Using Integers and Floats

by Sophia

WHAT'S COVERED

In this lesson, you will learn about using integer and float variables and values. Specifically, this lesson covers:

1. Integers and Number Systems

As a recap from the prior lesson, an integer is a numeric data type `int` that has no decimal points and can be a positive or a negative number.

EXAMPLE
For example, all of these are integers:

```
myFirstInt = 100
mySecondInt = 0
myThirdInt = -999
```

Something unique with how Python uses the integer or `int` data type is that there is no limit to how long the integer value can be. It's only constrained by the amount of memory on your computer. So, you could set an `int` variable like:

```
myInt = 123456789123456789123456789123456789123456789123456789
```

KEY CONCEPT

By default, any integer value without a prefix indicates that it is a base 10 number. A prefix is a set of characters prior to the integer value that indicates that the integer is of another base type, such as 0b for base 2, 0o for base 8, or 0x for base 16. Most individuals may only be familiar with base 10 number systems. **Base 10 numbers** are also called decimal values; this represents any number using 10 digits that go from 0 to 9. **Base 2 numbers** are called binary numbers and represent numbers using 2 digits with a 0 or a 1. This is important as computers store everything as 0s and 1s behind the scenes. There are also **base 8 numbers** called octal, which represent numbers using 8 digits going from 0 to 7. Lastly, we have **base 16 numbers** or hexadecimal. These represent any number using 10 digits and 6 characters going from 0 to 9 and then A, B, C, D, E, and F.

Since we're most familiar with the base 10 (decimal) numbers, let's look at how we write our numbers.

Base 10 Numbers (Decimal)

We start by writing from 0 and then go up to 9. Once we've used up all of our symbols, we add another symbol to the left and make the right symbol a 0, which is how we go from 9 to 10. Then, we go again, until we've used up all of our symbols on the right side, and then we increase the symbol to the left of it by one symbol, which goes from 19 to 20. We keep following that pattern until we've used up all of the symbols at 99. To move up, we make both of them 0 and add 1 to the left to have 100, and keep going there.

When we have 10 different symbols (0 to 9 in this case), we have each position worth 10 times more than the prior one, as you'll see in this quick example:

EXAMPLE

_ _ _ _ 1
_ _ _ 1 0
_ _ 1 0 0
_ 1 0 0 0
1 0 0 0 0

In essence, if we have a number like 483, we break it down into separate parts:

(4 * 100) + (8 * 10) + (3 * 1) = 483

This may seem obvious, but it is important because it also applies to other number systems.

Base 2 Numbers (Binary)

For example, with binary, we only have two digits that represent a number with 0 and 1. Then we're out of symbols, so we have to apply the same rules as the decimal system.

We start with 0 and then 1. Next, we make the right digit a 0 and add 1 to the left, meaning that our next number is '10'. Then we move the right digit to the next item, which is now 11. So, let's see what numbers we have so far:

EXAMPLE

0, 1, 10, 11

What happens next? We do the same thing as we did for the decimals, since we're out of symbols. We put 0s in both of those places and put a 1 at the left, so, 100. The binary system is based on 2 digits, and as such, each position is worth two more times than the prior position. It is similar to reading numbers in decimal, but instead of each number being 10 times the next, it's only 2 times.

EXAMPLE

For the conversion to decimal, these are the binary values:

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

For larger numbers, the binary system continues to add a digit to the left that is 2 times the amount of the digit to its right.

EXAMPLE

Here is a breakdown of the decimal number 563:

Binary		1	0	0	0	1	1	0	0	1	1
	...	512	256	128	64	32	16	8	4	2	1

512 + 32 + 16 + 2 + 1 = 563

You can see how the pattern functions, and it works the same for octal and hexadecimal numbers. With hexadecimal numbers, we have 10 digits and 6 letters to represent the values. Let's expand our table to include hexadecimal and octal numbers and run it up to decimal 30:

EXAMPLE

Decimal (Base 10)	Binary (Base 2)	Octal (Base 8)	Hexidecimal (Base 16)
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3

4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17
24	11000	30	18
25	11001	31	19
26	11010	32	1A
27	11011	33	1B
28	11100	34	1C
29	11101	35	1D
30	11110	36	1E

Using Different Bases in Python

To represent an integer in a different number system than decimal, we need to add a prefix.

For binary numbers, we add a prefix of 0b (number zero and letter b). For octal numbers, we add a prefix of 0o (number zero and letter o). For hexadecimal numbers, we add a prefix of 0x (number zero and letter x).

Using the table that we have above, let's output decimal 16 and its equivalent to see what the output results look like.

EXAMPLE

```
MyDecimalNum = 16
myBinaryNum = 0b10000
myOctNum = 0o20
myHexNum = 0x10

print("Decimal", MyDecimalNum)
print("Binary", myBinaryNum)
print("Octal", myOctNum)
print("Hexadecimal", myHexNum)
```

As expected, the output for all of them is the same:

```
Decimal 16
Binary 16
Octal 16
Hexadecimal 16
```

TERMS TO KNOW

Base 10 Numbers
Also called decimal values, these are numbers using 10 digits that go from 0 to 9.

Base 2 Numbers
Also called binary numbers, these are numbers using two digits, 0 or 1.

Base 8 Numbers
Also called octal numbers, these are numbers using 8 digits, going from 0 to 7.

Base 16 Numbers
Also called hexadecimal numbers, these are any numbers using 10 digits (0 to 9) and 6 letters (A, B, C, D, E, and F).

2. Floating-Point Numbers

The data type `float` represents a floating-point number. `Float` values generally have a decimal point. This is the key difference between an `int` and the `float` in Python. `Float` values can be positive or negative.

EXAMPLE

For example, these would all be valid decimal values:

```
myFirstFloat = 8.2
mySecondFloat = 0.0000001
myThirdFloat = -192.12387
```

The **type()** function takes a value or variable as input and returns the type of the value or variable. This is helpful with debugging since it shows what type the variable is set to. We'll talk about debugging in a later lesson. For now, we can check if a variable is set up as a `float` by using the `type()` function and printing it out.

```
myFirstFloat = 8.2
print(type(myFirstFloat))
```

The `type()` function identifies the value as `float`.

```
<class 'float'>
```

Assigning a value to a `float` is the same as with an `int` except the value being set contains a decimal point.

```
myFirstNumber = 13284 #this is an integer
myFirstFloat = 13.284 #this is a float
```

Since these values are represented as numbers, we can perform mathematical operations on them, as we'll see in the next lesson.

TERM TO KNOW

type()

The `type()` function takes a value or variable as input and returns the type of the value or variable.

SUMMARY

In this lesson, we learned more about **integers** and **various numbering systems** including binary (base 2) and decimal (base 10). We discussed that by default, Python treats an integer as a base 10 number unless there is a specific prefix included, in which case we can switch to other bases. Finally, we explored that a float data type represents a **floating-point number** that generally has a decimal point and can be either a positive or negative value.

Best of luck in your learning!

TERMS TO KNOW

Base 10 Numbers
Also called decimal values, these are numbers using 10 digits that go from 0 to 9.

Base 16 Numbers
Also called hexadecimal numbers, these are any numbers using 10 digits (0 to 9) and 6 letters (A, B, C, D, E, and F).

Base 2 Numbers
Also called binary numbers, these are numbers using two digits, 0 or 1.

Base 8 Numbers
Also called octal numbers, these are numbers using 8 digits, going from 0 to 7.

type()
The type() function takes a value or variable as input and returns the type of the value or variable.