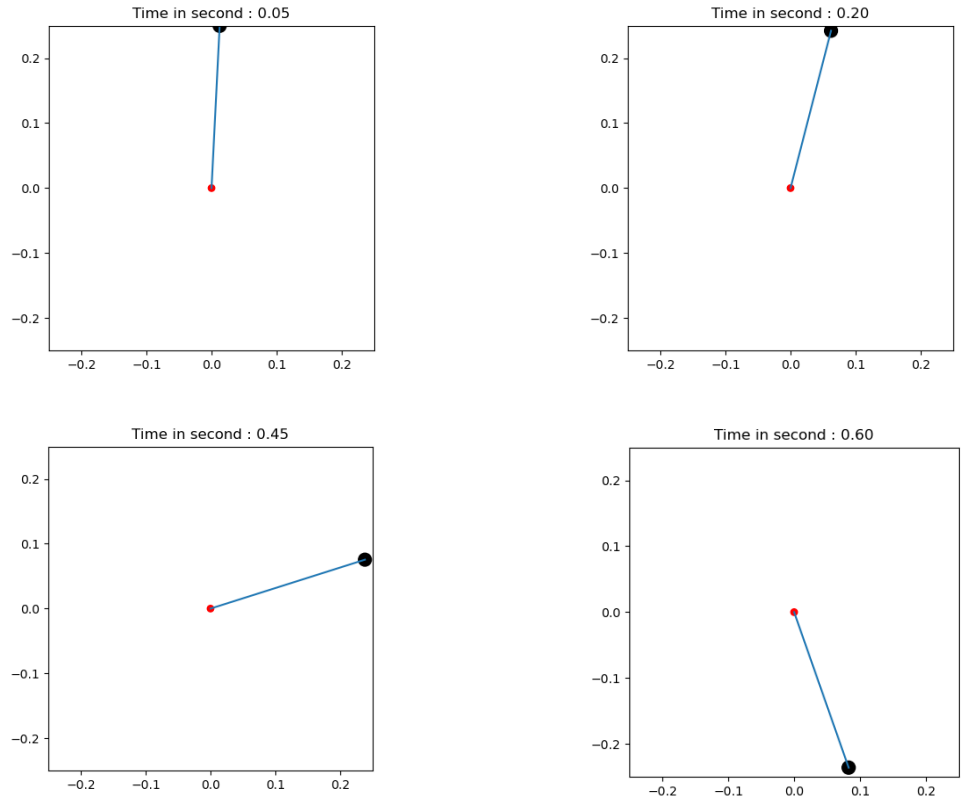
Nonlinear Pendulum Python Code Explanation

Pendulum တစ်ခု Nonlinear အတိုင်း ရွေ့သွားတဲ့ နေရာတွေကို လေ့လာဘို့အတွက် ပုံတွေ ထုတ်ပြီးတော့ simulation လုပ်ဘို့ Python Code တွေနဲ့ ရေးထားတဲ့ ပရိုဂရမ်ကို အသေးစိတ်ရှင်းပြထားတာပါ။ ရေးထားတဲ့ ပရိုဂရမ်ကို နားလည်အောင် ဖတ်ပြီးတော့ မိမိဖာသာ ပရိုဂရမ် ထဲက parameter တွေကို ပြောင်းပြီးတော့ ဆက်လက်လေ့လာလို့ ရပါတယ်။

နာရီချိန်သီးကို လက်တံနဲ့ ချိတ်ပြီတော့ အပေါ်ကနေ လွတ်ချလိုက်မယ်ဆိုရင် အချိန်ဘယ်လောက် ကြာရင် ဘယ်နေရာ ရောက်မလဲဆိုတာ ချမတွက်ဘဲ သိနိုင်ပါတယ်။ gravity acceleration ကို (G=9.806) ပါ ထည့်တွက်ထားတာကြောင့် G တန်ဖိုးကို 9.806 ထက် ပိုများ ပိုနည်းအောင် ပြောင်းပြီးတောလည်း လေ့လာလို့ ရပါတယ်။

အောက်က ပုံတွေက ပရိုဂရမ် ထုတ်ပေးထာတဲ့ ပုံတွေပါ။ အဲ့ပုံတွေ ရပြီးနောက် ဗွီဒီယိုဖိုင်၊ GIF ဖိုင် လုပ်လို့ ရပါတယ်။



ပရိုဂရမ် မလေ့လာချင် ရူပဗေဒနှင့် ဆိုင်တဲ့ အချက်အလက်တွေ အရင် ကြိုရှင်းပြပါရစေ။

```
# Parameters
G=9.806 # gravity acceleration
L=0.25 # pendulum length
T=1.4 # final time
V0=1 # initial velocity
theta0=math.pi # initial position
dt=0.05 # time step
```

Parameters

G=9.806 # gravity acceleration (ကမ္ဘာမြေဆွဲအားပါ)

L=0.25 # pendulum length (လက်တံ အရှည်ပါ)

T=1.4 # final time (စုစုပေါင်းကြာချိန်ပါ။ မိမိကြိုက်သလို ပြောင်းလို့ရပါတယ်။)

V0=1 # initial velocity (စဦးအလျှင်ပါ။)

theta0=math.pi # initial position (လက်တံရဲ့ထောင်ပါ။)

dt=0.05 # time step (တစ်ပုံနှင့် တစ်ပုံ အချိန်ဘယ်လောက်ကြာမလဲလို့ သတ်မှတ်ပေးဘို့ အချိန်ကွာခြားချက်ပါ။

ဘယ်ခါ လုပ်မလဲ။ တစ်နည်းအားဖြင့် ဘယ်နှစ်ပုံ ထုတ်မလဲ တွက်တာပါ။

```
# Total number of iterations
it = int(T/dt)
```

Vectors တွေတည်ဆောက် ဘို့ Initialize လုပ်ပါတယ်။

```
# Initialize vectors
theta=np.zeros(it)
x=np.zeros(it)
y=np.zeros(it)
t=np.zeros(it)
```

ချိန်သီးလေး ပထမဆုံးစရှိနေမဲ့ အခြေအနေတွေကို သတ်မှတ်ပေးလိုက်ပါတယ်။

```
# Initial conditions
theta[0]=theta0
theta[1]=V0*dt+theta[0]
x[0]=-L*math.sin(theta[0])
x[1]=-L*math.sin(theta[1])
y[0]=-L*math.cos(theta[0])
y[1]=-L*math.cos(theta[1])
t[0]=0
t[1]=dt
```

ချိန်သီးလေး 0.05 စက္ကန့်တိုင်းမှာ ရောက်နေမဲ့ (တည်ရှိနေမဲ့) နေရာကို တွက် loop ပတ်ပြီးတော့ တွက်ပါရင်း ပုံတွေကို plot လုပ်ပြီးတော့ သိမ်းဆည်းလိုက်ပါတယ်။ output အနေနဲ့ '.png' ဖိုင် ၂၅ ဖိုင် ထုတ်ပေးပါလိမ့်မယ်။

```
# Main time loop
for i in range(1,it-1):

    # Solves for theta
    theta[i+1]=((-dt**2)*G/L)*math.sin(theta[i])+2*theta[i]-theta[i-
1]
    t[i+1]=t[i]+dt
    x[i+1]=-L*math.sin(theta[i+1])
    y[i+1]=-L*math.cos(theta[i+1])

    # Figure plot
    fig, ax = plt.subplots()
    plt.axis([-L, L, -L, L])
    plt.plot([0,x[i]],[0,y[i]])
    #plt.title(i * dt)
    timing = i * dt
    plt.title("Time in second : %1.2f" % timing )
    circle1 = plt.Circle((0, 0), 0.005, color='r')
```

```
    circle2 = plt.Circle((x[i], y[i]), 0.01, color='k')
    ax.add_artist(circle1)
    ax.add_artist(circle2)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.savefig('figure-' + str(i) + '.png')
```

'.png' ဖိုင် ကို ဗီဒီယိုဖိုင်အဖြစ် ပြောင်းတာကိုတော့ ချန်ထားခဲ့ပါတယ်။

အပေါ်က ပရိုဂရမ်ကို မိမိဖာသာ ထဲက parameter တွေကို ပြောင်းပြီးတော့ ဆက်လက်လေ့လာလို့ ရပါတယ်။ အောက် Parameters တွေကို ပြောင်းလို့ ရပါတယ်။

```
# Parameters
G=9.806 # gravity acceleration
L=0.25 # pendulum length
T=1.4 # final time
V0=1 # initial velocity
theta0=math.pi # initial position
dt=0.05 # time step
```

အထူးသဖြင့် gravity acceleration , pendulum length, final time, initial velocity, time step တွေ က တန်ဖိုးတွေကို တစ်ခုချင်းစီ ပြောင်းပြီးတော့ ဘယ်လို အကျိုးသက်ရောက်မှုတွေ ရှိတယ်ဆိုတာ လေ့လာနိုင်ပါတယ်။

Ref: https://github.com/araujo88/nonlinear_pendulum

```
# Non-linear pendulum solver via explicit finite-difference scheme
# Author: Leonardo Antonio de Araujo
# E-mail: leonardo.aa88@gmail.com
# Date: 08/05/2020
```