Return vs Print

Quest

So, we have these two commands `print` and `return` that keep popping up all over the place. Both of these commands take a value and spit it out somewhere, but there is a huge difference between *printing* a value and *returning* it.

Return vs Print

```
# How is this function...
def meters_to_cm_case1(meters):
    return 100 * meters


# Different than this function?
def meters_to_cm_case2(meters):
    print(100 * meters)
```

The short answer is that `print` outputs a value to the console, while `return` outputs a value to a caller function.

The console is a visual tool for the programmer to see what is going on while the program is running. The console doesn't store or modify the values printed to it. The value is just pasted to the screen and that's it.

When a function returns a value, however, that value is passed back to the location of the function call as data, where it can be stored, modified, or even printed!

Let's see what happens if we run the two functions above side by side:

```
# How is this function...
def meters_to_cm_case1(meters):
    return 100 * meters


# Different than this function?
def meters_to_cm_case2(meters):
    print(100 * meters)


def main():
    print('running case1')
    case1 = meters_to_cm_case1(3)
    print('running case2')
    case2 = meters_to_cm_case2(3)
    print('case1 output: ' + str(case1))
    print('case2 output: ' + str(case2))


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

```
running case1
running case2
300
case1 output: 300
case2 output: None
```

▶ Run >_ Hide

Here's what's happening:

- The first function doesn't print anything to the console, but the value `100 * meters` is returned to the caller for later use (`case1`).
- The second function prints the value of `100 * meters` to the console, but then the program exits, and the value we printed is lost (which is why `None` is the value of `case2`).

This brings up another important difference. `print` does not end a function but `return` does! We can put multiple `print` statements back to back, and all of them will print something to the console. However, if we stacked several `return` statements on top of each other, only the first one would be executed.

```
# How is this function...
def multiple_returns():
    return "Howdy!"
    return "Howdy!"
    return "Howdy!"


# Different than this function?
def multiple_prints():
    print("Hey there!")
    print("Hey there!")
    print("Hey there!")


def main():
    print(multiple_returns())
    multiple_prints()


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

```
Howdy!
Hey there!
Hey there!
Hey there!
```

As you can see, we have to print the value of `multiple_returns()` for it to show up in the console. Even when we do this, only one `"Howdy!"` message shows up. This is because multiple returns are tricky. In the `multiple_returns()` function, the last two returns are not reached because each function can only return once. We will talk more about this in the **next section**.

Functions always return

We use `return` for two reasons: to stop a function early and to return information. If your function doesn't need to do either, you won't need to use the keyword `return`. Still, even without a `return` statement, your function returns no matter what. If the program ever reaches the last line of a function, it will automatically return to wherever that function was called (with a value of `None`). `print` does not work this way! A function is not guaranteed to print anything to the console. You have to specifically write a `print` statement to make that happen.