from tkinter import \*

import tkinter.messagebox as tkMessageBox

import sys,copy

from numpy import \*

xx=[]

yy=[]

class bcanvas:

 def \_\_init\_\_(self,root,width,height):

 self.root=root

 self.width=width

 self.height=height

 try:

 self.canvas = Canvas(self.root, width=self.width, height=self.height, bg = 'white')

 self.canvas.pack()

 self.clear()

 except:

 print('An error has occured in init!')

 def clear(self):

 try:

 self.canvas.create\_rectangle(0,0,self.width,self.height,fill='white')

 except:

 print('An error has occured in clear!')

 def drawnames(self):

 self.canvas.create\_text(25,10,text='Hour',font=("Helvetica","11","bold"))

 self.canvas.create\_text(25,self.height-40,text='Count',font=("Helvetica","11","bold"))

 #self.canvas.create\_text(25,self.height-30,text='Total:',font=("Helvetica","8","bold"))

 #self.canvas.create\_text(25,self.height-10,text='max',font=("Helvetica","12","bold"))

# self.canvas.create\_text(30,0,text='mean X =',font=("Times","15"))

 def drawv(self,v,x,xx,gen,r):

 ''' draw the blue circles and connecting lines. write info'''

 n=len(v)

 N.append(sum(v))

 print(gen,sum(v),sum(N))

 dy=0.5\*(self.height-60-2\*r)//n

 ylist = []

 for i in range(n):

 y = 2\*r + (i+0.5)\*(self.height-60-2\*r)//n

 ylist.append(y)

 if v[i]==1:

 if gen > 0:

 if (i//2)\*2 == i:

 self.canvas.create\_line((self.width\*xx+r,y+dy),(self.width\*x,y),width=1)

 else:

 self.canvas.create\_line((self.width\*xx+r,y-dy),(self.width\*x,y),width=1)

 self.canvas.create\_oval(self.width\*x-r,y-r,self.width\*x+r,y+r,outline='SkyBlue4',fill='LightSkyBlue1')

 self.canvas.create\_oval(self.width\*x-r/4,y-r/4,self.width\*x+r/2,y+r/2,outline='SkyBlue4',fill='LightSkyBlue3')

 #self.canvas.create\_oval(self.width\*x-r,y-r,self.width\*x+r,y+r,outline='white', fill='blue')

 #self.canvas.create\_oval(self.width\*x-r/4,y-r/4,self.width\*x+r/2,y+r/2,outline='white', fill='green')

 self.canvas.create\_text(self.width\*x,10,text=str(gen),font=("Helvetica","15","bold"))

 self.canvas.create\_text(self.width\*x,self.height-40,text=str(int(sum(v))),font=("Helvetica","12"))

 #self.canvas.create\_text(self.width\*x,self.height-30,text=str(int(sum(N))),font=("Helvetica","12"))

 #self.canvas.create\_text(self.width\*x,self.height-10,text=str(int(max(N))),font=("Helvetica","12"))

 return ylist

 def draww(self,v,ylist,x,xx,gen,r):

 ''' draw the blue circles and connecting lines, rescaling positions. write info'''

 n = len(v)

 N.append(sum(v))

 print(gen,sum(v),sum(N))

 nn = int(sum(v))

 dy=0.5\*(self.height-60-2\*r)//nn

 newylist = []

 j = 0

 for i in range(n):

 if v[i]==1:

 y = 2\*r + (j+0.5)\*(self.height-60-2\*r)//nn

 newylist.append(y)

 #yy = ylist[i/2]

 yy = ylist[i//2]

 j += 1

 self.canvas.create\_line((self.width\*xx+r,yy),(self.width\*x,y),width=1)

 self.canvas.create\_oval(self.width\*x-r,y-r,self.width\*x+r,y+r,outline='SkyBlue4',fill='LightSkyBlue1')

 self.canvas.create\_oval(self.width\*x-r/4,y-r/4,self.width\*x+r/2,y+r/2,outline='SkyBlue4',fill='LightSkyBlue3')

 #self.canvas.create\_oval(self.width\*x-r,y-r,self.width\*x+r,y+r,outline='white', fill='blue')

 #self.canvas.create\_oval(self.width\*x-r/4,y-r/4,self.width\*x+r/2,y+r/2,outline='white', fill='green')

 else:

 newylist.append(0)

 self.canvas.create\_text(self.width\*x,10,text=str(gen),font=("Helvetica","15","bold"))

 self.canvas.create\_text(self.width\*x,self.height-40,text=str(int(sum(v))),font=("Helvetica","12"))

 #self.canvas.create\_text(self.width\*x,self.height-30,text=str(int(sum(N))),font=("Helvetica","12"))

 #self.canvas.create\_text(self.width\*x,self.height-10,text=str(int(max(N))),font=("Helvetica","12"))

 return newylist

tree=Tk()

tree.title('Branching process')

top=Frame(tree)

top.pack(side=TOP)

bframe=Frame(tree)

#bframe.pack(side=LEFT)

bframe.pack(side=LEFT, fill=BOTH)

onoff=False

def toggle(event):

 global onoff

 if onoff:

 onoff=False

 else:

 onoff=True

# defaul values of p0,p1,p2

d = 0.25

p1 = 0.35

p2 = 1-d-p1

# Runs the animation when button is pressed

def run():

 global onoff

 global N

 N = []

 ngenerations=ngen.get()

 d = pdie.get()

 p1 = pone.get()

 p2 = ptwo.get()

 error=StringVar()

 if d+p1+p2>1:

 tkMessageBox.showerror(title='Probabilities',message='d+p1+p2>1, p2 will be adjusted')

 p2 = 1-p1-d

 ptwo.set(p2)

 if d+p1+p2<1:

 tkMessageBox.showerror(title='Probabilities',message='d+p1+p2<1, p2 will be adjusted')

 p2 = 1-p1-d

 ptwo.set(p2)

 n=n1.get()#int(input)

 print('===============')

 meanx = p1+2\*p2

 print(d,p1,p2,meanx)

 print('===start run===')

 cg.clear()

 v=ones(n)

 i=0

 cg.drawnames()

 xcoord=(0.5)/(ngenerations+1)

 r = 12-n

 cg.drawv(v,xcoord,xcoord,i,r)

 drawwide=False

 while sum(v) != 0 and i < ngenerations:

 vv=zeros(2\*len(v))

 for j in range(len(v)):

 if v[j]==1:

 x=random.random()

 if d<x<=d+p1/2:

 vv[2\*j]=1

 if d+p1/2<x<=d+p1:

 vv[2\*j+1]=1

 if d+p1<x<=1:

 vv[2\*j]=1

 vv[2\*j+1]=1

 i+=1

 if sum(vv)==0:

 tkMessageBox.Message(icon='info',type='ok',message='Population went extinct',title='Results').show()

 else:

 xcoord=(i+0.5)/(ngenerations+1)

 dx=1.0/(ngenerations+1)

 if i > 4 and sum(vv) > 3:

 drawwide=True

 if drawwide:

 ylist = cg.draww(vv,ylist,xcoord,xcoord-dx,i,r)

 else:

 ylist = cg.drawv(vv,xcoord,xcoord-dx,i,r)

 v=vv

 tree.update()

 tree.after(1500)

but=Button(top, text='Start/Stop',command=run)

but.bind('<Button-1>',toggle)

but.pack(side=LEFT)

# Now the sliders

pdie=Scale(top, orient=HORIZONTAL, length=214, from\_=0., to=1.00, tickinterval=.2, resolution=0.01,label='Die')

pdie.set(d)

pdie.pack(side=LEFT)

pone=Scale(top, orient=HORIZONTAL, length=214, from\_=0., to=1.00, tickinterval=.2, resolution=0.01,label='Survive')

pone.set(p1)

pone.pack(side=LEFT)

ptwo=Scale(top, orient=HORIZONTAL, length=214, from\_=0., to=1.00, tickinterval=.2, resolution=0.01,label='Divide')

ptwo.set(1-d-p1)

ptwo.pack(side=LEFT)

n1=Scale(top, orient=HORIZONTAL, length=184, from\_=0, to=8, tickinterval=2, resolution=1,label='Starting number of cells')

n1.set(1)

n1.pack(side=BOTTOM)

ngen=Scale(top, orient=HORIZONTAL, length=184, from\_=0, to=12, tickinterval=2, resolution=1,label='Number of generations')

ngen.set(6)

ngen.pack(side=BOTTOM)

# original totalwidth=1200, totalheight=500

totalwidth=1200

totalheight = 500

cg=bcanvas(tree,totalwidth,totalheight)

tree.mainloop()