Pygame lesson 5

As games become more complicated, with various characters each having a range of properties, it would be frustrating to keep track of all the properties in variables. That is where object orientation comes in.

This tutorial introduces you to OOP and explains why it is so useful. Do yourself a big favour – read it and watch the videos! You don't need to do any programming until the end of the tutorial. It will give you a nice background on properties, methods and the general idea behind OOP.
http://programarcadegames.com/index.php?chapter=introduction_to_classes&lang=en

Now for the ball example
Use the 'basic template' at the end of this tutorial, or from:
http://programarcadegames.com/python_examples/f.php?file=pygame_base_template.py

To create a 'ball' class, use this class definition near the top of your code:

```
class Ball():
    def __init__(self):
        # --- Class Attributes ---
        # Ball position
        self.x = 0
        self.y = 0

        # Ball's vector
        self.change_x = 0
        self.change_y = 0

        # Ball size
        self.size = 10

        # Ball color
        self.color = [255,255,255]

    # --- Class Methods ---
    def move(self):
        self.x += self.change_x
        self.y += self.change_y

    def draw(self, screen):
        pygame.draw.circle(screen, self.color, [self.x, self.y],
self.size )
```

This code should also go above the main loop – it creates a ball for you:

```
theBall = Ball()
```

```
theBall.x = 100
theBall.y = 100
theBall.change_x = 2
theBall.change_y = 1
theBall.color = [255,0,0]
```

Then this code goes inside the loop to draw the ball:

```
theBall.move()
theBall.draw(screen)
```

Groups of objects

If we wanted our program to create a several balls, and each tie through the loop move them and check if they hit the wall, it would be useful to put them in a group of some sort so we can iterate through each item in the group instead of having to write multiple versions of the same code. Obviously, right. Here is an example I made (based upon: http://programarcadegames.com/python_examples/f.php?file=bouncing_balls.py) which adds each ball to a ball list when the space bar is hit. I made each ball a different colour.

```
# http://programarcadegames.com/

import pygame
import random

# Define some colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)

SCREEN_WIDTH = 700
SCREEN_HEIGHT = 500
BALL_SIZE = 25

class Ball:
    """Class to keep track of a ball's location and vector."""
    def __init__(self):
        self.x = 0
        self.y = 0
        self.change_x = 0
        self.change_y = 0
        self.size=0
        self.color=(0,0,0)

def make_ball():
    """
    Function to make a new, random ball.
    """
    ball = Ball()
    # ball size and color
    ball.size=random.randrange(2,50)
    ball.color=(random.randrange(0,255), random.randrange(0,255),
random.randrange(0,255))

    # Starting position of the ball.
    # Take into account the ball size so we don't spawn on the edge.

    ball.x = random.randrange(BALL_SIZE, SCREEN_WIDTH - ball.size)
    ball.y = random.randrange(BALL_SIZE, SCREEN_HEIGHT - ball.size)

    # Speed and direction of rectangle
    ball.change_x = random.randrange(-2, 3)
    ball.change_y = random.randrange(-2, 3)

    return ball

def main():
    """
```

```
This is our main program.
"""
pygame.init()

# Set the height and width of the screen
size = [SCREEN_WIDTH, SCREEN_HEIGHT]
screen = pygame.display.set_mode(size)

pygame.display.set_caption("Bouncing Balls")

# Loop until the user clicks the close button.
done = False

# Used to manage how fast the screen updates
clock = pygame.time.Clock()

ball_list = []

ball = make_ball()
ball_list.append(ball)

# -------- Main Program Loop -----------
while not done:
    # --- Event Processing
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
        elif event.type == pygame.KEYDOWN:
            # Space bar! Spawn a new ball.
            if event.key == pygame.K_SPACE:
                ball = make_ball()
                ball_list.append(ball)
    # --- Logic
    for ball in ball_list:
        # Move the ball's center
        ball.x += ball.change_x
        ball.y += ball.change_y

        # Bounce the ball if needed
        if ball.y > SCREEN_HEIGHT - BALL_SIZE or ball.y < BALL_SIZE:
            ball.change_y *= -1
        if ball.x > SCREEN_WIDTH - BALL_SIZE or ball.x < BALL_SIZE:
            ball.change_x *= -1

    # --- Drawing
    # Set the screen background
    screen.fill(BLACK)

    # Draw the balls
    for ball in ball_list:
        pygame.draw.circle(screen, ball.color, [ball.x, ball.y], ball.size)

    # --- Wrap-up
    # Limit to 60 frames per second
    clock.tick(60)

    # Go ahead and update the screen with what we've drawn.
    pygame.display.flip()
```

```
    # Close everything down
    pygame.quit()

if __name__ == "__main__":
    main()
```

Challenge

You will notice that there are some dud balls that don’t move. See if you can figure out how to prevent that happening.

Basic template

```
#  Pygame base template for opening a window
# http://programarcadegames.com/

import pygame

# Define some colors
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
GREEN = (0, 255, 0)
RED = (255, 0, 0)

pygame.init()

# Set the width and height of the screen [width, height]
size = (700, 500)
screen = pygame.display.set_mode(size)

pygame.display.set_caption("My Game")

# Loop until the user clicks the close button.
done = False

# Used to manage how fast the screen updates
clock = pygame.time.Clock()

# -------- Main Program Loop -----------
while not done:
    # --- Main event loop
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True

    # --- Game logic should go here

    # --- Screen-clearing code goes here

    # Here, we clear the screen to white. Don't put other drawing commands
    # above this, or they will be erased with this command.

    # If you want a background image, replace this clear with blit'ing the
    # background image.
    screen.fill(WHITE)

    # --- Drawing code should go here

    # --- Go ahead and update the screen with what we've drawn.
    pygame.display.flip()

    # --- Limit to 60 frames per second
    clock.tick(60)

# Close the window and quit.
pygame.quit()
```