



# Functions

Chris Piech and Mehran Sahami  
CS106A, Stanford University

# Learn How To:

1. Write a function that takes in input
2. Write a function that gives back output



# Calling functions

`turn_right()`

`move()`

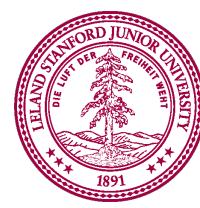
`input("string please! ")`

`print("hello world")`

`float("0.42")`

`canvas.create_rectangle(0, 0, 100, 100, "red")`

`math.sqrt(25)`



# Calling functions

`turn_right()`

`move()`

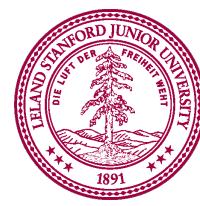
`result = input("string please!")`

`print("hello world")`

`float("0.42")`

`canvas.create_rectangle(0, 0, 100, 100, "red")`

`result = math.sqrt(25)`



# Defining a function

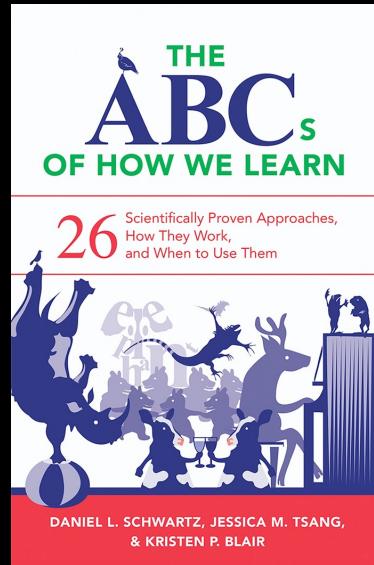
```
def turn_right():
    turn_left()
    turn_left()
    turn_left()
```



Big difference with python functions:  
Python functions can **take in data**, and can **return data!**



# Contrasting Case



*Thanks Dan Schwartz*

# Defining a function

```
def turn_right():
    turn_left()
    turn_left()
    turn_left()
```

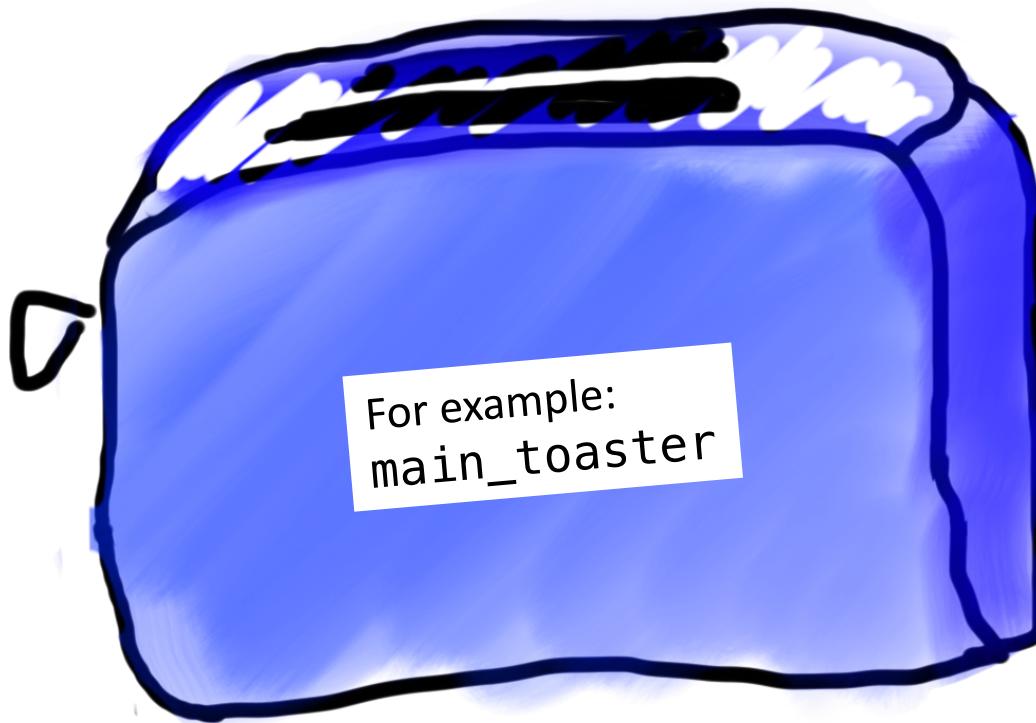
```
def move_n(n_moves):
    for i in range(n_moves):
        move()
```

```
def main():
    turn_right()
    move_n(10)
```

Big difference with python functions:  
Python functions can **take in data**, and can **return data!**



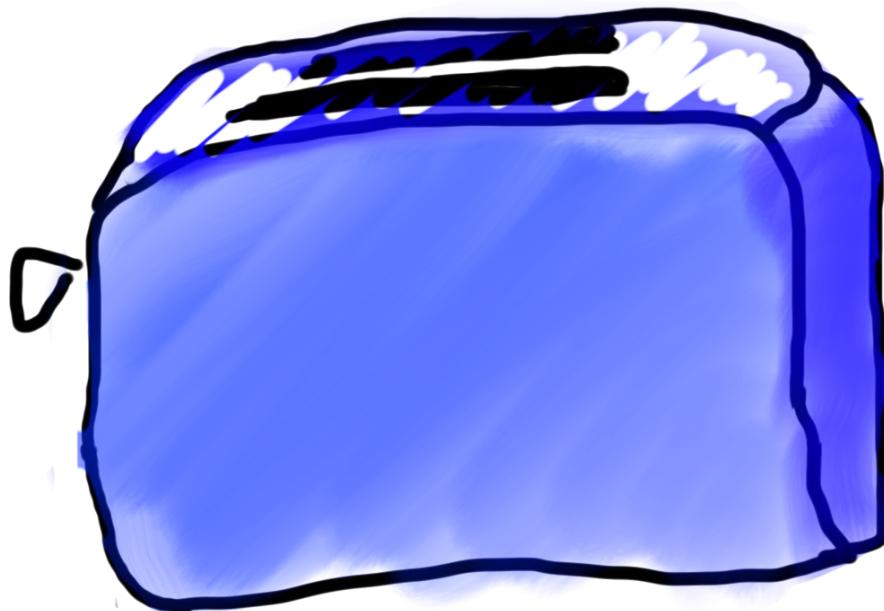
# Toasters are functions



# Toasters are functions



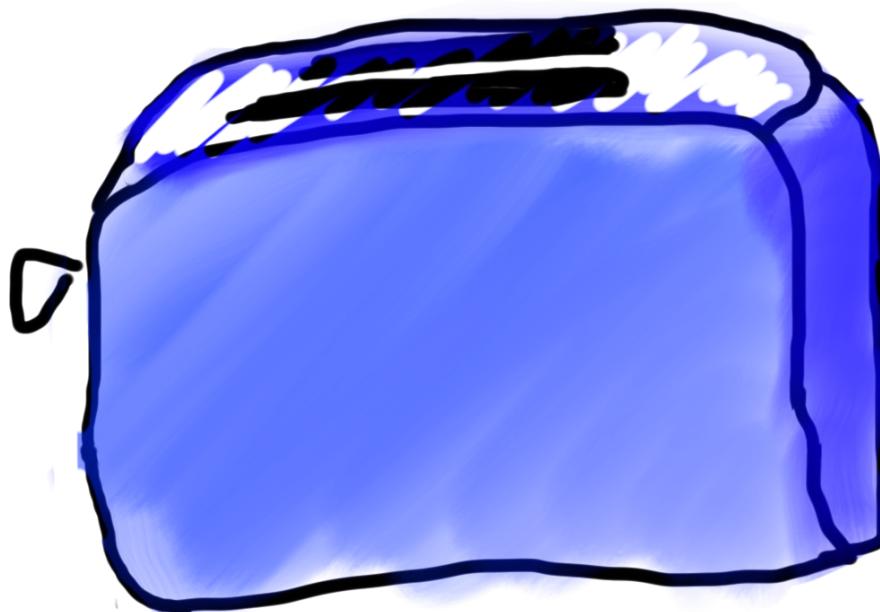
parameter



# Toasters are functions



parameter



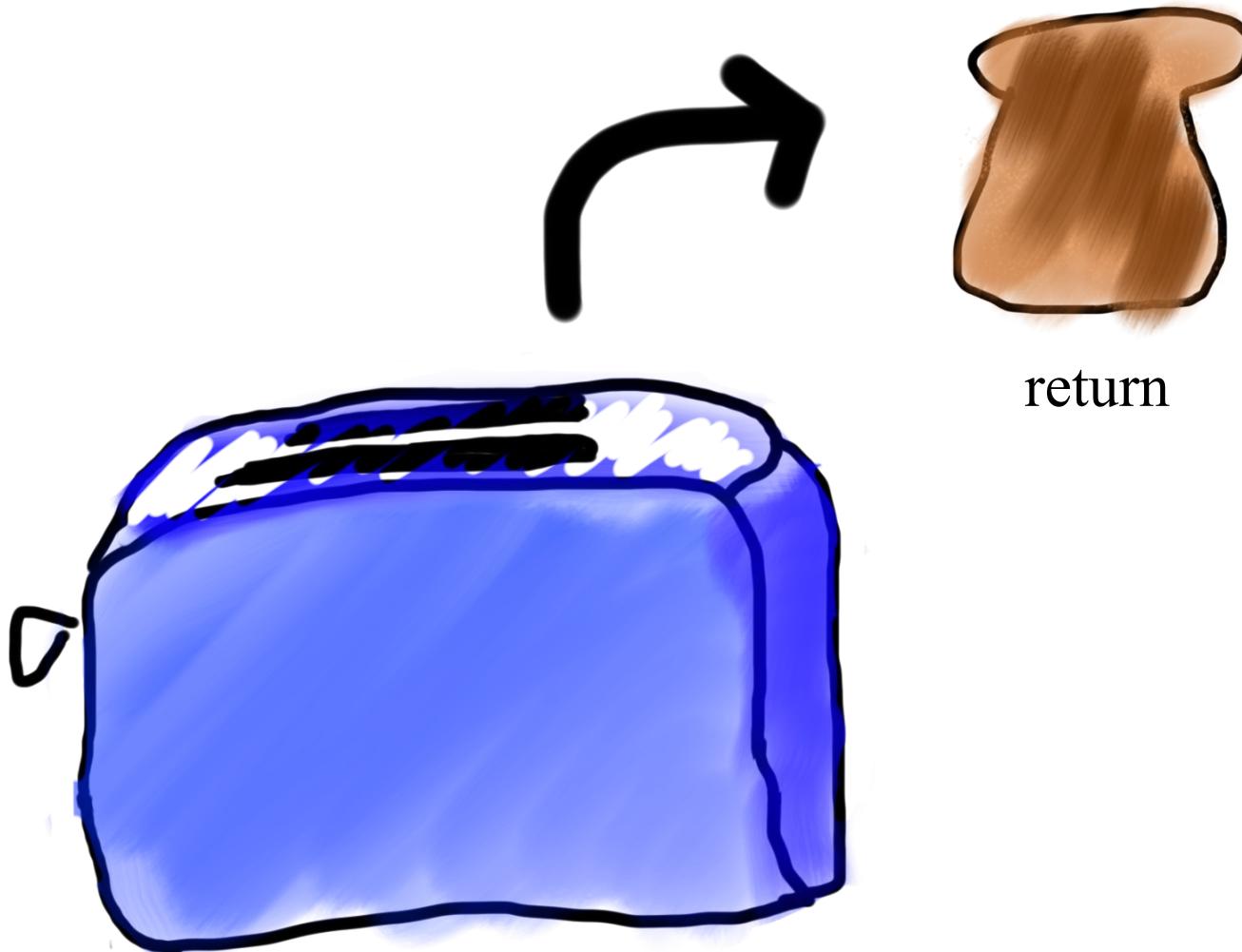
# Toasters are functions



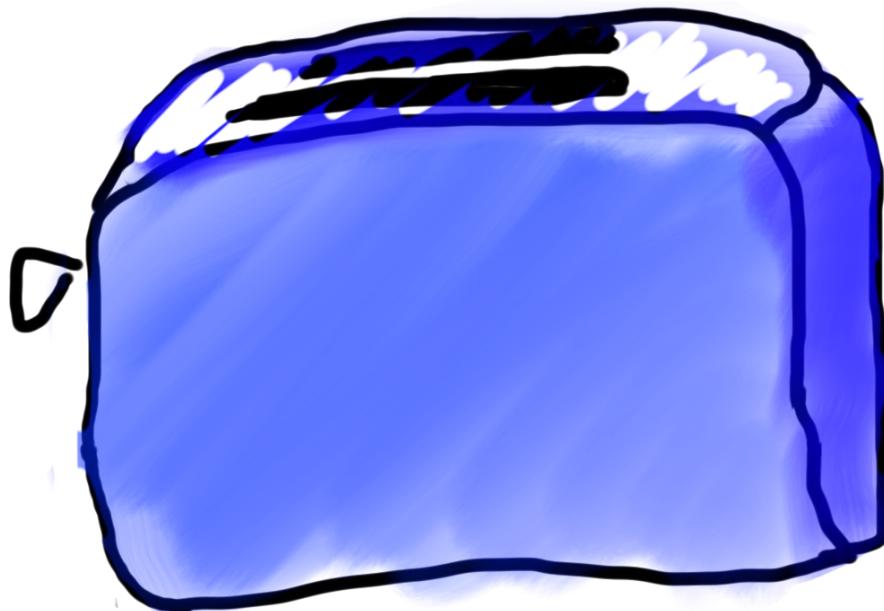
# Toasters are functions



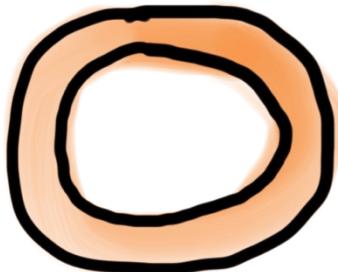
# Toasters are functions



# Toasters are functions



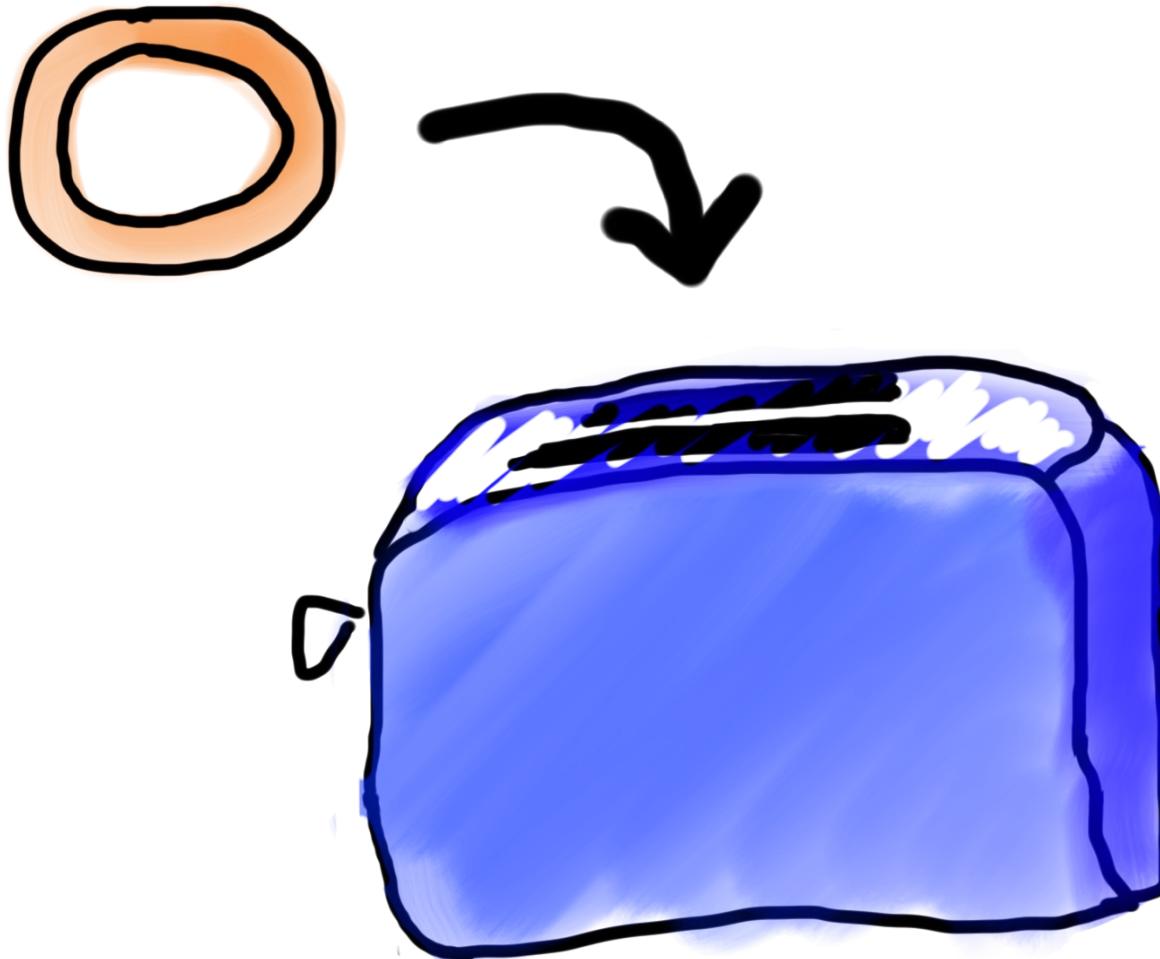
# Toasters are functions



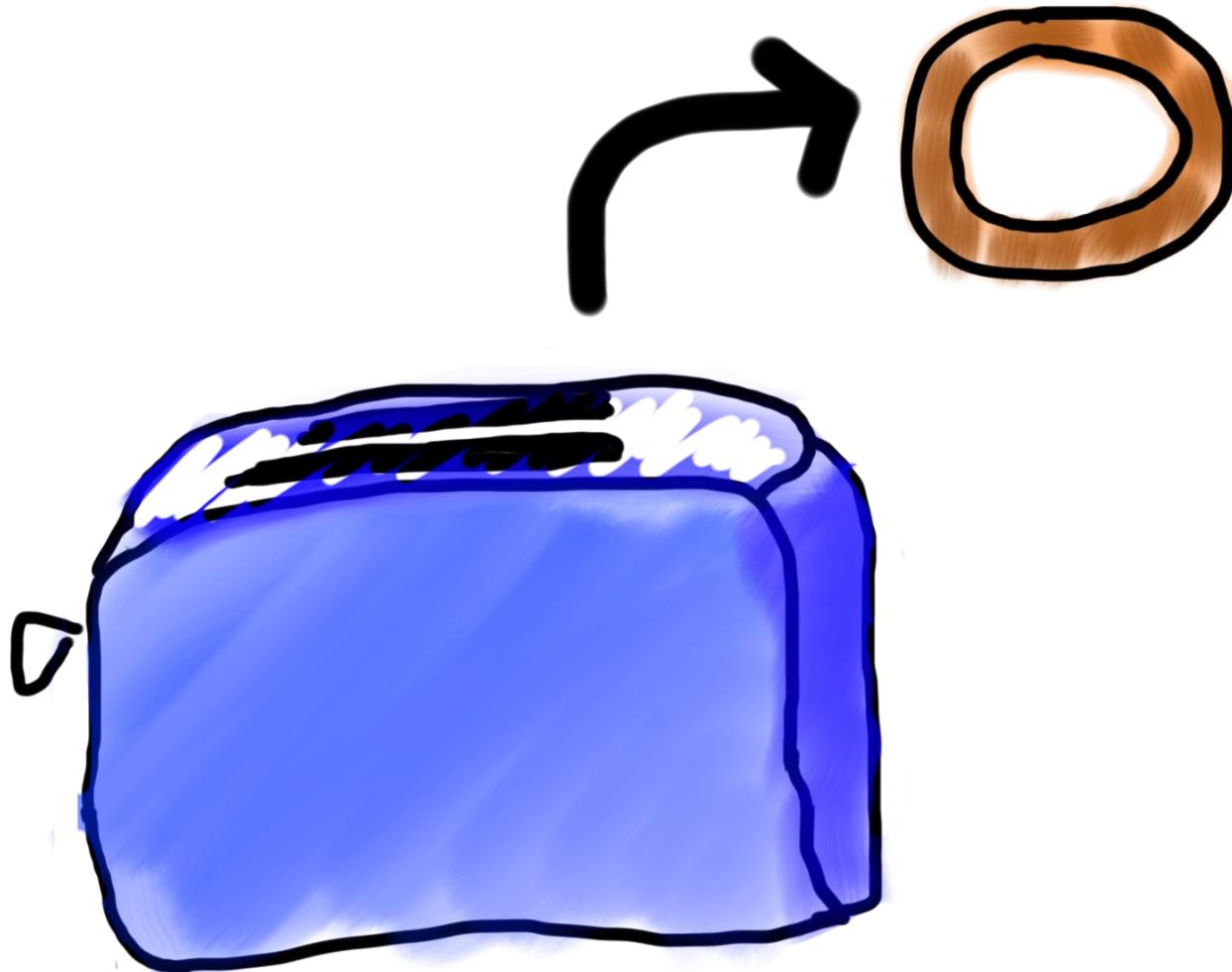
\* You don't need a second toaster if you want to toast bagels. Use the same one.



# Toasters are functions



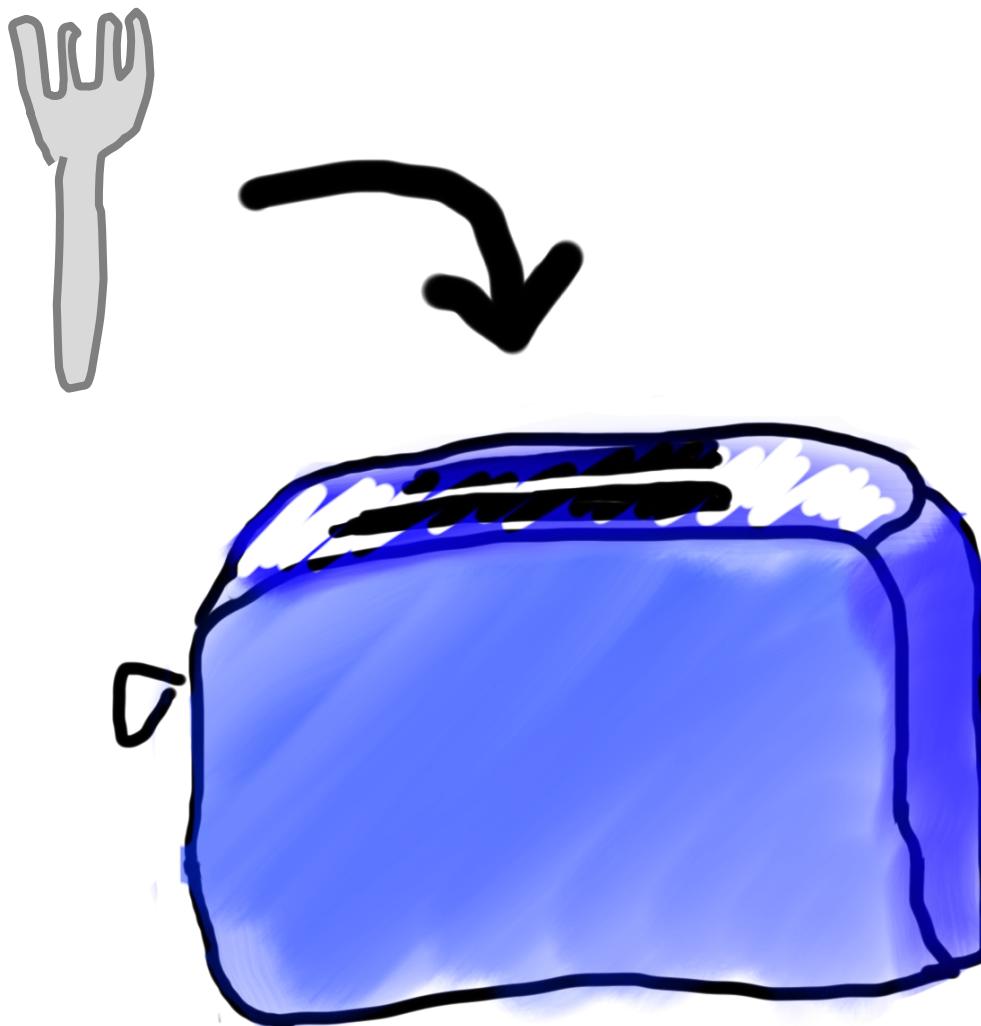
# Toasters are functions



# Toasters are functions



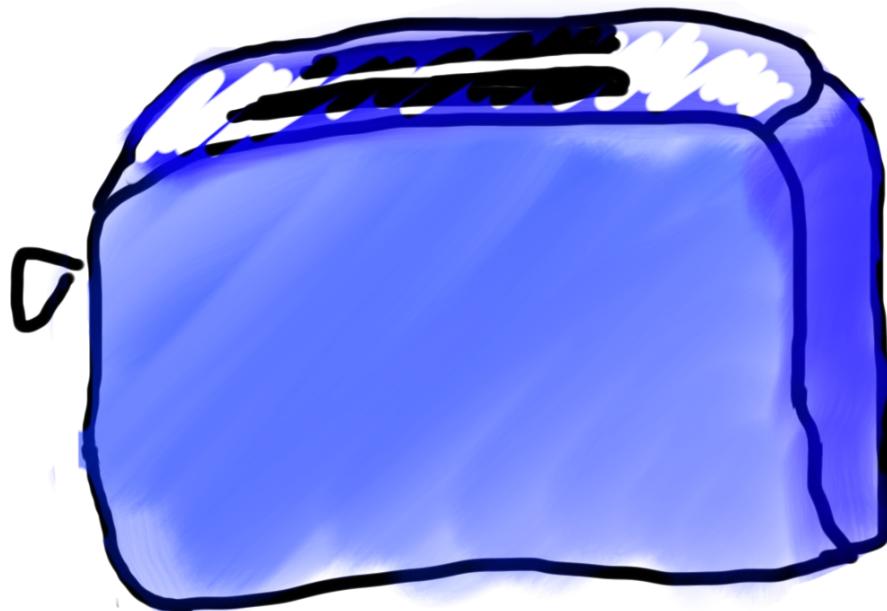
# Toasters are functions



# Toasters are functions



# functions are Like Toasters



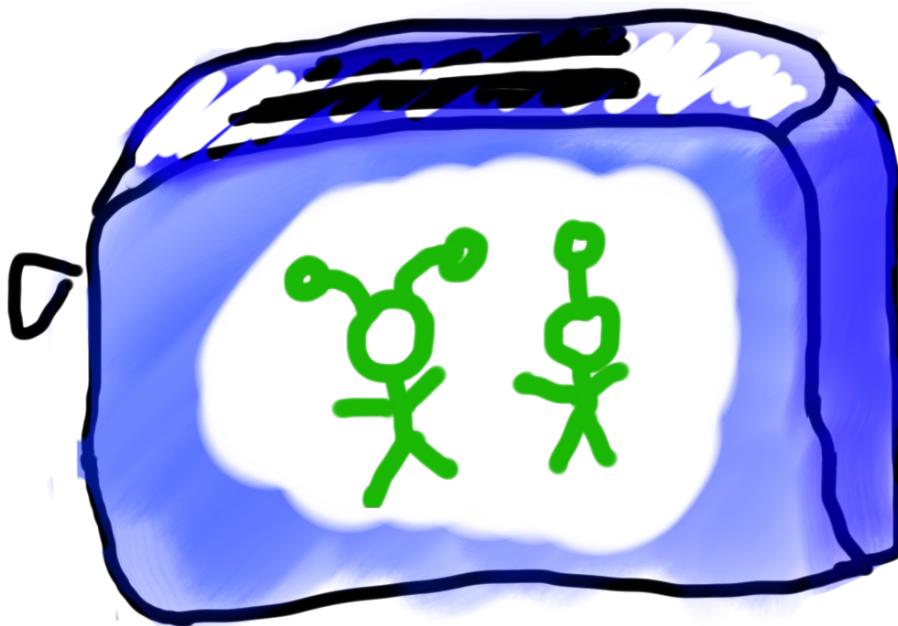
# functions are Like Toasters



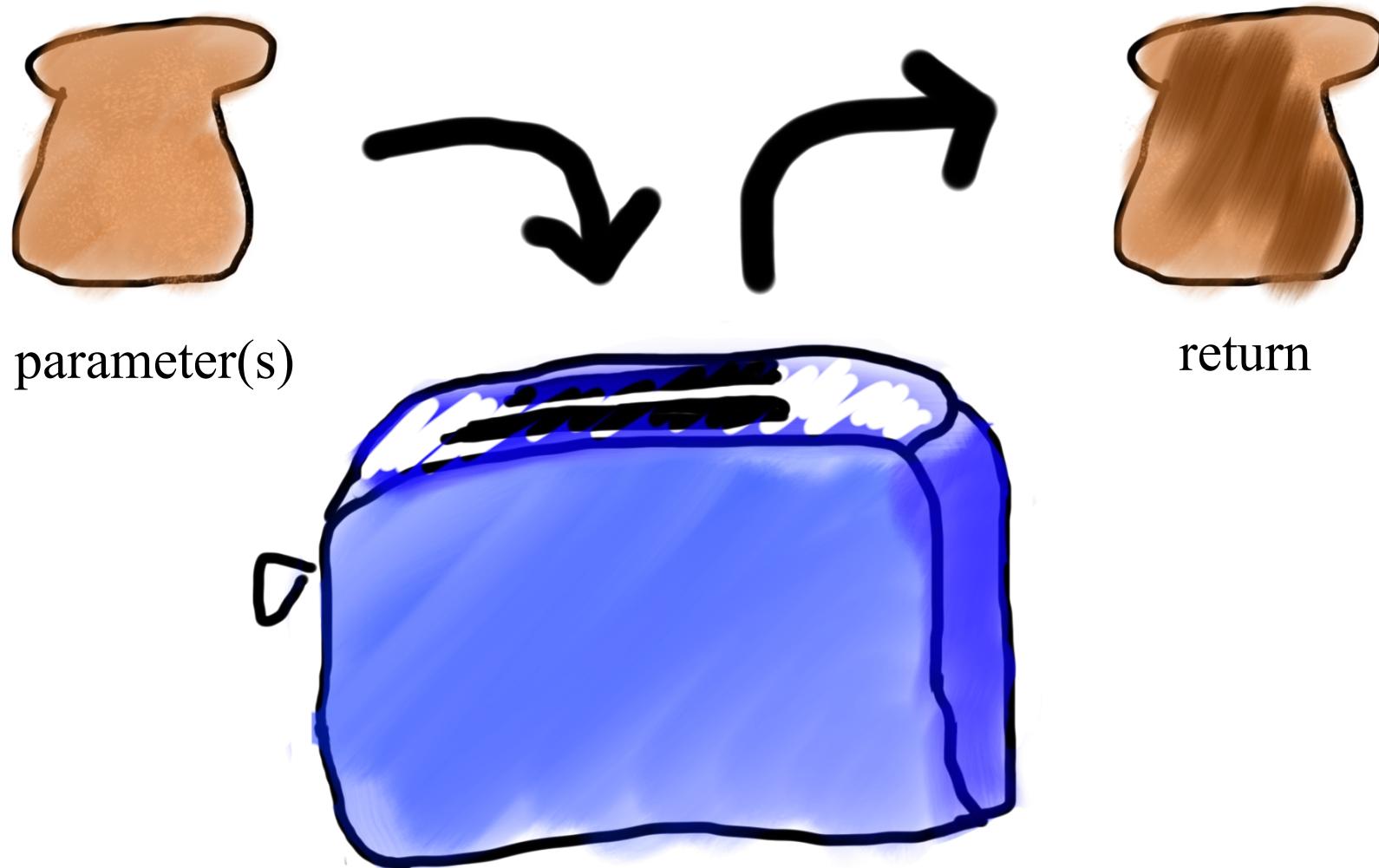
# functions are Like Toasters



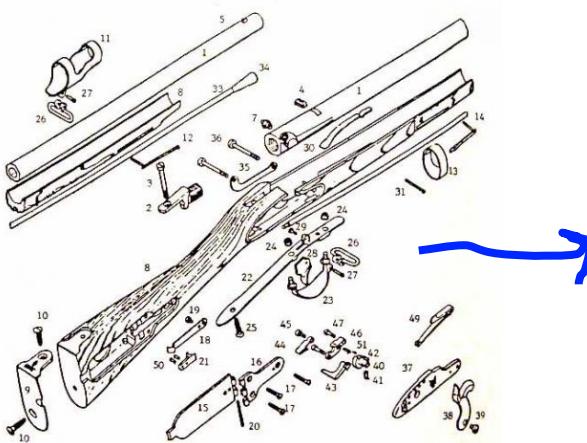
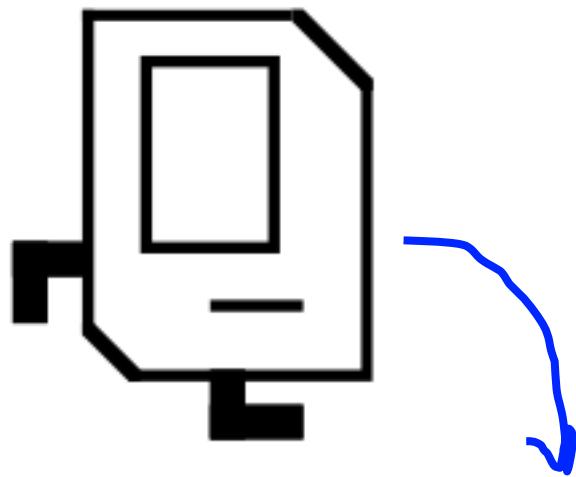
# functions are Like Toasters



# functions are Like Toasters



# Chance Connections



# Classic Challenge for CS106A



Perhaps the  
most  
underrated  
concept by  
students



# Anatomy of a function

```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```

```
def average(a, b):
    sum = a + b
    return sum / 2
```



# Anatomy of a function

```
def main():    function "call"  
    mid = average(5.0, 10.2)  
    print(mid)
```

function "definition"

```
def average(a, b):  
    sum = a + b  
    return sum / 2
```



# Anatomy of a function

```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```

name

```
def average(a, b):
    sum = a + b
    return sum / 2
```



# Anatomy of a function

```
def main():           Input given  
    mid = average(5.0, 10.2)  
    print(mid)
```

Input expected

```
def average(a, b):  
    sum = a + b  
    return sum / 2
```



# Anatomy of a function

```
def main():          Arguments  
    mid = average(5.0, 10.2)  
    print(mid)
```

```
Parameters  
def average(a, b):  
    sum = a + b  
    return sum / 2
```



# Anatomy of a function

```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```

```
def average(a, b):
```

```
    sum = a + b
    return sum / 2
```

body

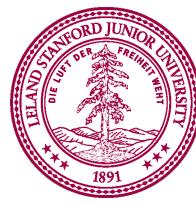


# Anatomy of a function

```
def main():    This call “evaluates” to the value returned  
    mid = average(5.0, 10.2)  
    print(mid)
```

```
def average(a, b):  
    sum = a + b  
    return sum / 2
```

Ends the function and gives back a value



# Anatomy of a function

Also a function call

```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```

```
def average(a, b):
    sum = a + b
    return sum / 2
```



# Anatomy of a function

No parameters (expects no input)

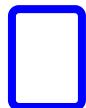
```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```

```
def average(a, b):
    sum = a + b
    return sum / 2
```



# Anatomy of a function

```
def main():
    mid = average(5.0, 10.2)
    print(mid)
```



When a function ends it “returns”

```
def average(a, b):
    sum = a + b
    return sum / 2
```



# Call Functions Many Times

```
def main():
    avg_1 = average(0, 10)
    avg_2 = average(8, 10)

    final = average(avg_1, avg_2)
    print("avg_1", avg_1)
    print("avg_2", avg_2)
    print("final", final)

def average(a, b):
    """
    returns the number which is half way between a and b
    """
    sum = a + b
    return sum / 2

if __name__ == '__main__':
    main()
```



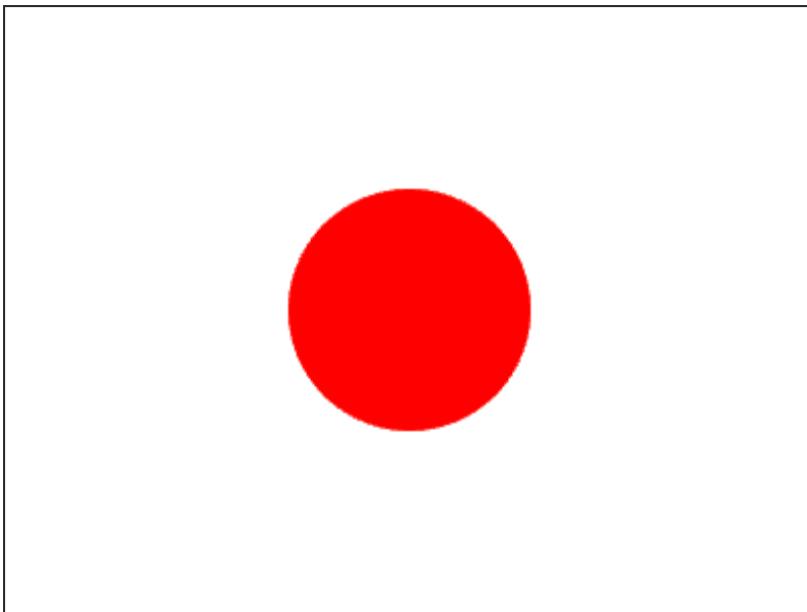
# Parameters



Parameters let you provide a function some information when you are calling it.

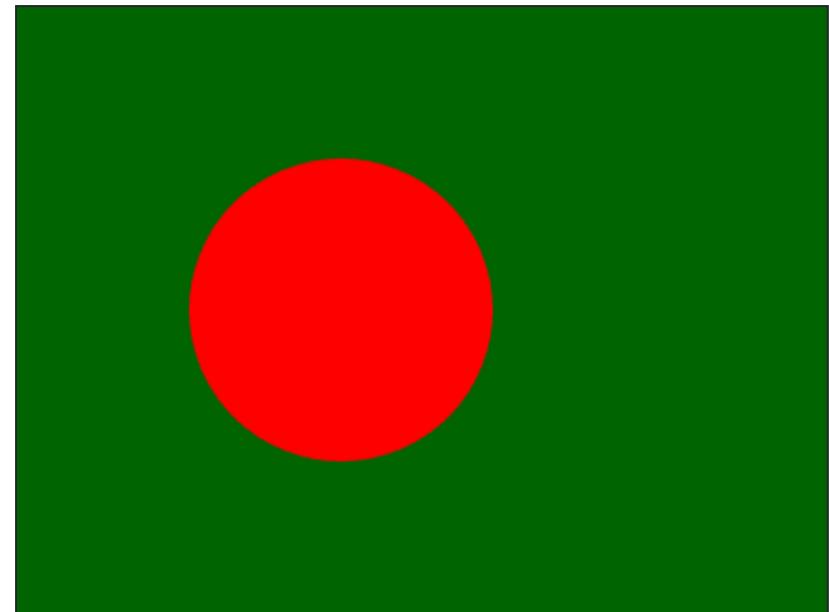


# What Function Do you Want?



Same:

- Both are circles
- Same color



Different:

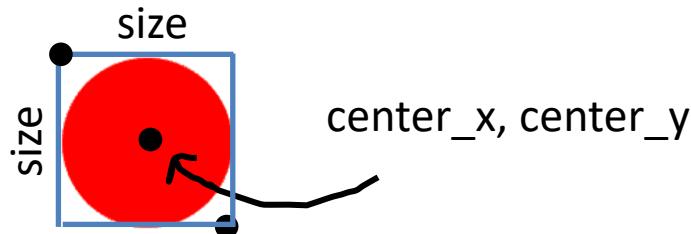
- Position
- Size



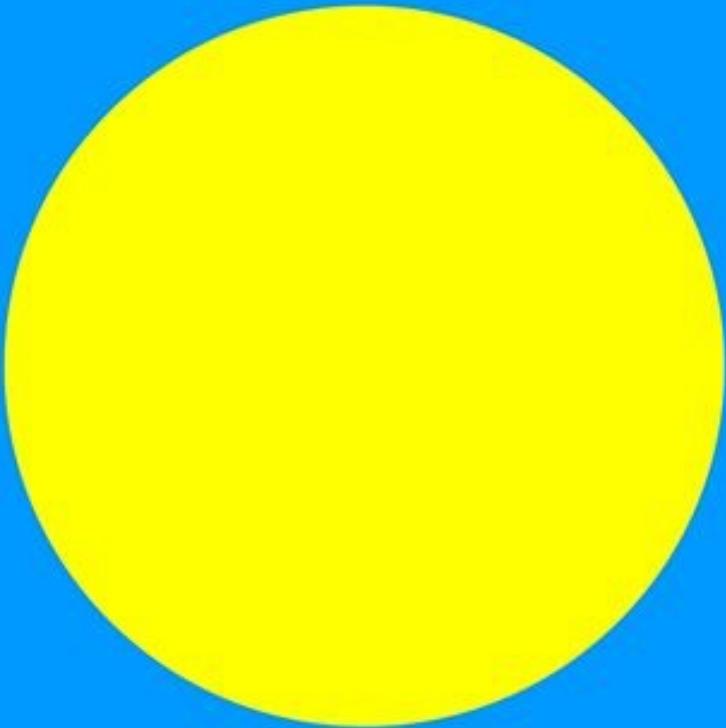
# What Function Do you Want?

```
def draw_red_circle(canvas, center_x, center_y, size):
    """
    Draws a circle on the given canvas. The circle will have its center
    at center_x, center_y. It will be size pixels tall and size pixels wide.
    """

    left_x = center_x - size/2
    top_y = center_y - size/2
    right_x = left_x + size
    bottom_y = top_y + size
    canvas.create_oval(left_x, top_y, right_x, bottom_y, 'red')
```



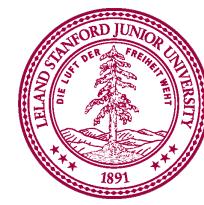
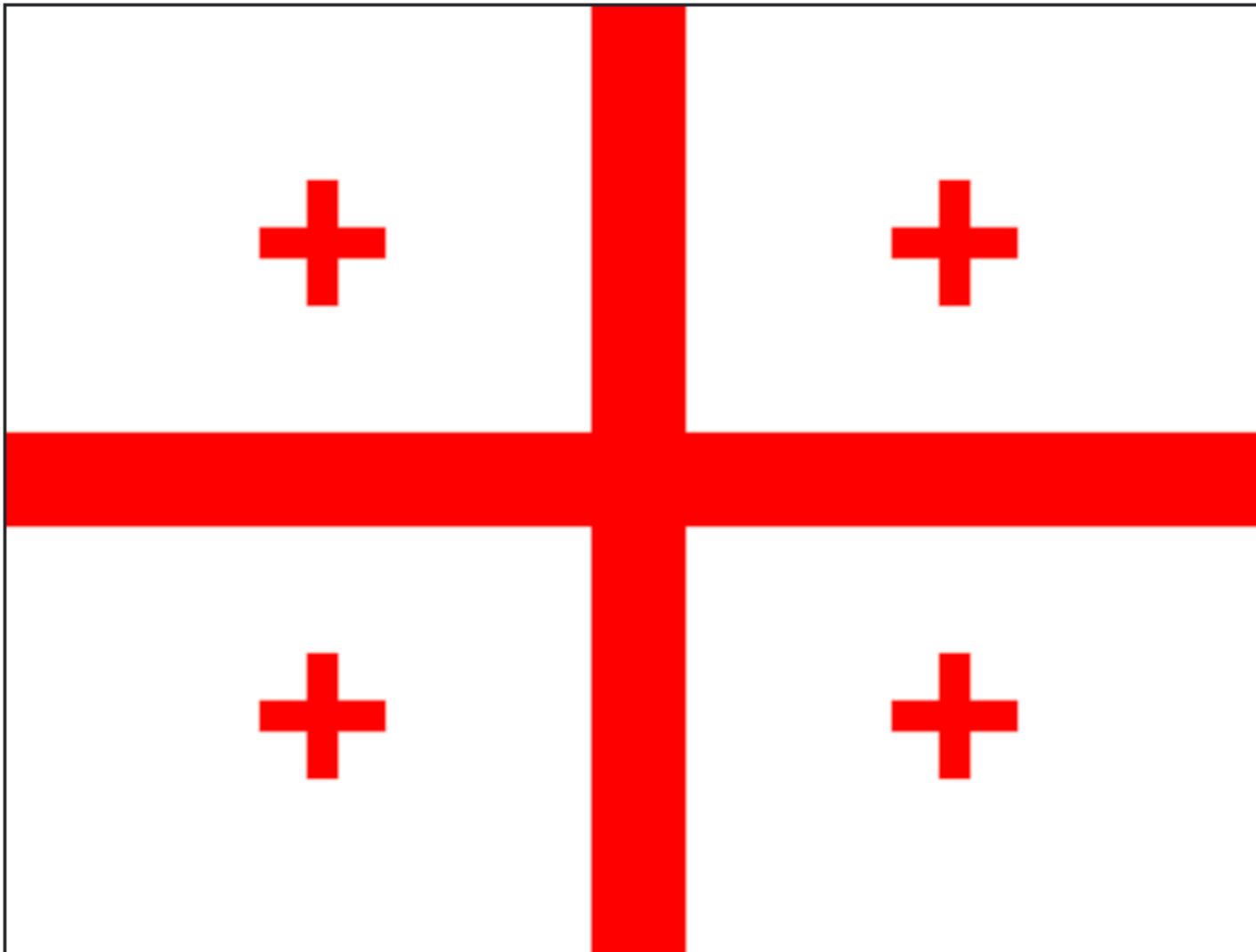
# What Function Do you Want?



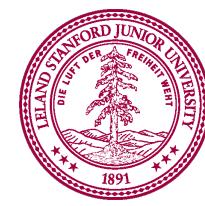
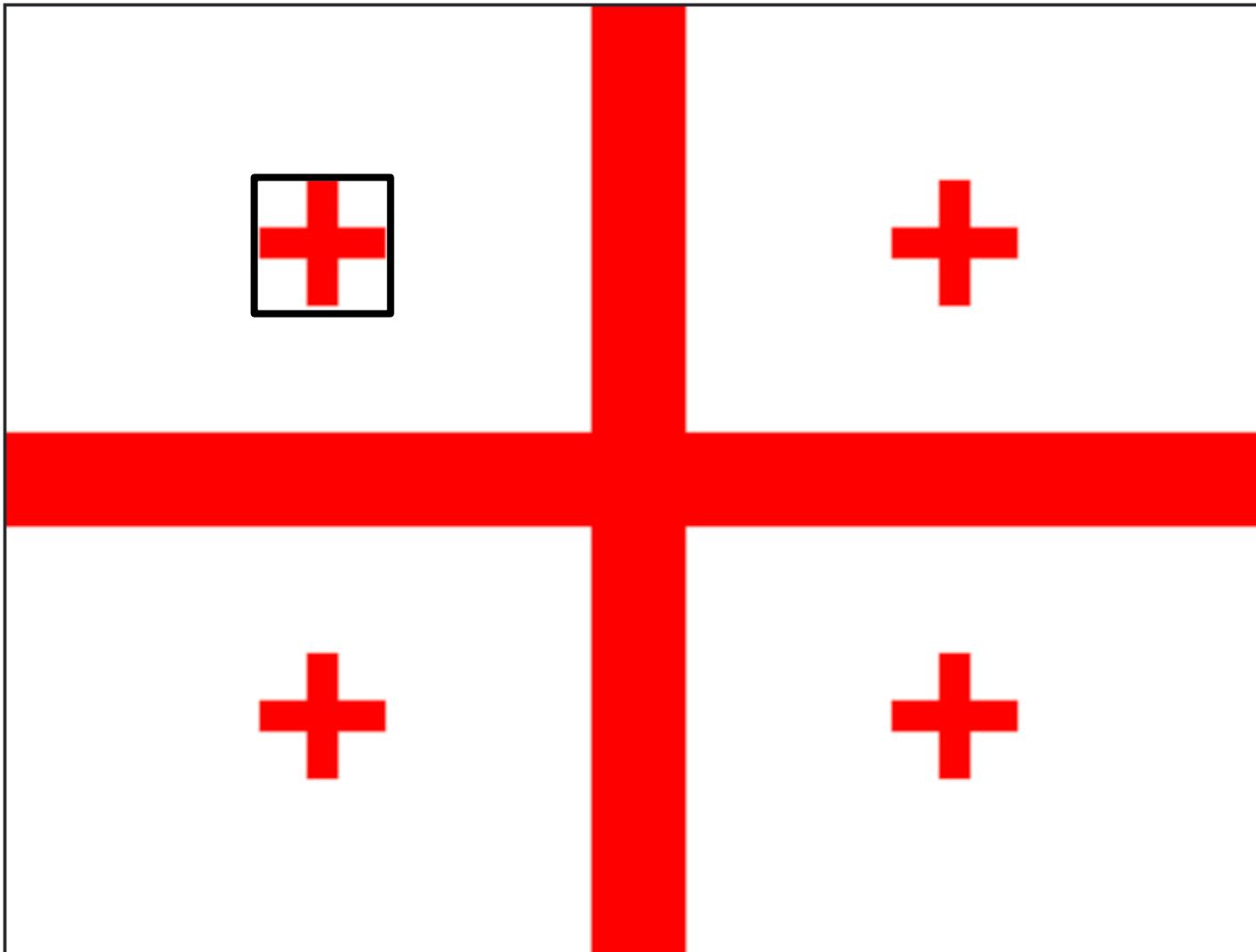
draw\_circle  
Is the turn\_right of graphics

Advanced Version

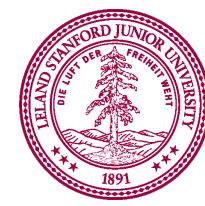
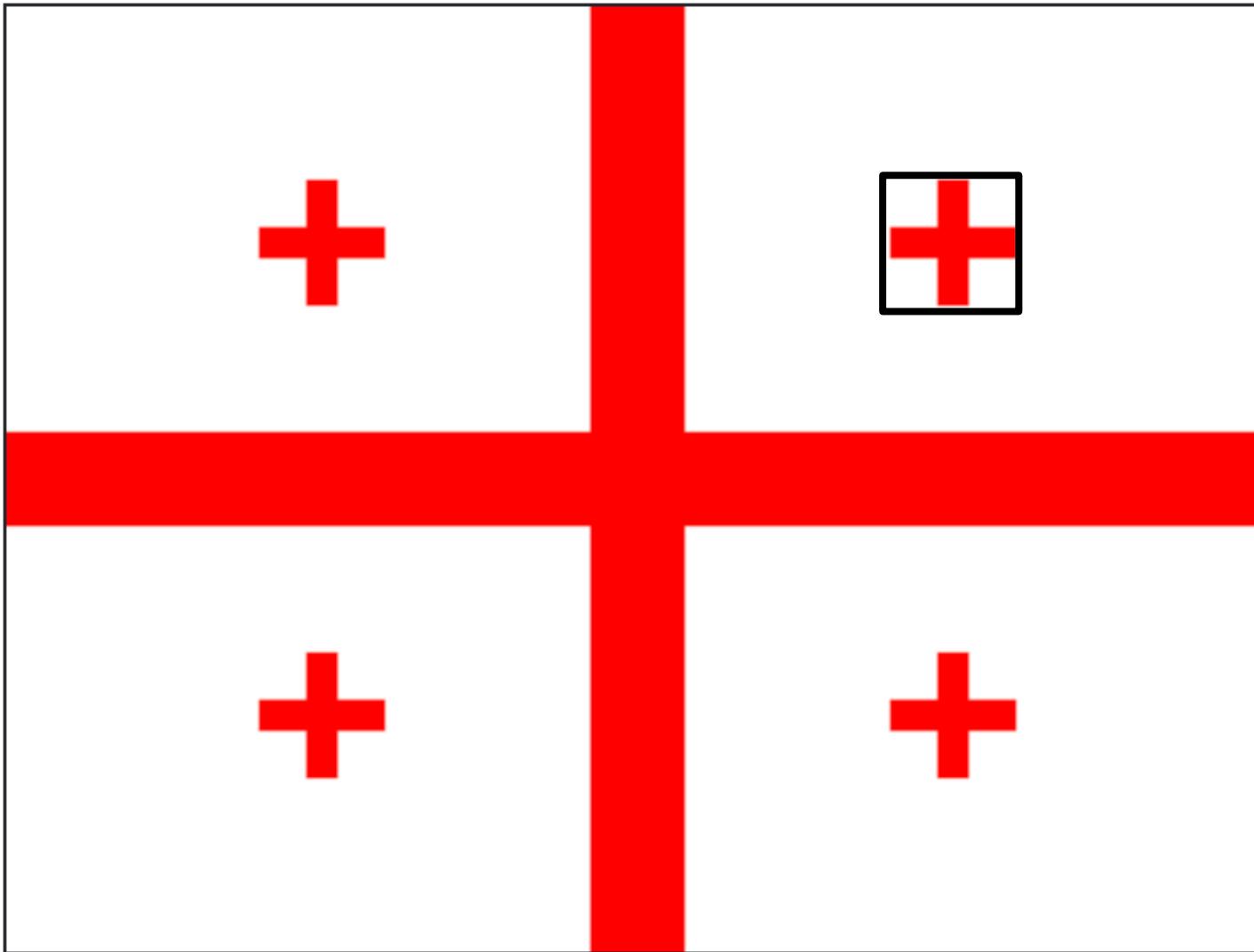
# What Function Do you Want?



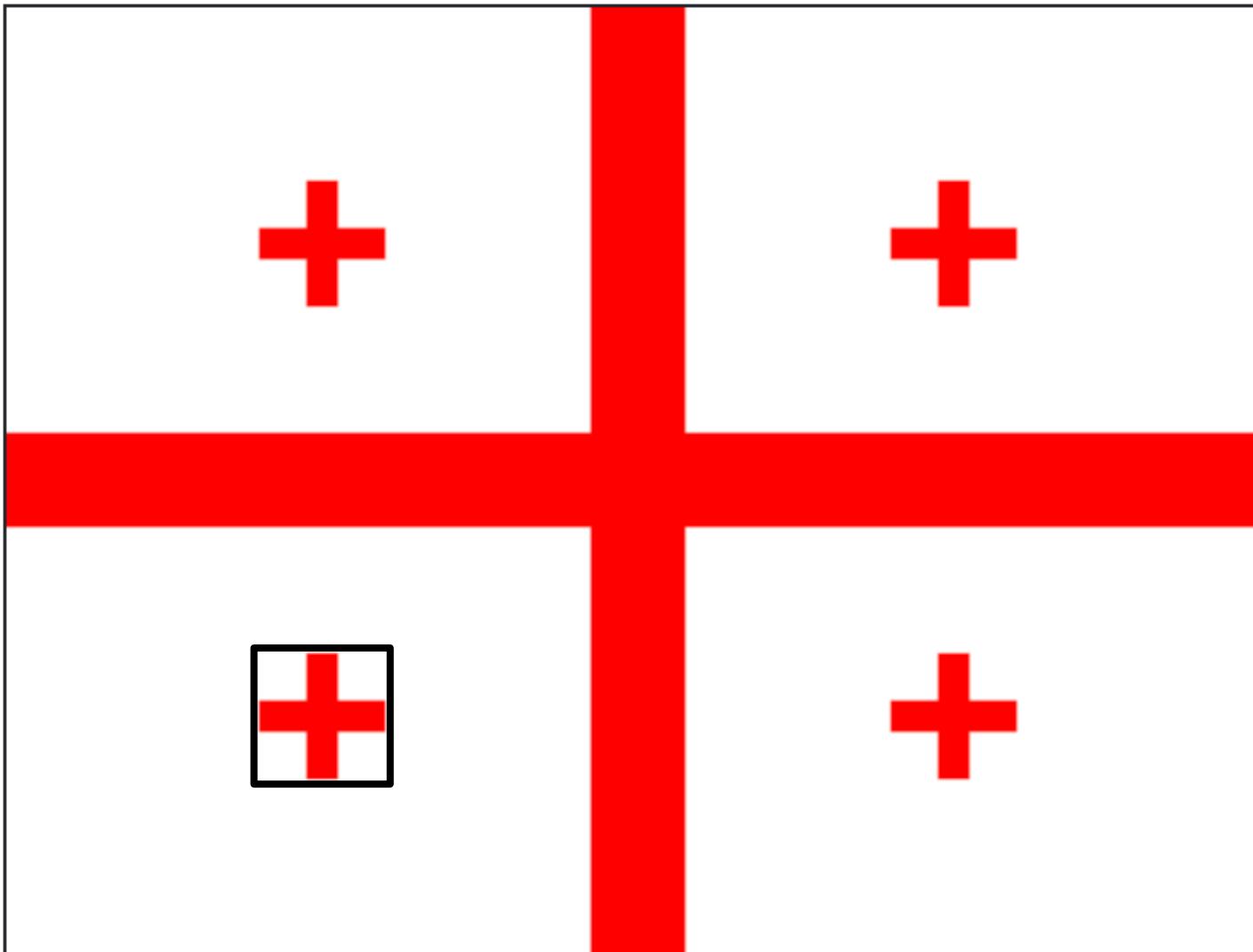
# What Function Do you Want?



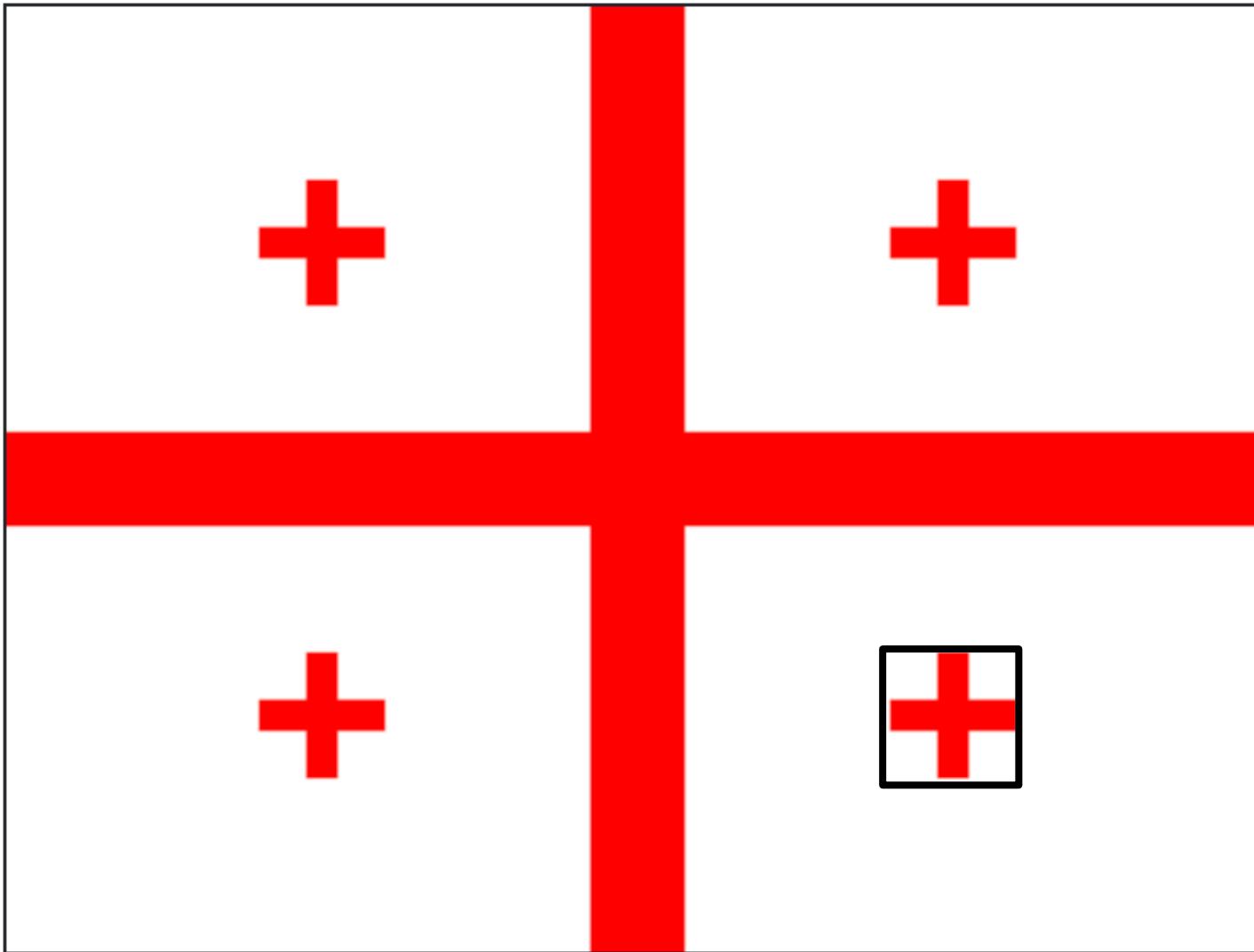
# What Function Do you Want?



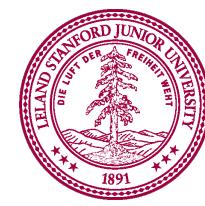
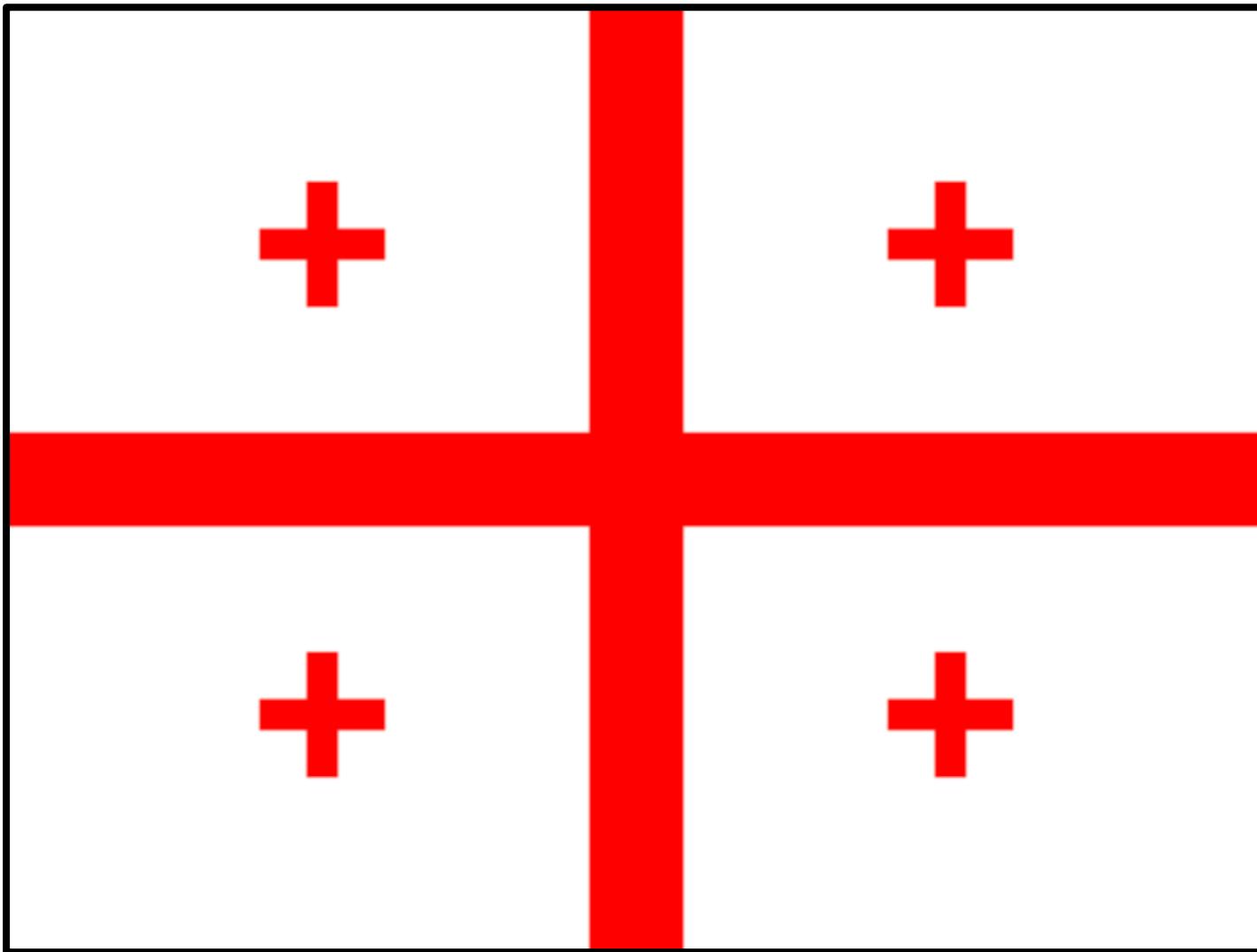
# What Function Do you Want?



# What Function Do you Want?



# What Function Do you Want?



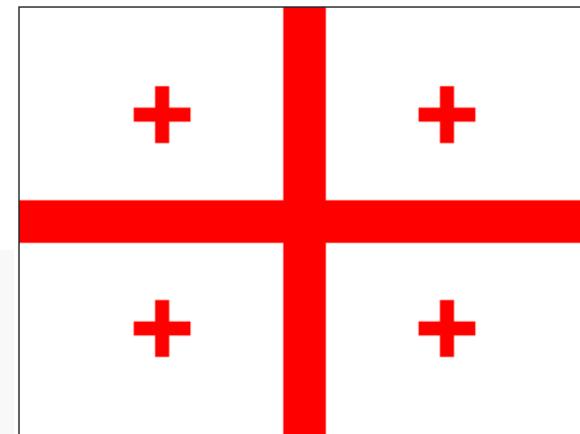
# What Function Do you Want?

```
def draw_plus(canvas, x_1, y_1, x_2, y_2, width):
```

```
def draw_georgia_flag(canvas):
    # some calculations for where the pluses go!
    x_left = CANVAS_WIDTH * 1/4
    x_right = CANVAS_WIDTH * 3/4
    y_top = CANVAS_HEIGHT * 1/4
    y_bottom = CANVAS_HEIGHT * 3/4

    # four calls to draw_plus
    draw_plus(canvas, x_left - 20, y_top - 20, x_left+20, y_top+20, 10)
    draw_plus(canvas, x_right - 20, y_top - 20, x_right+20, y_top+20, 10)
    draw_plus(canvas, x_left - 20, y_bottom - 20, x_left+20, y_bottom+20, 10)
    draw_plus(canvas, x_right - 20, y_bottom - 20, x_right+20, y_bottom+20, 10)

    # big background plus
    draw_plus(canvas, 0, 0, CANVAS_WIDTH, CANVAS_HEIGHT, 30)
```



is\_odd

# Is Odd

Code in Place    Code in Place

localhost:3000/cip3/ide/a/iseven

IDE | Is Even

Run

Replay Mode

Variable	Value
value	17
remainder	1
to_return	True

Replay Test Center >

When you are finished with this task:

Mark Complete

main.py

```
13     if value == 2:
14         return False
15     return True
16
17
18 def is_odd_v2(value):
19     # Slightly too complex
20     remainder = value % 2
21     if remainder == 1:
22         return True
23     else:
24         return False
25
26 def is_odd(value):
27     remainder = value % 2
28     to_return = remainder == 1
29     return to_return
30
31
32
33
34 if __name__ == '__main__':
35     main()
```

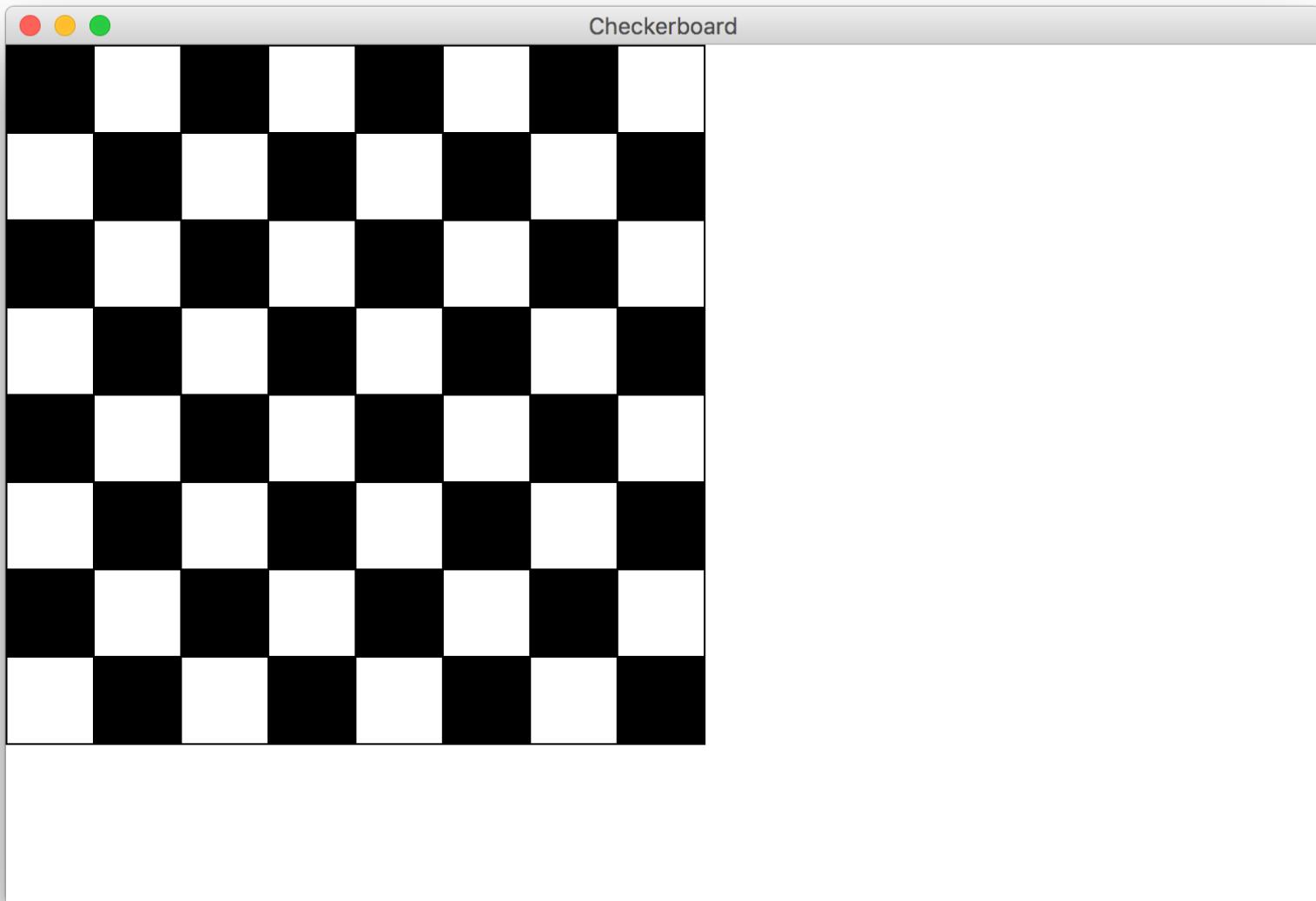
Replay Terminal

```
10 even
11 odd
12 even
13 odd
14 even
15 odd
16 even
```

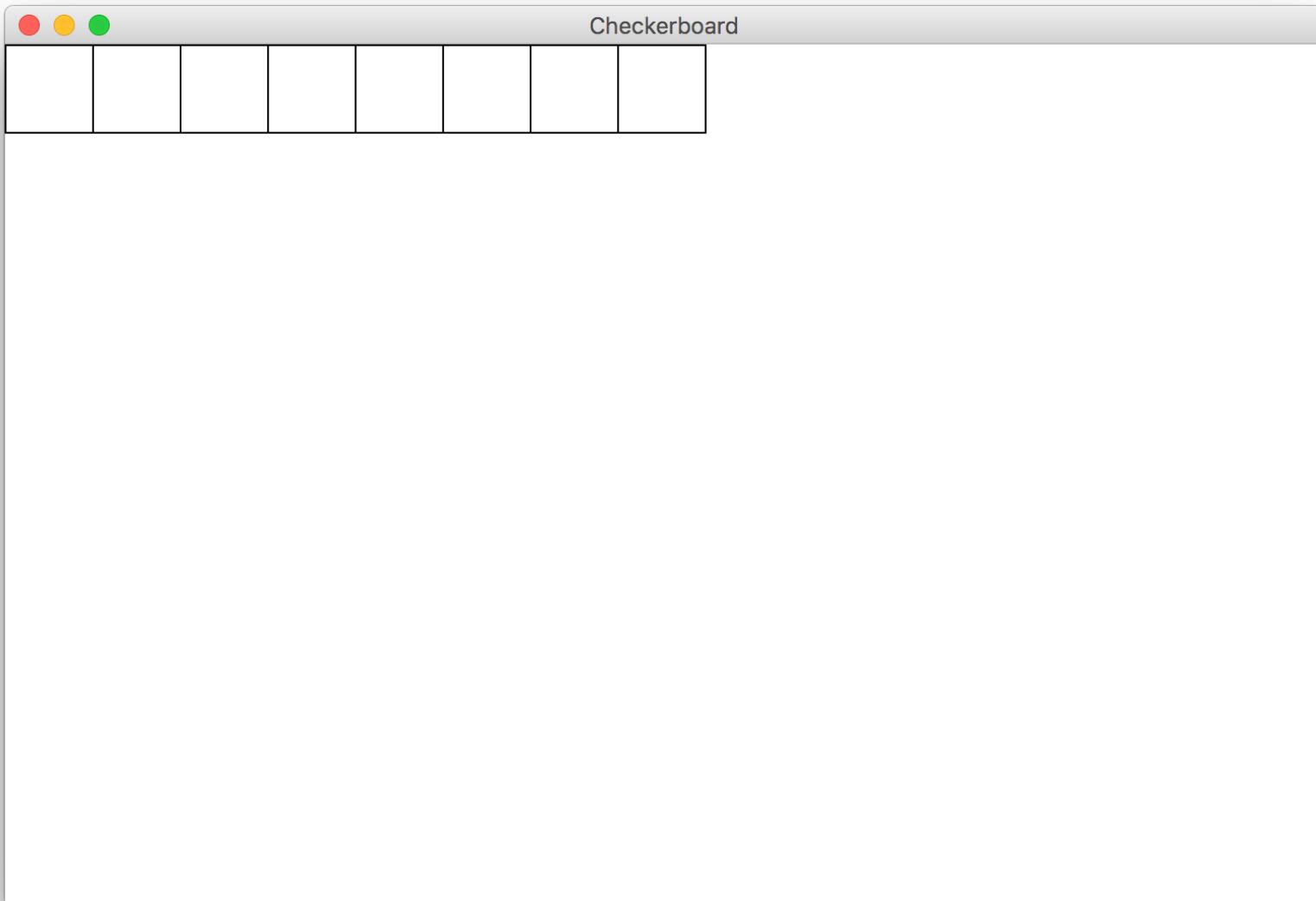


# Chessboard

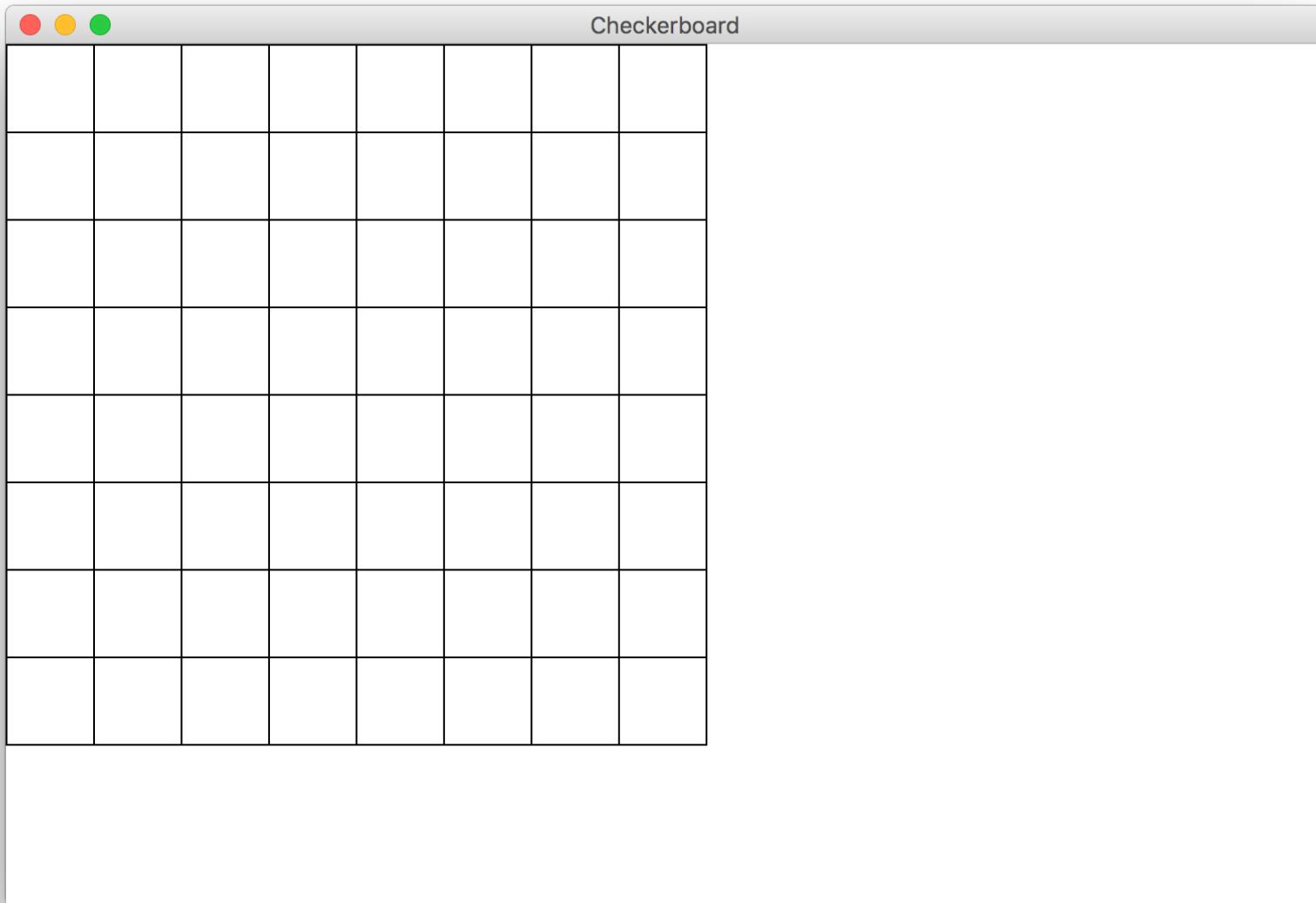
# Goal



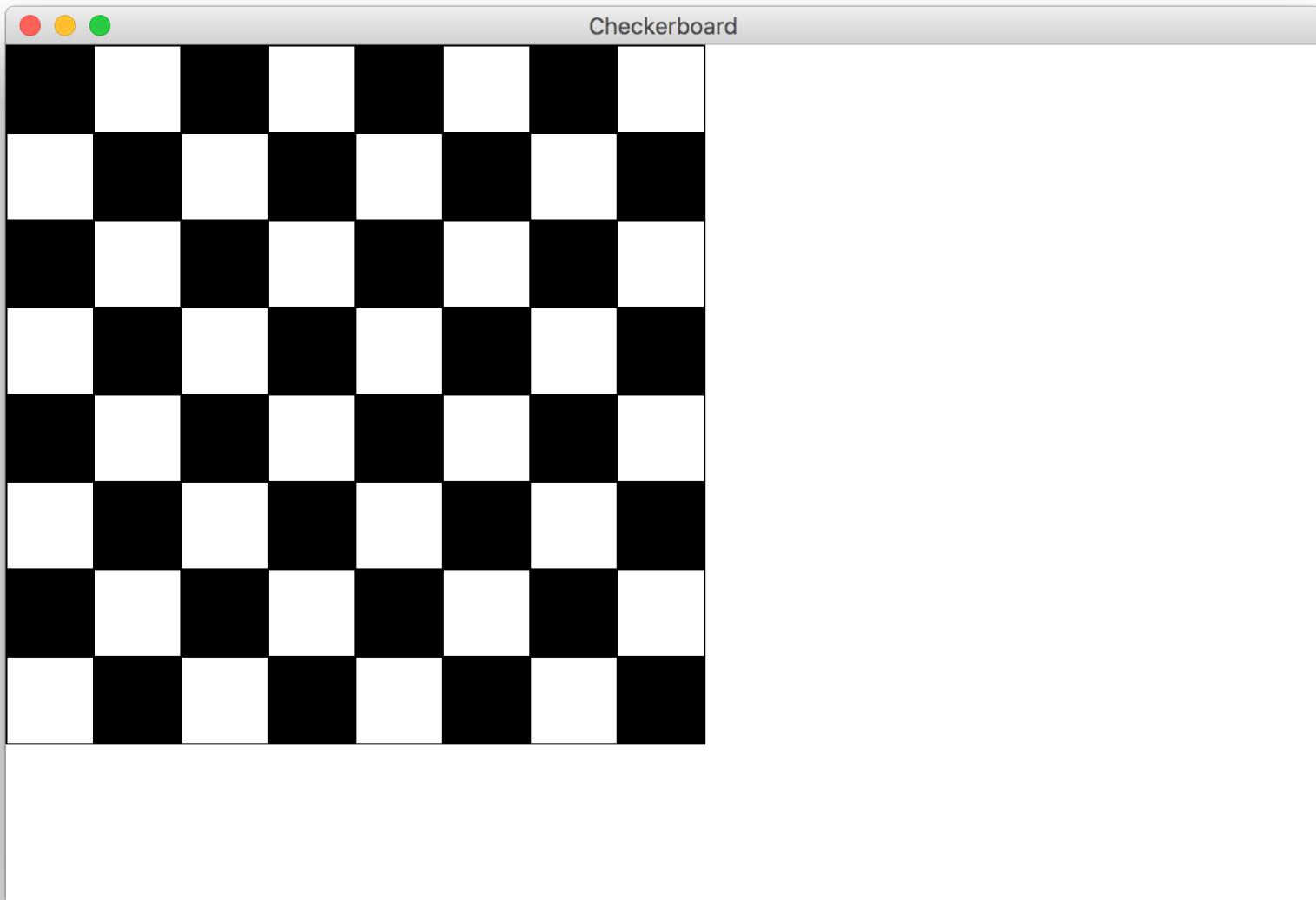
# Milestone 1



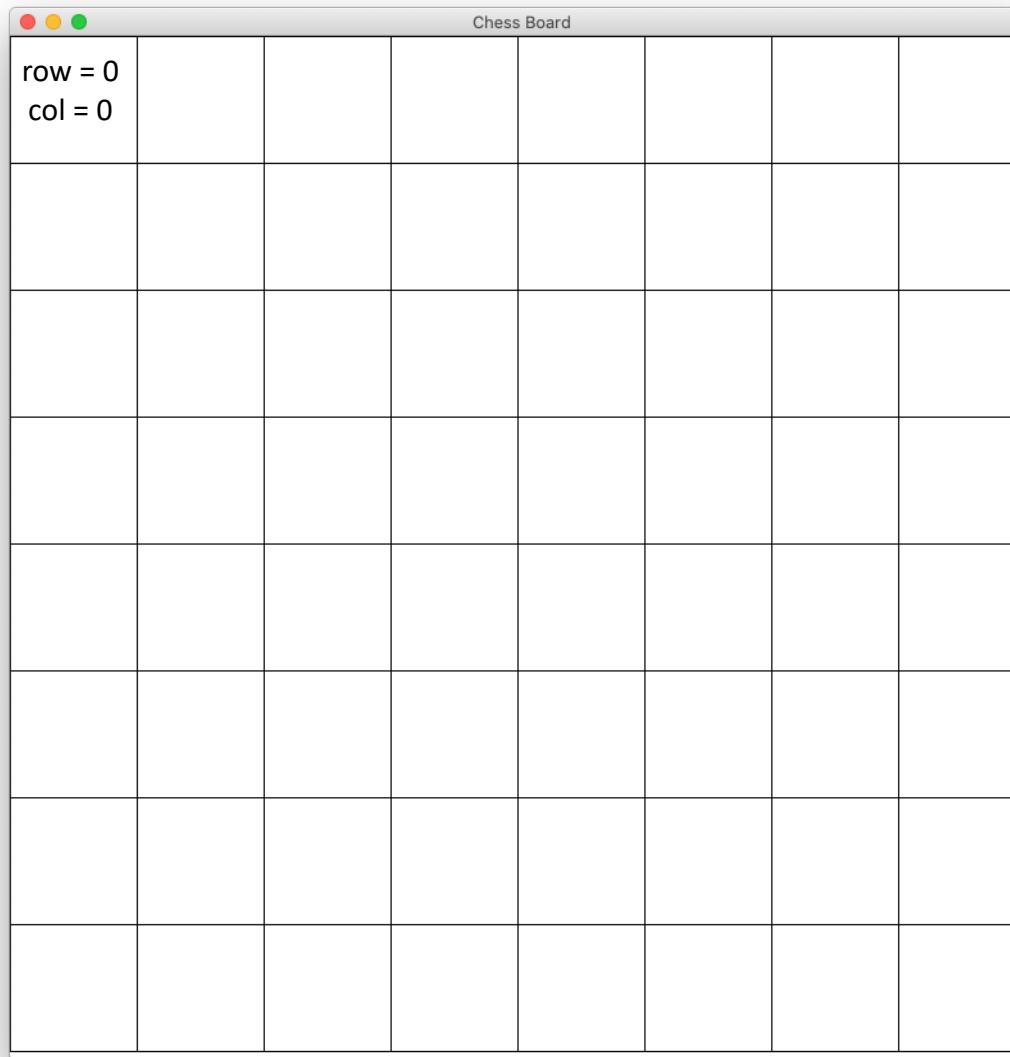
# Milestone 2



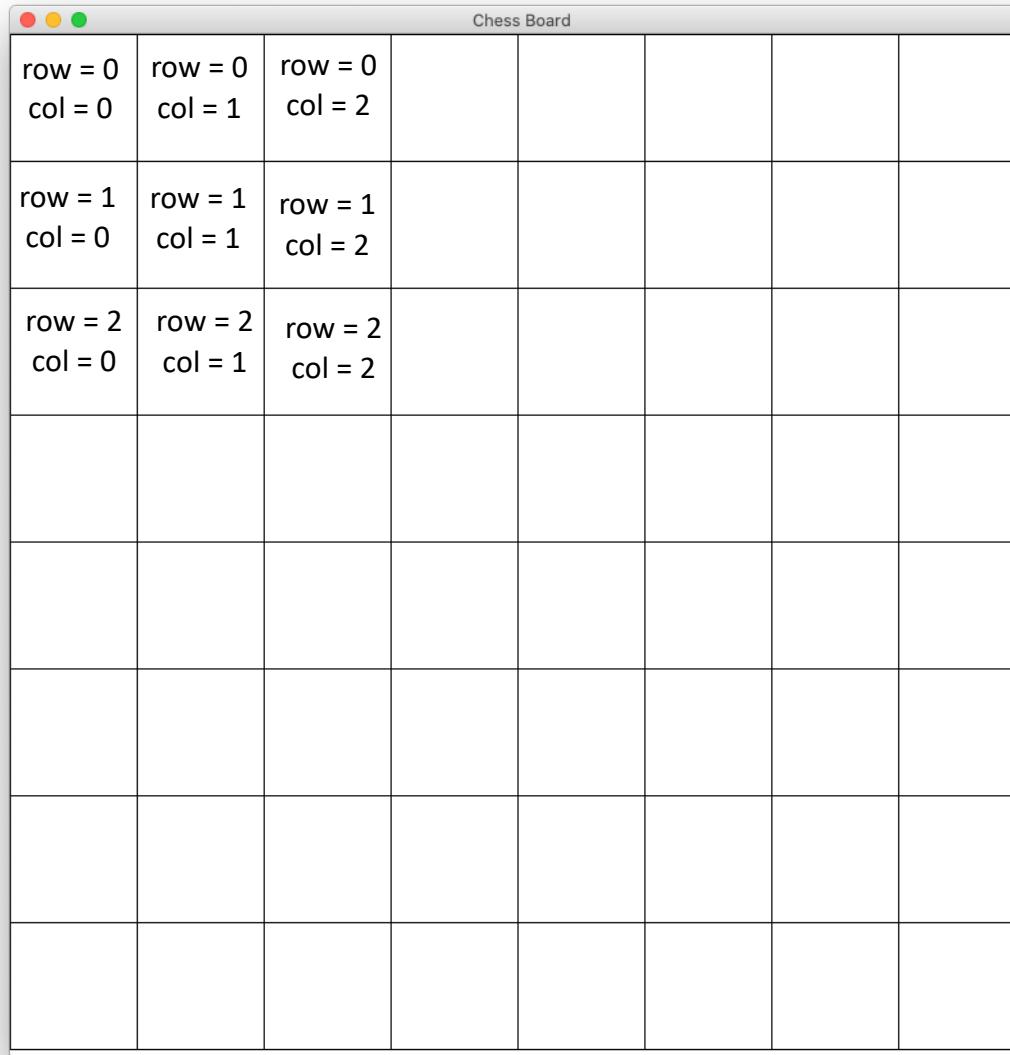
# Milestone 3



# Milestone 3



# Milestone 3



# Row + Col

Chess Board

0	1	2					
1	2	3					
2	3	4					

