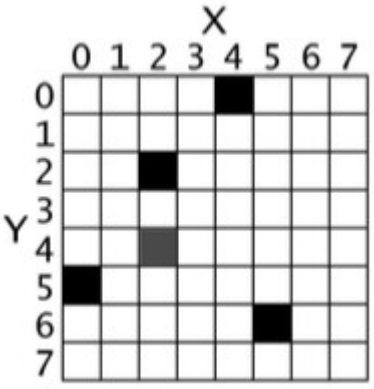
Pygame Basics

Pygame is a set of Python modules designed for writing video games. It runs on a variety of platforms, including Windows, Linux, and OS X. Like any module, use `import` to access it.

In order to draw objects, display images, write text, and so on, we must first create a **surface**. A surface is a two-dimensional array of pixels, each of which has a set of coordinates (*x*, *y*). Unlike in mathematics, the lop left pixel is (0, 0), and the values of *x* and *y* increase as the surface moves right and *down*. In Pygame, an instance of the `Surface` class can be created using `set_mode(WIDTH, HEIGHT)` from the `pygame.display` module. Pygame will need to be initialized, so be sure to call `pygame.init()` at the start of any program.



Once the surface has been created, the remainder of a typical Pygame program is a loop that contains three main components: an **event loop**, which handles things like key presses and mouse clicks; code for the **game logic**, which describes the rules and structure of the game; and **drawing commands**, which display graphics on the surface. These will be discussed later.

Basic Structure For a Pygame Program

```
# Imported modules and initialization (add as necessary)
import pygame
pygame.init()

# Create a surface for drawing
SURF_WIDTH, SURF_HEIGHT = 400, 300
SCREEN = pygame.display.set_mode((SURF_WIDTH, SURF_HEIGHT))
pygame.display.set_caption("My Pygame Program")

# MAIN GAME LOOP
running = True
while running:
    # Check event queue for actionable items
    for event in pygame.event.get():
        # Exit the program
        if event.type == pygame.QUIT:
            running = False

    # Game logic goes here
    # Drawing commands go here

    # Update the display and increment the clock
    pygame.display.update()

# Shut down pygame
pygame.quit()
```

Pygame Basics

Answer the following questions.

1. What are the coordinates of the five darkened pixels in the sample surface?

2. What is the purpose of the `While True` statement on line 10 of the Basic Structure example?

Write programs that accomplish each task, using appropriate programming conventions.

3. Create a 500 x 400 pixel surface with the caption "Hello There!". Use the `fill()` method of `Surface` to change the background colour to blue, (0, 0, 255). Add this method call where the drawing commands should go, just before `pygame.display.update()`.

4. Create a 250 x 250 pixel surface, filled black, with the caption "Random Pixels". Use the `set_at()` method of `Surface` to change the colour of a random pixel – from (0, 0) to (250, 250) -- to a random RGB value, such as (255, 105, 180). Watch as the surface slowly fills with coloured dots.

5. The `set_clip()` method takes four arguments (`left`, `top`, `width`, `height`) to define the area on the surface that can be drawn on (known as the **clipping area**). Modify your program above to set the clipping area to a smaller portion of the surface, to verify that portions outside of the clipping area are not coloured.

6. Using the `fill()` method, change the background colour of the surface so that it cycles through every possible RGB combination from (0, 0, 0) to (255, 255, 255). To prevent flickering, import the `time` module and call the `sleep()` method with an argument of 0.1 seconds.