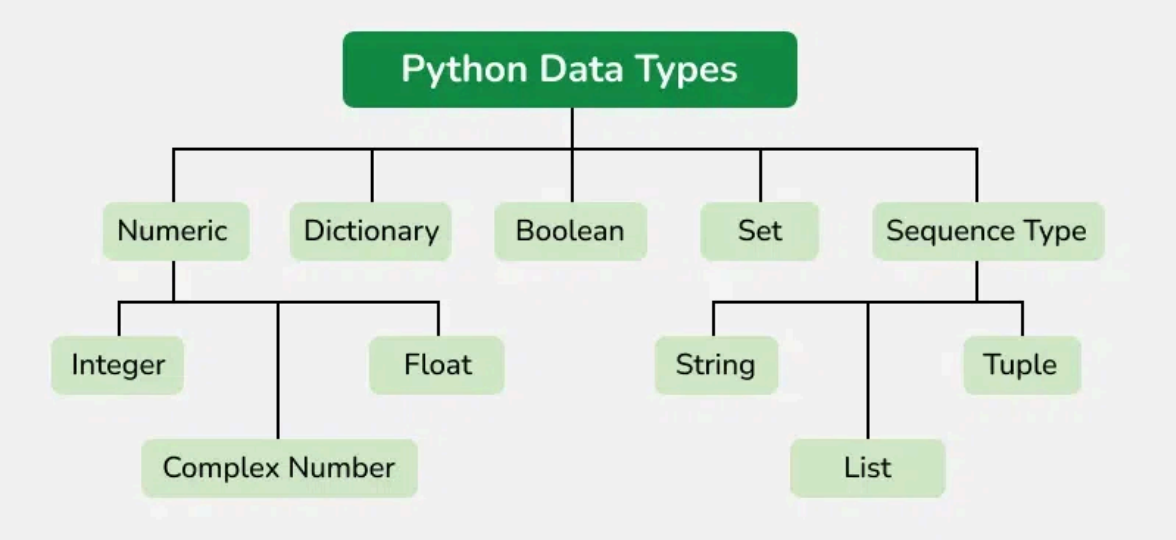
Python Data Types

Last Updated : 15 Oct, 2025

Data types in Python are a way to classify data items. They represent the kind of value, which determines what operations can be performed on that data. Since everything is an object in Python programming, Python data types are classes and variables are instances (objects) of these classes.

The following are standard or built-in data types in Python:

- **Numeric:** int, float, complex
- **Sequence Type:** string, list, tuple
- **Mapping Type:** dict
- **Boolean:** bool
- **Set Type:** set, frozenset
- **Binary Types:** bytes, bytearray, memoryview



DataTypes

Below code assigns variable 'x' different values of few Python data types - int, float, list, tuple and string. Each assignment replaces previous value, making 'x' take on data type and value of most recent assignment.

```
x = 50  # int
x = 60.5  # float
x = "Hello World"  # string
x = ["geeks", "for", "geeks"]  # list
x = ("geeks", "for", "geeks")  # tuple
```

1. Numeric Data Types

Python numbers represent data that has a numeric value. A numeric value can be an integer, a floating number or even a complex number. These values are defined as int, float and complex classes.

- **Integers:** value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). There is no limit to how long an integer value can be.

- **Float:** value is represented by float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- **Complex Numbers:** It is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example - 2+3j

```
a = 5
print(type(a))

b = 5.0
print(type(b))

c = 2 + 4j
print(type(c))
```

Loading Playground...

Output

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

2. Sequence Data Types

A sequence is an ordered collection of items, which can be of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence data types of Python:

String Data Type

Python Strings are arrays of bytes representing Unicode characters. In Python, there is no character data type, a character is a string of length one. It is represented by str class.

Strings in Python can be created using single quotes, double quotes or even triple quotes. We can access individual characters of a String using index.

```
s = 'Welcome to the Geeks World'
print(s)

# check data type
print(type(s))

# access string with index
print(s[1])
print(s[2])
print(s[-1])
```

Loading Playground...

Output

```
Welcome to the Geeks World
<class 'str'>
e
l
d
```

List Data Type

Lists are similar to arrays found in other languages. They are an ordered and mutable collection of items. It is very flexible as items in a list do not need to be of the same type.

Creating a List in Python

Lists in Python can be created by just placing sequence inside the square brackets[].

```
# Empty list
a = []

# list with int values
a = [1, 2, 3]
print(a)

# list with mixed values int and String
b = ["Geeks", "For", "Geeks", 4, 5]
print(b)
```

Loading Playground...

Output

```
[1, 2, 3]
['Geeks', 'For', 'Geeks', 4, 5]
```

Access List Items

In order to access the list items refer to index number. In Python, negative sequence indexes represent positions from end of the array. Instead of having to compute offset as in List[len(List)-3], it is enough to just write List[-3]. Negative indexing means beginning from end, -1 refers to last item, -2 refers to second-last item, etc.

```
a = ["Geeks", "For", "Geeks"]
print("Accessing element from the list")
print(a[0])
print(a[2])

print("Accessing element using negative indexing")
print(a[-1])
print(a[-3])
```

Loading Playground...

Output

```
Accessing element from the list
Geeks
Geeks
Accessing element using negative indexing
Geeks
Geeks
```

Tuple Data Type

Tuple is an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable. Tuples cannot be modified after it is created.

Creating a Tuple in Python

In Python, tuples are created by placing a sequence of values separated by a 'comma' with or without the use of parentheses for grouping data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, lists, etc.).

> ***Note:*** *Tuples can also be created with a single element, but it is a bit tricky. Having one element in the parentheses is not sufficient, there must be a trailing* ***'comma'*** *to make it a tuple.*

```
# initiate empty tuple
tup1 = ()

tup2 = ('Geeks', 'For')
print("\nTuple with the use of String: ", tup2)
```

Loading Playground...

Output

```
Tuple with the use of String:  ('Geeks', 'For')
```

> ***Note*** *- The creation of a Python tuple without the use of parentheses is known as Tuple Packing.*

Access Tuple Items

In order to access tuple items refer to the index number. Use the index operator [ ] to access an item in a tuple.

```
tup1 = (1, 2, 3, 4, 5)

# access tuple items
print(tup1[0])
print(tup1[-1])
print(tup1[-3])
```

Loading Playground...

Output

```
1
5
3
```

3. Boolean Data Type

Python Boolean Data type is one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true) and those equal to False are falsy (false). However non-Boolean objects can be evaluated in a Boolean context as well and determined to be true or false. It is denoted by class bool.

```
print(type(True))
print(type(False))
print(type(true))
```

Loading Playground...

Output:

<class 'bool'>
<class 'bool'>

```
Traceback (most recent call last):
  File "/home/7e8862763fb66153d70824099d4f5fb7.py", line 8, in
    print(type(true))
NameError: name 'true' is not defined
```

Truthy and Falsy Values

In Python, truthy and falsy values are values that evaluate to True or False in a Boolean context. Truthy values behave like True, while falsy values behave like False when used in conditions.

```
if 1:
    print("1 is truthy")

if not 0:
    print("0 is falsy")
```

Loading Playground...

Output

```
1 is truthy
0 is falsy
```

4. Set Data Type

In Python Data Types, Set is an unordered collection of data types that is iterable, mutable, and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.

Create a Set in Python

Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by a 'comma'. The type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

```
# initializing empty set
s1 = set()

s1 = set("GeeksForGeeks")
print("Set with the use of String: ", s1)

s2 = set(["Geeks", "For", "Geeks"])
print("Set with the use of List: ", s2)
```

Loading Playground...

Output

```
Set with the use of String:  {'s', 'o', 'F', 'G', 'e', 'k', 'r'}
Set with the use of List:  {'Geeks', 'For'}
```

Access Set Items

Set items cannot be accessed by referring to an index, since sets are unordered the items have no index. But we can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the keyword in.

```
set1 = set(["Geeks", "For", "Geeks"]) #Duplicates are removed automatically
print(set1)

# loop through set
for i in set1:
   print(i, end=" ") #prints elements one by one

# check if item exist in set
print("Geeks" in set1)
```

Loading Playground...

Output

```
{'For', 'Geeks'}
For Geeks True
```

5. Dictionary Data Type

A dictionary in Python is a collection of data values, used to store data values like a map, unlike other Python Data Types, a Dictionary holds a key: value pair. Key-value is provided in dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon : , whereas each key is separated by a 'comma'.

Create a Dictionary in Python

Values in a dictionary can be of any datatype and can be duplicated, whereas keys can't be repeated and must be immutable. The dictionary can also be created by the built-in function **dict().**

> ***Note*** *- Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.*

```
# initialize empty dictionary
d = {}

d = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print(d)

# creating dictionary using dict() constructor
d1 = dict({1: 'Geeks', 2: 'For', 3: 'Geeks'})
print(d1)
```

Loading Playground...

Output

```
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
```

Accessing Key-value in Dictionary

In order to access items of a dictionary refer to its key name. Key can be used inside square brackets. Using get() method we can access dictionary elements.

```
d = {1: 'Geeks', 'name': 'For', 3: 'Geeks'}

# Accessing an element using key
print(d['name'])
```

```
# Accessing a element using get
print(d.get(3))
```

Loading Playground...

Output

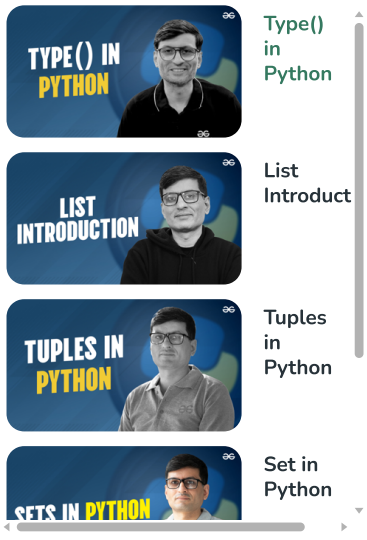
```
For
Geeks
```

Recommended Problems:

- Input In Python
- Find index
- Decimal to Binary
- Type Conversion
- TypeCast And Double It

Related Links:

- Quiz on Python Data Type
- Convert integer to string
- Convert String to Int
- Convert float to exponential
- Convert int to exponential
- Primitive vs Non Primitive Data Types



Type() in Python

Visit Course

Suggested Quiz

9 Questions

Which of the following is a mutable data type in Python?

A int

B str

C list

D tuple

Login to View Explanation

1/9

< Previous Next >

Comment

N nikhila... + Follow

272

Article Tags: Python Python-datatype