**Python**

**Python** is a widely used programming language that offers several unique features and advantages compared to languages like **Java** and **C++.**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas the other languages use punctuations. It has fewer syntactical constructions than other languages.

• Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

• Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

• Python is Object-Oriented: Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

• Python is a Beginner's Language: Python is a great language for the beginnerlevel programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

In the late 1980s, [**Guido van Rossum**](https://en.wikipedia.org/wiki/Guido_van_Rossum) dreamed of developing Python. The first version of **Python 0.9.0 was released in 1991**.

**What is Python**

Python is a general-purpose, dynamically typed, high-level, compiled and interpreted, garbage-collected, and purely object-oriented programming language that supports procedural, object-oriented, and functional programming.

## Features of Python:

* **Easy to use and Read -** Python's syntax is clear and easy to read, making it an ideal language for both beginners and experienced programmers. This simplicity can lead to faster development and reduce the chances of errors.
* **Dynamically Typed** - The data types of variables are determined during run-time. We do not need to specify the data type of a variable during writing codes.
* **High-level** - High-level language means human readable code.
* **Compiled and Interpreted** - Python code first gets compiled into bytecode, and then interpreted line by line. When we download the Python in our system form [org](https://www.python.org/) we download the default implement of Python known as CPython. CPython is considered to be Complied and Interpreted both.
* **Garbage Collected** - Memory allocation and de-allocation are automatically managed. Programmers do not specifically need to manage the memory.
* **Purely Object-Oriented** - It refers to everything as an object, including numbers and strings.
* **Cross-platform Compatibility** - Python can be easily installed on Windows, macOS, and various Linux distributions, allowing developers to create software that runs across different operating systems.
* **Rich Standard Library** - Python comes with several standard libraries that provide ready-to-use modules and functions for various tasks, ranging from **web development** and **data manipulation** to **machine learning** and **networking**.
* **Open Source** - Python is an open-source, cost-free programming language. It is utilized in several sectors and disciplines as a result.

# Python Applications

Python is known for its general-purpose nature that makes it applicable in almost every domain of software development. Python makes its presence in every emerging field. It is the fastest-growing programming language and can develop any application.

### Web Applications

### 2) Desktop GUI Applications

### 3) Software Development

### 4) Business Applications

# Python Variables

A variable is the name given to a memory location. A value-holding Python variable is also known as an identifier.

Since Python is an infer language that is smart enough to determine the type of a variable, we do not need to specify its type in Python.

Variable names must begin with a letter or an underscore, but they can be a group of both letters and digits.

The name of the variable should be written in lowercase. Both Rahul and rahul are distinct variables.

* The variable's first character must be an underscore or alphabet (\_).
* Every one of the characters with the exception of the main person might be a letter set of lower-case(a-z), capitalized (A-Z), highlight, or digit (0-9).
* White space and special characters (!, @, #, %, etc.) are not allowed in the identifier name. ^, &, \*).
* Identifier name should not be like any watchword characterized in the language.
* Names of identifiers are case-sensitive; for instance, my name, and MyName isn't something very similar.
* Examples of valid identifiers: a123, \_n, n\_9, etc.
* Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values

* Python doesn't tie us to pronounce a variable prior to involving it in the application. It permits us to make a variable at the necessary time.
* In Python, we don't have to explicitly declare variables. The variable is declared automatically whenever a value is added to it.
* The equal (=) operator is utilized to assign worth to a variable.

# Python Operators

## Introduction:

The operator is a symbol that performs a specific operation between two operands, according to one definition. Operators serve as the foundation upon which logic is constructed in a program in a particular programming language. In every programming language, some operators perform several tasks. Same as other languages, Python also has some operators, and these are given below

* Arithmetic operators
* Comparison operators
* Assignment Operators
* Logical Operators
* Bitwise Operators
* Membership Operators
* Identity Operators
* Arithmetic Operators

## Arithmetic Operators

Arithmetic operators used between two operands for a particular operation. There are many arithmetic operators. It includes the exponent (\*\*) operator as well as the + (addition), - (subtraction), \* (multiplication), / (divide), % (reminder), and // (floor division) operators.

Consider the following table for a detailed explanation of arithmetic operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| **+ (Addition)** | It is used to add two operands. For example, if a = 10, b = 10 => a+b = 20 |
| **- (Subtraction)** | It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 5 => a - b = 15 |
| **/ (divide)** | It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2.0 |
| **\* (Multiplication)** | It is used to multiply one operand with the other. For example, if a = 20, b = 4 => a \* b = 80 |
| **% (reminder)** | It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0 |
| **\*\* (Exponent)** | As it calculates the first operand's power to the second operand, it is an exponent operator. |
| **// (Floor division)** | It provides the quotient's floor value, which is obtained by dividing the two operands. |

## Comparison operator

Comparison operators mainly use for comparison purposes. Comparison operators compare the values of the two operands and return a true or false Boolean value in accordance. The example of comparison operators are ==, !=, <=, >=, >, <. In the below table, we explain the works of the operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| == | If the value of two operands is equal, then the condition becomes true. |
| != | If the value of two operands is not equal, then the condition becomes true. |
| <= | The condition is met if the first operand is smaller than or equal to the second operand. |
| >= | The condition is met if the first operand is greater than or equal to the second operand. |
| > | If the first operand is greater than the second operand, then the condition becomes true. |
| **<** | If the first operand is less than the second operand, then the condition becomes true. |

## Assignment Operators

Using the assignment operators, the right expression's value is assigned to the left operand. There are some examples of assignment operators like =, +=, -=, \*=, %=, \*\*=, //=. In the below table, we explain the works of the operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | It assigns the value of the right expression to the left operand. |
| += | By multiplying the value of the right operand by the value of the left operand, the left operand receives a changed value. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30. |
| -= | It decreases the value of the left operand by the value of the right operand and assigns the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10. |
| \*= | It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if a = 10, b = 20 => a\* = b will be equal to a = a\* b and therefore, a = 200. |
| %= | It divides the value of the left operand by the value of the right operand and assigns the reminder back to the left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0. |
| \*\*= | a\*\*=b will be equal to a=a\*\*b, for example, if a = 4, b =2, a\*\*=b will assign 4\*\*2 = 16 to a. |
| //= | A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a. |

## Bitwise Operators

The two operands' values are processed bit by bit by the bitwise operators. The examples of Bitwise operators are bitwise OR (|), bitwise AND (&), bitwise XOR (^), negation (~), Left shift (<<), and Right shift (>>).

|  |  |
| --- | --- |
| **Operator** | **Description** |
| & (binary and) | A 1 is copied to the result if both bits in two operands at the same location are 1. If not, 0 is copied. |
| | (binary or) | The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1. |
| ^ (binary xor) | If the two bits are different, the outcome bit will be 1, else it will be 0. |
| ~ (negation) | The operand's bits are calculated as their negations, so if one bit is 0, the next bit will be 1, and vice versa. |
| << (left shift) | The number of bits in the right operand is multiplied by the leftward shift of the value of the left operand. |
| >> (right shift) | The left operand is moved right by the number of bits present in the right operand. |

## Logical Operators

The assessment of expressions to make decisions typically uses logical operators. The examples of logical operators are and, or, and not. In the case of logical AND, if the first one is 0, it does not depend upon the second one. In the case of logical OR, if the first one is 1, it does not depend on the second one. Python supports the following logical operators. In the below table, we explain the works of the logical operators.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| and | The condition will also be true if the expression is true. If the two expressions a and b are the same, then a and b must both be true. |
| or | The condition will be true if one of the phrases is true. If a and b are the two expressions, then an or b must be true if and is true and b is false. |
| not | If an expression **a** is true, then not (a) will be false and vice versa. |

## Membership Operators

The membership of a value inside a Python data structure can be verified using Python membership operators. The result is true if the value is in the data structure; otherwise, it returns false.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| in | If the first operand cannot be found in the second operand, it is evaluated to be true (list, tuple, or dictionary). |
| not in | If the first operand is not present in the second operand, the evaluation is true (list, tuple, or dictionary). |

## Identity Operators

## [Python identity operators](https://www.tutorialspoint.com/python/python_identity_operators.htm) compare the memory locations of two objects.

|  |  |
| --- | --- |
| **Operator** | **Description** |
| is | If the references on both sides point to the same object, it is determined to be true. |
| is not | If the references on both sides do not point at the same object, it is determined to be true. |