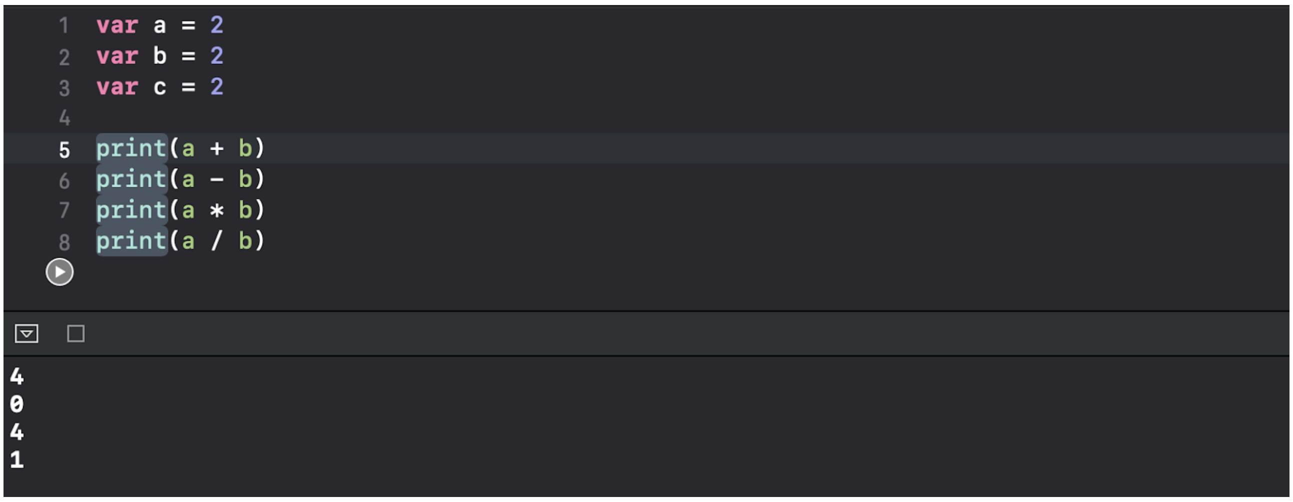
In this chapter, we'll examine some basic math operations that you can use with constants, variables, and equations in general. Swift covers your basic math operations: addition, subtraction, multiplication, and division (syntax for all shown below).



```
var a = 2
var b = 2
var c = 2

print(a + b)
print(a - b)
print(a * b)
print(a / b)
```

```
4
0
4
1
```

You can also chain these operators together, so something like the following would evaluate to 8:

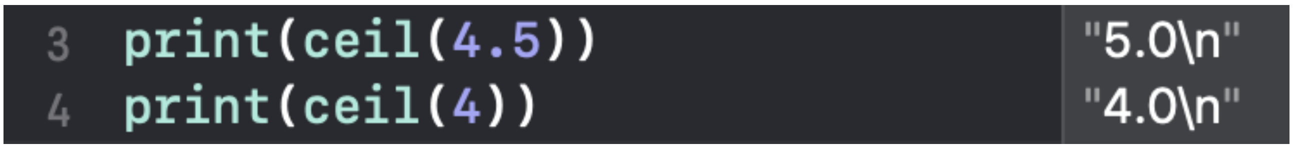
```
print(a * b * c)
```

Besides the basic math operators, there are several operators in Swift you might not realize exist. For example, we can use the power function for exponentials, as demonstrated below. This function accepts two numbers, either doubles or ints, the base and the exponent to raise the base to. Thus, the example below evaluates to 8.

```
pow(2, 3)
```

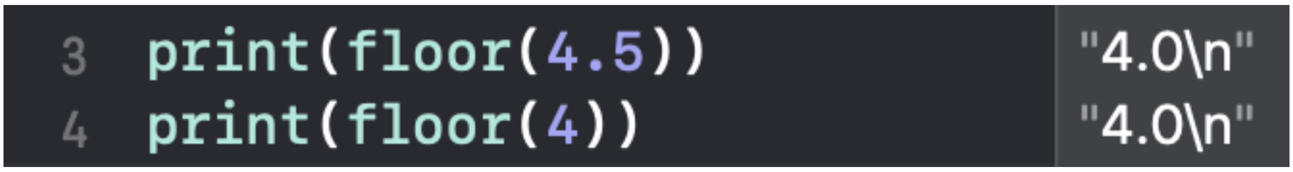
Another common function at our disposal is `sqrt()`. For example, `sqrt(16)` would give us 4.

One operator that you might not be familiar with is ceiling, or `ceil()`, which rounds any number up to the next whole number. In the example below, it rounds 4.5 up to 5. Whole numbers get rounded up to themselves.



```
print(ceil(4.5))        "5.0\n"
print(ceil(4))          "4.0\n"
```

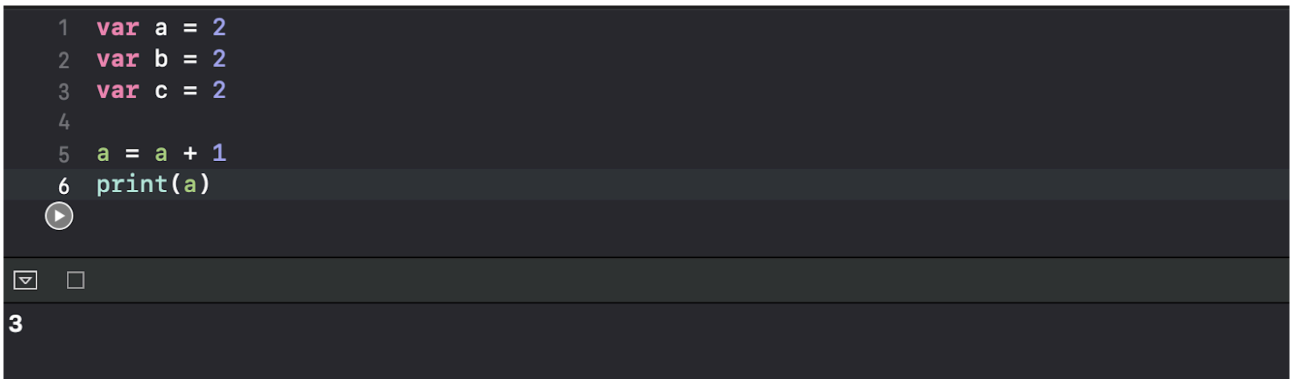
Similarly, there is `floor()`, which rounds any number down to the next whole number. Below, we have `floor()` rounding 4.5 down to 4. Again, whole numbers are roun
ded down to themselves.



```
print(floor(4.5))       "4.0\n"
print(floor(4))         "4.0\n"
```

Incrementing and Decrementing Numbers

Let's say you want to increment a variable by one, in this case the variable *a*. As shown below, you would take *a*, add 1 to it, and reassign it back to *a*. If we run this code in a playground, we would see that *a* now contains 3



```
var a = 2
var b = 2
var c = 2

a = a + 1
print(a)
```

```
3
```

The equivalent, shorthand way of writing the same thing is to use += instead:

```
a += 1
```

Similarly, you can use the shorthand way of decrementing numbers, as shown below:

```
a -= 1
```

This shorthand also works with multiplication (*=) and division (/=). This shorthand exists because you tend to modify variables through simple math operations quite often when expressing your logic or writing algorithms in your code.