Multiple Returns

Quest

Welcome to part 2 of our saga on return statements. We know what `return` statements are (statements that give back information from the function) and we know how to distinguish them from `print` statements. So what is left to learn? Well, as the title of this section suggests, it is possible to have multiple return statements in a function. WHAT?! 😲 How is this possible? Let's see below.

```
def get_max(num1, num2):
    '''
    gets the maximum of the two input numbers
    params:
        num1 (int or float): the first input number
        num2 (int or float): the second input number
    returns: the maximum of num1 and num2
    '''
    if num1 > num2:
        return num1
    return num2

def main():
    print(get_max(4, 7))

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

```
7
```

▶ Run >_ Hide

This code is completely valid. Let's discuss why. Based on what we know about if statements, any code not in an attached else statement should also run after the body of the if has finished executing (given that the condition was true). So why doesn't the second return statement run? As we've said before, return statements end the function. This means that if the conditional is true and `return num1` runs, everything after that will not be executed. We never get to `return num2` even though it is not in an else.

Let's look at another example. This example doesn't have multiple return statements in code, but it has a single return statement within a for loop that runs multiple times. What will this do?

[⚠ Buggy Code]

```
def count_to_num(num):
    '''
    counts from 1 to num
    params: num: the number to count to
    returns: the counted numbers
    '''
    for i in range(1, num+1):
        return i

def main():
    print(count_to_num(5))

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

```
1
```

▶ Run >_ Hide

Huh. That's not what we wanted. We wanted our function to count all the way from one to the inputted number, but it only returned the first number. Even in for loops, the return statements end the function. The rest of the for loop is not executed, so the only number that is returned is 1. In this particular case, the issue can be fixed by changing the return statement to a print statement.

```
def count_to_num(num):
    '''
    counts from 1 to num
    params: num: the number to count to
    returns: the counted numbers
    '''
    for i in range(1, num+1):
        print(i)

def main():
    count_to_num(5)

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

```
1
2
3
4
5
```

▶ Run >_ Hide

This solution does not actually lead to returning multiple things, but avoids having to do that all

together. We will see a few ways to return multiple things in later sections. Below is another example of how **not** to return multiple things:

[⚠ Buggy Code]

```
def divisible(num1, num2):
    '''
    checks to see if num1 is evenly divisible by num2 (assuming that num1 is
    larger than num2)

    params:
        num1 (int): dividend (number being divided)
        num2 (int): divisor (number dividing into num1)
    returns: true and the quotient if the number goes in evenly, false and the
    remainder otherwise
    '''
    divides_evenly = False
    result = num1 % num2

    if result == 0:
        divides_evenly = True
        result = num1 / num2

    return divides_evenly
    return result


def main():
    print(divisible(18, 6))


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

```
True
```

▶ Run >_ Hide

As you can see, no error occurs, but the second return statement is completely ignored. If statements and conditionals allow for multiple return statements, but they cannot just exist side by side.