

```

1 import tkinter
2 import math
3 from time import sleep
4 import random
5
6 class Window:
7     def __init__(self, xdim, ydim, scale):
8         self.root = tkinter.Tk()
9         self.canvas = tkinter.Canvas(self.root, bg="white", height=ydim*scale,
width=xdim*scale)
10         self.scale = scale
11         self.canvas.pack()
12
13     def drawLine(self, x, y, length, angleToVertical):
14         self.canvas.create_line(x*self.scale, y*self.scale,
(x+length*math.sin(angleToVertical))*self.scale,
(y+length*math.cos(angleToVertical))*self.scale, width=3)
15
16     def drawLineColour(self, x1, y1, x2, y2, colour):
17         self.canvas.create_line(x1*self.scale, y1*self.scale, x2*self.scale,
y2*self.scale, width=1, fill=colour)
18
19 class Pendulum:
20     def __init__(self, x, y, length, mass, theta, order):
21         self.x = x
22         self.y = y
23         self.v = 0
24         self.length = length
25         self.mass = mass
26         self.theta = theta
27         self.order = order
28
29     def draw(self, win):
30         win.drawLine(self.x, self.y, self.length, self.theta)
31
32     def getEnd(self):
33         return self.x+self.length*math.sin(self.theta),
self.y+self.length*math.cos(self.theta)
34
35     def simulate(self, timestep, val):
36         self.v += timestep*val
37         self.theta += timestep*self.v
38         self.theta = self.theta%(2*math.pi)
39
40 global sim, follow
41 sim = 1
42 follow = []
43
44 def update(win, pendulums, points, showOther=True):
45     global sim
46     win.canvas.delete("all")
47     for a in range(sim):
48         if len(points[a]) > 1:
49             for i in range(len(points[a])-1):
50                 if a in follow:
51                     win.drawLineColour(points[a][i][0], points[a][i][1], points[a]
[i+1][0], points[a][i+1][1], "blue")
52                 elif showOther:

```

```

53         win.drawLineColour(points[a][i][0],points[a][i][1],points[a]
[i+1][0],points[a][i+1][1],"red")
54         for p in pendulums[a]:
55             p.draw(win)
56
57         win.canvas.create_text(5*math.ceil(math.log(sim,10)),10,fill="black",text=str(sim)
)
58         win.root.update()
59 timestep = 0.05
60 initalAngle = math.pi/2 + 0.6*(random.random()-0.5)
61
62 win = Window(400,400,2)
63 allPairs = []
64 for i in range(sim):
65
66     allPairs.append([Pendulum(200,200,90,10,initalAngle,1),Pendulum(200+90*math.sin(in
italAngle),200+90*math.cos(initalAngle),90,10,2*initalAngle,1)])
67 ps = []
68 points = [[] for n in range(sim)]
69 update(win, allPairs, points)
70 while True:
71     for n in range(sim):
72         try:
73             ps = allPairs[n]
74         except IndexError:
75             break
76         try:
77             calc1 = (-ps[1].mass*math.cos(ps[0].theta-
ps[1].theta)*ps[0].length*math.pow(ps[0].v,2)*math.sin(ps[0].theta-ps[1].theta)+
78                 ps[1].mass*math.cos(ps[0].theta-
ps[1].theta)*9.81*math.sin(ps[1].theta)-
79                 ps[1].mass*ps[1].length*math.pow(ps[1].v,2)*math.sin(ps[0].theta-ps[1].theta)-
80                 (ps[0].mass+ps[1].mass)*9.81*math.sin(ps[0].theta))/(ps[0].length*
(ps[0].mass+ps[1].mass)-ps[1].mass*math.pow(math.cos(ps[0].theta-ps[1].theta),2))
81             ps[0].simulate(timestep, calc1)
82             ps[1].x, ps[1].y = ps[0].getEnd()
83             calc2 = (ps[0].mass+ps[1].mass)*
(ps[0].length*math.pow(ps[0].v,2)*math.sin(ps[0].theta-ps[1].theta)+
84                 ((math.pow(ps[1].v,2)*math.sin(ps[0].theta-ps[1].theta)*math.cos(ps[0].theta-
ps[1].theta)*ps[1].mass*ps[1].length)/(ps[0].mass+ps[1].mass))+
85                 math.cos(ps[0].theta-
ps[1].theta)*9.81*math.sin(ps[0].theta)-
86                 9.81*math.sin(ps[1].theta))/(ps[0].length*
(ps[0].mass+ps[1].mass)*math.pow(math.sin(ps[0].theta-ps[1].theta),2)))
87             ps[1].simulate(timestep, calc2)
88             points[n].append(list(ps[1].getEnd()))
89             if len(points[n]) > 500:
90                 points[n].pop(0)
91         except OverflowError:
92             sim -= 1
93             allPairs.pop(n)
94             points.pop(n)
95         update(win, allPairs, points)
96

```

```

1 import tkinter
2 import math
3 from time import sleep
4 import random
5
6 class Window:
7     def __init__(self, xdim, ydim, scale):
8         self.root = tkinter.Tk()
9         self.canvas = tkinter.Canvas(self.root, bg="white", height=ydim*scale,
width=xdim*scale)
10         self.scale = scale
11         self.canvas.pack()
12
13     def drawLine(self, x, y, length, angleToVertical):
14         self.canvas.create_line(x*self.scale, y*self.scale,
(x+length*math.sin(angleToVertical))*self.scale,
(y+length*math.cos(angleToVertical))*self.scale, width=3)
15
16     def drawLineColour(self, x1, y1, x2, y2, colour):
17         self.canvas.create_line(x1*self.scale, y1*self.scale, x2*self.scale,
y2*self.scale, width=1, fill=colour)
18
19 class Pendulum:
20     def __init__(self, x, y, length, mass, theta, order):
21         self.x = x
22         self.y = y
23         self.v = 0
24         self.length = length
25         self.mass = mass
26         self.theta = theta
27         self.order = order
28
29     def draw(self, win):
30         win.drawLine(self.x, self.y, self.length, self.theta)
31
32     def getEnd(self):
33         return self.x+self.length*math.sin(self.theta),
self.y+self.length*math.cos(self.theta)
34
35     def simulate(self, timestep, val):
36         self.v += timestep*val
37         self.theta += timestep*self.v
38         self.theta = self.theta%(2*math.pi)
39
40 global sim, follow
41 sim = int(math.pow(2, 11))
42 #follow = [0, 1, 3, 15, 31, 63, 127, 255]
43 follow = []
44
45 def update(win, pendulums, points, showBeams=False, showOther=True):
46     global sim
47     win.canvas.delete("all")
48     for a in range(sim):
49         if len(points[a]) > 1:
50             for i in range(len(points[a])-1):
51                 if a in follow:
52                     win.drawLineColour(points[a][i][0], points[a][i][1], points[a]
[i+1][0], points[a][i+1][1], "blue")
53                 elif showOther:

```

```

54         win.drawLineColour(points[a][i][0],points[a][i][1],points[a]
[i+1][0],points[a][i+1][1],"red")
55         if showBeams:
56             for p in pendulums[a]:
57                 p.draw(win)
58
59         win.canvas.create_text(5*math.ceil(math.log(sim,10)),10,fill="black",text=str(sim
))
60         win.root.update()
61 timestep = 0.1
62 initalAngle = math.pi*0.25
63 angleStep = 1.5*math.pi/sim
64
65 win = Window(400,400,2)
66 allPairs = []
67 for i in range(sim):
68     allPairs.append([Pendulum(200,200,90,10,initalAngle+angleStep*i,1),Pendulum(200+9
0*math.sin(initalAngle),200+90*math.cos(initalAngle),90,10,2*
(initalAngle+angleStep*i),1)])
69 ps = []
70 points = [[] for n in range(sim)]
71 update(win, allPairs, points)
72
73 while True:
74     for n in range(sim):
75         try:
76             ps = allPairs[n]
77         except IndexError:
78             break
79         try:
80             calc1 = (-ps[1].mass*math.cos(ps[0].theta-
ps[1].theta)*ps[0].length*math.pow(ps[0].v,2)*math.sin(ps[0].theta-ps[1].theta)+
81                 ps[1].mass*math.cos(ps[0].theta-
ps[1].theta)*9.81*math.sin(ps[1].theta)-
82             ps[1].mass*ps[1].length*math.pow(ps[1].v,2)*math.sin(ps[0].theta-ps[1].theta)-
83             (ps[0].mass+ps[1].mass)*9.81*math.sin(ps[0].theta))/(ps[0].length*
(ps[0].mass+ps[1].mass)-ps[1].mass*math.pow(math.cos(ps[0].theta-ps[1].theta),2))
84             ps[0].simulate(timestep, calc1)
85             ps[1].x, ps[1].y = ps[0].getEnd()
86             calc2 = (ps[0].mass+ps[1].mass)*
(ps[0].length*math.pow(ps[0].v,2)*math.sin(ps[0].theta-ps[1].theta)+
87             ((math.pow(ps[1].v,2)*math.sin(ps[0].theta-ps[1].theta)*math.cos(ps[0].theta-
ps[1].theta)*ps[1].mass*ps[1].length)/(ps[0].mass+ps[1].mass))+
88             math.cos(ps[0].theta-
ps[1].theta)*9.81*math.sin(ps[0].theta)-
89             9.81*math.sin(ps[1].theta))/(ps[0].length*
(ps[0].mass+ps[1].mass*math.pow(math.sin(ps[0].theta-ps[1].theta),2)))
90             ps[1].simulate(timestep, calc2)
91             points[n].append(list(ps[1].getEnd()))
92             if len(points[n]) > 2:
93                 points[n].pop(0)
94         except OverflowError:
95             sim -= 1
96             allPairs.pop(n)

```

```
97         points.pop(n)
98     update(win, allPairs, points)
99
100
```