Skills Network

Working with Types in Python

Estimated time needed: **10** minutes

Objectives

After completing this lab you will be able to:

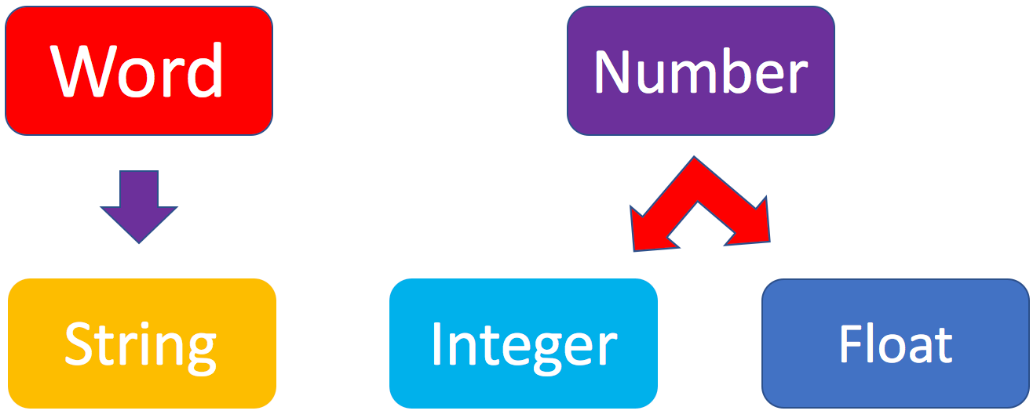
- Work with various types of data in Python
- Convert the data from one type to another

Table of Contents

Types of objects in Python
- Integers
- Floats
- Converting from one object type to a different object type
- Boolean data type
- Exercise: Types

Types of objects in Python

Python is an object-oriented language. There are many different types of objects in Python. Let's start with the most common object types: *strings*, *integers* and *floats*. Anytime you write words (text) in Python, you're using *character strings* (strings for short). The most common numbers, on the other hand, are *integers* (e.g. -1, 0, 100) and *floats*, which represent real numbers (e.g. 3.14, -42.0).



The following code cells contain some examples.

```
In [ ]: # Integer

        11
```

```
In [ ]: # Float

        2.14
```

```
In [ ]: # String

        "Hello, Python 101!"
```

You can get Python to tell you the type of an expression by using the built-in `type()` function. You'll notice that Python refers to integers as `int`, floats as `float`, and character strings as `str`.

```
In [ ]: # Type of 12

        type(12)
```

```
In [ ]: # Type of 2.14

        type(2.14)
```

```
In [ ]: # Type of "Hello, Python 101!"

        type("Hello, Python 101!")
```

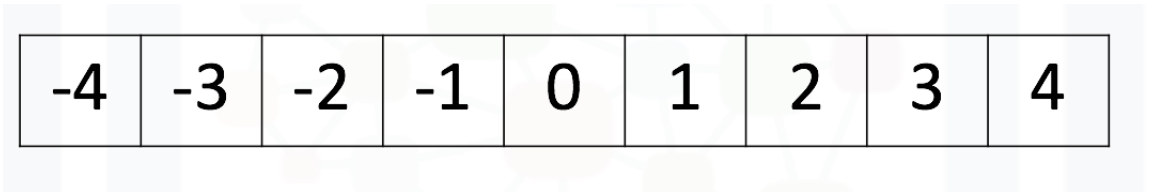
In the code cell below, use the `type()` function to check the object type of `12.0`.

```
In [ ]: # Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

Integers

Here are some examples of integers. Integers can be negative or positive numbers:



We can verify this is the case by using, you guessed it, the `type()` function:

In [ ]:
```python
# Print the type of -1

type(-1)
```

In [ ]:
```python
# Print the type of 4

type(4)
```

In [ ]:
```python
# Print the type of 0

type(0)
```

Floats

Floats represent real numbers; they are a superset of integer numbers but also include "numbers with decimals". There are some limitations when it comes to machines representing real numbers, but floating point numbers are a good representation in most cases. You can learn more about the specifics of floats for your runtime environment, by checking the value of `sys.float_info`. This will also tell you what's the largest and smallest number that can be represented with them.

Once again, can test some examples with the `type()` function:

In [ ]:
```python
# Print the type of 1.0

type(1.0) # Notice that 1 is an int, and 1.0 is a float
```

In [ ]:
```python
# Print the type of 0.5

type(0.5)
```

In [ ]:
```python
# Print the type of 0.56

type(0.56)
```

In [ ]:
```
# System settings about float type
import sys
sys.float_info
```

Converting from one object type to a different object type

You can change the type of the object in Python; this is called typecasting. For example, you can convert an *integer* into a *float* (e.g. 2 to 2.0).

Let's try it:

In [ ]:
```
# Verify that this is an integer

type(2)
```

Converting integers to floats

Let's cast integer 2 to float:

In [ ]:
```
# Convert 2 to a float

float(2)
```

In [ ]:
```
# Convert integer 2 to a float and check its type

type(float(2))
```

When we convert an integer into a float, we don't really change the value (i.e., the significand) of the number. However, if we cast a float into an integer, we could potentially lose some information. For example, if we cast the float 1.1 to integer we will get 1 and lose the decimal information (i.e., 0.1):

In [ ]:
```
# Casting 1.1 to integer will result in loss of information

int(1.1)
```

Converting from strings to integers or floats

Sometimes, we can have a string that contains a number within it. If this is the case, we can cast that string that represents a number into an integer using `int()`:

In [ ]:
```
# Convert a string into an integer

int('1')
```

But if you try to do so with a string that is not a perfect match for a number, you'll get an error. Try the following:

In [ ]:
```
# Convert a string into an integer with error

int('1 or 2 people')
```

You can also convert strings containing floating point numbers into *float* objects:

In [ ]:
```
# Convert the string "1.2" into a float

float('1.2')
```

[Tip:] Note that strings can be represented with single quotes (`'1.2'`) or double quotes (`"1.2"`), but you can't mix both (e.g., `"1.2'`).

Converting numbers to strings

If we can convert strings to numbers, it is only natural to assume that we can convert numbers to strings, right?

In [ ]:
```
# Convert an integer to a string

str(1)
```

And there is no reason why we shouldn't be able to make floats into strings as well:

In [ ]:
```
# Convert a float to a string

str(1.2)
```

Boolean data type

Boolean is another important type in Python. An object of type *Boolean* can take on one of two values: `True` or `False` :

In [ ]:
```
# Value true

True
```

Notice that the value `True` has an uppercase "T". The same is true for `False` (i.e. you must use the uppercase "F").

In [ ]:
```
# Value false

False
```

When you ask Python to display the type of a boolean object it will show `bool` which stands for *boolean*:

In [ ]:
```
# Type of True

type(True)
```

In [ ]:
```
# Type of False

type(False)
```

We can cast boolean objects to other data types. If we cast a boolean with a value of `True` to an integer or float we will get a one. If we cast a boolean with a value of `False` to an integer or float we will get a zero. Similarly, if we cast a 1 to a Boolean, you get a `True`. And if we cast a 0 to a Boolean we will get a `False`. Let's give it a try:

In [ ]:
```
# Convert True to int

int(True)
```

In [ ]:
```
# Convert 1 to boolean

bool(1)
```

In [ ]:
```
# Convert 0 to boolean

bool(0)
```

In [ ]:
```
# Convert True to float

float(True)
```

Exercise: Types

What is the data type of the result of: `6 / 2`?

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

What is the type of the result of: `6 // 2`? (Note the double slash `//`.)

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

What is the type of the result of: `"Hello, World!"`

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

What is the type of the result of: `"hello" == "world"`

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

Write the code to convert the following number representing employeeid **"1001"** to an integer

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

Write the code to convert this number representing financial value **"1234.56"** to a floating point number

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

Write the code to convert this phone number **123-456-7890** to a string

In [ ]:
```
# Write your code below. Don't forget to press Shift+Enter to execute the cell
```

▶ Click here for the solution

Congratulations, you have completed your hands-on lab on Types in Python.

Author

Joseph Santarcangelo

Other contributors

Mavis Zhou

toggle##

toggle|

toggle|---|---|---|---|

toggle|

toggle|

toggle|

{toggle}##

{toggle}|

{toggle}|

{toggle}|

{toggle}|