'''

using Tkinter to create a marquee/ticker

uses a display width of 20 characters

'''

import time

import tkinter as tk

root = tk.Tk()

# width --> width in chars, height --> lines of text

text\_width = 20

text = tk.Text(root, width=text\_width, height=1, bg='yellow')

text.pack()

# use a proportional font to handle spaces correctly

text.config(font=('courier', 48, 'bold'))

s1 = "We don't really care why the chicken crossed the road. "

s2 = "We just want to know if the chicken is on our side of the "

s3 = "road or not. The chicken is either for us or against us. "

s4 = "There is no middle ground here. (George W. Bush)"

# pad front and end of text with spaces

s5 = ' ' \* text\_width

# concatenate it all

s = s5 + s1 + s2 + s3 + s4 + s5

for k in range(len(s)):

# use string slicing to do the trick

ticker\_text = s[k:k+text\_width]

text.insert("1.1", ticker\_text)

root.update()

# delay by 0.22 seconds

time.sleep(0.22)

root.mainloop()

''' tk\_circles\_animate101.py

draw 2 circles via canvas.create\_oval(x0, y0, x1, y1, option, ...)

and animate them using canvas.move() and canvas.after()

'''

import tkinter as tk

CANVAS\_WIDTH = 400

CANVAS\_HEIGHT = 400

# circle starting center coordinates and radius

CIRCLE\_X = 50

CIRCLE\_Y = 50

CIRCLE\_RADIUS = 50

# fix animation rate, time in milliseconds

STEP\_TIME = 25

STEP\_X = 1

STEP\_Y = 1

#''' recursive function '''

def move\_circle():

canvas.move("orange\_circles", STEP\_X, STEP\_Y)

x0, y0, x1, y1 = canvas.bbox("orange\_circles")

if x1 > CANVAS\_WIDTH:

return

canvas.after(STEP\_TIME, move\_circle)

def circle(x, y, r):

# form a bounding square using center (x,y) and radius r

# upper left corner (ulc) and lower right corner (lrc) coordinates of square

ulc = x-r, y-r

lrc = x+r, y+r

# give the circle a tag name for reference

canvas.create\_oval(ulc, lrc, tag="orange\_circles", fill='orange')

root = tk.Tk()

root.title("Animated Circle")

# ulc position of rootwindow

root.geometry("+{}+{}".format(150, 80))

# create a canvas to draw on

canvas = tk.Canvas(root, width=CANVAS\_WIDTH, height=CANVAS\_HEIGHT, bg='lightblue')

canvas.pack()

circle(CIRCLE\_X, CIRCLE\_Y, CIRCLE\_RADIUS)

circle(CIRCLE\_X, CIRCLE\_Y-75, CIRCLE\_RADIUS)

move\_circle()

root.mainloop()

#Displays a red circle bouncing around the canvas window.

from tkinter import \*

window = Tk()

canvas = Canvas(window, width = 400, height = 300)

canvas.pack()

x0 = 10

y0 = 50

x1 = 60

y1 = 100

i = 0

deltax = 2

deltay = 3

which = canvas.create\_oval(x0,y0,x1,y1,fill="red", tag='redBall')

while True:

canvas.move('redBall', deltax, deltay)

canvas.after(20)

canvas.update()

if x1 >= 400:

deltax = -2

if x0 < 0:

deltax = 2

if y1 > 300:

deltay = -3

if y0 < 0:

deltay = 3

x0 += deltax

x1 += deltax

y0 += deltay

y1 += deltay

window.mainloop()

from tkinter import \*

import random

import time

tk = Tk()

tk.title("Game")

tk.resizable(0, 0)

tk.wm\_attributes("-topmost", 1)

canvas = Canvas(tk, width=500, height=400, bd=0, highlightthickness=0)

canvas.pack()

tk.update()

class Ball:

def \_\_init\_\_(self, canvas, paddle, color):

self.canvas = canvas

self.paddle = paddle

self.id = canvas.create\_oval(10, 10, 25, 25, fill=color)

self.canvas.move(self.id, 245, 100)

starts = [-3, -2, -1, 1, 2, 3]

random.shuffle(starts)

self.x = starts[0]

self.y = -3

self.canvas\_height = self.canvas.winfo\_height()

self.canvas\_width = self.canvas.winfo\_width()

self.hit\_bottom = False

def draw(self):

self.canvas.move(self.id, self.x, self.y)

pos = self.canvas.coords(self.id)

if pos[1] <= 0:

self.y = 3

if self.hit\_paddle(pos) == True:

self.y = -3

if pos[3] >= self.canvas\_height:

self.hit\_bottom = True

if pos[0] <= 0:

self.x = 3

if pos[2] >= self.canvas\_width:

self.x = -3

def hit\_paddle(self, pos):

paddle\_pos = self.canvas.coords(self.paddle.id)

if pos[2] >= paddle\_pos[0] and pos[0] <= paddle\_pos[2]:

if pos[3] >= paddle\_pos[1] and pos[3] <= paddle\_pos[3]:

return True

return False

class Paddle:

def \_\_init\_\_(self, canvas, color):

self.canvas = canvas

self.id = canvas.create\_rectangle(0, 0, 100, 10, fill=color)

self.canvas.move(self.id, 200, 300)

self.x = 0

self.canvas\_width = self.canvas.winfo\_width()

self.canvas.bind\_all('<KeyPress-Left>', self.turn\_left)

self.canvas.bind\_all('<KeyPress-Right>', self.turn\_right)

def turn\_left(self, evt):

self.x = -2

def turn\_right(self, evt):

self.x = 2

def draw(self):

self.canvas.move(self.id, self.x, 0)

pos = self.canvas.coords(self.id)

if pos[0] <= 0:

self.x = 0

elif pos[2] >= self.canvas\_width:

self.x = 0

paddle = Paddle(canvas, 'blue')

ball = Ball(canvas, paddle, 'red')

while 1:

if ball.hit\_bottom == False:

ball.draw()

paddle.draw()

tk.update\_idletasks()

tk.update()

time.sleep(0.01)