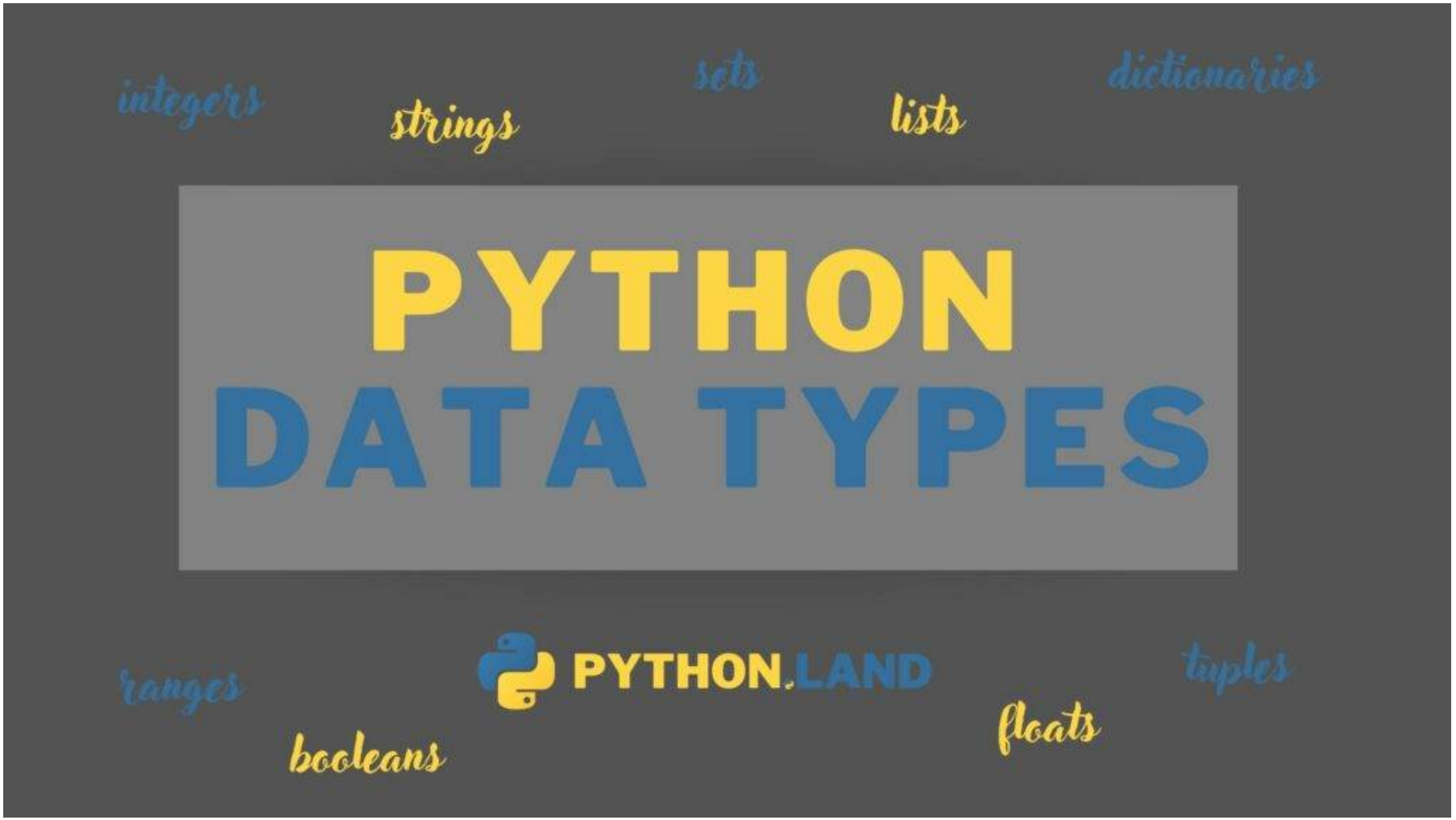
Previous: Python Packages

Next: Python Integer ›

Home » Python Data Types



Python Data Types

August 19, 2022

In this section, we take a close look at the most important Python data types. Python provides us with several native data types to store and process data. These are essential building blocks that you need to know well. When thinking about a problem, part of the solution is often choosing the right data type.

Knowing what each data type is capable of will make it much easier to pick the right one.

Table of Contents [hide]

Beginners Python Course (2023) **Modules, Packages, And Virtual Environments (2023)**

Basic and advanced Python data types

We make a distinction between basic types and the more advanced data structures. The basic data types in Python store a single value, like a number or a piece of text. The basic data types in Python are:

- Integers
- Floating point numbers
- Complex numbers
- Booleans
- Strings

Next, we have the more advanced Python data types. They can store many items, like lists of items or key-value pairs:

- Tuples
- Lists
- Ranges
- Dictionaries
- Sets

These types all have distinctive features that set them apart from the others. E.g., ranges are efficiently calculated on the fly, tuples can not be modified (while lists can), and sets allow you to do mathematical set calculations.

Mutability in Python

Python data types can be categorized into two classes: **mutable** and **immutable**. Or, more accurately: hashable and unhashable. An object is mutable if we can change the data it holds, and it's immutable if we can't change it. Examples of immutable data types in Python are:

- All numbers (integers, floats, complex numbers)
- Booleans
- Strings
- Tuples

Python data types that are mutable are:

- Lists
- Dictionaries
- Sets

Why and when are data types mutable?

We didn't dive into all these types yet, but let's take a list as an example. Without knowing the exact details, I can tell you that you can add more items to a list, remove items, and replace them. These are not surprising features for a list, right? So a list can be changed; hence it is mutable.

However, an integer is simply a number, like 2, 3, and 4. You can't change a number; it is what it is. I can almost hear you thinking now. "But I can change a Python variable, even if it's an integer!" You're right, but that's something different though.

Let's look at an example where we assign the string 'hello' to a variable called `mystring`, and then change it:

```
>>> mystring = 'hello'
>>> mystring =  'world'
```

What we are doing is reassigning a new value to a variable. We're not changing the string 'hello' itself.

There's another way to explain this. A variable points to a spot in your computer's memory. That is what we call a pointer. In the first instance, `mystring` points to a spot in memory where we stored the string 'hello.' After changing `mystring` to 'world,' it points to another spot in memory where we store the word 'world.' We didn't change the string 'hello.'

In fact, we can prove this by doing the following:

```
>>> mystring = 'hello'
>>> mycopy = mystring
>>> mystring =  'world'
>>> print(mycopy)
'hello'
```

We created a second variable pointing to the string 'hello'. When we changed `mystring` to point to a different string, we could still reference the previous string because `mycopy` also points to the 'hello' string's location in memory.

This is different from a list, for example. If the variable `mylist` points to a list structure in memory, and we change that list, it still points to the same list structure. All we do is change the list structure itself (its content). Python doesn't replace the list but modifies it instead.

How to check the Python data type?

There's a built-in function called `type` which you can use to check data types in Python. Let's look at some examples of `type` at work:

```
>>> type(3)
<class 'int'>
>>> type('hello')
<class 'str'>
>>> type([1,2,3])
<class 'list'>
```

If you are experimenting in the REPL, `type` is a valuable function that can give you more insight into what's happening under the hood!

Get certified with our courses

Learn Python properly through small, easy-to-digest lessons, progress tracking, quizzes to test your knowledge, and practice sessions. Each course will earn you a downloadable course certificate.

‹ Previous: Python Packages

Next: Python Integer