ITP 115 – Programming in Python

Drawing with Canvas

Canvas Widget

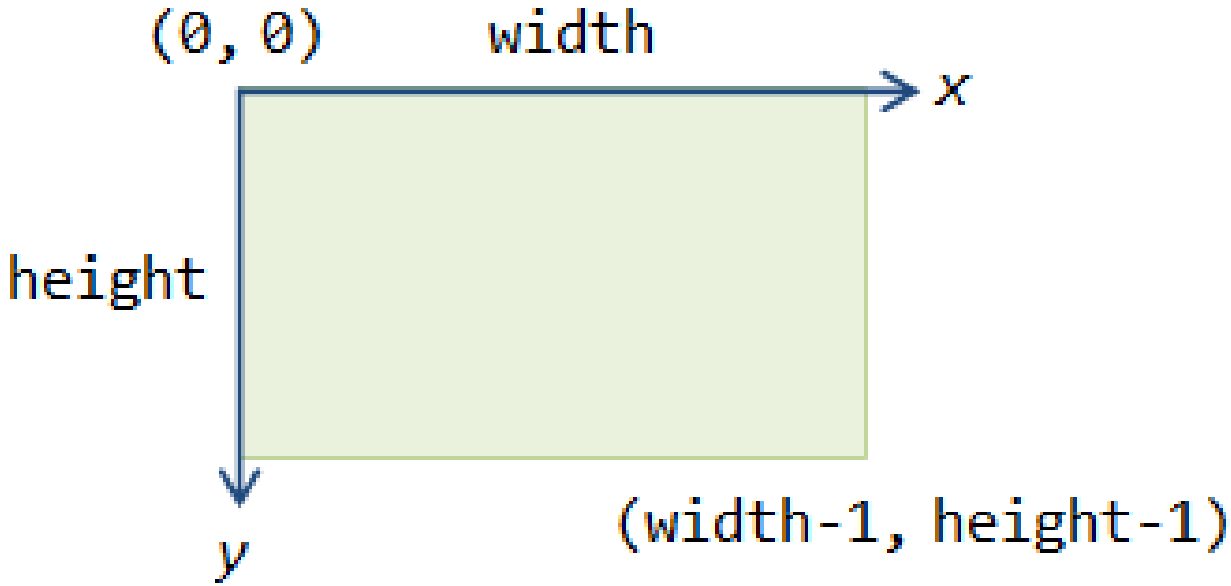
- The Canvas widget is blank widget that you can draw shapes on

- Create and modify assests within a Canvas
  - Contains shapes, graphs, etc.

Canvas Widget

- Syntax

```
c = Canvas(root, width = x, height = y,
           bg = "color")
```



Canvas Widget

- Example (within a class)

```
self.canvas = Canvas(self,
            width = 1000,
            height = 700,
            bg = "black")
```

*For the remainder of the presentation, **self** will be omitted*

Shapes

- You can draw various shapes on the canvas
- Specify **top-left** coordinates and **bottom-right** coordinates
- Can specify other attributes

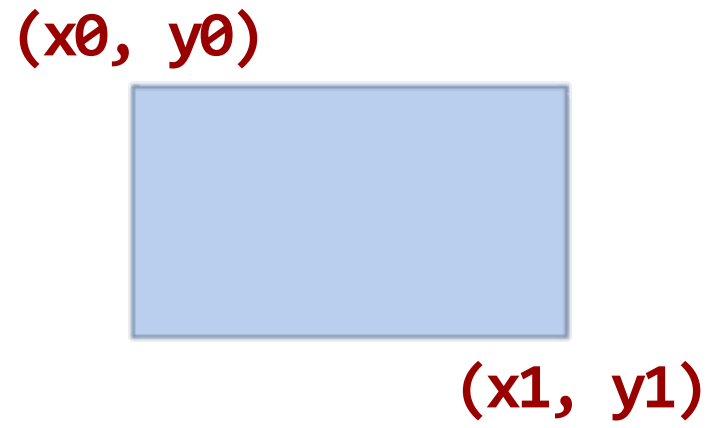
Shapes

- Rectangle

```
rect1 = canvas.create_rectangle(x0, y0, x1, y1)
```

- Other attributes

```
outline="color"
fill="color"
```



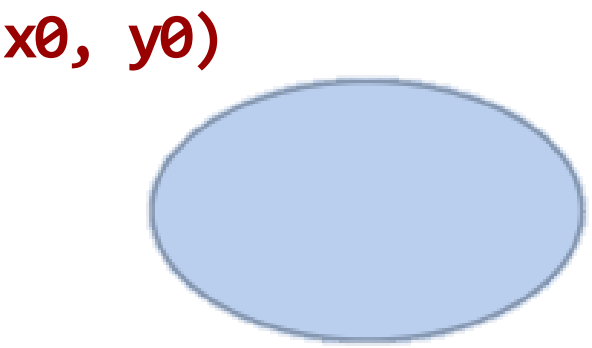
Shapes

- Oval

```
oval1 = canvas.create_oval(x0, y0, x1, y1)
```

- Other attributes

```
outline="color"
fill="color"
```



Special Note about Creating Shapes

- `canvas.create_xxx(...)` returns an (int) id

- We need to store this if we want to access the shape later

Exercise

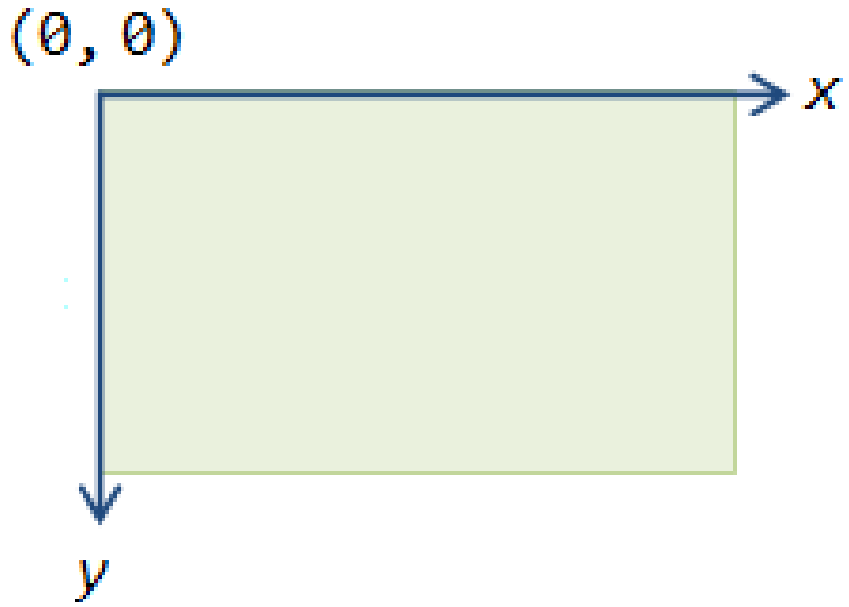
- Example 1 – shapes

Moving Shapes

- Syntax

```
canvas.move(shapeId, changeX, changeY)
```

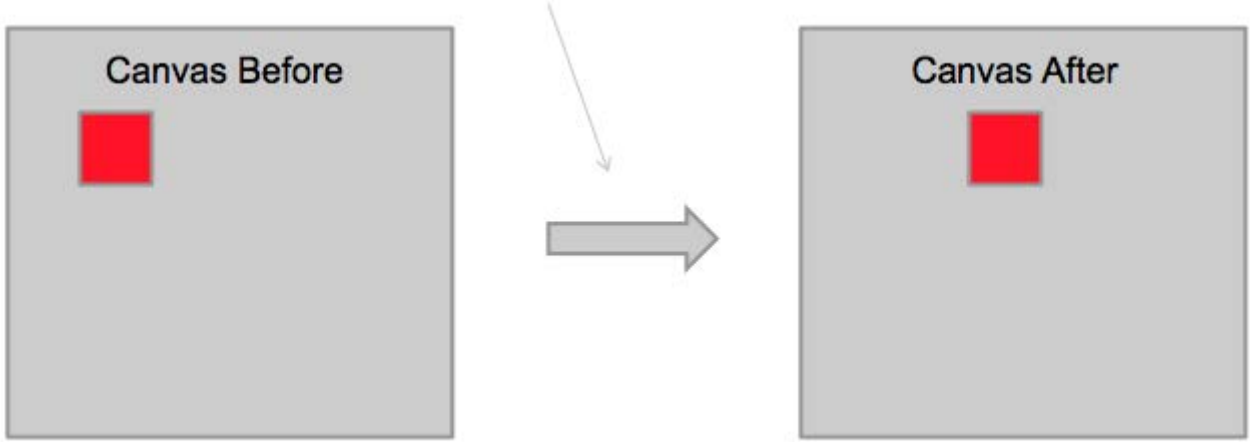
- Remember
  - Right is postive
  - Left is negative
  - Up is negative
  - Down is positive



Moving Shapes

- Example

```
canvas.move(redSquare, 20, 0)
```



How do we get things to happen?

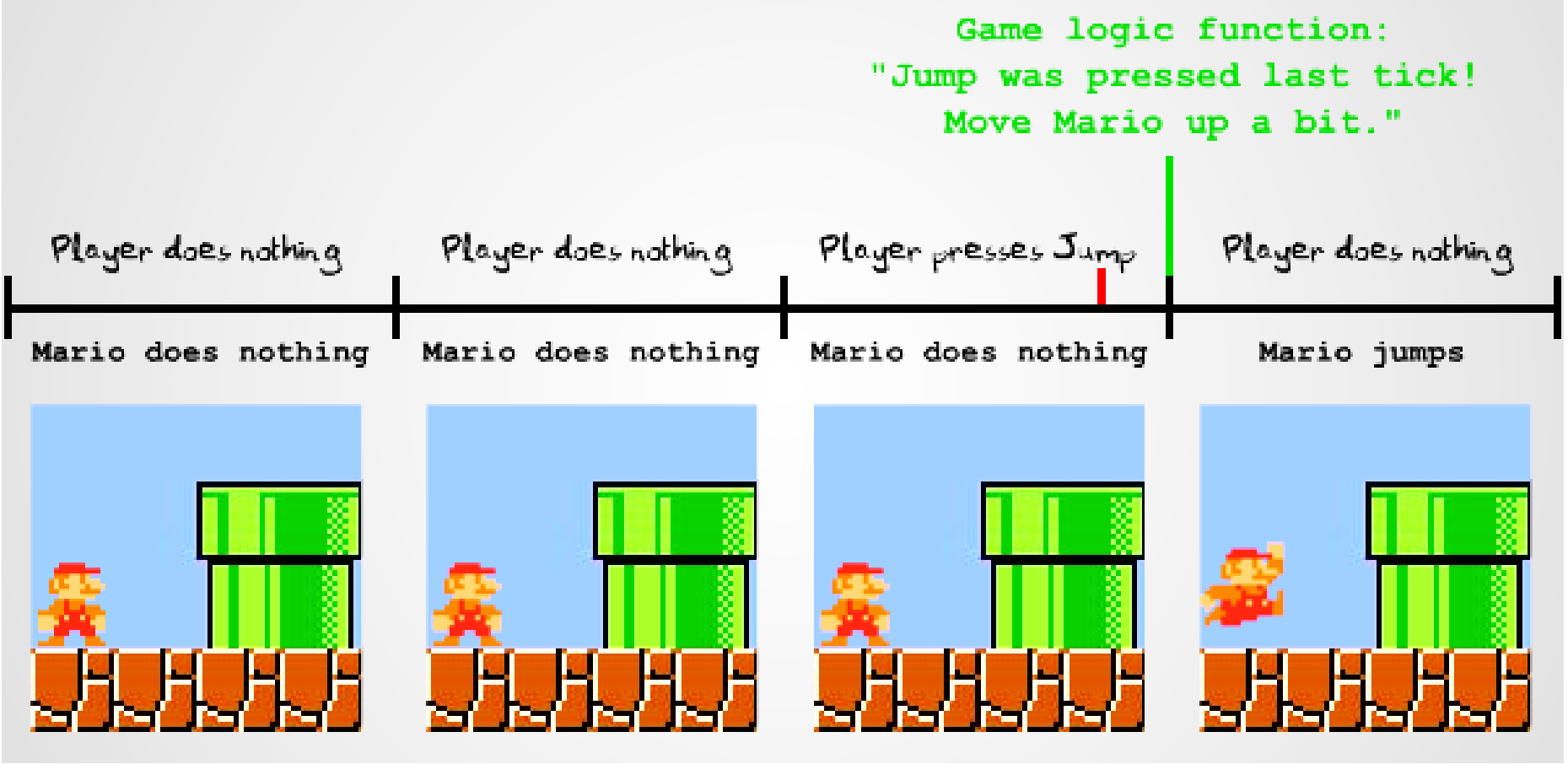
- The Game Loop!
- Key bindings

The Game Loop

- Most game are built around a giant **while** loop
  - Just like event-driven GUIs
- This loop is timed to run VERY fast (milliseconds)
- Everything that happens in the game is in the loop
  - Did player get health pack? *(if yes, update health)*
  - Did player get hit by enemy? *(if yes, lose health)*
  - Did player reach the end of the level *(if yes, show next level)*

Ex: Super Mario Bros. Game Loop

Ex: Super Mario Bros. Game Loop



code.tutsplus.com

Ex: Super Mario Bros. Game Loop

- Move all enemies
- Is User pressing direction?
  - *Move Mario left or right*
- Is User pressing jump?
  - *Mario starts jump*
- Is Mario in the air already?
  - *Continue jump*
- Is Mario touching enemy?
  - *Mario dies*
- Is Mario landing on enemy?
  - *Enemy dies*

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies
```

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies
```

But wait!
We don't want an "out of control" while loop

We want this to run every X milliseconds

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies
```

The solution?

Have this method set a timer for this same method to be called again in X milliseconds

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies

    after(X, gameLoop)
```

The solution?

Have this method set a timer for this same method to be called again in X milliseconds

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies

    after(X, gameLoop)
```

after *is a Python function to set a timer to call a function after a certain delay*

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies

    after(X, gameLoop)
```

function / method to call

gameLoop()

```
def gameLoop():
    #code we want to happen every X milliseconds
    #things like enemies moving
    #spawning enemies

    after(X, gameLoop)
```

Delay (in milliseconds)

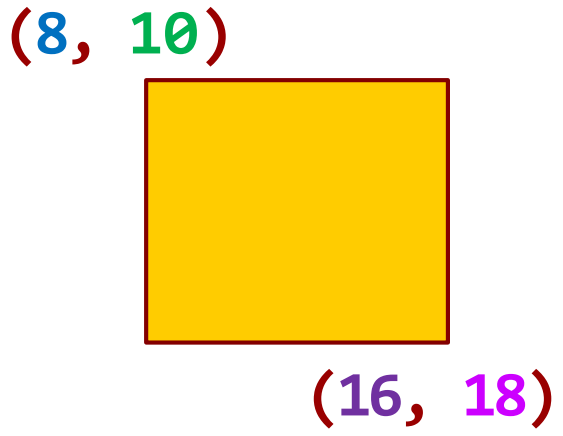
Exercise

- Example 2 – timer

How do we know where a shape is?

```
coords = canvas.coords(shape)
```

```
coords[0] is left X: 8
coords[1] is top Y: 10
coords[2] is right X: 16
coords[3] is bottom Y: 18
```



Exercise

- Example 3 – advanced timer

User Input (w/ respect to Canvas)

- Binding Keys

```
canvas.bind("<Key>", methodName)
```

- Focusing on window

```
canvas.focus_set()
```

- Responding to keyboard input

```
def methodName(event):
    #code to happen on key press
    canvas.update()
```

How do we know where a shape is?

`canvas.coords(shapeId)`

- Returns a tuple that has the top-left and bottom-right (x, y) points that define the shape

Exercise

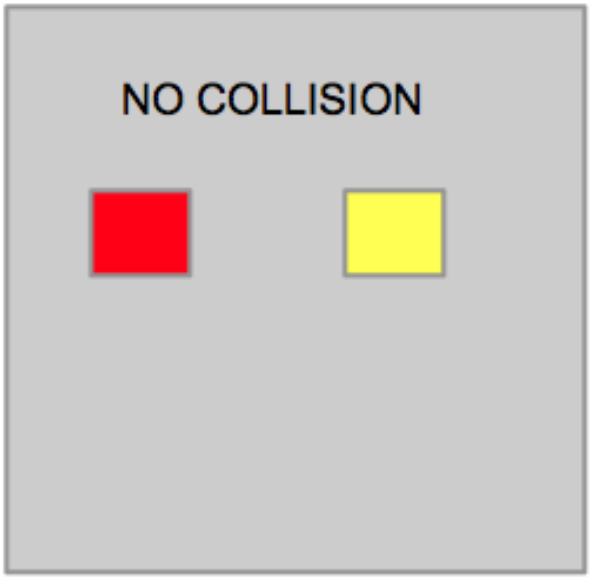
- Example 4 - keybinding

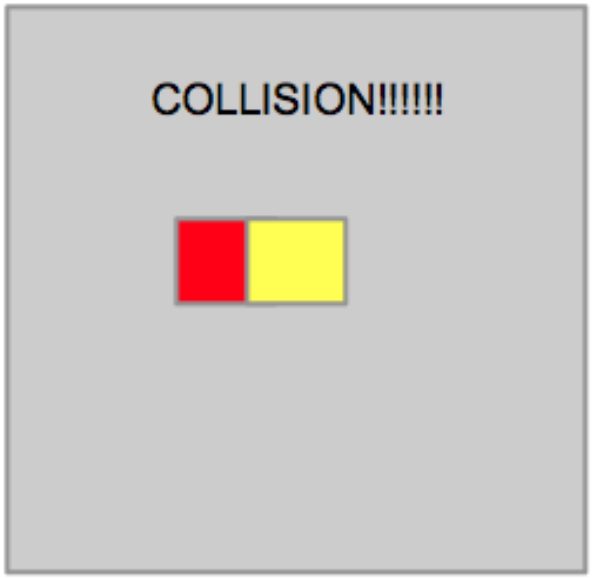
Collision Detection

- What is considered a collision?
- How can we detect collisions on Canvas?
- How should we implement collision detection?

Collision Detection

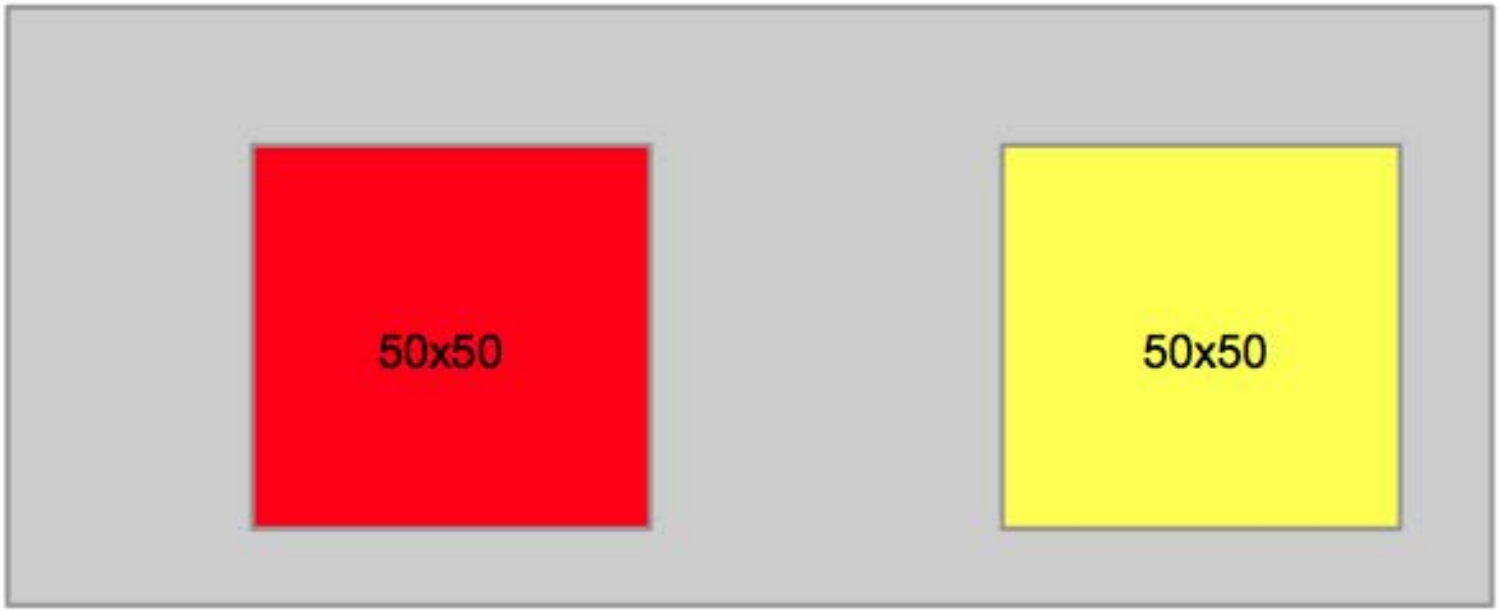
- A collision occurs when two objects *on the screen* intersect





Collision Detection

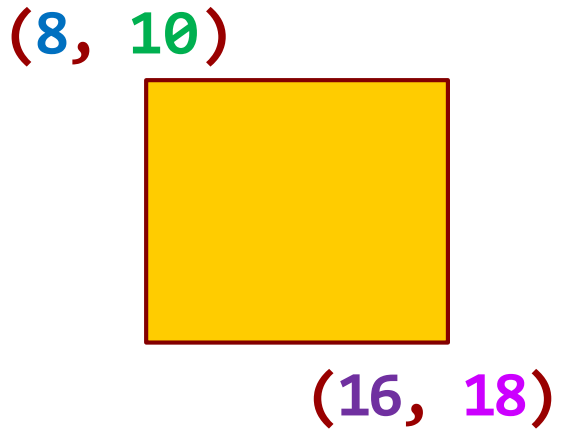
- We need to be careful how we check for collisions



Collision Detection

```
coords = canvas.coords(shape)
```

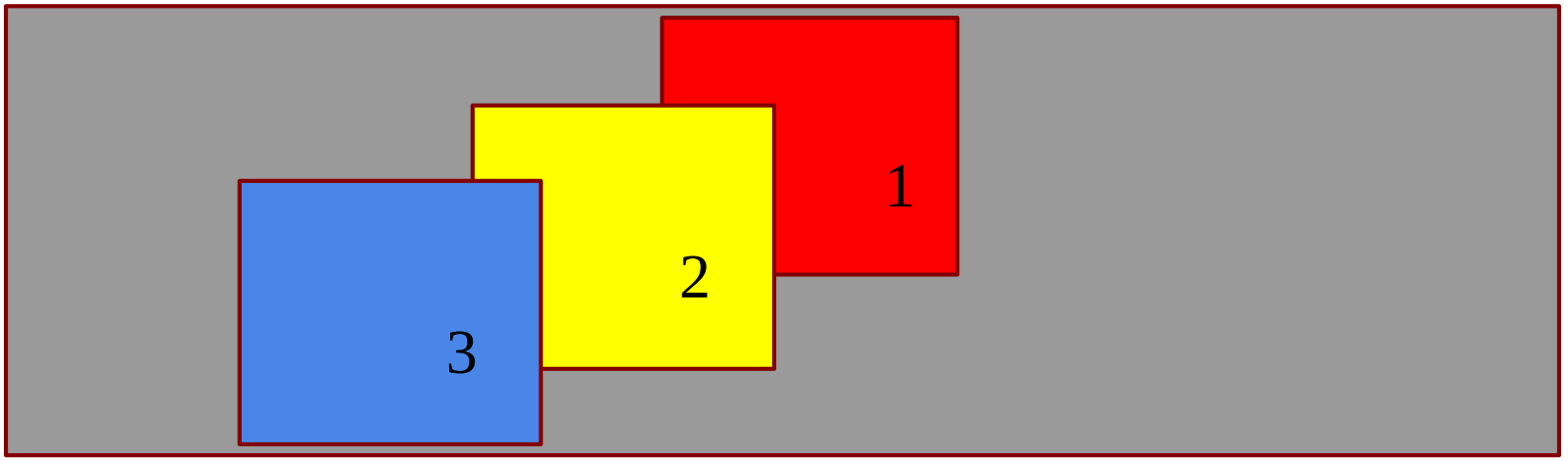
```
coords[0] is left X: 8
coords[1] is top Y: 10
coords[2] is right X: 16
coords[3] is bottom Y: 18
```



Collision Detection

- Find tuple of all id's of overlapping coordinates

```
collisionList =
  canvas.find_overlapping(x0,y0,x1,y1)
```



Exercise

- Example 5 - collision detection with a single object

- Example 6 - collision detection with multiple objects

Loading Images

- Images act like rectangles
- Only .gif files can be used

Two Stesp to Loading Images

1. Load the image file

```
imageVar = PhotoImage(file="filename.gif")
```

2. Draw the image on the screen

```
canvas.create_image(leftXCoord, topYCoord,
                    image=imageVar)
```

Exercise

- Example 7 – Mario!