# Python Data Types

Python data types are different in some aspects from other programming languages. It is simple to understand and easy to use. Because Python is interpreted programming language and Python interpreter can determine which type of data are storing, so no need to define the data type of memory location.

Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

- Numbers
- String
- List
- Tuple
- Dictionary

## Numbers

Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.

Number stores numeric values. Python creates Number objects when a number is assigned to a variable. Complex numbers are written with a "j" as the imaginary part. To verify the type of any object in Python, use the type() function.

| Example | Output |
|---|---|
| a = 10   #integer type<br>print(a)<br>print(type(a))<br><br>a = 20   #integer type<br>print(a)<br>print(type(a))<br><br>a = 10.12   #float type<br>print(a)<br>print(type(a))<br><br>a= 10+0j   #complex type<br>print(a)<br>print(type(a)) | 10<br><type 'int'><br>20<br><type 'int'><br>10.12<br><type 'float'><br>10+0j<br><type 'complex'> |

## String

The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end. The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

| **Example** | **Output** |
|---|---|
| str = 'Hello Python'<br>print str<br>print str[0]<br>print str[1:4]<br>print str[4:]<br>print str * 3<br>print str + "PYTHON" | Hello Python<br>H<br>ell<br>o Python<br>Hello PythonHello PythonHello Python<br>Hello PythonPYTHON |

## List

Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) works with the list in the same way as they were working with the strings.

| Example | Output |
|---|---|
| list = [ 'abc', 123 , 'hi', 10.12 , 10 ]<br><br>print list<br>print list[0]<br>print list[2:4]<br>print list[2:]<br>print list * 2<br>print list + list | ['abc', 123, 'hi', 10.12, 10]<br>abc<br>['hi', 10.12]<br>['hi', 10.12, 10]<br>['abc', 123, 'hi', 10.12, 10, 'abc', 123, 'hi', 10.12, 10]<br><br>['abc', 123, 'hi', 10.12, 10, 'abc', 123, 'hi', 10.12, 10] |

## Tuple

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses (). A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

| Example | Output |
|---|---|
| tuple = ( 'abc', 123 , 'hi', 10.12 , 10 )<br><br>print tuple<br>print tuple[0]<br>print tuple[2:4]<br>print tuple[3:]<br>print tuple * 2<br>print tuple + tuple | ('abc', 123, 'hi', 10.12, 10)<br><br>abc<br><br>('hi', 10.12)<br><br>(10.12, 10)<br><br>('abc', 123, 'hi', 10.12, 10, 'abc', 123, 'hi', 10.12, 10)<br><br>('abc', 123, 'hi', 10.12, 10, 'abc', 123, 'hi', 10.12, 10) |

# Python Dictionary

Dictionary is an ordered set of a key-value pair of items. They work like associative arrays or hashes and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

| **Example** | **Output** |
|---|---|
| d = {1:'abc', 2:'xyz', 3:'pqr'}; | abc |
| print(d[1]); | pqr |
| print(d[3]); | {1: 'abc', 2: 'xyz', 3: 'pqr'} |
| print (d); | [1, 2, 3] |
| print (d.keys()); | ['abc', 'xyz', 'pqr'] |
| print (d.values()); | |