**Data model**

[The standard type hierarchy](https://docs.python.org/3.9/reference/datamodel.html#objects-values-and-types)

Below is a list of the types that are built into Python. Extension modules (written in C, Java, or other languages, depending on the implementation) can define additional types.

**None**

This type has a single value. There is a single object with this value. This object is accessed through the built-in name None. It is used to signify the absence of a value in many situations, e.g., it is returned from functions that don’t explicitly return anything. Its truth value is false.

**Sequences**

These represent finite ordered sets indexed by non-negative numbers. The built-in function [len()](https://docs.python.org/3.9/library/functions.html" \l "len" \o "len) returns the number of items of a sequence. When the length of a sequence is *n*, the index set contains the numbers 0, 1, …, *n*-1. Item *i* of sequence *a* is selected by a[i].

**Immutable sequences**

An object of an immutable sequence type cannot change once it is created. (If the object contains references to other objects, these other objects may be mutable and may be changed; however, the collection of objects directly referenced by an immutable object cannot change.)

The following types are immutable sequences:

[**Strings**](https://docs.python.org/3/library/stdtypes.html?highlight=Lists#str)

A string is a sequence of values that represent Unicode code points. All the code points in the range U+0000 - U+10FFFF can be represented in a string. Python doesn’t have a char type; instead, every code point in the string is represented as a string object with length 1.

Textual data in Python is handled with [str](https://docs.python.org/3/library/stdtypes.html?highlight=Lists#str) objects, or *strings*. String literals are written in a variety of ways:

* Single quotes: 'allows embedded "double" quotes'
* Double quotes: "allows embedded 'single' quotes".
* Triple quoted: '''Three single quotes''', """Three double quotes"""

Triple quoted strings may span multiple lines - all associated whitespace will be included in the string literal.

[**Tuples**](https://docs.python.org/3/library/stdtypes.html?highlight=tuple#tuple)

The items of a tuple are arbitrary Python objects. Tuples of two or more items are formed by comma-separated lists of expressions. A tuple of one item (a ‘singleton’) can be formed by affixing a comma to an expression (an expression by itself does not create a tuple, since parentheses must be usable for grouping of expressions). The parentheses are optional, except in the empty tuple case, or when they are needed to avoid syntactic ambiguity.

Tuples may be constructed in a number of ways:

* Using a pair of parentheses to denote the empty tuple: ()
* Using a trailing comma for a singleton tuple: a, or (a,)
* Separating items with commas: a, b, c or (a, b, c)
* Using the [tuple()](https://docs.python.org/3/library/stdtypes.html?highlight=tuple#tuple) built-in: tuple() or tuple(iterable)

tupleInitSingleton = 'a',

tupleInitSingleton[0] + 'b'

[**Bytes**](https://docs.python.org/3/library/stdtypes.html?highlight=Bytes#bytes)

A bytes object is an immutable array. The items are 8-bit bytes, represented by integers in the range 0 <= x < 256. Bytes literals (like b'abc') and the built-in [bytes()](https://docs.python.org/3.9/library/stdtypes.html#bytes) constructor can be used to create bytes objects. Also, bytes objects can be decoded to strings via the [decode()](https://docs.python.org/3.9/library/stdtypes.html#bytes.decode) method.

**Mutable sequences**

Mutable sequences can be changed after they are created. The subscription and slicing notations can be used as the target of assignment and [del](https://docs.python.org/3.9/reference/simple_stmts.html#del) (delete) statements.

There are currently two intrinsic mutable sequence types:

[**Lists**](https://docs.python.org/3/library/stdtypes.html?highlight=Lists#lists)

The items of a list are arbitrary Python objects. Lists are formed by placing a comma-separated list of expressions in square brackets. (Note that there are no special cases needed to form lists of length 0 or 1.)

Lists may be constructed in several ways:

* Using a pair of square brackets to denote the empty list: []
* Using square brackets, separating items with commas: [a], [a, b, c]
* Using a list comprehension: [x for x in iterable]
* Using the type constructor: list() or list(iterable)

[**Byte Arrays**](https://docs.python.org/3/library/stdtypes.html?highlight=Bytearray#bytearray)

A bytearray object is a mutable array. They are created by the built-in [bytearray()](https://docs.python.org/3.9/library/stdtypes.html" \l "bytearray" \o "bytearray) constructor. Aside from being mutable (and hence unhashable), byte arrays otherwise provide the same interface and functionality as immutable [bytes](https://docs.python.org/3.9/library/stdtypes.html#bytes) objects.

The extension module [array](https://docs.python.org/3.9/library/array.html#module-array) provides an additional example of a mutable sequence type, as does the [collections](https://docs.python.org/3.9/library/collections.html#module-collections) module.

**Set types**

These represent unordered, finite sets of unique, immutable objects. As such, they cannot be indexed by any subscript. However, they can be iterated over, and the built-in function [len()](https://docs.python.org/3.9/library/functions.html" \l "len" \o "len) returns the number of items in a set. Common uses for sets are fast membership testing, removing duplicates from a sequence, and computing mathematical operations such as intersection, union, difference, and symmetric difference.

For set elements, the same immutability rules apply as for dictionary keys. Note that numeric types obey the normal rules for numeric comparison: if two numbers compare equal (e.g., 1 and 1.0), only one of them can be contained in a set.

There are currently two intrinsic set types:

[**Sets**](https://docs.python.org/3/library/stdtypes.html?highlight=Dictionaries#set)

These represent a **mutable** set. They are created by the built-in [set()](https://docs.python.org/3.9/library/stdtypes.html#set) constructor and can be modified afterwards by several methods, such as add().

Sets can be created by several means:

* Use a comma-separated list of elements within braces: {'jack', 'sjoerd'}
* Use a set comprehension: {c for c in 'abracadabra' if c not in 'abc'}
* Use the type constructor: set(), set('foobar'), set(['a', 'b', 'foo'])

Like other collections, sets support x in set,

[**Frozen sets**](https://docs.python.org/3/library/stdtypes.html?highlight=Dictionaries#frozenset)

These represent an **immutable** set. They are created by the built-in [frozenset()](https://docs.python.org/3.9/library/stdtypes.html" \l "frozenset" \o "frozenset) constructor. As a frozenset is immutable and [hashable](https://docs.python.org/3.9/glossary.html" \l "term-hashable), it can be used again as an element of another set, or as a dictionary key.

**Mappings**

These represent finite sets of objects indexed by arbitrary index sets. The subscript notation a[k] selects the item indexed by k from the mapping a; this can be used in expressions and as the target of assignments or [del](https://docs.python.org/3.9/reference/simple_stmts.html#del) statements. The built-in function [len()](https://docs.python.org/3.9/library/functions.html" \l "len" \o "len) returns the number of items in a mapping.

There is currently a single intrinsic mapping type:

[**Dictionaries**](https://docs.python.org/3/library/stdtypes.html?highlight=Dictionaries#dict)

These represent finite sets of objects indexed by nearly arbitrary values. The only types of values not acceptable as keys are values containing lists or dictionaries or other mutable types that are compared by value rather than by object identity, the reason being that the efficient implementation of dictionaries requires a key’s hash value to remain constant. Numeric types used for keys obey the normal rules for numeric comparison: if two numbers compare equal (e.g., 1 and 1.0) then they can be used interchangeably to index the same dictionary entry.

Dictionaries preserve insertion order, meaning that keys will be produced in the same order they were added sequentially over the dictionary. Replacing an existing key does not change the order, however removing a key and re-inserting it will add it to the end instead of keeping its old place.

**Dictionaries are mutable**; they can be created by the {...} notation (see section [Dictionary displays](https://docs.python.org/3.9/reference/expressions.html#dict)).

The extension modules [dbm.ndbm](https://docs.python.org/3.9/library/dbm.html" \l "module-dbm.ndbm" \o "dbm.ndbm: The standard \"database\" interface, based on ndbm. (Unix)) and [dbm.gnu](https://docs.python.org/3.9/library/dbm.html" \l "module-dbm.gnu" \o "dbm.gnu: GNU's reinterpretation of dbm. (Unix)) provide additional examples of mapping types, as does the [collections](https://docs.python.org/3.9/library/collections.html#module-collections) module.

*Changed in version 3.7:*Dictionaries did not preserve insertion order in versions of Python before 3.6. In CPython 3.6, insertion order was preserved, but it was considered an implementation detail at that time rather than a language guarantee.

Dictionaries can be created by several means:

* Use a comma-separated list of key: value pairs within braces: {'jack': 4098, 'sjoerd': 4127} or {4098: 'jack', 4127: 'sjoerd'}
* Use a dict comprehension: {}, {x: x \*\* 2 for x in range(10)}
* Use the type constructor: dict(), dict([('foo', 100), ('bar', 200)]), dict(foo=100, bar=200)

## Glossary

## Container

## Containers are any object that holds an arbitrary number of other objects. Generally, containers provide a way to access the contained objects and to iterate over them.

## Python’s general purpose built-in containers, [dict](https://docs.python.org/3.9/library/stdtypes.html" \l "dict" \o "dict), [list](https://docs.python.org/3.9/library/stdtypes.html#list), [set](https://docs.python.org/3.9/library/stdtypes.html#set), and [tuple](https://docs.python.org/3.9/library/stdtypes.html#tuple).