This is what we have done so far in the Ball animation

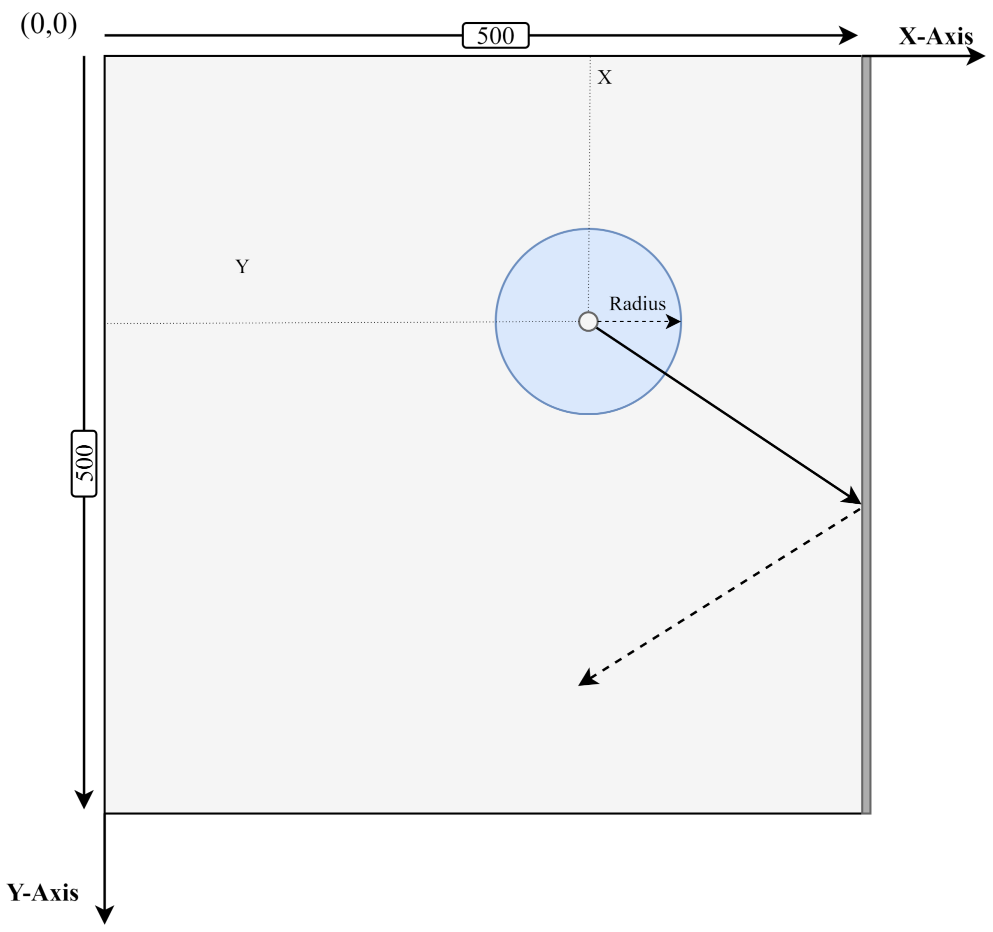
The balls move out of the screen as there is nothing stoping it from doing that.

In [9]:

```
import pygame
from random import randint

pygame.init()

screen = pygame.display.set_mode([500, 500])   # creates a screen with the said size

clock = pygame.time.Clock()


# creating the Base/Parent cirlce class
class Circle():
    def __init__(self):
        self.x = randint(0,500)
        self.y = randint(0,500)
        self.r = randint(10,50)
        self.color = (randint(0,255),randint(0,255),randint(0,255))
        self.x_speed = randint(-2,2)
        self.y_speed = randint(-2,2)

    def move(self):
        self.x = self.x+self.x_speed
        self.y = self.y+self.y_speed


# Inheriting from the parent circle class and creating the child class for FasCtCircle
class FastCircle(Circle):
    def  __init__(self):
        super().__init__()

    # redefinig the move() method
    def move(self):
        self.x = self.x + (self.x_speed*2)
        self.y = self.y + (self.y_speed*2)

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)


# Inheriting from the parent circle class and creating the child class for SlowCircle
class SlowCircle(Circle):
    def  __init__(self):
        super().__init__()

    #no need to define the move method as we can use the one from the parent class

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)

# List to store all the circles
cir = []

# creating FastCircles and adding them to the list
for i in range(5):
    cir.append(FastCircle())

# creating SlowCircles and adding them to the list
for i in range(5):
    cir.append(SlowCircle())


run = True
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    screen.fill((255,255,255))



    # creating individual cirlce objects using the draw method
    for i in range(10):
        cir[i].draw()

    # Moving the circle objects
    for i in range(10):
        cir[i].move()



    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

Lets try building on the ball animation and create a wallpaper out of it

1. The balls should bounce off the walls after hitting it
2. color of the ball should change after it hits the wall



Detecting the ball hitting the wall can be done using a simple formula Every wall on the screen will a separate formula for detection a ball collision with it

Below formula detects the collision of the ball with the right hand side wall
`x+radius > 500`

If the above condition becomes true it means the ball has collided with the right hand side wall and the ball should revers its direction. This reversal of direction has to happen only in the horizontal direction ie we resverse the x speed of the ball. As the ball can retains its speed in the vertical direction.

Just like the above condition for the right hand side we can come up with condition for all the walls on the screen.

The Below code shows how to detect if the ball has collided with any wall on the screen.

This has to be added as a method to the circle class as all the balls requires to detect if it has hit the wall and reverse the direction.

Task 1:

The balls should bounce off the walls after hitting it

In [ ]:
```python
def wall_collide(self):
    # detecting collision with the right wall
    if self.x+self.r > 500:
        # reversing the speed which also reverses the direction of the balls movement
        self.x_speed = self.x_speed * -1

    # detecting collision with the left wall
    if self.x-self.r < 0:
        # reversing the speed which also reverses the direction of the balls movement
        self.x_speed = self.x_speed * -1

    # detecting collision with the bottom wall
    if self.y+self.r > 500:
        # reversing the speed which also reverses the direction of the balls movement
        self.y_speed = self.y_speed * -1

    # detecting collision with the top wall
    if self.y-self.r < 0:
        # reversing the speed which also reverses the direction of the balls movement
        self.y_speed = self.y_speed * -1
```

Putting it all together

In [4]:
```python
import pygame
from random import randint

pygame.init()

screen = pygame.display.set_mode([500, 500])   # creates a screen with the said size

clock = pygame.time.Clock()


# creating the Base/Parent cirlce class
class Circle():
    def __init__(self):
        self.x = randint(0,500)
        self.y = randint(0,500)
        self.r = randint(10,50)
        self.color = (randint(0,255),randint(0,255),randint(0,255))
        self.x_speed = randint(-2,2)
        self.y_speed = randint(-2,2)

    def move(self):
        self.x = self.x+self.x_speed
        self.y = self.y+self.y_speed

    def wall_collide(self):
        if self.x+self.r > 500:
            self.x_speed = self.x_speed * -1
        if self.x-self.r < 0:
            self.x_speed = self.x_speed * -1
        if self.y+self.r > 500:
            self.y_speed = self.y_speed * -1
        if self.y-self.r < 0:
            self.y_speed = self.y_speed * -1
```

```python
# Inheriting from the parent circle class and creating the child class for FasCtCircle
class FastCircle(Circle):
    def  __init__(self):
        super().__init__()

    # redefinig the move() method
    def move(self):
        self.x = self.x + (self.x_speed*2)
        self.y = self.y + (self.y_speed*2)

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)


# Inheriting from the parent circle class and creating the child class for SlowCircle
class SlowCircle(Circle):
    def  __init__(self):
        super().__init__()

    #no need to define the move method as we can use the one from the parent class

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)

# List to store all the circles
cir = []

# creating FastCircles and adding them to the list
for i in range(50):
    cir.append(FastCircle())

# creating SlowCircles and adding them to the list
for i in range(50):
    cir.append(SlowCircle())


run = True
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    #screen.fill((255,255,255))



    # creating individual cirlce objects using the draw method
    for i in range(100):
        cir[i].draw()

    # Moving the circle objects
    for i in range(100):
        cir[i].move()

    # chekcing if the balls are colliding with the wall
    for i in range(100):
        cir[i].wall_collide()
    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

Task 2:

Try changing the color of the ball as soon as it hits the wall. Choose any random color

---

Solution:

In [5]:

```
import pygame
from random import randint

pygame.init()

screen = pygame.display.set_mode([500, 500])   # creates a screen with the said size

clock = pygame.time.Clock()


# creating the Base/Parent cirlce class
class Circle():
    def __init__(self):
        self.x = randint(0,500)
        self.y = randint(0,500)
        self.r = randint(10,50)
        self.color = (randint(0,255),randint(0,255),randint(0,255))
        self.x_speed = randint(-2,2)
        self.y_speed = randint(-2,2)

    def move(self):
        self.x = self.x+self.x_speed
        self.y = self.y+self.y_speed

    def wall_collide(self):
        if self.x+self.r > 500:
            self.x_speed = self.x_speed * -1
            self.color = (randint(0,255),randint(0,255),randint(0,255))
        if self.x-self.r < 0:
            self.x_speed = self.x_speed * -1
            self.color = (randint(0,255),randint(0,255),randint(0,255))
        if self.y+self.r > 500:
            self.y_speed = self.y_speed * -1
            self.color = (randint(0,255),randint(0,255),randint(0,255))
        if self.y-self.r < 0:
            self.y_speed = self.y_speed * -1
            self.color = (randint(0,255),randint(0,255),randint(0,255))
```

```python
# Inheriting from the parent circle class and creating the child class for FasCtCircle
class FastCircle(Circle):
    def  __init__(self):
        super().__init__()

    # redefinig the move() method
    def move(self):
        self.x = self.x + (self.x_speed*2)
        self.y = self.y + (self.y_speed*2)

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r,2)


# Inheriting from the parent circle class and creating the child class for SlowCircle
class SlowCircle(Circle):
    def  __init__(self):
        super().__init__()

    #no need to define the move method as we can use the one from the parent class

    # drawing circles with thin edges
    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)

# List to store all the circles
cir = []

# creating FastCircles and adding them to the list
for i in range(5):
    cir.append(FastCircle())

# creating SlowCircles and adding them to the list
for i in range(5):
    cir.append(SlowCircle())


run = True
while run:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            run = False

    #screen.fill((255,255,255))



    # creating individual cirlce objects using the draw method
    for i in range(10):
        cir[i].draw()

    # Moving the circle objects
    for i in range(10):
        cir[i].move()

    # chekcing if the balls are colliding with the wall
    for i in range(10):
        cir[i].wall_collide()
    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

Task 3 :

Lets try fixing a few issue with the wallpaper animation

1. Fix the issue where the balls spawns too close to the edge and gets stuck over there.(**Hint : Chnage the initial x and y coordinates of the ball**)
2. Fix the issue in which the ball remains stationary on the screen. (**Hint : try using random.choices()**)

REVISION

- Concept of : OOP, Class, object,Method,Attributes,Inheritance, polymorphism
- All programs in sequence
- Animation (effect)
- Wallpaper making

HOMEWORK

1. Try making the balls collide with each other

In [ ]:

```

```