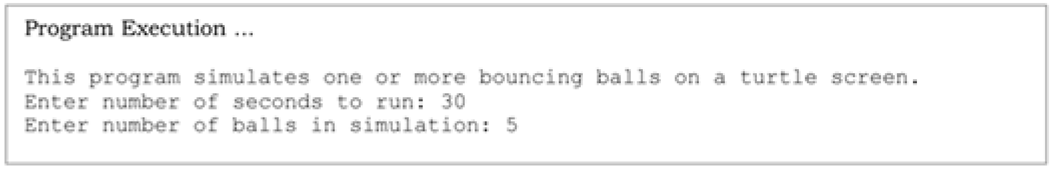
Bouncing Balls Program

The following is a program that displays one or more bouncing balls within a turtle screen. This program utilizes the following programming features.

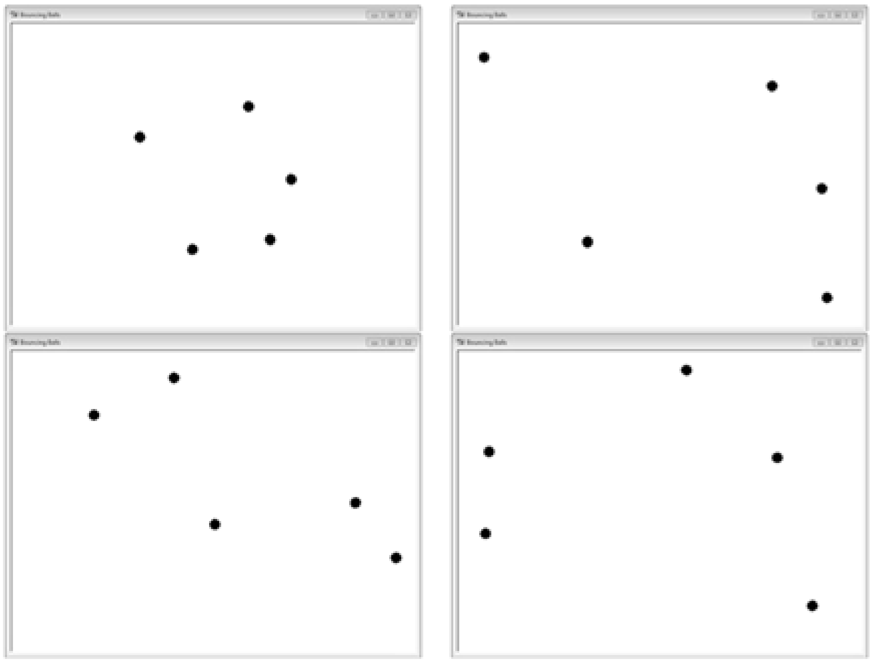
- turtle module
- time module
- random module

Example execution of the program is givenbelow:



Program Execution ...

```
This program simulates one or more bouncing balls on a turtle screen.
Enter number of seconds to run: 30
Enter number of balls in simulation: 5
```



Task: In IDLE, open a new project and save at as BouncingBalls_yourLastName. Copy the code from the sample on the next pages. Test and revise the program, as needed.

```python
# Bouncing Balls Simulation Program

import turtle
import random
import time

def atLeftEdge(ball, screen_width):
    if ball.xcor() < -screen_width / 2:
        return True
    else:
        return False

def atRightEdge(ball, screen_width):
    if ball.xcor() > screen_width / 2:
        return True
    else:
        return False

def atTopEdge(ball, screen_height):
    if ball.ycor() > screen_height / 2:
        return True
    else:
        return False

def atBottomEdge(ball, screen_height):
    if ball.ycor() < -screen_height / 2:
        return True
    else:
        return False

def bounceBall(ball, new_direction):
    if new_direction == 'left' or new_direction == 'right':
        new_heading = 180 - ball.heading()
    elif new_direction == 'down' or new_direction == 'up':
        new_heading =  360 - ball.heading()

    return new_heading

def createBalls(num_balls):
    balls = []
    for k in range(0, num_balls):
        new_ball = turtle.Turtle()
        new_ball.shape('circle')
        new_ball.fillcolor('black')
        new_ball.speed(0)
        new_ball.penup()
        new_ball.setheading(random.randint(1, 359))
        balls.append(new_ball)

    return balls

```

```
52 # ---- main
53 # program greeting
54 print('This program simulates bouncing balls in a turtle screen')
55 print('for a specified number of seconds')
56
57 # init screen size
58 screen_width = 800
59 screen_height = 600
60 turtle.setup(screen_width, screen_height)
61
62 # create turtle window
63 window = turtle.Screen()
64 window.title('Bouncing Balls')
65
66 # prompt user for execution time and number of balls
67 num_seconds = int(input('Enter number of seconds to run: '))
68 num_balls = int(input('Enter number of balls in simulation: '))
69
70 # create balls
71 balls = createBalls(num_balls)
72
73 # set start time
74 start_time = time.time()
75
76 # begin simulation
77 terminate = False
78
79 while not terminate:
80     for k in range(0, len(balls)):
81         balls[k].forward(15)
82
83         if atLeftEdge(balls[k], screen_width):
84             balls[k].setheading(bounceBall(balls[k], 'right'))
85         elif atRightEdge(balls[k], screen_width):
86             balls[k].setheading(bounceBall(balls[k], 'left'))
87         elif atTopEdge(balls[k], screen_height):
88             balls[k].setheading(bounceBall(balls[k], 'down'))
89         elif atBottomEdge(balls[k], screen_height):
90             balls[k].setheading(bounceBall(balls[k], 'up'))
91
92         if time.time() - start_time > num_seconds:
93             terminate = True
94
95 # exit on close window
96 turtle.exitonclick()
```

Notes:

On lines 63–64, the turtle screen is created and its reference value assigned to variable `window`. The title of the window is assigned through a call to the `title` method. Following that, the user is prompted to enter the number of seconds for the simulation, as well as the number of simultaneously bouncing balls.

Function `createBalls` is called (on line 71) to create and return a list of turtle objects with a ball shape. The function definition (lines 39–50) initializes an empty list named `balls` and creates the requested number of balls one-by-one, each appended to the list, by use of the for loop at line 41. Each ball is created with shape `'circle'`, fill color of `'black'`, speed of 0 (fastest speed), and with pen attribute `'up'`. In addition, the initial heading of each turtle is set to a random angle between 1 and 359 (line 47).

Back in the main program section at line 74, the current time (in seconds) is obtained from a call to method `time` of the `time` module: `time.time()`. The current time value is stored in variable `start_time`. (The current time is the number of seconds since the "epoch," which is January 1, 1970. This will be discussed further in the Horse Racing program that follows.) The while loop beginning on line 79 begins the simulation. The loop iterates as long as Boolean vari- able `terminate` is `False` (initialized to `False` on line 77). The for loop at line 80 moves each of the specified number of balls a small distance until reaching one of the four edges of the window (left, right, top, or bottom edge). Boolean functions `atLeftEdge`, `atRightEdge`, `atTopEdge`, and `atBottomEdge` are used to determine when a ball is at an edge (defined in lines 7–29). Func- tion `bounceBall` is called to bounce the ball in the opposite direction it is heading, and returns the new heading of the ball, passed as the argument to that ball's `setheading` method. Finally, on line 92 a check is made to determine whether the user-requested simulation time has been exceeded. If so, Boolean variable `terminate` is set to `True`, and the program terminates. Because of the call to `exitonclick()` on line 96, the program will properly shut down when the close button of the turtle window is clicked.