Python is one of the most popular programming languages. It’s simple to use, packed with features and supported by a wide range of libraries and frameworks. Its clean syntax makes it beginner-friendly.

- A high-level language, used in data science, automation, AI, web development and more.
- Known for its readability, which means code is easier to write, understand and maintain.
- Backed by strong library support, we don’t have to build everything from scratch.

Why Learn Python?

- Requires fewer lines of code compared to other programming languages like Java.
- Provides Libraries / Frameworks like Django, Flask and many more for Web Development, and Pandas, Tensorflow, Scikit-learn and many more for, AI/ML, Data Science and Data Analysis
- Cross-platform, works on Windows, Mac and Linux without major changes.
- Used by top tech companies like Google, Netflix and NASA.
- Many Python coding job opportunities in Software Development, Data Science and AI/ML.

Famous Application Built using Python

- **YouTube:** World's largest video-sharing platform uses Python for features like video streaming and backend services.
- **Instagram:** This popular social media app relies on Python's simplicity for scaling and handling millions of users.
- **Spotify:** Python is used for backend services and machine learning to personalize music recommendations.
- **Dropbox:** The file hosting service uses Python for both its desktop client and server-side operations.
- **Netflix:** Python powers key components of Netflix's recommendation engine and content delivery systems (CDN).
- **Google:** Python is one of the key languages used in Google for web crawling, testing and data analysis.
- **Uber:** Python helps Uber handle dynamic pricing and route optimization using machine learning.
- **Pinterest:** Python is used to process and store huge amounts of image data efficiently.

Advantages of Python

1. **Presence of third-party modules:** Python has a rich ecosystem of third-party modules and libraries that extend its functionality for various tasks.
2. **Extensive support libraries:** Python boasts extensive support libraries like NumPy for numerical calculations and Pandas for data analytics, making it suitable for scientific and data-related applications.

3. **Open source and large active community base:** Python is open source, and it has a large and active community that contributes to its development and provides support.
4. **Versatile, easy to read, learn, and write:** Python is known for its simplicity and readability, making it an excellent choice for both beginners and experienced programmers.
5. **Dynamically typed language:** Python is dynamically typed, meaning you don't need to declare data types explicitly, making it flexible but still reliable.
6. **Object-Oriented and Procedural programming language:** Python supports both object-oriented and procedural programming, providing versatility in coding styles.
7. **Portable and interactive:** Python is portable across operating systems and interactive, allowing real-time code execution and testing.

Disadvantages of Python

1. **Performance:** Python is an interpreted language, which means that it can be slower than compiled languages like C or Java. This can be an issue for performance-intensive tasks.
2. **Global Interpreter Lock:** The Global Interpreter Lock (GIL) is a mechanism in Python that prevents multiple threads from executing Python code at once. This can limit the parallelism and concurrency of some applications.

3. **Memory consumption:** Python can consume a lot of memory, especially when working with large datasets or running complex algorithms.
4. **Dynamically typed:** Python is a dynamically typed language, which means that the types of variables can change at runtime. This can make it more difficult to catch errors and can lead to bugs.
5. **Packaging and versioning:** Python has a large number of packages and libraries, which can sometimes lead to versioning issues and package conflicts.
6. **Lack of strictness:** Python's flexibility can sometimes be a double-edged sword. While it can be great for rapid development and prototyping, it can also lead to code that is difficult to read and maintain.
7. **Not Ideal for System Programming / Embedded Systems / Mobile Development / Frontend:** For real-time constraints, low-level systems programming (drivers, OS kernels, embedded microcontrollers), or very tight performance latency, Python typically isn’t used. There is comparatively weak support or adoption for mobile apps or browser-side code

Python Variables

In Python, variables are used to store data that can be referenced and manipulated during program execution. A variable is essentially a name that is assigned to a value.

Rules for Naming Variables

To use variables effectively, we must follow Python's naming rules:

1. Variable names can only contain letters, digits and underscores (_).
2. A variable name cannot start with a digit.
3. Variable names are case-sensitive like myVar and myvar are different.
4. Avoid using Python keywords like if, else, for as variable names.

Python Operators

In Python programming, Operators in general are used to perform operations on values and variables.

- **Operators:** Special symbols like -, + , * , /, etc.
- **Operands:** Value on which the operator is applied

Arithmetic Operators

Python Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication and division.

In Python 3.x the result of division is a floating-point while in Python 2.x division of 2 integers was an integer. To obtain an integer result in Python 3.x floored (// integer) is used.

Comparison Operators

In Python, Comparison (or Relational) operators compares values. It either returns True or False according to the condition.

Logical Operators

Python Logical operators perform **Logical AND**, **Logical OR** and **Logical NOT** operations. It is used to combine conditional statements.

The precedence of Logical Operators in Python is as follows:

1. Logical not
2. logical and
3. logical or

Bitwise Operators

Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

Bitwise Operators in Python are as follows:

1. Bitwise NOT
2. Bitwise Shift
3. Bitwise AND
4. Bitwise XOR
5. Bitwise OR

Assignment Operators

Python Assignment operators are used to assign values to the variables. This operator is used to assign the value of the right side of the expression to the left side operand.

dentity Operators

In Python, **is** and **is not** are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

is	*True if the operands are identical*
is not	*True if the operands are not identical*

Membership Operators

In Python, **in** and **not in** are the membership operators that are used to test whether a value or variable is in a sequence.

in	*True if value is found in the sequence*
not in	*True if value is not found in the sequence*

Ternary Operator

In Python, Ternary operators also known as conditional expressions are operators that evaluate something based on a condition being true or false. It was added to Python in version 2.5.

It simply allows testing a condition in a **single line** replacing the multiline if-else, making the code compact.

Python Keywords

Keywords in Python are special reserved words that are part of the language itself. They define the rules and structure of Python programs which means you cannot use them as names for your variables, functions, classes or any other identifiers.

The list of keywords are:
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Python Data Types

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

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

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.

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.

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.

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.

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.

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.

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'.

Conditional Statements in Python

Conditional statements in Python are used to execute certain blocks of code based on specific conditions. These statements help control the flow of a program, making it behave differently in different situations.

Short Hand if

Short-hand if statement allows us to write a single-line if statement.

If else Conditional Statement

If Else allows us to specify a block of code that will execute if the condition(s) associated with an if or elif statement evaluates to False. Else block provides a way to handle all other cases that don't meet the specified conditions.

Short Hand if-else

The short-hand if-else statement allows us to write a single-line if-else statement

elif Statement

elif statement in Python stands for "else if." It allows us to check multiple conditions, providing a way to execute different blocks of code based on which condition is true. Using elif statements makes our code more readable and efficient by eliminating the need for multiple nested if statements.

Loops in Python - For, While and Nested Loops

Loops in Python are used to repeat actions efficiently. The main types are For loops (counting through items) and While loops (based on conditions).

For Loop

For loops is used to iterate over a sequence such as a list, tuple, string or range. It allow to execute a block of code repeatedly, once for each item in the sequence.

While Loop

In Python, a while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition becomes false, the line immediately after the loop in the program is executed.

Nested Loops

Python programming language allows to use one loop inside another loop which is called nested loop.

Loop Control Statements

Loop control statements change execution from their normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed. Python supports the following control statements.

Continue Statement

The continue statement in Python returns the control to the beginning of the loop.

Break Statement

The break statement in Python brings control out of the loop.

Pass Statement

We use pass statement in Python to write empty loops. Pass is also used for empty control statements, functions and classes.

Functions

In this section of Python 3 tutorial we'll explore Python function syntax, parameter handling, return values and variable scope. Along the way, we'll also introduce versatile functions like range(), map, filter and lambda functions.

Python Functions

Python Functions are a block of statements that does a specific task. The idea is to put some commonly or repeatedly done task together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.

Types of Function Arguments

Python supports various types of arguments that can be passed at the time of the function call. In Python, we have the following function argument types in Python, Let's explore them one by one.

1. Default Arguments

A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.

2. Keyword Arguments

In keyword arguments, values are passed by explicitly specifying the parameter names, so the order doesn’t matter.

3. Positional Arguments

In positional arguments, values are assigned to parameters based on their order in the function call.

4. Arbitrary Arguments

In Python Arbitrary Keyword Arguments, *args and **kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:

- ***args** in Python (Non-Keyword Arguments)
- ****kwargs** in Python (Keyword Arguments)

Function within Functions

A function defined inside another function is called an inner function (or nested function). It can access variables from the enclosing function’s scope and is often used to keep logic protected and organized.

Anonymous Functions

In Python, an anonymous function means that a function is without a name. As we already know the def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions.