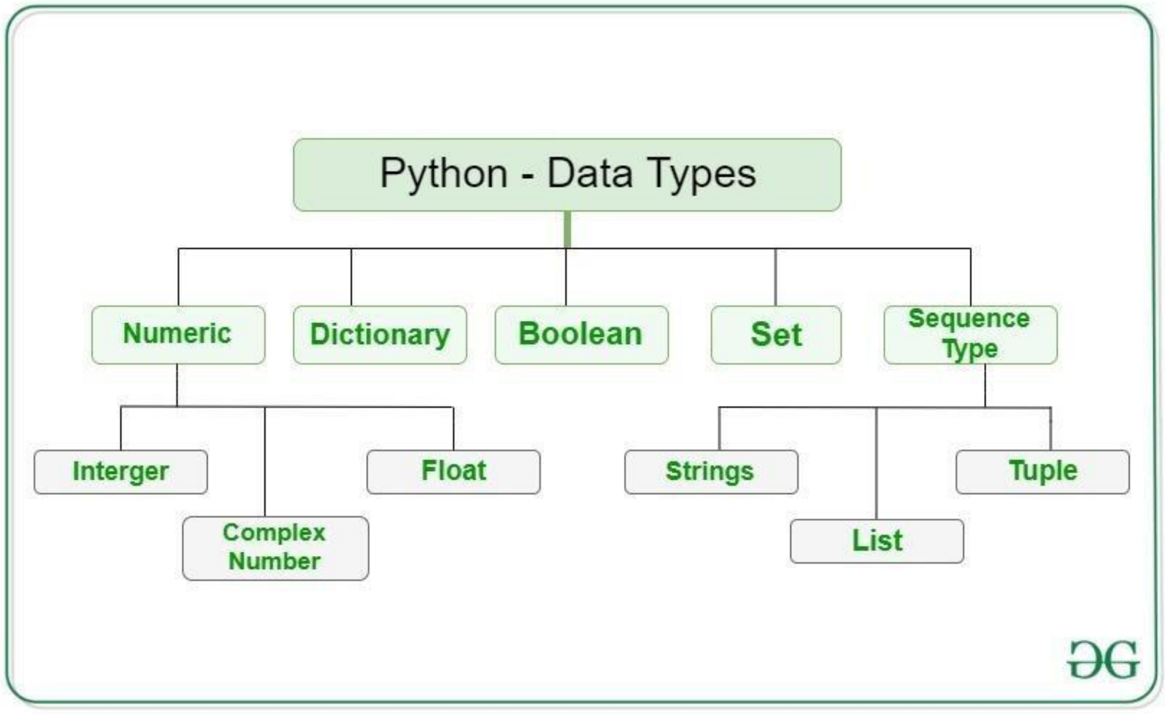
Python Data Types

-

Python Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular 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 the standard or built-in data types in Python:

- **Numeric**
- **Sequence Type**
- **Boolean**
- **Set**
- **Dictionary**
- **Binary Types**



1. Numeric Data Types in Python

The numeric data type in Python represents the 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 Python int, Python float, and Python complex classes in Python.

- **Integers** – This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.

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

2. Sequence Data Types in Python

The sequence Data Type in Python is the ordered collection of similar or different Python data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence data types of Python:

- Python String
- Python List
- Python Tuple

String Data Type

Strings in Python are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote, or triple-quote. In Python, there is no character data type Python, a character is a string of length one. It is represented by str class.

Creating String

Strings in Python can be created using single quotes, double quotes, or even triple quotes.

Accessing elements of String

In Python programming, individual characters of a String can be accessed by using the method of Indexing. Negative Indexing allows negative address references to access characters from the back of the String, e.g. -1 refers to the last character, -2 refers to the second last character, and so on.

List Data Type

Lists are just like arrays, declared in other languages which is an ordered collection of data. It is very flexible as the 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 the sequence inside the square brackets[].

List Methods in Python

S.no	Method	Description
1	append()	Used for adding elements to the end of the List.
2	copy()	It returns a shallow copy of a list
3	clear()	This method is used for removing all items from the list.
4	count()	These methods count the elements.
5	extend()	Adds each element of an iterable to the end of the List
6	index()	Returns the lowest index where the element appears.
7	insert()	Inserts a given element at a given index in a list.

Tuple Data Type

Just like a list, a tuple is also an ordered collection of Python objects. The only difference between a tuple and a list is that tuples are immutable i.e. tuples cannot be modified after it is created. It is represented by a tuple class.

Creating a Tuple in Python

In Python Data Types, tuples are created by placing a sequence of values separated by a „comma" with or without the use of parentheses for grouping the 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.

Python has two built-in methods that you can use on tuples.

Method	**Description**
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

3. Boolean Data Type in Python

Python Data type with 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 the class bool.

Note – True and False with capital „T" and „F" are valid booleans otherwise python will throw an error.

4. Set Data Type in Python

In Python Data Types, a 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.

Access Set Items

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

Note – To know more about sets, refer to Python Sets.

5. Dictionary Data Type in Python

A dictionary in Python is an unordered collection of data values, used to store data values like a map, unlike other Python Data Types that hold only a single value as an element, a Dictionary holds a key: value pair. Key-value is provided in the 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

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by „comma‟. 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().** An empty dictionary can be created by just placing it in curly braces{}. **Note** – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

Accessing Key-value in Dictionary

In order to access the items of a dictionary refer to its key name. Key can be used inside square brackets. There is also a method called **get()** that will also help in accessing the element from a dictionary.

Python Operators

In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into different types of **Python operators.**

- OPERATORS: These are the special symbols. Eg- + , * , /, etc.
- OPERAND: It is the value on which the operator is applied.

Types of Operators in Python

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators
6. Identity Operators and Membership Operators

Python Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations such as addition, subtraction, multiplication, etc.

Assume variable a holds 10 and variable b holds 20, then

Operator	Name	Example
+	Addition	a + b = 30
-	Subtraction	a – b = -10
*	Multiplication	a * b = 200
/	Division	b / a = 2
%	Modulus	b % a = 0
**	Exponent	a**b =10**20
//	Floor Division	9//2 = 4

Python Comparison Operators

Comparison operators compare the values on either side of them and decide the relation among them. They are also called Relational operators.

Assume variable **a** holds 10 and variable **b** holds 20, then

Operator	Name	Example
==	Equal	(a == b) is not true.
!=	Not equal	(a != b) is true.

>	Greater than	(a > b) is not true.
<	Less than	(a < b) is true.
>=	Greater than or equal to	(a >= b) is not true.
<=	Less than or equal to	(a <= b) is true.

Python Assignment Operators

Assignment operators are used to assign values to variables. Following is a table which shows all Python assignment operators.

Operator	**Example**	**Same As**
=	a = 10	a = 10
+=	a += 30	a = a + 30
-=	a -= 15	a = a - 15
*=	a *= 10	a = a * 10
/=	a /= 5	a = a / 5
%=	a %= 5	a = a % 5
**=	a **= 4	a = a ** 4
//=	a //= 5	a = a // 5
&=	a &= 5	a = a & 5
|=	a |= 5	a = a | 5

^=	a ^= 5	a = a ^ 5
>>=	a >>= 5	a = a >> 5
<<=	a <<= 5	a = a << 5

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. These operators are used to compare binary numbers.

There are following Bitwise operators supported by Python language

Operator	**Name**	**Example**
&	AND	a & b
|	OR	a | b
^	XOR	a ^ b
~	NOT	~a
<<	Zero fill left shift	a << 3
>>	Signed right shift	a >> 3

Python Logical Operators

Python logical operators are used to combile two or more conditions and check the final result. There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

Operator	**Name**	**Example**

and	AND	a and b
or	OR	a or b
not	NOT	not(a)

Python Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below −

Operator	Description	Example
in	Returns True if it finds a variable in the specified sequence, false otherwise.	a in b
not in	returns True if it does not finds a variable in the specified sequence and false otherwise.	a not in b

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below −

Operator	Description	Example
is	Returns True if both variables are the same object and false otherwise.	a is b
is not	Returns True if both variables are not the same object and false otherwise.	a is not b

Python Variables

Python Variable is containers that store values. Python is not "statically typed". We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. It is the basic unit of storage in a program. In this article, we will see how to define a variable in Python.

In a Python class, we can define three types of methods:

- Instance methods
- Class methods
- Static methods

Instance methods

Instance methods are the most used methods in a Python class. These methods are only accessible through class objects. If we want to modify any class variable, this should be done inside an instance method.

Class methods

Class methods are usually used to access class variables. You can call these methods directly using the class name instead of creating an object of that class.

To declare a class method, we need to use the `@classmethod` decorator. Also, as in the case of instance methods, `self` is the keyword used to access the class variables. In class methods, we use use the `cls` variable to refer to the class.

Static methods

Static methods are usually used as a utility function or when we do not want an inherited class to modify a function definition. These methods do not have any relation to the class variables and instance variables; so, are not allowed to modify the class attributes inside a static method.

To declare a static method, we need to use the `@staticmethod`. Again, we will be using the `cls` variable to refer to the class. These methods can be accessed using the class name as well as class objects.