My Invaders Game

Week 1

This week, we are going to build a simple game using object-oriented programming techniques with Python. Have a look at `My Invaders Project - Week01.py`

Import Pygame

First, we need to import the pygame module

```
import pygame
```

Create Player Class

We need to create a class to represent the player. We use the `__init__()` constructor to initialise all our variables and load the rocket.png image.

```
class Rocket():
    def __init__(self, screen):
        self.screen = screen
        self.image = pygame.image.load('rocket.png')
        self.rect = self.image.get_rect()
        self.screen_rect = screen.get_rect()
        self.rect.centerx = self.screen_rect.centerx
        self.rect.bottom = self.screen_rect.bottom
```

We also need a method that will draw the player's ship – the rocket – on the screen.

```
    def draw(self):
        self.screen.blit(self.image, self.rect)
```

Create Main Program

Now we need to create the main program. First we initialise pygame and create a screen.

```
pygame.init()
screen = pygame.display.set_mode((800, 600))
```

Create a rocket object from the class Rocket. We pass our "screen" we created above to the Rocket class.

```
rocket = Rocket(screen)
```

Load our background image

```
background = pygame.image.load("bg.jpg")
```

Create our clock so the game runs at a specific speed

```
clock = pygame.time.Clock()
```

Turn on keyboard repeat, so the rocket continues to move if you hold down the key

```
pygame.key.set_repeat(1, 25)
```

Initialise our running variable to 1 which means the game is running

```
running = 1
```

Create the Game Loop

Here we need to create a while loop to keep the game running. We'll run the program until the variable running = 0. We'll set the clock to 25 which means that for every second 25 frames should pass.

```
while running:
    clock.tick(25)
    screen.blit(background, (0, 0))
```

Next, we need an event handler that will check for key presses. We want to monitor the left and right arrow keys.

```
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = 0
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_RIGHT:
                rocket.rect.centerx += 10
            if event.key == pygame.K_LEFT:
                rocket.rect.centerx -= 10
```

Now, we can draw our rocket on the screen

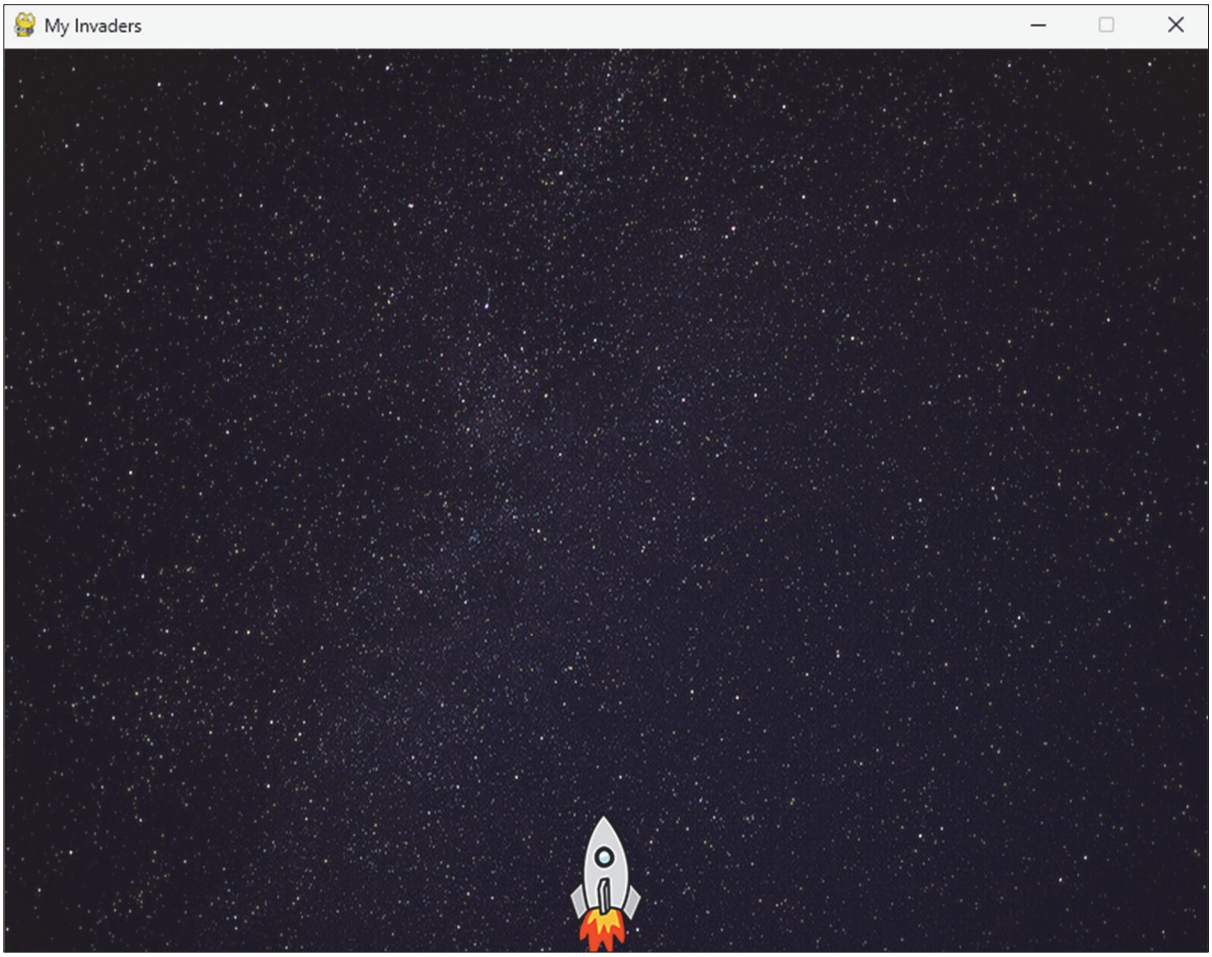
```
rocket.draw()
```

Then update the screen

```
pygame.display.update()
```

Run the Program

When you run the program, you'll be able to move your ship left and right.



Week 2

This week, we are going to add some UFOs to the project and make them fly around the screen. Have a look at `My Invaders Project - Week02.py`

We can do this by adding a new class for the UFO.

```
class UFO():
    def __init__(self, screen):
        super(UFO, self).__init__()
        self.screen = screen


        self.speed = [5, 5]
```

We can create some attributes for the x and y direction and current position on the screen

```
        #ufo direction
        self.ufo_X_dir = 0
        self.ufo_Y_dir = 0

        #ufo position
        self.ufo_X_pos = 0
        self.ufo_Y_pos = 0
```

We can also add an image of the UFO

```
        self.image = pygame.image.load('ufo.png')
        self.rect = self.image.get_rect()
```

We set the UFO to appear in a random position on the screen between 100 and 600 pixels for x and y

```
        self.rect.x = random.randint(100,600)
        self.rect.y = random.randint(100,600)
```

Add attributes to store the exact position on the screen.

```
        self.x = float(self.rect.x)
```

Now we can add a method to draw the UFO at a position

```
    def draw(self):
        self.screen.blit(self.image, self.rect)
```

Add another method to move the UFO by given x and y coords

```
    def move(self):
        self.rect.move_ip(self.speed) #ufo_rect.move_ip (x, y)
```

We need to make sure the UFO bounced off the 4 edges of the screen

Left edge

```
        if self.rect.left < 0:
            self.speed[0] = -self.speed[0]
            self.ufo_X_dir=self.ufo_X_dir-1
```

Right edge

```
        if self.rect.right > 800:
            self.speed[0] = -self.speed[0]
            self.ufo_X_dir=self.ufo_X_dir+1
```

Top edge

```
        if self.rect.top < 0:
            self.speed[1] = -self.speed[1]
            self.ufo_Y_dir=self.ufo_Y_dir-1
```

Bottom edge

```
        if self.rect.bottom > 600:
            self.speed[1] = -self.speed[1]
            self.ufo_Y_dir=self.ufo_Y_dir+1
```

Now set position of UFO center to the UFO rectangle

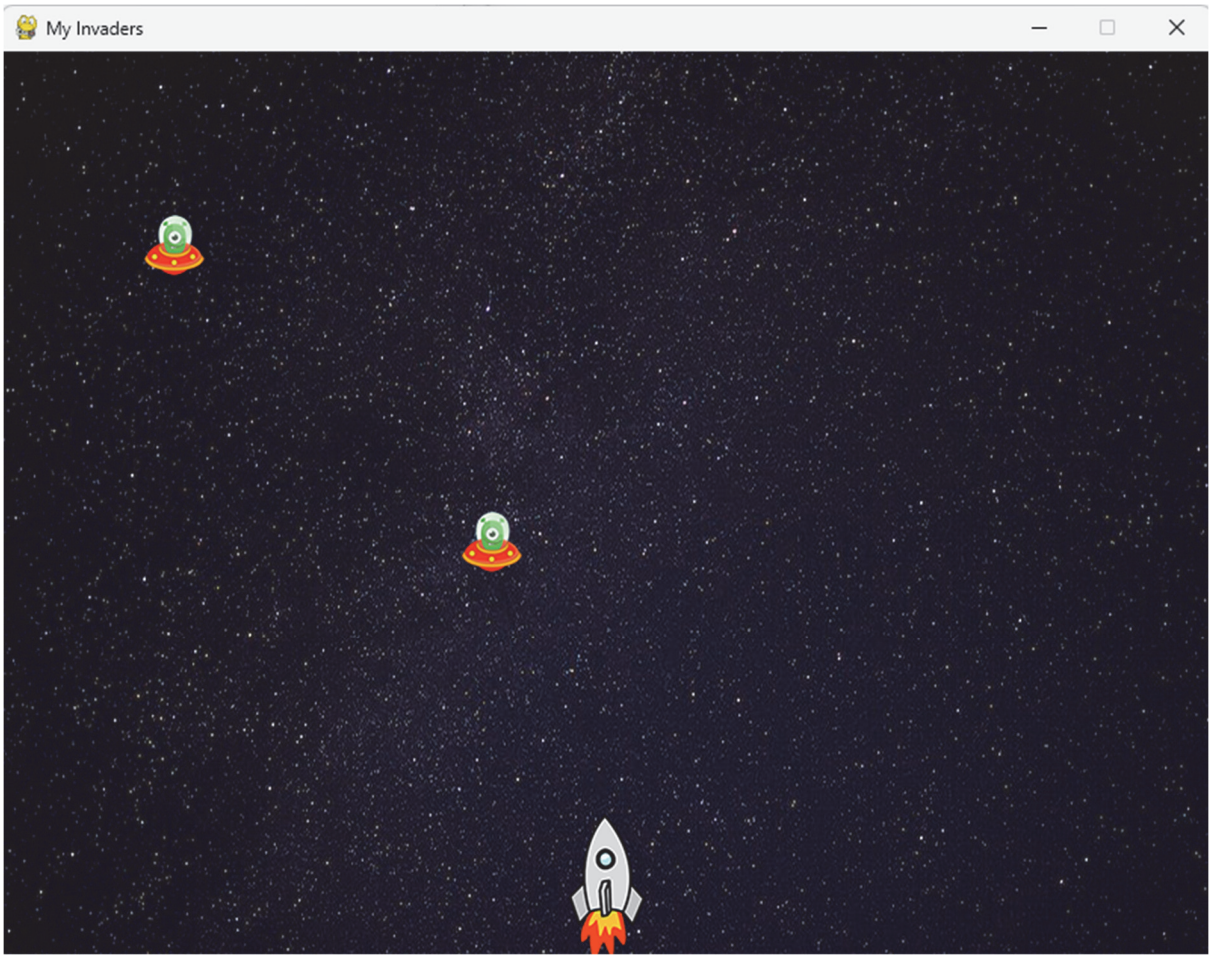
```
        self.ufo_X_pos = self.rect.centerx
        self.ufo_Y_pos = self.rect.centery
```

In our code, we can create some UFOs

```
ufo = UFO(screen)
ufo2 = UFO(screen)
```

Run the Program

When you run the program, you'll be able to move your ship left and right. The UFOs will float around the screen.



Week 3

This week, we are going to add some bullets to the rocket ship for the player to fire at the UFOs. Have a look at `My Invaders Project - Week03.py`

We can add another class to create the bullets

```
class Bullet():
    def __init__(self, screen, rocket):
        self.screen = screen
```

Load the image of the bullet and assign to a rectangle

```
        self.image = pygame.image.load('bullet.png')
        self.rect = self.image.get_rect()
```

Store the bullet's y start position ie at nose of rocket ship

```
        self.y = 480
```

We need to know if the bullet has been fired or not… Initialise the state to not fired (nofire)

```
        self.state='nofire'
```

Now add a method to draw the bullet on the screen.

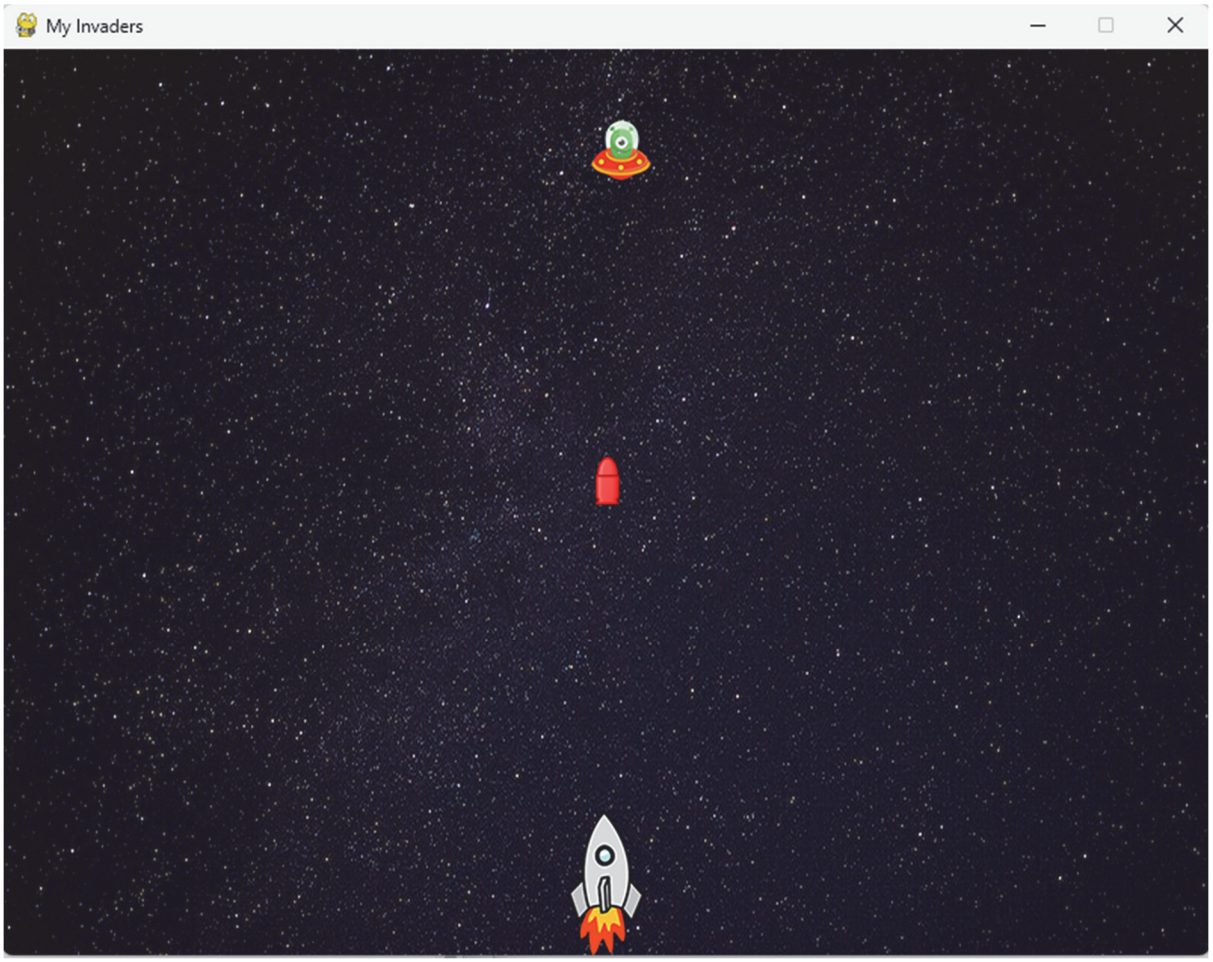
```
    def draw(self):
        # Draw the ufo at its current location.
        self.screen.blit(self.image, self.rect)
```

Add another method to fire the bullet, starting at the rocket's x position on the screen.

```
    def fire(self, rocketxpos):
        self.screen.blit(self.image, (rocketxpos, self.y))
        self.y -= 10
        if self.y < 0:
            self.state = 'nofire'
            self.y = 480
```

Run the Program

When you run the program, you'll be able to move your ship left and right. The UFOs will float around the screen. You can also fire a bullet from the rocket ship

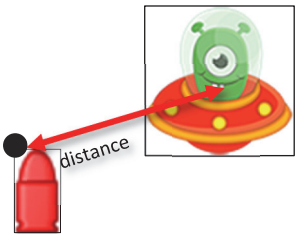


Week 4

This week, we are going to add some collision detection to the bullets and UFO. Have a look at `My Invaders Project - Week04.py`

To do this, first we need to add a new method to the bullets to check to see if they collide with the UFO. We take the x pos and y pos of the UFO, along with the x and y position of the bullet and run it through a formula to calculate the distance between them.

$$distance = \sqrt{(\text{ufo\_X\_pos} - \text{bullet\_X})^2 + (\text{ufo\_Y\_pos} - \text{bullet\_Y})^2}$$



We can write a method as follows. If the distance is below 35 pixels, then the function will return true.

```
def isCollision(self, ufo):
    distance = math.sqrt(math.pow(ufo.ufo_X_pos - self.x, 2) +
    (math.pow(ufo.ufo_Y_pos - self.y, 2)))
    if distance < 35:
        return True
    else:
        return False
```

Then in the main code we can call the method in the bullet object. This method returns true if it's a hit as we saw above. If the isCollision method returns true we draw the explosion.png image on the screen at the same position of the UFO.

```
if bullet.isCollision(ufo):
    screen.blit(exp, (ufo.rect.x, ufo.rect.y)) #show explosion image
```

After a hit, we place UFO back on screen in random position

```
    ufo.rect.x = random.randint(0, 736)
    ufo.rect.y = random.randint(50, 300)
```

Run the Program

When you run the program, you'll be able to move your ship left and right. The UFOs will float around the screen. You can also fire a bullet from the rocket ship. When a bullet hits the UFO you'll see the explosion and the UFO will re-appear on a random position.