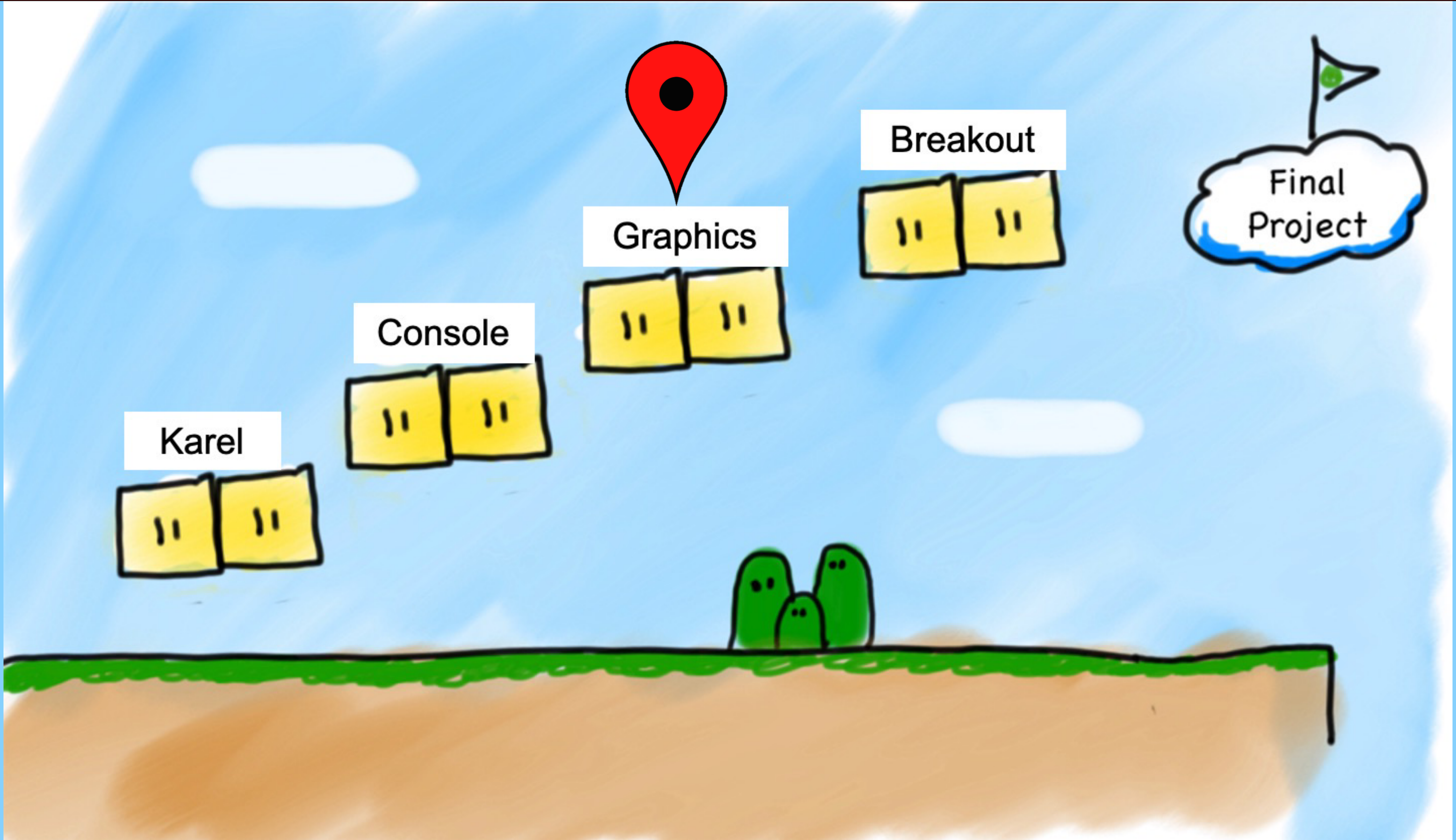
CS Bridge, Lecture 7

Graphics

Learning Goals

1. Learn how we can make graphical programs
2. Learn about the different ways to draw shapes on-screen

We Are Here



Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

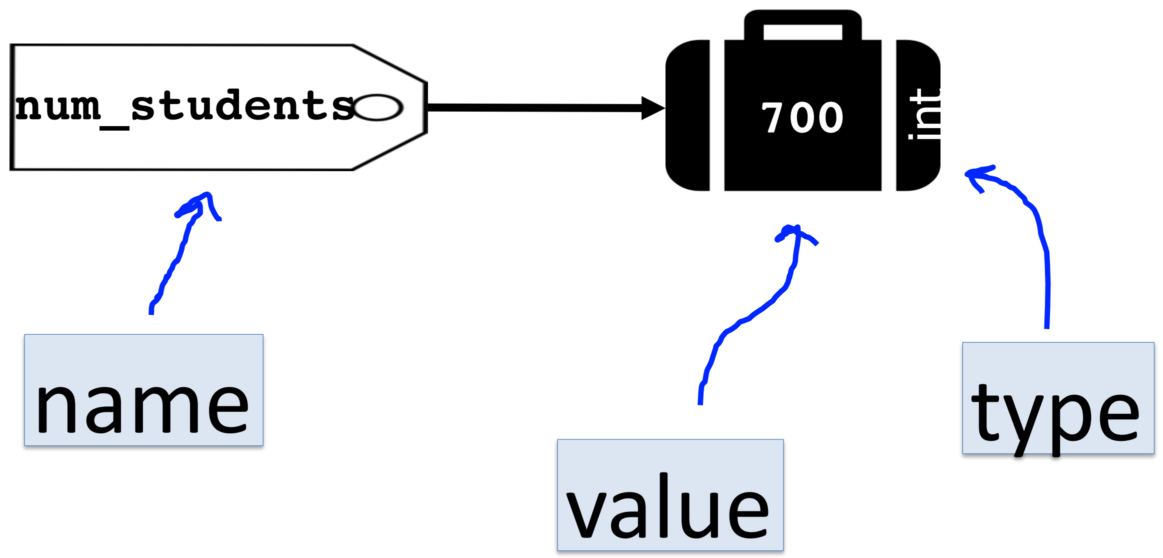
Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

Review: Variables

A variable is a "suitcase". We can use it to store a value.

```
num_students = 700
```



Review: Variables

We can change the values of variables and use them in expressions.

```
num_flowers = 5
flowers_planted = 6
num_flowers = num_flowers + flowers_planted
```

Review: User Input

We can ask the user for input and store it in a variable.

```
num1 = input("Enter first number: ")
```

Remember that user input is always *text* ("string").

```
num1 = input("Enter first number: ")
print(int(num1) + 2)
```

Review: Printing

We can print out information to the user:

```
num1 = random.randint(1, 6)
print("You rolled a " + str(num1))
```

Remember we must convert to a string before adding to another string.

```
num1 = random.randint(1, 6)
print("You rolled a " + num1) # error
```

Review: Control Flow

We can use if/elif/else to conditionally perform tasks:

```
if condition1:
    statement
    ...
elif condition2:
    statement
    ...
else:
    statement
    ...
```

Runs the first group of statements if ***condition1*** is true; otherwise, runs the second group of statements if ***condition2*** is true. Otherwise, runs the third group of statements.

Review: Control Flow

We can use while loops to repeat until some condition is false:

```
while condition:
    statement
    statement
    ...
```

Repeats the statements in the body until ***condition*** is no longer true. Each time, we execute *all* statements, and **then** check the condition.

Conditions in Python

```
while condition:
    body
```

```
if condition:
    body
```

The condition should be a “boolean” which is either `True` or `False`

Relational Operators

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=`	does not equal	`3.2 != 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`
`>=`	greater than or equal to	`5.0 >= 5.0`	`true`

Relational Operators

Operator	Meaning	Example	Value
`==`	equals	`1 + 1 == 2`	`true`
`!=`	does not equal	`3.2 != 2.5`	`true`
`<`	less than	`10 < 5`	`false`
`>`	greater than	`10 > 5`	`true`
`<=`	less than or equal to	`126 <= 100`	`false`
`>=`	greater than or equal to	`5.0 >= 5.0`	`true`

Relational Operators

```
if 1 < 2:
    print("1 is less than 2!")
```

```
num = int(input("Enter a number: "))
if num == 0:
    print("That number is 0!")
else:
    print("That number is not 0.")
```

Practice: Sentinel Loops

- **sentinel**: A value that signals the end of user input.
  - **sentinel loop**: Repeats until a sentinel value is seen.

- Example: Write a program that prompts the user for numbers until the user types -1, then output the sum of the numbers.
  - In this case, -1 is the sentinel value.

```
Type a number: 10
Type a number: 20
Type a number: 30
Type a number: -1
Sum is 60
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
   sum += num
   num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
    sum += num
    num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
    sum += num
    num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
    sum += num
    num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
    sum += num
    num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Practice: Sentinel Loops

```
# fencepost problem!
# ask for number - post
# add number to sum - fence

sum = 0
num = int(input("Enter a number: "))
while num != -1:
    sum += num
    num = int(input("Enter a number: ")

print("Sum is " + str(sum))
```

Review: Control Flow

We can use for loops to repeat a certain number of times:

```
for i in range(max):
    statement
    statement
    ...
```

Repeats the statements in the body ***max*** times.

Using the For Loop Variable

This is a **variable** that, every time through the loop, becomes the next value in the range 0…49.

```
for i in range(50):
    print(i * 2)
```

<u>Question</u>: **what does this code do?**

Using the For Loop Variable

This is a **variable** that, every time through the loop, becomes the next value in the range 0,2,4,6,8,..98.

```
for i in range(0, 100, 2):
    print(i)
```

<u>Question</u>: **what does this code do?**

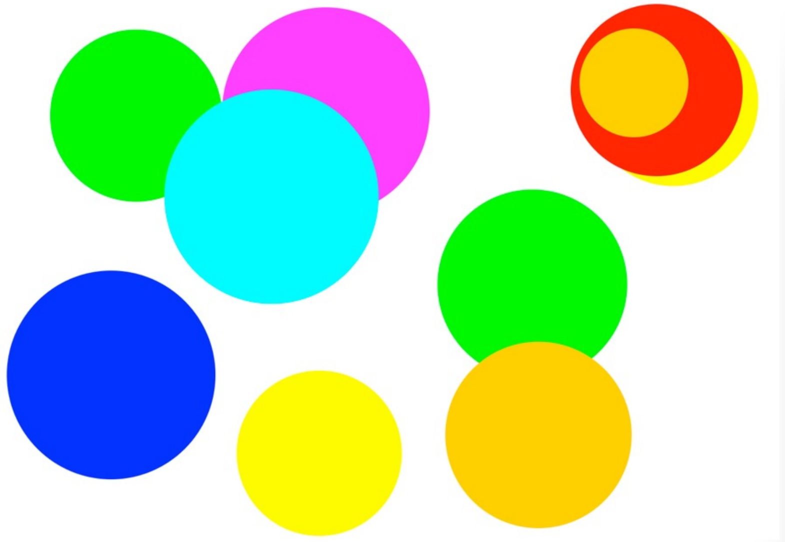
Concepts So Far

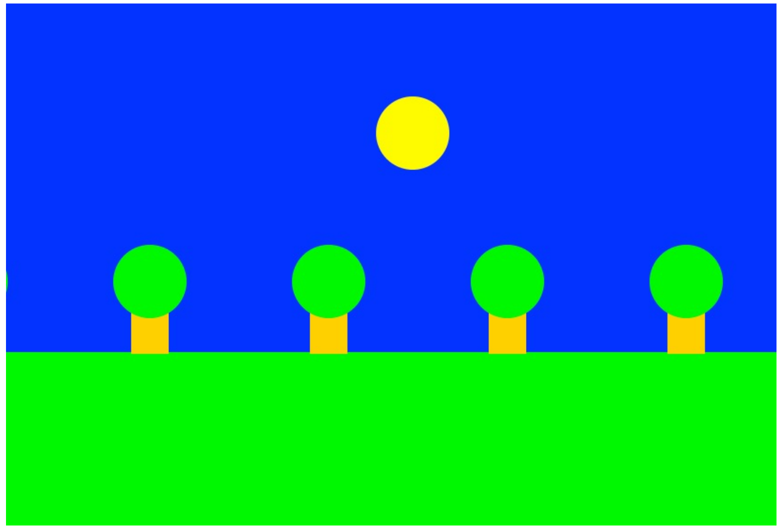
- We have covered many concepts quickly, and they can be challenging to understand!
- If you have any questions, or feel stuck with any of the concepts, please visit us in office hours
- There is a “conceptual help” room where we can walk through any concepts again

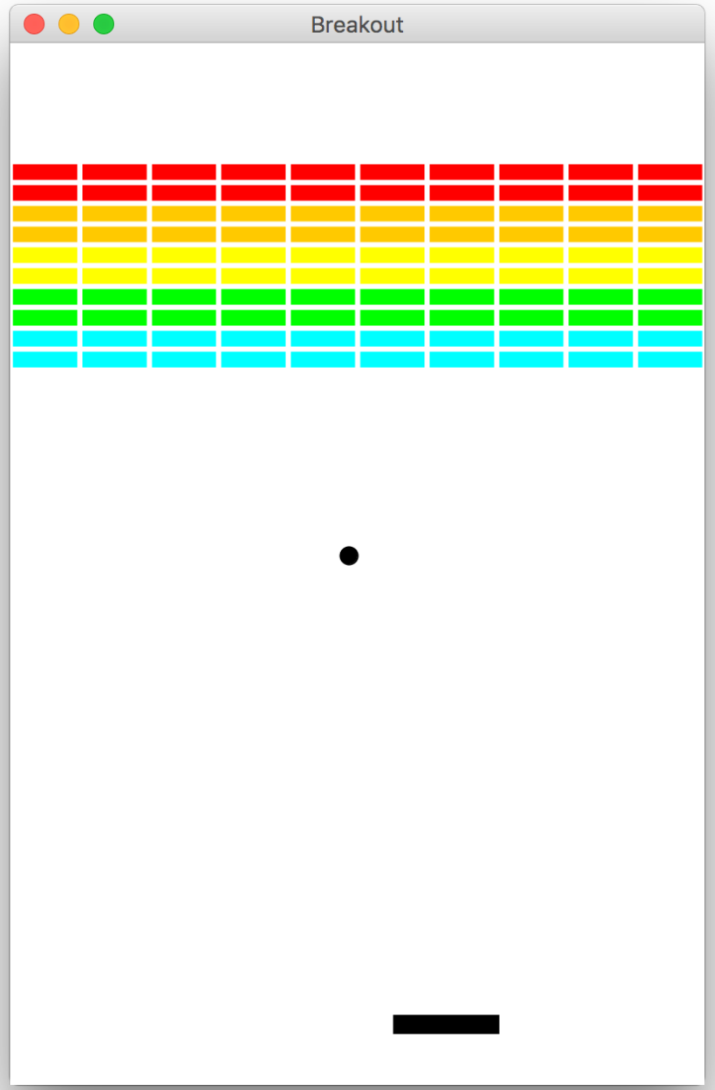
Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

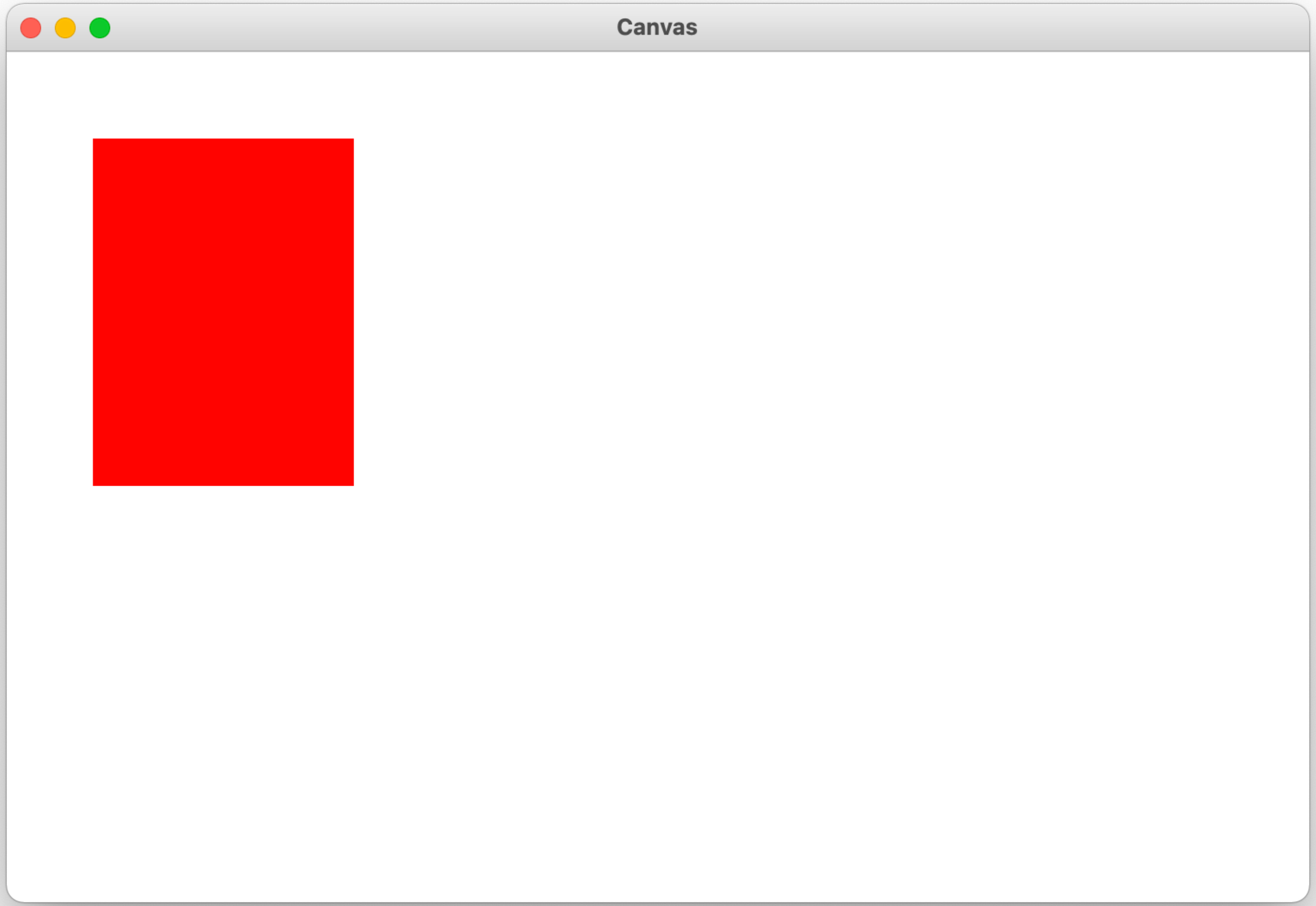
Graphics Programs







Our First Graphics Program



Our First Graphics Program

```
from graphics import Canvas    # Import graphics library


def main():
    canvas = Canvas()
    rect = canvas.create_rectangle(50, 50, 200, 250)
    canvas.set_color(rect, 'red')
    canvas.mainloop()
```

Our First Graphics Program

```
from graphics import Canvas    # Import graphics library


def main():
    canvas = Canvas()
    rect = canvas.create_rectangle(50, 50, 200, 250)
    canvas.set_color(rect, 'red')
    canvas.mainloop()
```

Our First Graphics Program

```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```

Our First Graphics Program

```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```

Our First Graphics Program

```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```

Our First Graphics Program

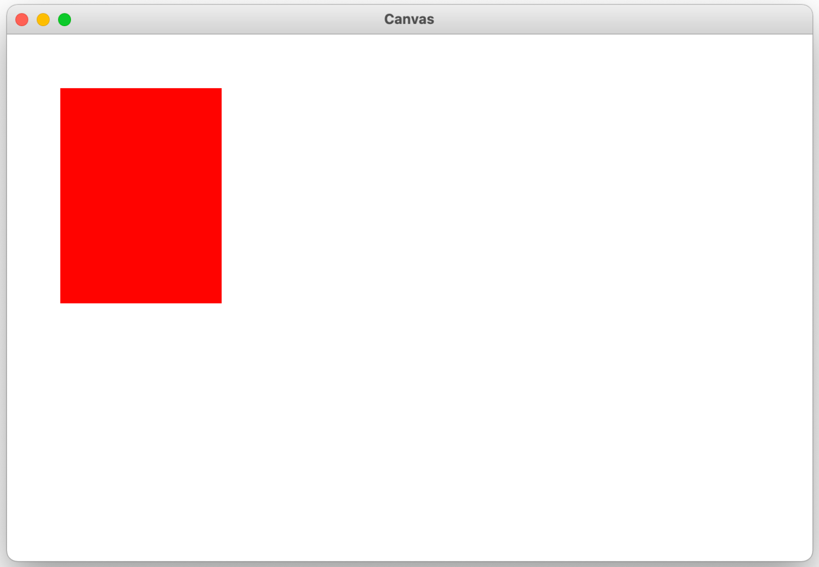
```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```

Our First Graphics Program

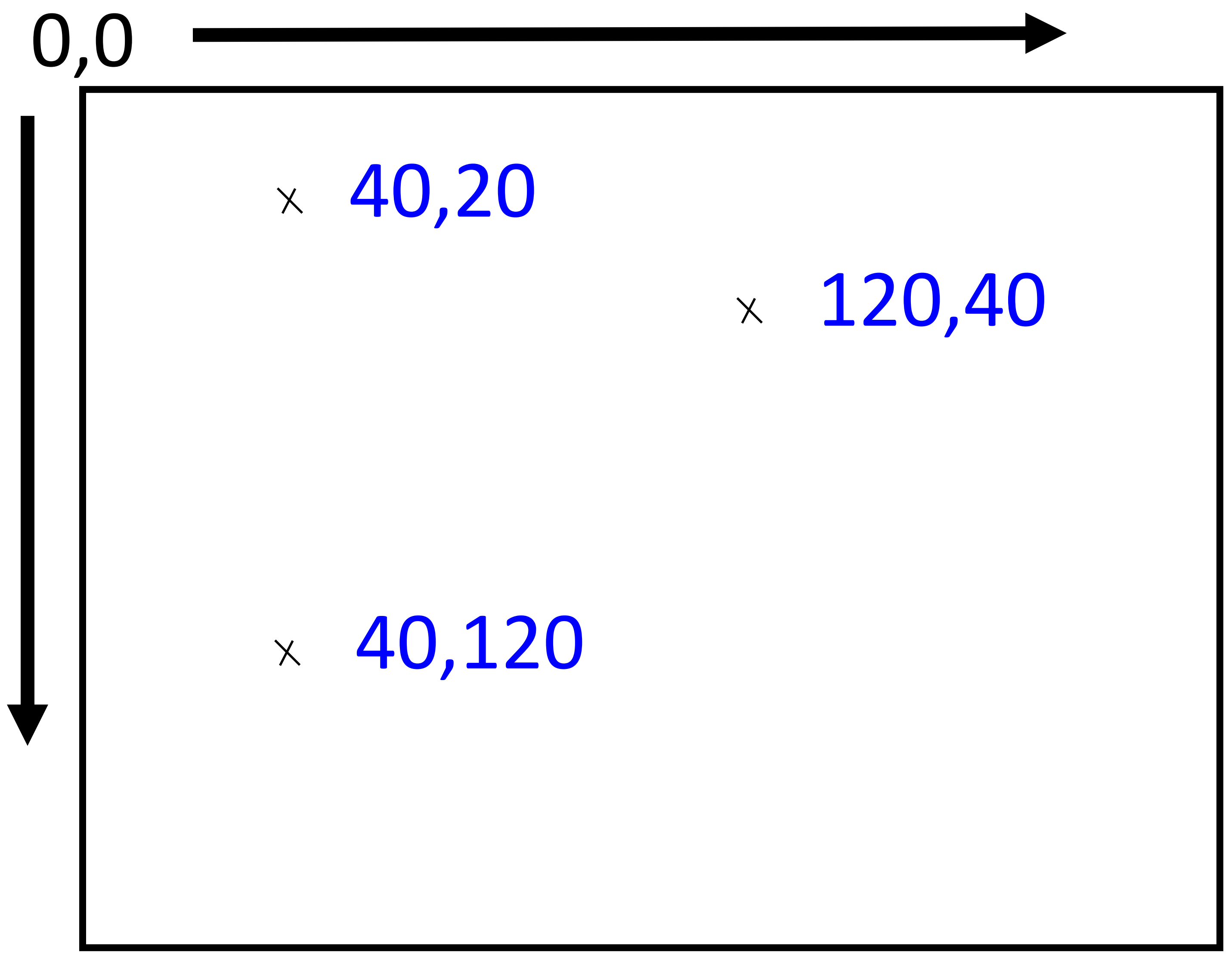
```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```

Our First Graphics Program

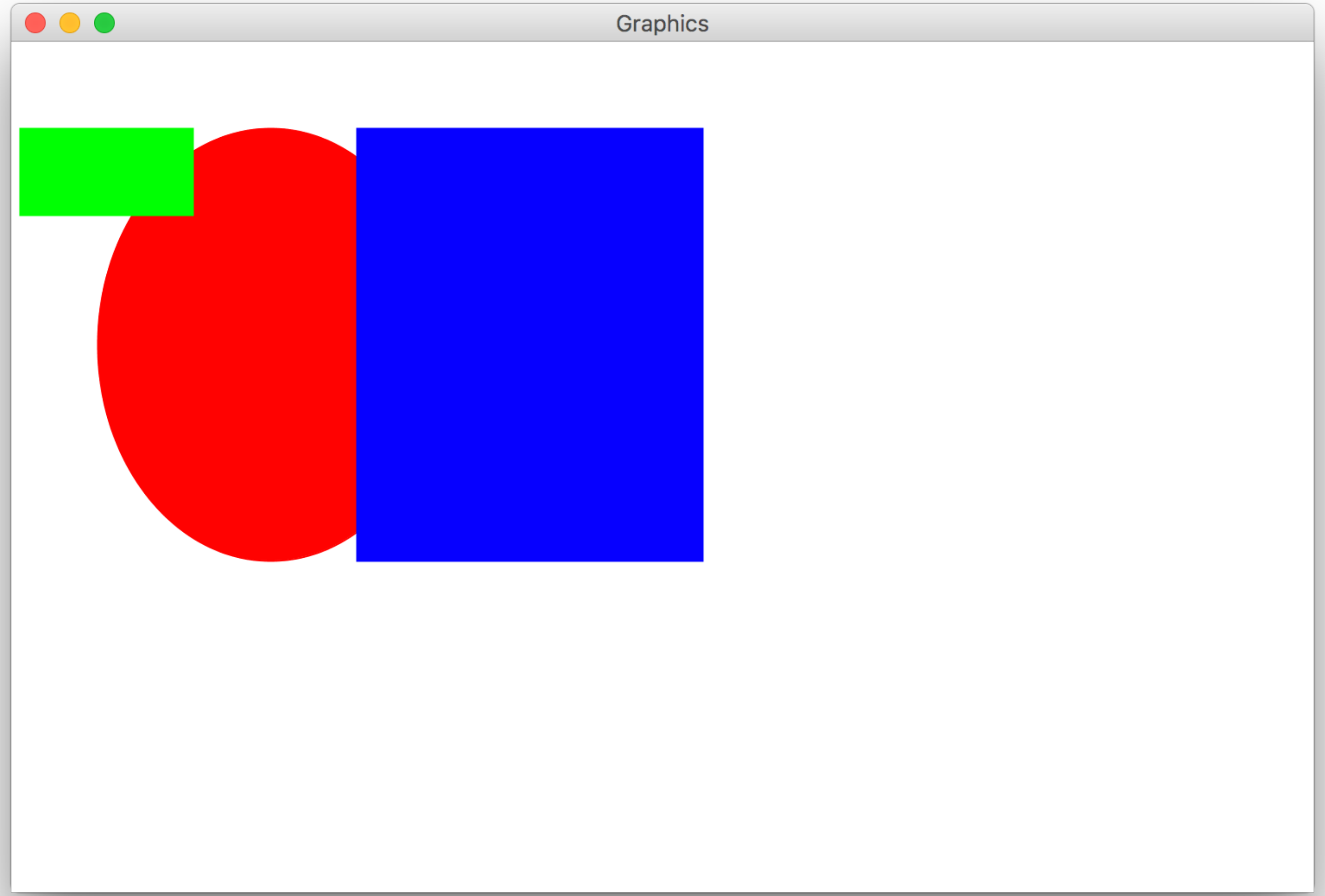
```
# Create a new graphical canvas window
canvas = Canvas()

# Create a rect from (50, 50) to (200, 250)
rect = canvas.create_rectangle(50, 50, 200, 250)

# Set some properties
canvas.set_color(rect, 'red')

# Draw the canvas
canvas.mainloop()
```



The Graphics Canvas

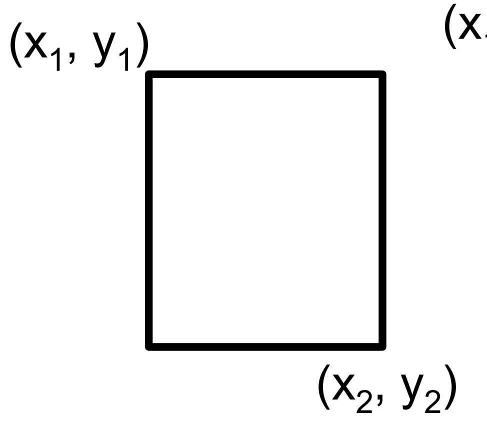


Collage Model

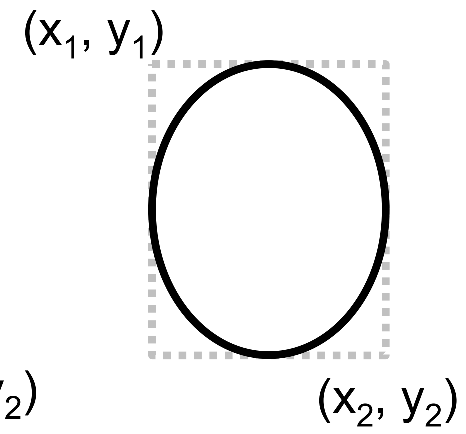


Graphical Objects

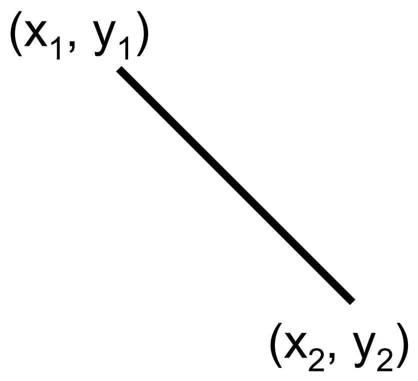
`rectangle`



`oval`



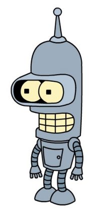
`line`



`text`

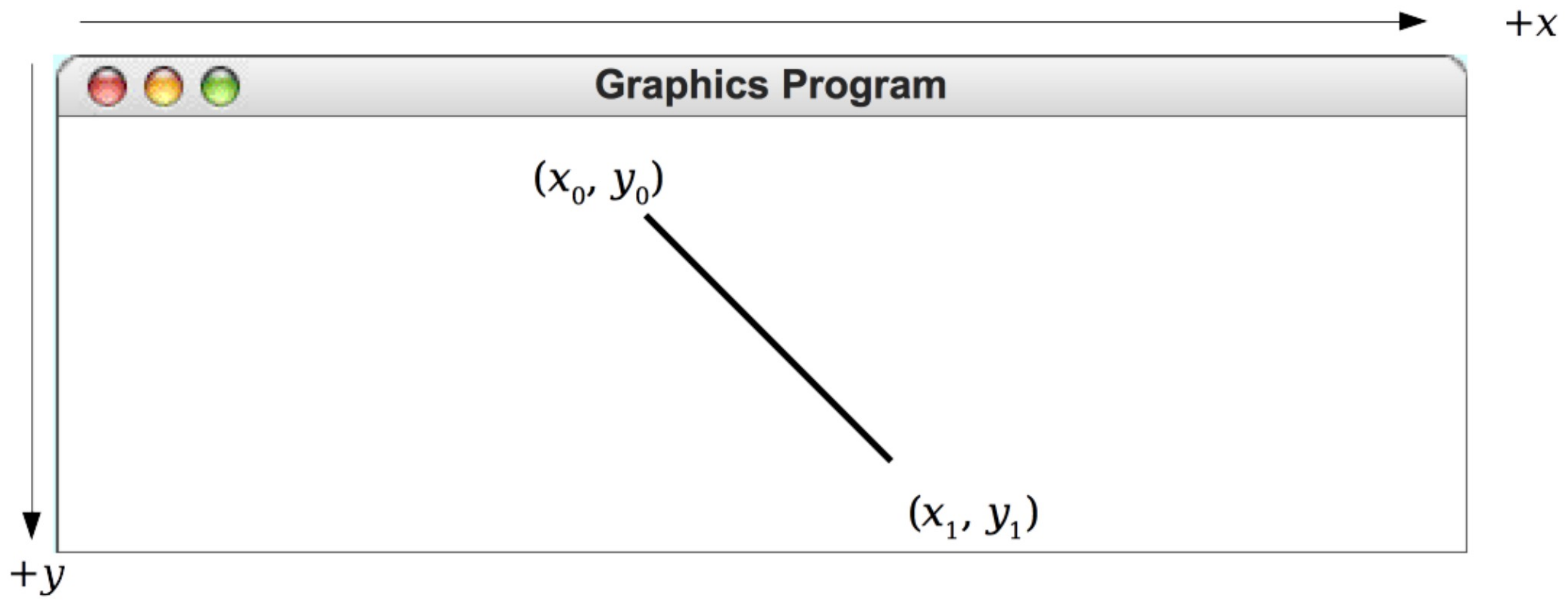
Hello there!

`image`



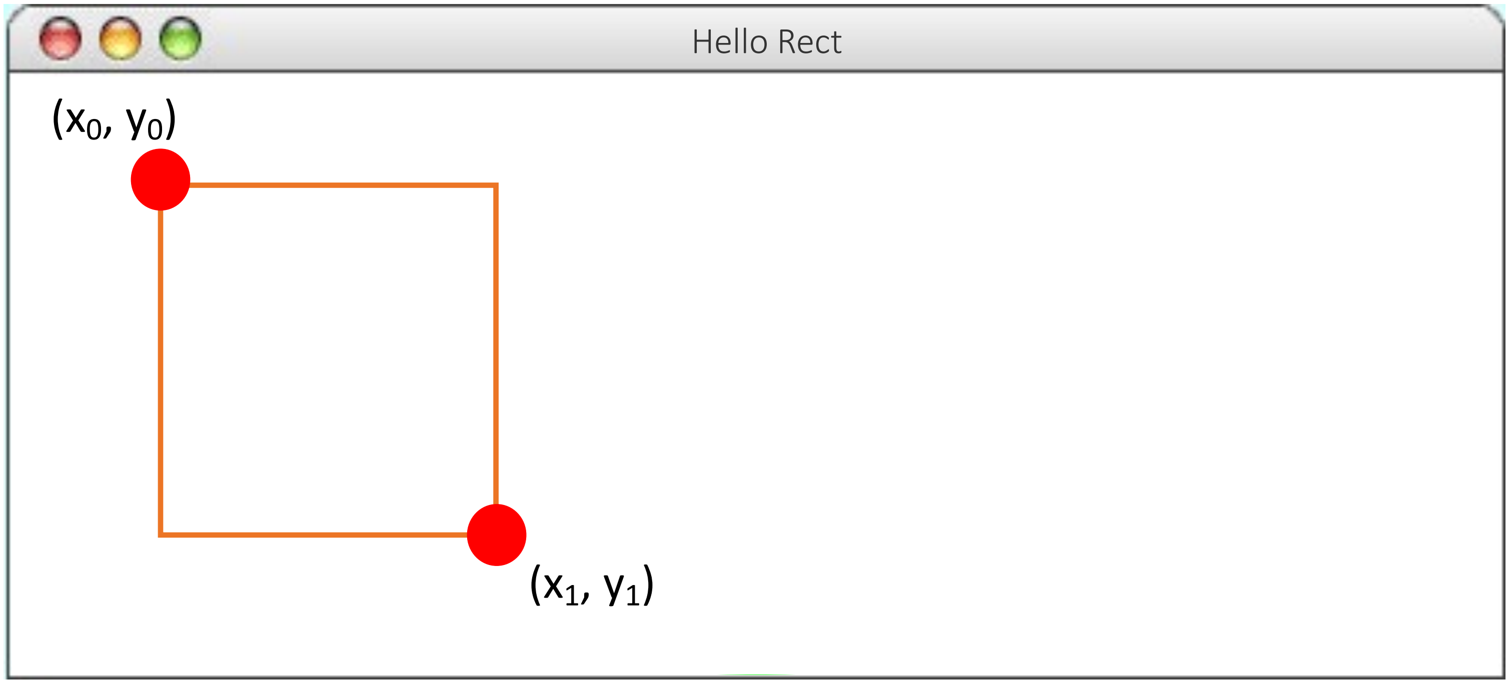
Drawing Lines

canvas`.create_line(`x_0, y_0, x_1, y_1`)`



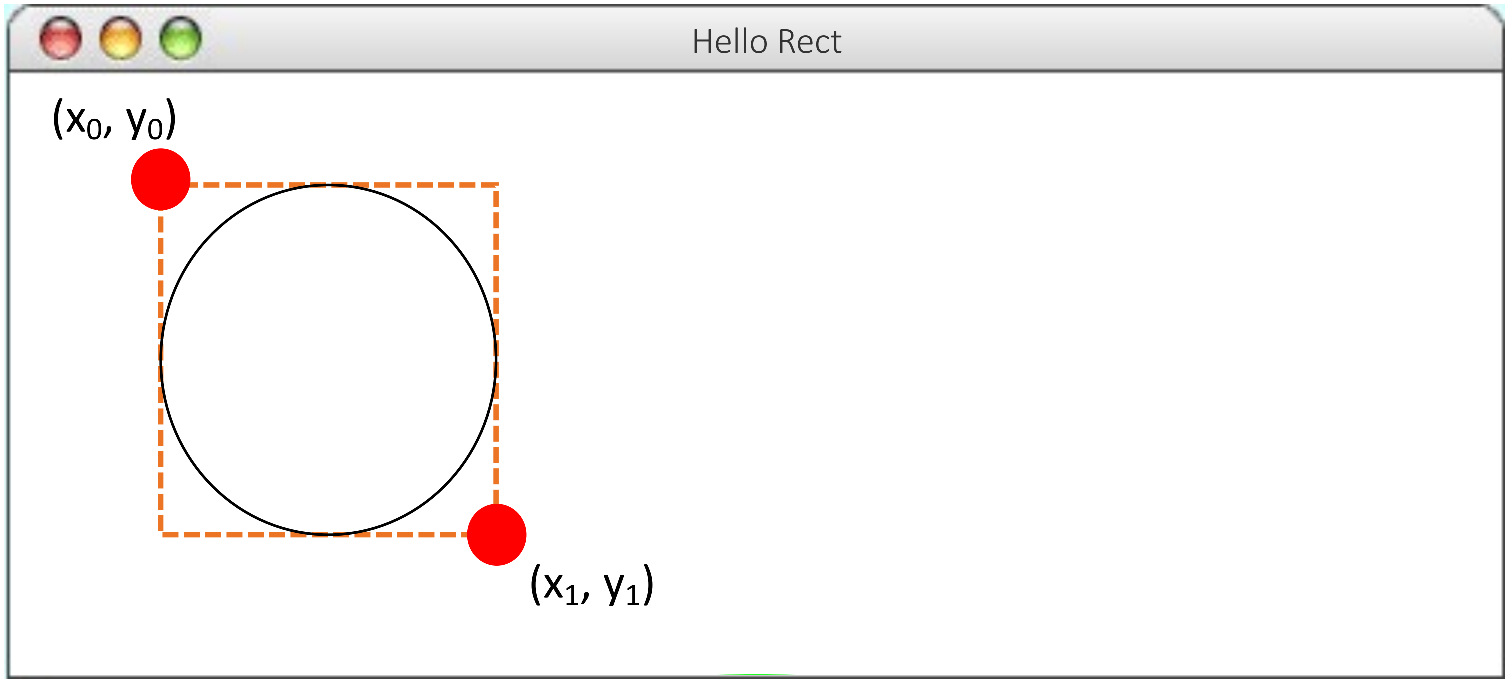
Drawing Rectangles

canvas`.create_rectangle(`x_0, y_0, x_1, y_1`)`



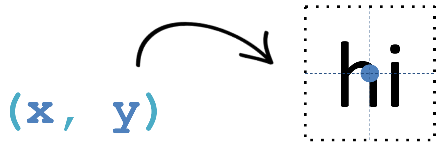
Drawing Ovals

canvas`.create_oval(`x_0, y_0, x_1, y_1`)`



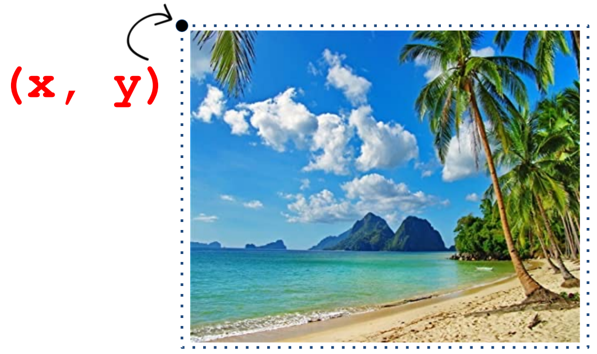
Drawing Text

canvas.**`create_text(`***x, y, "text"***`)`**

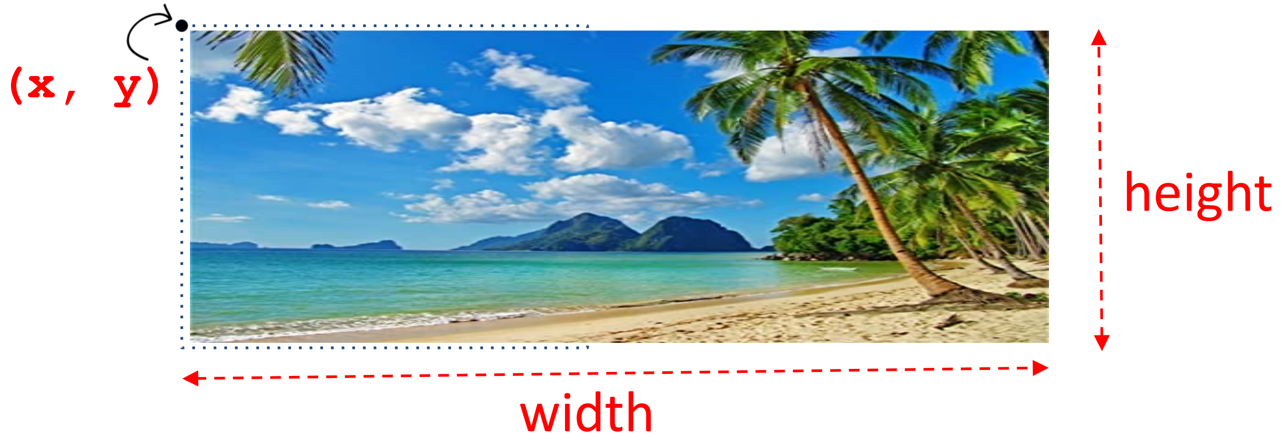


Drawing Images

canvas.**create_image(***x, y, “name of the file”***)**



canvas.**create_image(***x, y, width, height, “name of the file”***)**



Creating Graphical Objects

You can create graphical objects as follows:

canvas`.create_line(`x_0, y_0, x_1, y_1`)`
Creates a new line connecting (x0, y0) and (x1, y1).

canvas`.create_rectangle(`x_0, y_0, x_1, y_1`)`
Creates a new rectangle on the canvas the size of this bounding box.

canvas`.create_oval(`x_0, y_0, x_1, y_1`)`
Creates a new oval on the canvas contained within this bounding box.

canvas`.create_text(`*x, y, text*`)`
Creates text on the canvas with the specified contents, centered at (x, y).

canvas`.create_image(`*x, y, filepath*`)`
Creates a new image on the canvas from the specified file, with top-left corner at (x, y).

canvas`.create_image(`*x, y, width, height, filepath*`)`
Creates a new image on the canvas from the specified file, with top-left corner at (x, y) and the specified width and height.

Operations on Graphical Objects

canvas.**moveto**(*object, x, y*) Sets the location of obj to the specified coordinates.
canvas.**set_color**(*object, color*) Sets the outline and fill color (if applicable) of the object.
canvas.**set_outline_color**(*object, color*) Sets the outline color of the object.
canvas.**set_fill_color**(*object, color*) Sets the fill color of the object.
canvas.**set_font**(*object, font, size*) Sets the font and font size for the given text object.
canvas.**delete**(*object*) Deletes the object from the canvas

See the Graphics reference under the "Resources" tab on the course website for the full list!

Operations on the Canvas

- We can perform some operations with the Canvas itself:

Method	Description
canvas.get_canvas_width(), *canvas*.get_canvas_height()	Get the width and height of the Canvas window
`canvas.set_canvas_title(text)`	Sets the text in the window title bar
`canvas.set_canvas_background_color(color)`	Sets the background color of the canvas

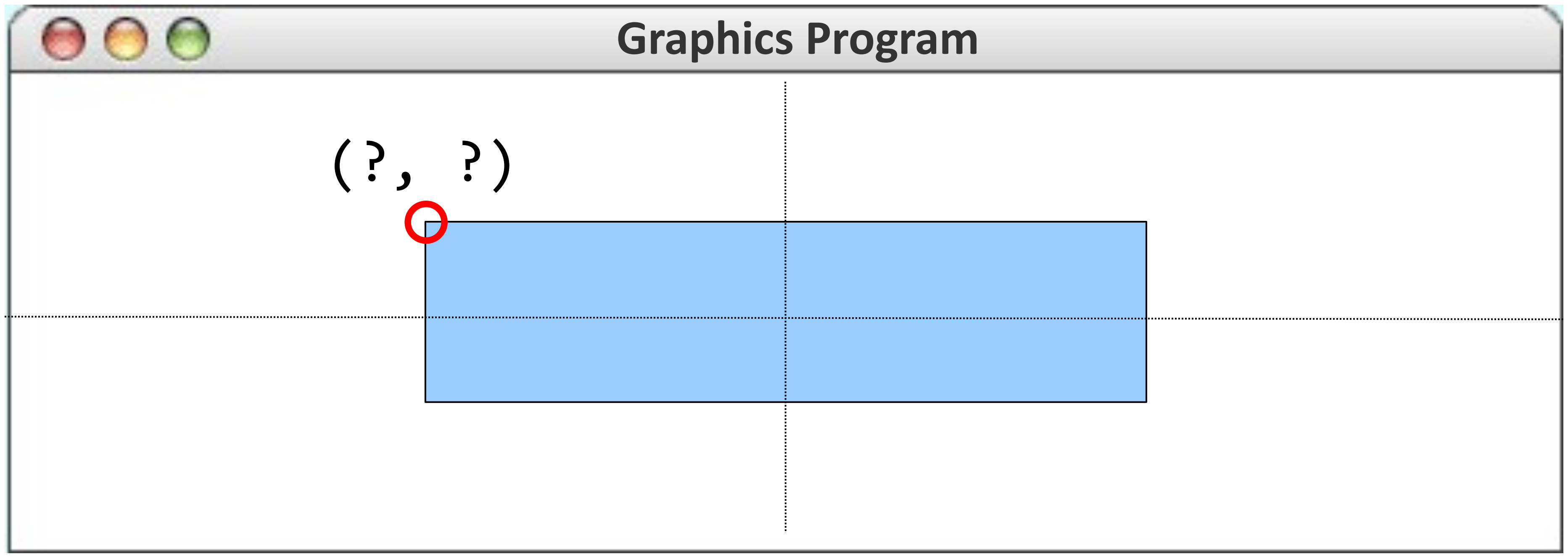
- You can optionally specify the Canvas size (width/height) when you create it:

```
def main():
    canvas = Canvas(500, 400)
    rect = canvas.create_rectangle(50, 50, 200, 250)
    ...
    canvas.mainloop()
```

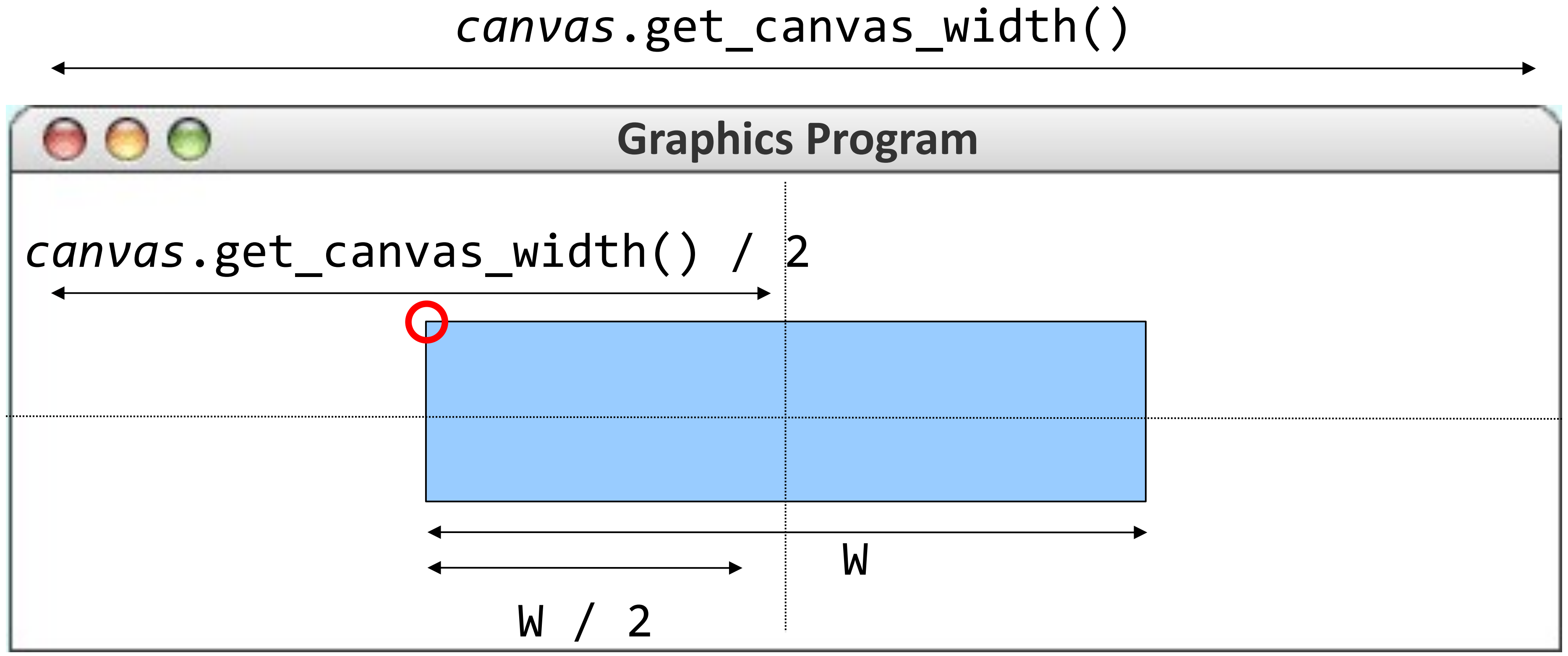
Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

Practice: Centering

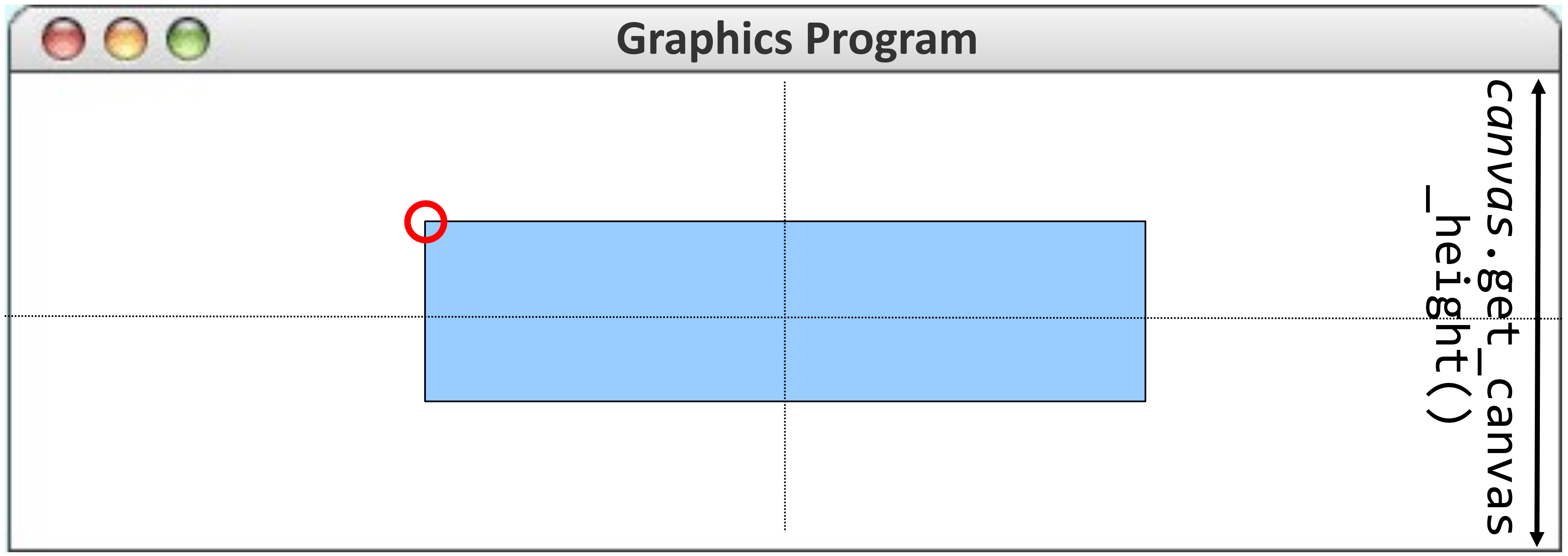


Practice: Centering



rectangle's x value = *canvas*.get_canvas_width() / 2 – W / 2

Practice: Centering



rectangle's y value = *canvas*.get_canvas_height() / 2 – H / 2

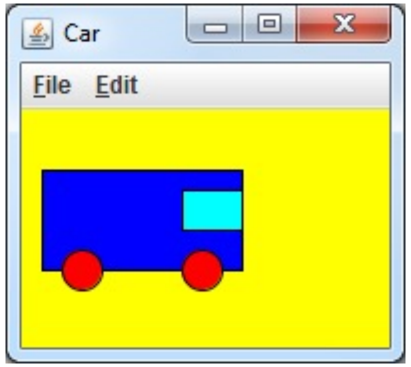
Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

Practice: Car

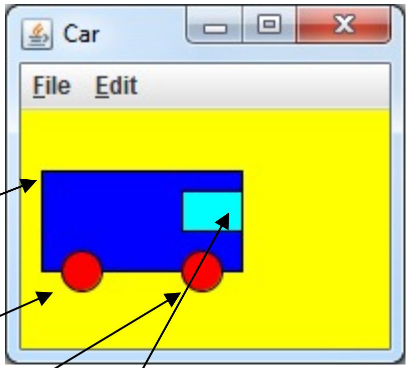
Write a graphical program named **Car** that draws a figure that looks (kind of) like a car.

- Red wheels at (20, 70) and (80, 70), size 20x20
- Cyan windshield at (80, 40), size 30x20
- Blue body at (10, 30), size 100x50
- yellow background



Car Solution

```
# When 2 shapes occupy the same pixels, the last one drawn "wins"
def main():
    canvas = Canvas()
    canvas.set_canvas_title("Car")
    canvas.set_canvas_background_color("yellow")

    # Car body
    body = canvas.create_rectangle(10, 30, 110, 80)
    canvas.set_fill_color(body, "blue")

    # Car wheels
    wheel1 = canvas.create_oval(20, 70, 40, 90)
    canvas.set_fill_color(wheel1, "red")
    wheel2 = canvas.create_oval(80, 70, 100, 90)
    canvas.set_fill_color(wheel2, "red")

    # Windshield
    windshield = canvas.create_rectangle(80, 40, 110, 60)
    canvas.set_fill_color(windshield, "cyan")

    canvas.mainloop()
```

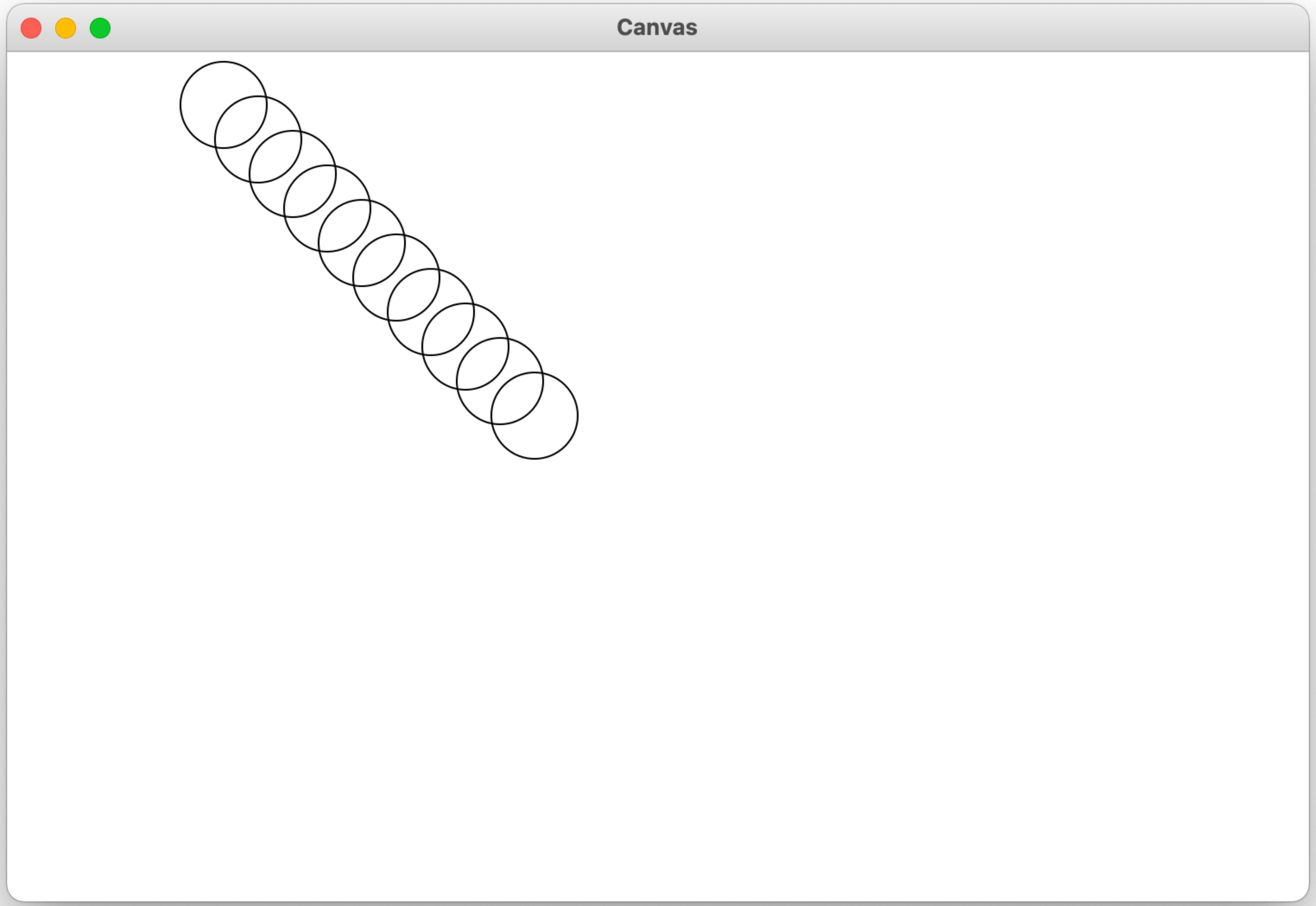


Lecture Plan

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

Practice: Drawing w/ Loops

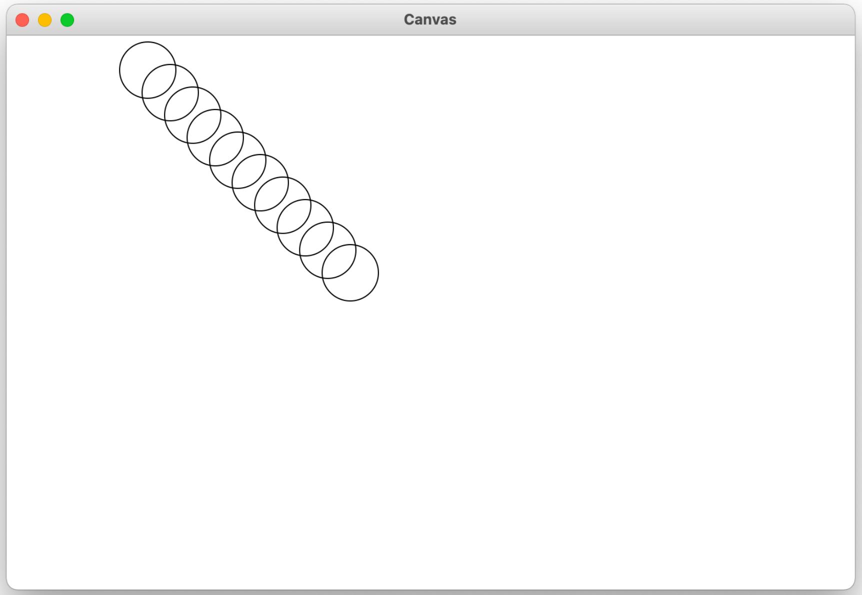
We can combine loops and graphics to draw cool patterns:



Practice: Drawing w/ Loops

We can combine loops and graphics to draw cool patterns:

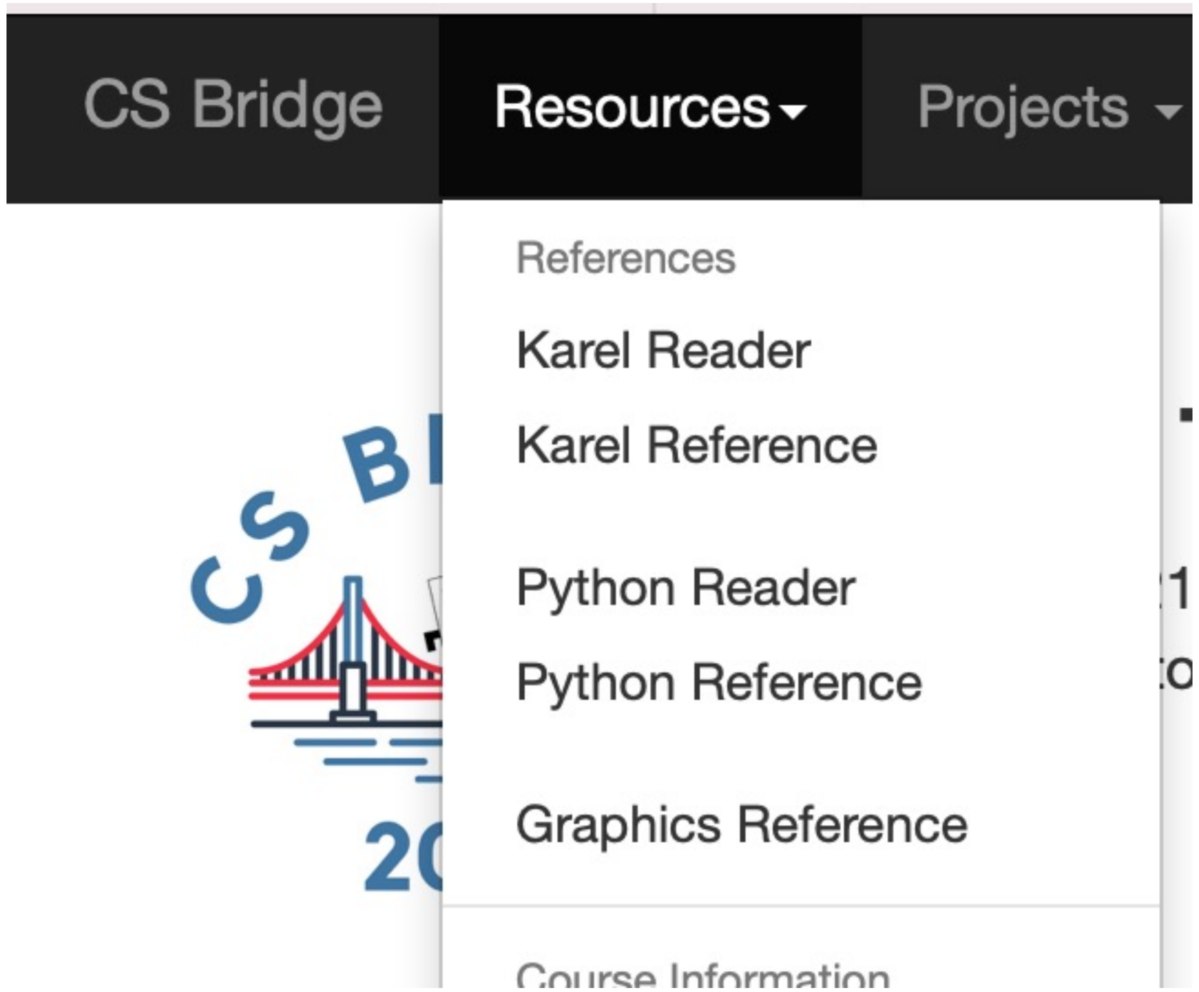
```
def main():
    canvas = Canvas()
    for i in range(10):
        circle_x = 100 + 20 * i
        circle_y = 5 + 20 * i
        canvas.create_oval(circle_x, circle_y, circle_x + 50, circle_y + 50)
    canvas.mainloop()
```



Lecture Recap

- **Review:** Python So Far
- Graphics Programs
- Practice: Centering Objects
- Practice: Drawing a Car
- Practice: Graphics and Loops

Graphics Resources


CS Bridge

Resources

Projects

References

Karel Reader

Karel Reference

Python Reader

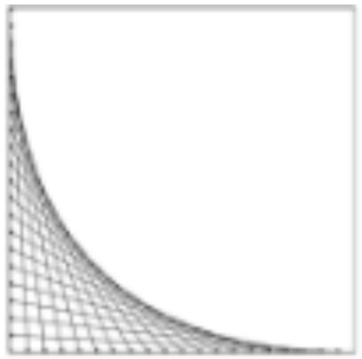
Python Reference

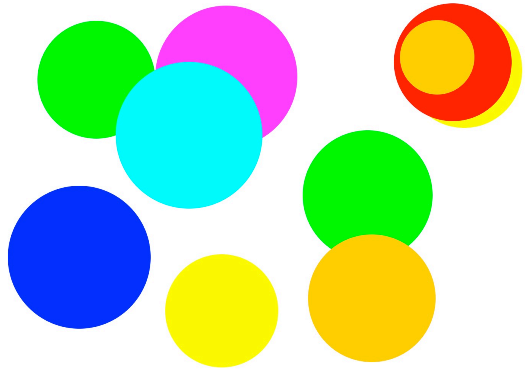
Graphics Reference

Course Information

Rest Of Today

- **Quickstart:** Make your own art!
- **Section:** Combine loops and graphics to make beautiful patterns
- **Project:** Use randomness to draw colorful circle art that is different each time





What’s Next?

- Time for your section’s quickstart time!
- Check your section’s Ed group for more information