



Graphical User Interface

Today's questions

How can we create graphical
programs?

Today's topics

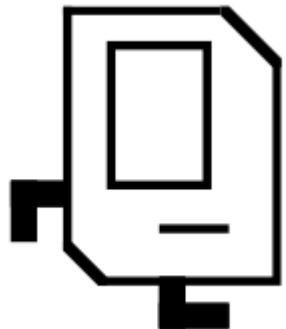
1. Review
2. Graphics

Review

Meet Karel the Robot!



Karel Knows 4 Commands



move

turn_left

put_beeper

pick_beeper

“functions”

For Loops

Instead, use a **for** loop:

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

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

Control Flow: While Loops

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

Repeats the statements in the body until *condition* is no longer true.
Each time, Karel executes *all statements*, and **then** checks the condition.

Possible Questions

front_is_clear()	front_is_blocked()
beepers_present()	no_beeper_present()
beepers_in_bag()	no_beeper_in_bag()
left_is_clear()	left_is_blocked()
right_is_clear()	right_is_blocked()
facing_north()	not_facing_north()
facing_south()	not_facing_south()
facing_east()	not_facing_east()
facing_west()	not_facing_west()

This is taken from the [Karel Reference](#).

If/ Else Statements

What if we want to do one thing if some condition is true, and another otherwise? We can add an **else** statement:

```
if condition:  
    statement  
    statement  
    ...  
elif condition:  
    statement  
    statement  
else:  
    statement  
    statement  
    ...
```

Top-Down Design - Decomposition

- Start from a large task and break it up into smaller pieces
- Ok to write your program in terms of commands that don't exist yet
- **Goal:** make our programs easily readable by humans
 - Commenting
 - Decomposition
- E.g. Going to the moon

Using Python

- **Python must be installed and configured prior to use**
 - One of the items installed is the Python interpreter
- **Python interpreter can be used in two modes:**
 - Interactive mode: enter statements on keyboard
 - Script mode: save statements in Python script

Create, Modify, Use

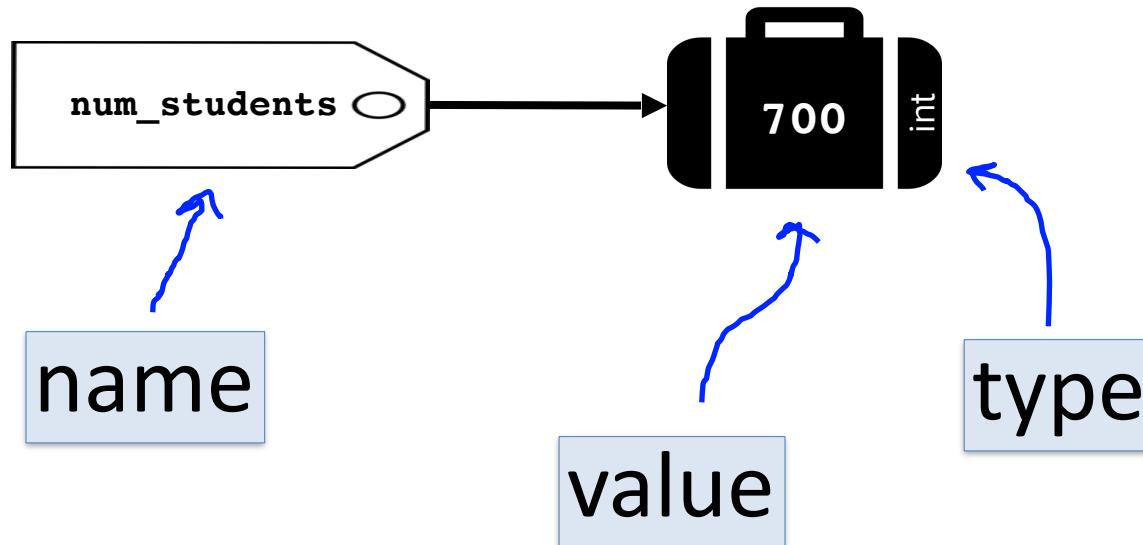
```
# Create a variable, of type int
# called age with the value 30.
age = int(input('What is your age: '))

# Modify age to be one greater.
age = age + 1

# Use the value in age (output it)
print("age is: " + age)
```

Variable Analogy

```
num_students = 700
```



input function

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

- **input** command gets text input from the user
- Prints text specified in double/single quotes
 - Then waits for user input
 - Here, user input from **input** is put in a variable (**num1**)
 - The user input is considered text, even if user entered a number

Types

All Python objects have a type!

- Integers - numbers with no decimals

```
num_flowers = 5
```

- Floats - numbers with decimals

```
fraction = 0.2
```

- Booleans - true or false

```
is_raining_today = True
```

- Strings - collection of characters

```
myName = 'Ayca'
```

Binary Operators

+ Addition

- Subtraction

* Multiplication

/ Division

There are others too!

% Remainder // Int Division

** Exponentiation

Comparison Operators

Operator	Meaning	Example	Value
<code>==</code>	equals	<code>1 + 1 == 2</code>	True
<code>!=</code>	does not equal	<code>3.2 != 2.5</code>	True
<code><</code>	less than	<code>10 < 5</code>	False
<code>></code>	greater than	<code>10 > 5</code>	True
<code><=</code>	less than or equal to	<code>126 <= 100</code>	False
<code>>=</code>	greater than or equal to	<code>5.0 >= 5.0</code>	True

* All have equal precedence

Random Integers

```
secret_number = random.randint(1, 99)
```

Minimum value

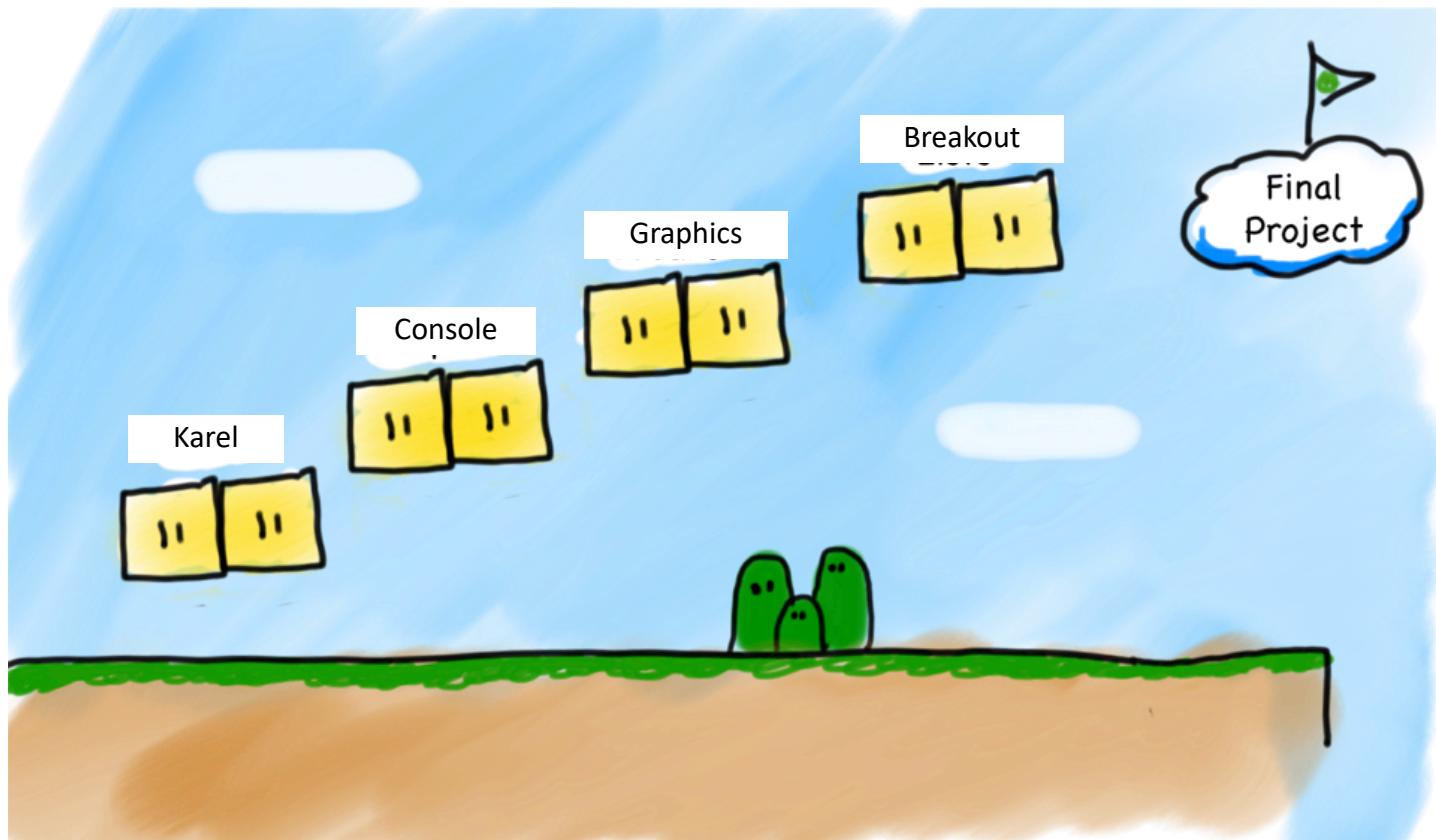


Maximum value

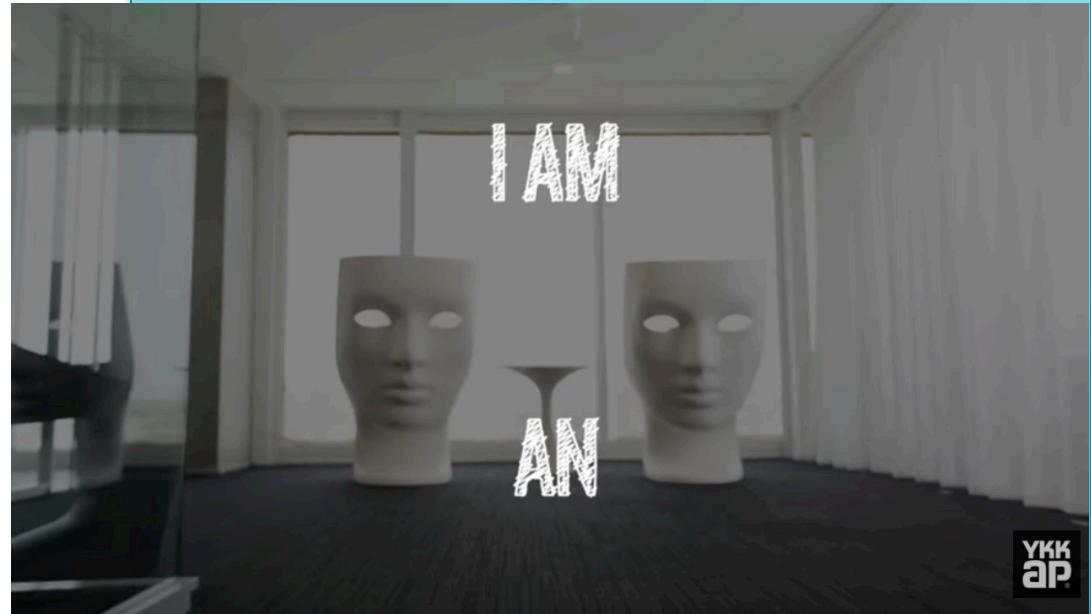


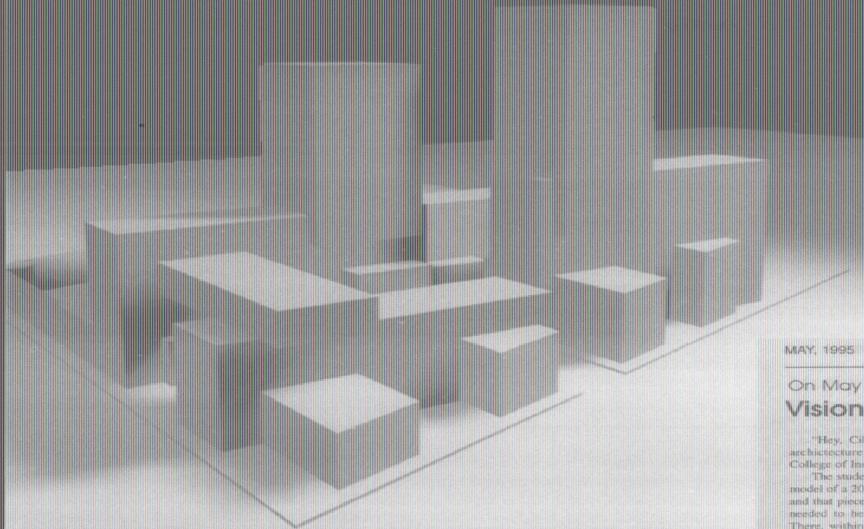
Minimum and max values are includes

What have we learned so far?



Fun fact!





MAY, 1995

On May 3, Student Architects Present Designs for Visions of Housing and a "Y" on

"Hey, Cihan," called a fellow architecture student. "Is the College of Insurance around here?"

The students were readying the model of a 20-block area of Tribeca and that piece, freshly painted, was indeed the missing component. There, within the six-by-eight-foot model stood miniature versions of P.S. 234, Stryvencs High School and the bridge leading to it, Borough of Manhattan Community College, Washington Market Park, the nearby apartment houses along Greenwich Street, and buildings across the street's 310 Greenwich St., which, as in real life, stood awkwardly above it all.

What remained empty were the spaces surrounding P.S. 234. Once at a time, the men and women, third-year graduate architecture students at the New Jersey Institute of Technology in Newark, filled those lots with their own, widely varying models of housing and (much-needed) indoor recreation facility. Each had his or her own concept of how these structures could be integrated into the city-owned lots — the largest remaining, vacant property and the site of past battles over the building of a competing home for the Commodities Exchange.

The assignment is only partly



At the New Jersey Institute of Technology, a housing and a "Y" on the empty lots next to Professor Tribeca architect Julian Weiss.

The model and designs by these architects, students provide, for the first time, an opportunity for Tribeca residents to visualize how the neighborhood might look if that trade off takes place. It also underscores the belief held by many in the community that a "Y"-type indoor recreation facility is an essential part of new residential construction.

The students will present their designs at 6 p.m. May 3 in the P.S. 234 auditorium.

A massive residential complex was designed for downtown Manhattan. The complex included a "Y"-type indoor recreation facility.

The proposed design is community and has been Trip on May 1995.



How can we generate graphical programs?

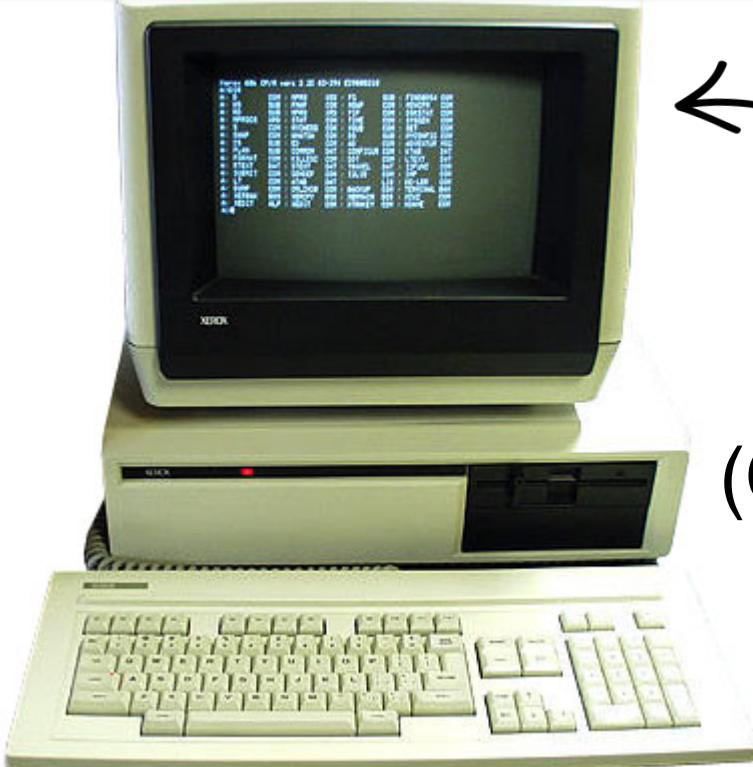
GUI libraries
(modules)!

Definition

Graphical User Interface (GUI)

Visual elements that you interact with within applications (windows, buttons, scroll bars, etc.)

Console - Command Line



No GUI! All text-based

(Old computers used to only have a command line!)

Gui - Event Driven Program

- In text-based environments, programs determine the order in which things happen
 - The user can only enter data in the order requested by the program
- GUI environment is event-driven
 - The user determines the order in which things happen
 - User causes events to take place and the program responds to the events

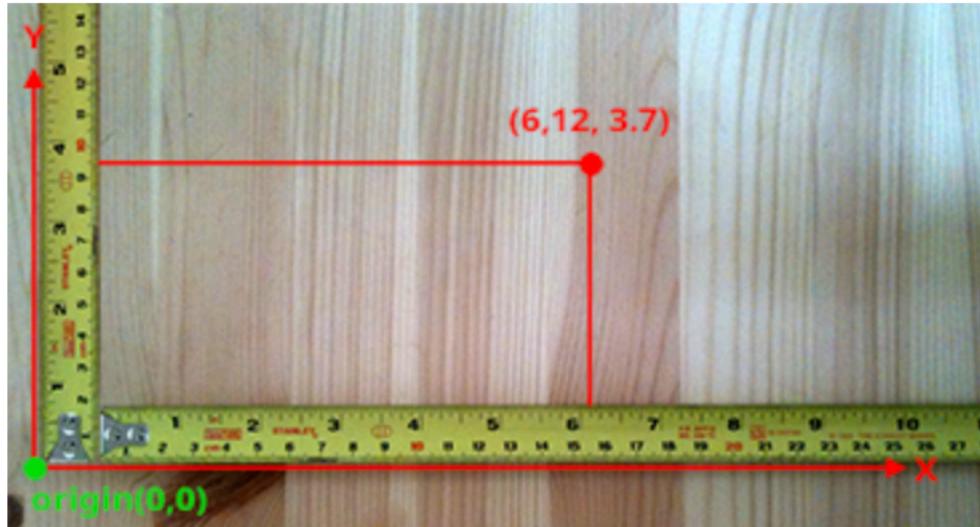
How does computer
graphics get created?

Gui - Event Driven Program

- Windows, Mac OS, Linux each have their own GUI systems
- **GUI libraries** are modules that support the GUI (i.e. their functions allow you to interact with your computer's GUI system)
- **tkinter** is Python's standard GUI package
 - Many other GUI libraries are built on top of **tkinter!**

Drawing in Real Life

- Draw objects using Cartesian coordinate system



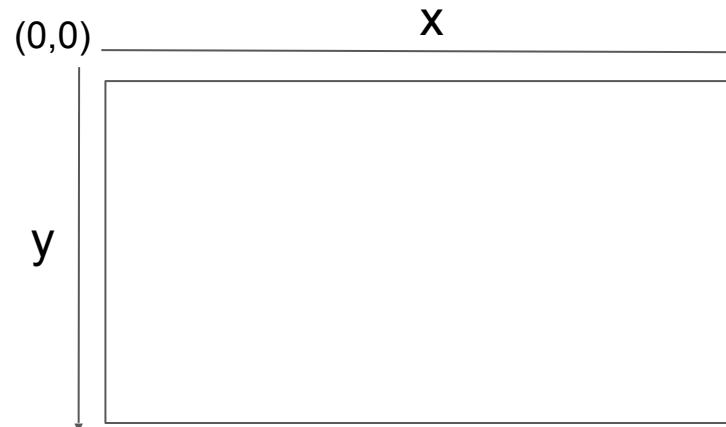
Drawing Computer Graphics

- When creating computer graphics we first somehow describe the shape (and the appearance) of objects
- Place objects on a background called “Canvas”
- Similar to “Felt board” metaphor -
 - The underlying model is similar to that of a collage in which an artist creates a composition by taking various objects and assembling them on a background canvas.



Canvas

- Everything gets drawn on a **canvas**
 - In today's examples, we'll provide the code for creating the canvas.



Canvas

0,0

x 40,20

x 120,40

x 40,120

CANVAS_WIDTH

CANVAS_HEIGHT

Canvas

the following `main` creates a canvas with kitle with default size

```
def main():
    canvas = Canvas()
```

Canvas

the following `main` creates a canvas with title with default size

```
def main():
    canvas = Canvas()
    canvas.set_canvas_title("Programming is Awesome")
```



Canvas

the following `main` creates a canvas with title with default size

```
def main():
    canvas = Canvas(800, 600)
    canvas.set_canvas_title("Programming is Awesome")
```



How do we draw simple shapes?

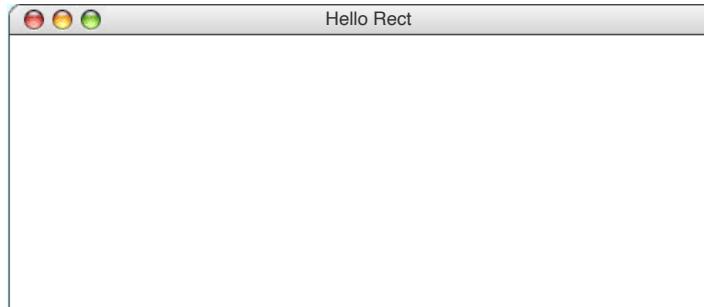
Drawing Simple Shapes

- `canvas.create_text()`
- `canvas.create_line()`
- `canvas.create_rectangle()`
- `canvas.create_oval()`

Draw a Rectangle

the following `main` method displays a blue square

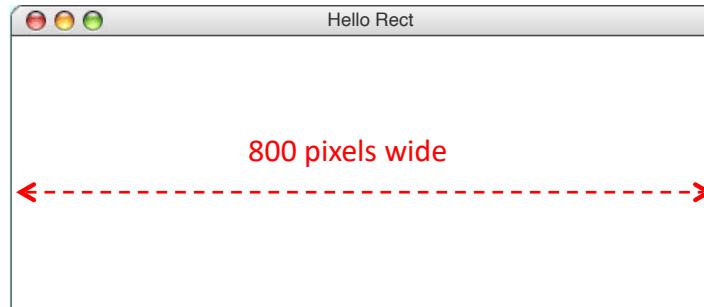
```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
```



Draw a Rectangle

the following `main` method displays a blue square

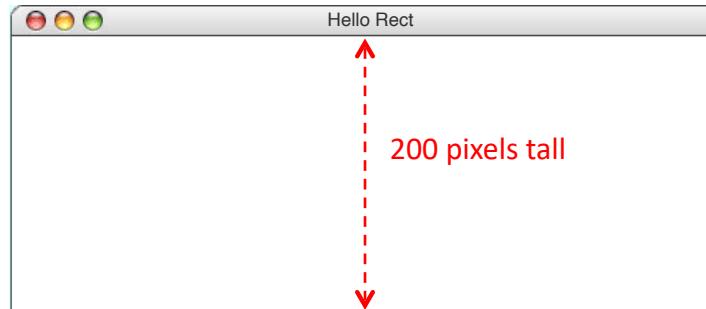
```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
```



Draw a Rectangle

the following `main` method displays a blue square

```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
```



Draw a Rectangle

the following `main` method displays a blue square

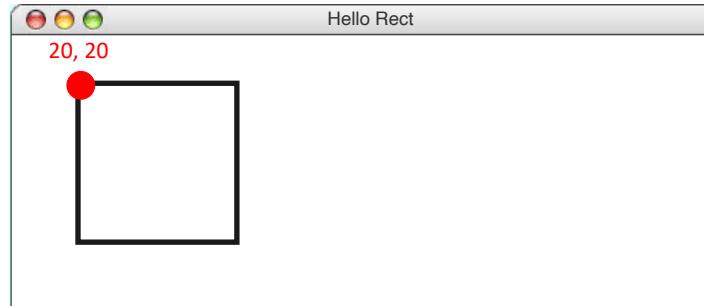
```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
```



Draw a Rectangle

the following `main` method displays a blue square

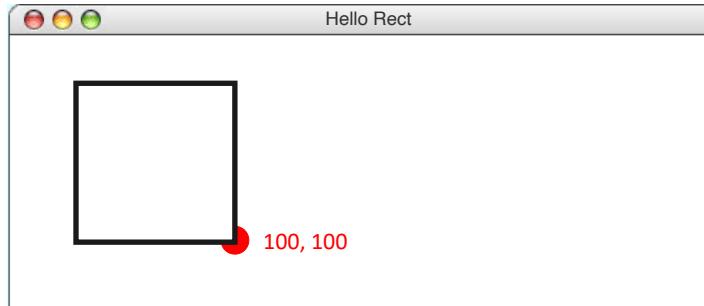
```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
    canvas.create_rectangle(20, 20, 100, 100)
```



Draw a Rectangle

the following `main` method displays a blue square

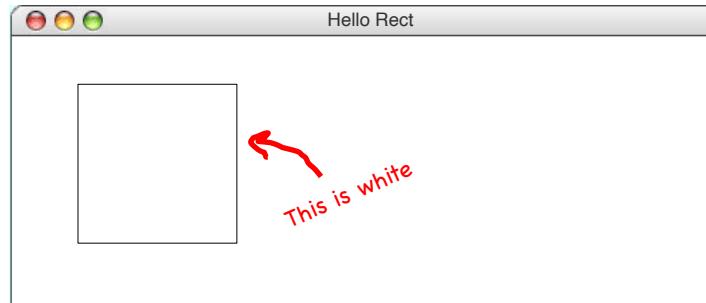
```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
    canvas.create_rectangle(20, 20, 100, 100)
```



Draw a Rectangle

the following `main` method displays a blue square

```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
    canvas.create_rectangle(20, 20, 100, 100)
```

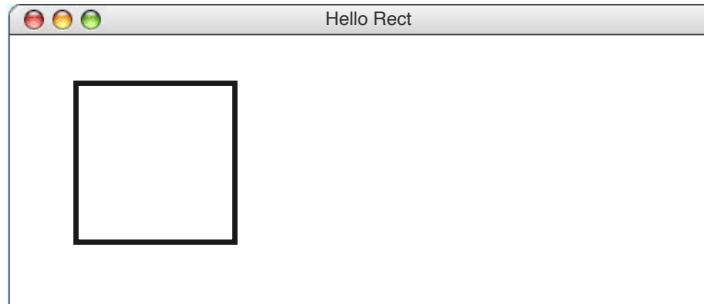


Aside: Named Arguments
This argument is named as filled. It allows functions to have arguments which you can ignore if you want a default value.

Draw a Rectangle

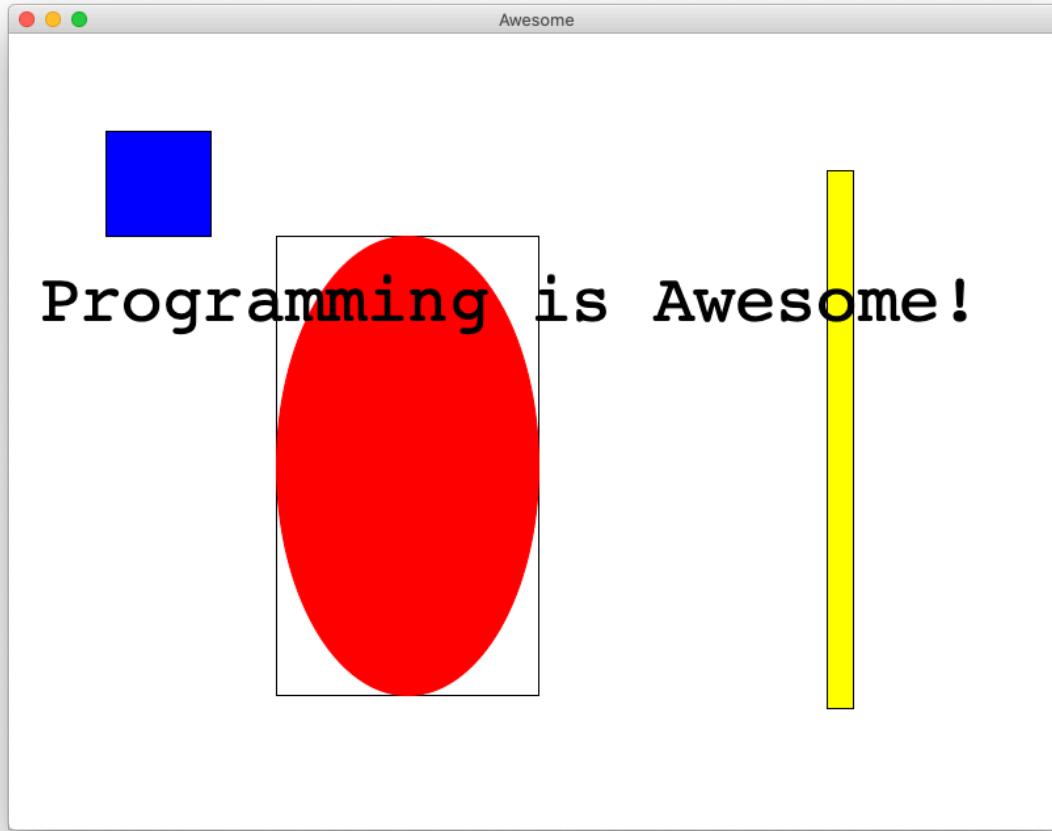
the following `main` method displays a blue square

```
def main():
    canvas = Canvas(800, 200)
    canvas.set_canvas_title("Hello Rect")
    canvas.create_rectangle(20, 20, 100, 100)
    canvas.mainloop()
```



*Mainloop is a method
will loop forever, waiting
for events from the user,
until the user exits the
program*

Rectangle, Oval and Text



Drawing Line

- `canvas.create_line(x1, y1, x2, y2)`

Drawing Line

- `canvas.create_line(x1, y1, x2, y2)`



Drawing Oval

- `canvas.create_oval(x1, y1, x2, y2)`



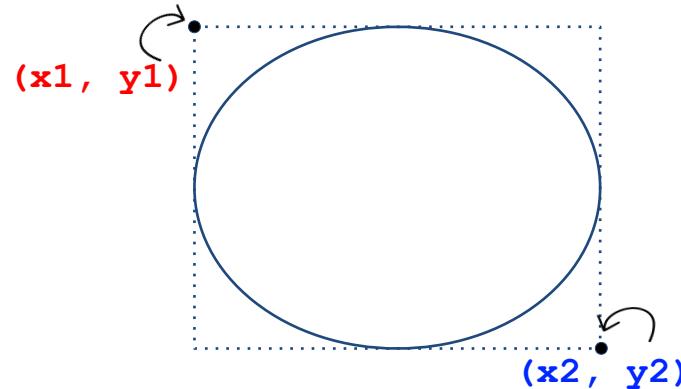
Drawing Line

- `canvas.create_line(x1, y1, x2, y2)`



Drawing Oval

- `canvas.create_oval(x1, y1, x2, y2)`



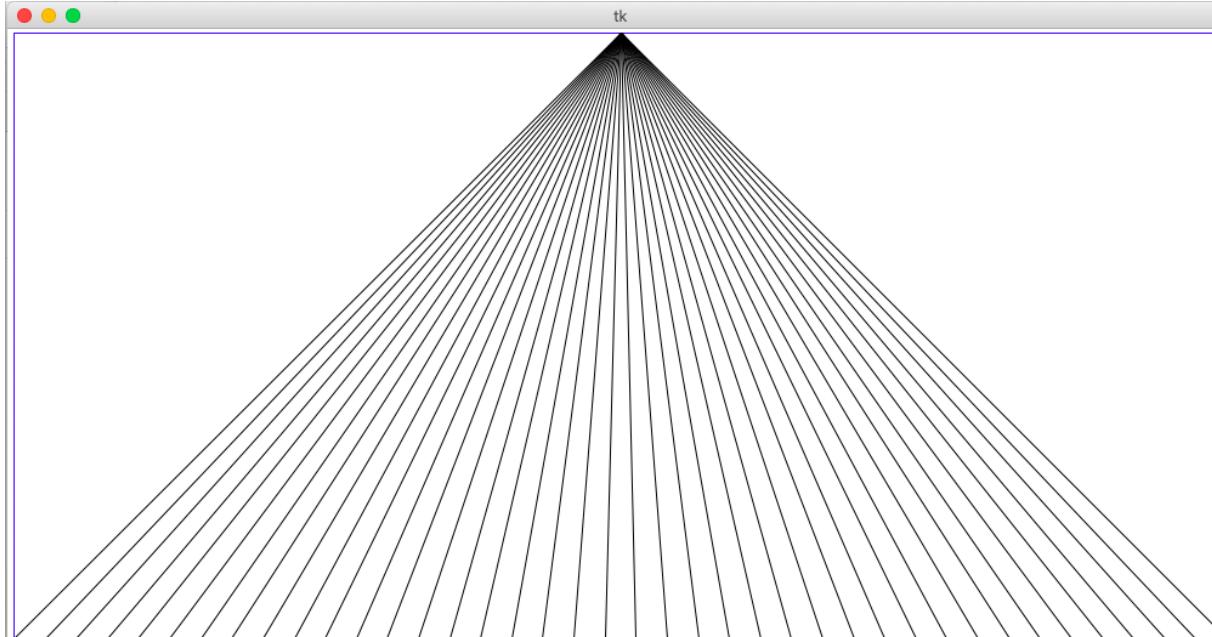
Drawing Text

- `canvas.create_text(x, y, 'hi')`



Proportional math:
`draw_pyramid()`

draw_pyramid()



[demo]

Drawing Image

- `canvas.create_image(x, y, "name of the file")`



Get Canvas Information

```
def main():

    canvas = Canvas(800, 200)
    canvas.set_title('My Program')
```

Get Canvas Information

```
def main():

    canvas = Canvas(800, 200)
    canvas.set_title('My Program')

    canvas_width = canvas.get_canvas_width()

    canvas_height = canvas.get_canvas_height()
```

Get Canvas Information

```
def main():

    canvas = Canvas(800, 200)
    canvas.set_title('My Program')

    canvas_width = canvas.get_canvas_width()

    canvas_height = canvas.get_canvas_height()

    #create centered text
    Label = canvas.create_text(canvas_width /2,
    canvas_height/2, 'I am centered')
```

Get Object Information

```
def main():

    canvas = Canvas(800, 200)
    rect = canvas.create_rectangle(5, 10, 50, 100)
    print(canvas.get_width(rect) #print 50
    print(canvas.get_height(rect) #print 100
    print(canvas.get_left_x(rect) #print 5
    print(canvas.get_top_y(rect) #print 10
```

Update Object Information

```
def main():
    canvas = Canvas(400, 200)
    canvas.set_title('My Program')
    rect = canvas.create_rectangle(5, 10, 50, 100)
```



Update Object Information

```
def main():

    canvas = Canvas(400,200)
    canvas.set_title('My Program')
    rect = canvas.create_rectangle(5, 10, 50, 100)
    canvas.set_outline_color(rect, 'red')
    canvas.set_fill_color(rect, 'yellow')
```



What is next?

Day 5: Graphics and Functions

Morning Project [[here](#)]

	Programming is Awesome	Graphics
	Quickstart	
	SL Notes	
	Random Circles	Graphics and Randomness
	Project	
	SL Notes	

Resources ▾

Project ▾

- Karel Reader
- Karel Reference
- Python Reader
- Python Reference
- Graphics Reference**
- Course Information
- General Info
- [Student] Course FAQs
- Section Info

- Install PyCharm
- How to use PyCharm

- Submission/Office Hours