Functions and Parameters and Return Values OH MY

Using Functions & Parameters with Graphics

We can create Functions to draw a shape that we're planning to draw repeatedly!

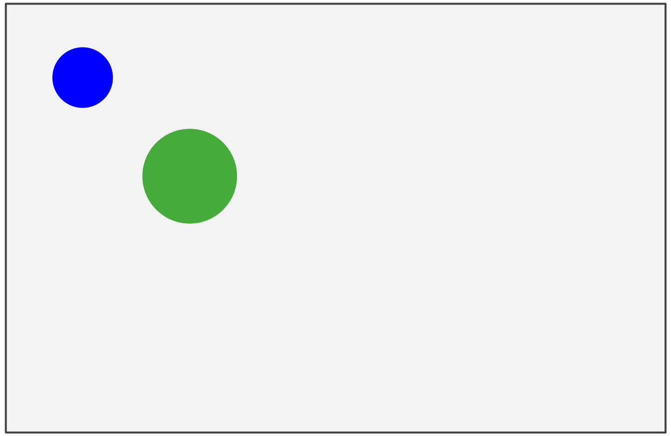
```
def draw_circle(radius, color, x, y):
    circle = Circle(radius)
    circle.set_color(color)
    circle.set_position(x, y)
    add(circle)
```

Using Functions & Parameters with Graphics

We can create Functions to draw a shape that we're planning to draw repeatedly!

```
def draw_circle(radius, color, x, y):
    circle = Circle(radius)
    circle.set_color(color)
    circle.set_position(x, y)
    add(circle)

draw_circle(20, Color.blue, 100, 100)
draw_circle(50, Color.green, 200, 200)
```



What’s going to be printed here?

Assume the user types in **asdf** when prompted.

```
x = int("45")
print(x)
y = input("Say something: ")
print(y)
```

What’s going to be printed here?

Assume the user types in **asdf** when prompted.

```
x = int("45")
print(x)
y = input("Say something: ")
print(y)
```

```
45
asdf
```

What’s going to be printed **here**?

```
def print_sum(num1, num2):
    print(num1 + num2)

print(print_sum(5, 29))
```

What’s going to be printed **here**?

```
def print_sum(num1, num2):
    print(num1 + num2)

print(print_sum(5, 29))
```

```
34
None
```

Sometimes, a function will equal something

Some Functions, like `int()` or `input()` can equal something when we call them. Others, like the ones we've been defining so far, will not equal anything - they'll equal **`None`**.

This is because some Functions do something called **`return`**ing a value. We can make our functions do the same by using a new keyword:

`return`!

How to use return

```python
def get_sum(num1, num2):
    total = num1 + num2
    return total

get_sum(2, 5)
```

How to use return

```
def get_sum(num1, num2):
    total = num1 + num2
    return total
```

~~get_sum(2, 5)~~ 7

How to use return

```
def get_sum(num1, num2):
    total = num1 + num2
    return total
```

~~get_sum(2, 5)~~ 7

```
 print(get_sum(2, 5))
```

How to use return

```
def get_sum(num1, num2):
    total = num1 + num2
    return total
```

~~get_sum(2, 5)~~ 7

print(~~get_sum(2, 5)~~)7

```
7
```

Why is this useful?

Using `return` lets us make changes to the code outside our Functions based on code inside of them!

```
def add_one(x):
    return x + 1


num1 = 5
print(num1)
num1 = add_one(num1)
print(num1)
```

Not using return

When a Function does not have a `return` value, the default `return` value is **`None`**. This is why we see what we do when we try to print a Function that doesn’t have a `return` in it.

Let’s Practice!

```
def add_two(x):
    return x + 2

def multiply_by_three(x):
    return x * 3

def do_something(x):
    return add_two(x) + multiply_by_three(x)

print(do_something(10))
```