Animation in PyGame:

- Animation is basically a series of images which is being shown at a particular frame rate.
- **Lets try making the ball move around on the screen.**
- To make the ball move, all we need to do is change the x and y coordinate of the ball and keep on doing this as the while loop is running.

In [1]:
```
#TASK 1:  moving Ball on screen
```

In [5]:
```
import pygame
import time

pygame.init()

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

clock = pygame.time.Clock()

# variables used to represent the center location of the circle
x = 250
y = 250

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

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

    pygame.draw.circle(screen, (0, 0, 255), (x, y), 75)

    pygame.display.flip()

    # changing the x and y coordinate of the circle
    x = x+1
    y = y+1

    clock.tick(30)


pygame.quit()
```

EXPLAINATION OF CODE

- The x and y coordinate of the circle keeps updating in the while loop itself.

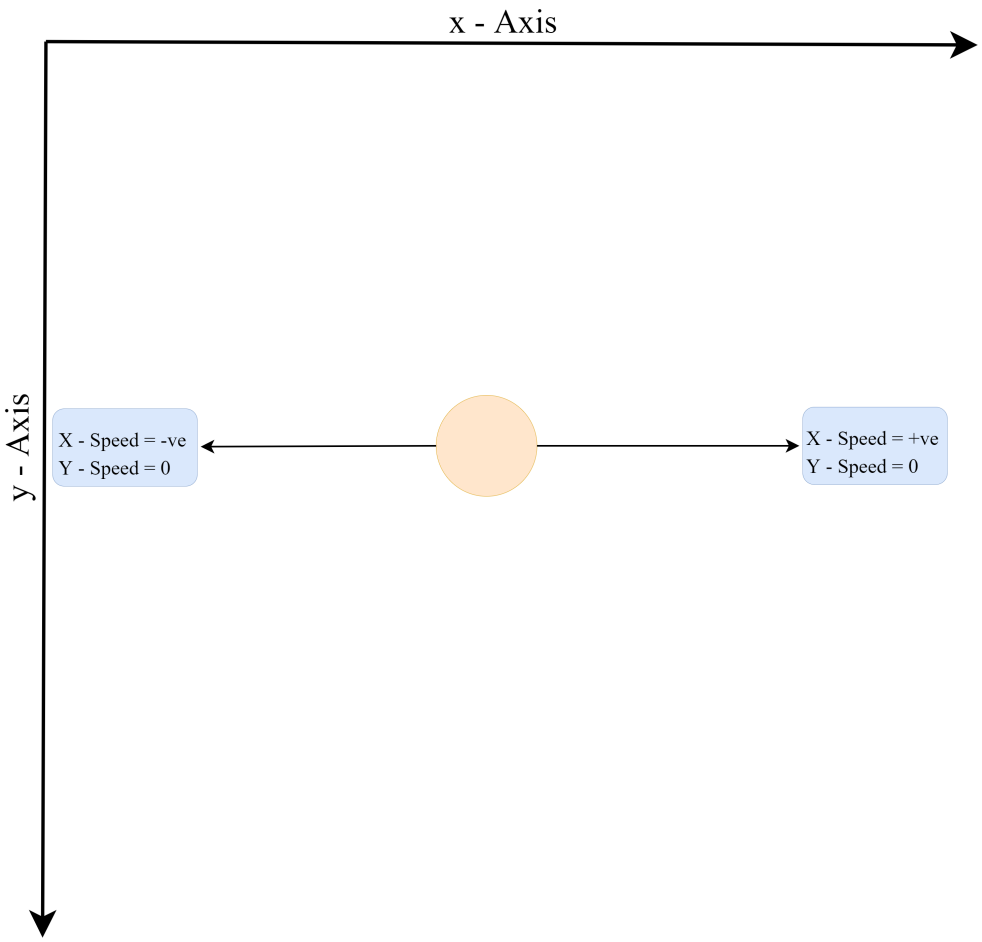
- This causes the circle to be redrawn in the new location evrey time the loop runs once.
- And since we have set the fps of the game loop by using `clock.tick()` the ball will move by 30 times every second.

You can also try running the code without the `screen.fill((255,255,255))`.
This will cause the ball to leave a trail behind. So in order to avoid the trail behind we fill the screen with the white color again and the redraw the ball. These steps are already being followed in the while loop.

```
x = x+1
y = y+1
```

The `+1` controls the speed at whicih the ball is moving ahead.

The direction of the ball is determined by wheather the ball's x and y coordinate is increasing or decreasing.



In [3]:

```
#task 2 : Add  4 more balls each with different, color, speed and moving in different d
#         Also make sure the balls have different sizesPractice problem

import pygame
import time
pygame.init()
screen = pygame.display.set_mode([500, 500])   # creates a screen with the said size
clock = pygame.time.Clock()
```

```
x1 = 250
y1 = 250

x2 = -100
y2 = 100

x3 = 50
y3 = -50

x4 = -150
y4 = -150


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

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

    pygame.draw.circle(screen, (0, 0, 255), (x1, y1), 75)

    pygame.draw.circle(screen, (0, 255, 255), (x2, y2), 25)
    pygame.draw.circle(screen, (10, 40, 255), (x3, y3), 60)
    pygame.draw.circle(screen, (255, 0, 255), (x4, y4), 10)

    pygame.display.flip()

    # changing the x and y coordinate of the circle
    x1 = x1+10
    y1 = y1+20

    x2 = x2+2
    y2 = y2-2

    x3 = x3-5
    y3 = y3+5

    x4 = x4+10
    y4 = y4+10

    clock.tick(30)


pygame.quit()
```

The OOP's based approach.

After solving the above pratice problem you might have noticed that how difficult it is to maintain all the variables associated with differnet balls and to keep track of them gets really difficult.

This is a good example where the OOP's based approach can come in handy. We can create circle objects using the circle class.

We can start by identifying the attributes of a cirlce so that we can define these inside the class itself.

1. x and y coordinate
2. radius
3. color

We also need the draw method to draw/show the cicle on the screen itself.

1. draw()

We are using random module to select all the prameters associated with the cirlce objetc randomly when the object is created.

By adding the attributes associated with an object into the class we have effectively acheived encapsulation

This solves one of the major issue that we faced in the practice problem which is to keep track of all the variables associated with individual circles and drawing the circles itself.

In [8]:

```
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 cirlce class
class Circle():
    def __init__(self):
        self.x = randint(-50,500)
        self.y = randint(0,500)
        self.r = randint(10,50)
        self.color = (randint(0,255),randint(0,255),randint(0,255)) # (randint(0,255),r

    def draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)


# creating circle objects using the Circle class
c1 = Circle()
c2 = Circle()
c3 = Circle()
c4 = Circle()
c5 = Circle()
c6 = Circle()
c7 = Circle()
c8 = Circle()
c9 = Circle()
c10 = Circle()

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

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

    # creating indivisual cirlce objects using the draw method
    c1.draw()
    c2.draw()
```

```
    c3.draw()
    c4.draw()
    c5.draw()
    c6.draw()
    c7.draw()
    c8.draw()
    c9.draw()
    c10.draw()
    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

The next step is to identify the methods associated with the object. Methods are functionalities that are performed by the object itself.

In case of the example that we are doing the methods that a cirlce object can have is to move around. So calling the move methods will make the ball move in a certial direction with a certain speed. We will have to define the x and y speed attributes which can be used by the move() method to move the ball around by manipulating the x and y coordinate of the ball itself.

Attributes to be defined

1. x speed
2. y speed

Methods to be defined

3. move()

In [6]:

```
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 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 draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)

    def move(self):
```

```
        self.x = self.x+self.x_speed
        self.y = self.y+self.y_speed


# creating circle objects using the Circle class
c1 = Circle()
c2 = Circle()
c3 = Circle()
c4 = Circle()
c5 = Circle()


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

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



    # creating indivisual cirlce objects using the draw method
    c1.draw()
    c2.draw()
    c3.draw()
    c4.draw()
    c5.draw()



    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

Running the above code won't make the ball's move around as we have not called the move method on individual circle objects

In [10]:

```
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 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 draw(self):
        pygame.draw.circle(screen, self.color, (self.x, self.y), self.r)

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


# creating circle objects using the Circle class
c1 = Circle()
c2 = Circle()
c3 = Circle()
c4 = Circle()
c5 = Circle()


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

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



    # creating indivisual cirlce objects using the draw method
    c1.draw()
    c2.draw()
    c3.draw()
    c4.draw()
    c5.draw()

    # Moving the circle objects
    c1.move()
    c2.move()
    c3.move()
    c4.move()
    c5.move()



    pygame.display.flip()

    clock.tick(30)


pygame.quit()
```

REVISION

- Animation using speed variation
- different object with size, color variation
- OOP approach to create differnt object using class
- Animation with differnt object using OOP

HOMEWORK

1. Animate rectacngles and circles together on the screen.

In [ ]: