# COMPARISON OF C AND PYTHON

## INTRODUCTION

In the modern age we live in, our lives are so tightly connected with digital technologies. From mobile phones, video games to digital signage in the malls, softwares are everywhere, and it is all because of the power of the programming languages. But all of this did not happen suddenly; it's an evolution that has been going on for many years. Even though most of today's software depends on object-oriented programming languages, the start of the journey of programming languages can be said that it began with the C programming language, which is a procedural programming language.

In the pages that follow, I am going to try to explain as much as possible the inner workings of these two programming languages, C and Python, that power the very foundation of our world. Through this article, I will examine many topics, drawing comparisons and contrasts.

## BACKGROUND

The C programming language is a general purpose, low level, and procedural programming language with static typing features. It is famous for its efficiency, portability and vital role in embedded systems and a lot more. C was initially developed at Bell Labs by Ritchie int 1970s to develop tools designed for Unix systems. Through the years, it has become one of the most used programming languages in the world.

The Python programming language is a high level, interpreted and multi-paradigm programming language with dynamic typing features. Python was released in 1991 thanks to the efforts of Guido van Rossum. He wanted to develop a language focused on code readability and simplicity that takes fewer lines to write compared to C-like languages. This feature makes python one of the most popular programming languages.

## SYNTAX

Syntax in programming languages is very similar to the grammar in a language. It includes a set of rules indicating how a programmer must write their code, containing symbols, keywords, and various expressions to make sure the code is valid. Not following these rules leads to resulting in syntax errors. In other words, it prevents code execution.

C and Python share a lot of syntactic resemblances, but they also have significant differences due to their distinct design principles and programming paradigms.

### PUNCTUATION IN C AND PYTHON

**Semicolons (;):** In C, semicolons are required at the end of each statement. However, Python does not use semicolons, it uses the indentation to keep track of statement termination (e.g., int x = 3;).

**Parentheses (()):** Parentheses are used in both C and Python when grouping the arithmetic expressions, function parameters and calling a function (e.g., (2 + 5)*2).

**Curly Braces ({}):** Curly braces are used in code blocks such as loops, conditionals, functions, structures (e.g., if(condition) {/*code*/}). On the other hand, Python does not use curly braces. Rather, it relies on indentation for code blocks.

**Commas (,):** Commas are used both in C and Python to separate items such as function parameters, arrays (list in python) (e.g., function in python -> def foo(a,b,c) )

**Period (.):** Periods are used in C to access the members of a structure or a union. It is used for accessing the attributes of an object and methods in Python.
(e.g., (In Python) object.foo() , (In C) object**.**member = 5;)

**Square Brackets ([]):** Square Brackets are used when declaring, initializing arrays and accessing the elements of the arrays in C. Square Brackets are used when declaring and accessing elements of a list structure in python.(e.g., (In C)int arr[5]={1,2,3}; (In Python) my_list = **[**1,2,3**]**)

**Colon (:):** Colons are used in switch statements in C. Colons are used to exhibit the indented code blocks such as in loops, conditionals, functions, classes in python. (e.g., if condition**:**)

**Hash / Preprocessor Directives (#):** '#' is used in Python to indicate the comments (e.g., #comment). '#' is used in C right before the statements like '#define' and '#include' for code inclusion.

**Asterisk** (*): Asterisk is used both in C and Python when computing multiplication. Specifically, in Python, it is used when raising a number to its power using a double asterisk and is used when multiplying sequences to repeat elements. It is also specifically used in C to declare a pointer variable and is used when dereferencing a pointer which the python does not have.
(e.g., (In C) int *ptr = 5; (In python) res = 2**4) # 16)

## COMMON TOKENS IN C AND PYTHON

**Keywords:** Keywords are predefined words in a programming language. Each of them is meant to perform a specific task. They cannot be used for naming a variable since compilers need to detect them. The C language supports 32 keywords while Python supports 33 keywords. Here are the keywords.

| C | | | | Python | | | | |
|---|---|---|---|---|---|---|---|---|
| auto | double | int | struct | False | await | else | import | pass |
| break | else | long | switch | None | break | except | in | raise |
| case | enum | register | typedef | True | class | finally | is | return |
| char | extern | return | union | and | continue | for | lambda | try |
| const | float | short | unsigned | as | def | from | nonlocal | while |
| continue | for | signed | void | assert | del | global | not | with |
| default | goto | sizeof | volatile | async | elif | if | or | yield |
| do | if | static | while | | | | | |

**Identifiers**: Like an identity belongs to a unique person, identifiers apply the same rule in programming languages. In Python and C, an identifier is a name given to a function, variable, struct(for c), class(for Python). Once defined, those identifiers can be used in the program to refer to that specific value. But there are some rules for an identifier to be named for both C and Python. Here are some of them.

- It must begin with a letter or underscore.
- It can not be a keyword.
- It can not contain white space.

**Operators**: Operators are tokens that, when applied to variables or objects, they trigger specific computations. There are different types of operators and most of them are the same in the way they are written, but some operators are written differently in C and Python.

**a) Arithmetic Operators:** They perform mathematical tasks in both C and Python. They are written the same in C and Python.

| Operator | C | Python |
|---|---|---|
| Addition | + | + |
| Subtraction | - | - |
| Multiplication | * | * |
| Division | / | / |
| Modulus | % | % |

**b) Rational Operators:** They are used to analyze the relationship between two values. They are written the same in C and Python.

| Operator | C | Python |
|---|---|---|
| Equal to | == | == |
| Not equal to | != | != |
| Less than | < | < |
| Greater than | > | > |
| Less than or equal to | <= | <= |
| Greater than or equal to | >= | >= |

**c) Logical Operators:** They are used to perform logical operations using Boolean values. They are written differently in C and Python.

| Operator | C | Python |
|---|---|---|
| Logical and | && | and |
| Logical or | \|\| | or |
| Logical not | ! | Not |

**d) Assignment Operators:** They are used to assign values to variables. They are mostly the same in C and Python. But Python lacks increment and decrement operators. Here are a few examples.

| Operator | C | Python |
|---|---|---|
| Assignment | = | = |
| Addition assignment | += | += |
| Increment | ++ | Does not exist |
| Decrement | -- | Does not exist |

**e) Bitwise Operators:** They are used to compare binary numbers. They are used the same way in C and Python. Here are a few examples.

| Operantor | C | Python |
|---|---|---|
| Bitwise and | & | & |
| Bitwise or | \| | \| |
| Bitwise xor | ^ | ^ |

**f) Conditional (Ternary) Operator:** It is used for inline condition checking. It is used differently in C and Python.

| Operator | C | Python |
|---|---|---|
| Ternary | condition ? x : y | x if condition else y |

**SPECIFIC TOKENS IN PYTHON**

**Literals:** Literals are a way of expressing a constant value in a computer program's code. Here are the literals in Python.

**a) Integer Literals:** Integer values are numeric values that do not include fractional pieces and

it represents integer type values such as decimal (e.g., 999), hexadecimal (e.g.,0x1B) in Python.

**b) Float Literal:** A float literal is used to represent fractional numbers in Python. (e.g., 3.14)

**c) Complex Literal:** Complex literal is used to represent complex numbers that contains real and imaginary parts (e.g., z = 3.14 + 2.5j).

**d) String Literal:** A string literal is enclosed in double or single quotes to represent a set of characters (e.g., "Hello Word").

**e) Boolean Literal:** A boolean literal can represent two values which are true and false. The value of true is 1 and false is 0. This literal does not exist in the standard library of the C language.
(e.g., var = 1==1 #true)

**f) Collection Literal:** Collection literals are used to store collections of dates in the Python code. There are 4 types of collection literals in python.

| Collection | Definition | Example |
|---|---|---|
| List | Ordered collections within square brackets | exampleList = [1,2] |
| Tuple | Immutable and ordered collections within parentheses | exampleTuple = (1,2) |
| Dictionary | Collection of data in a key-value format within curly braces | exampleDictionary = {'A':1,'B':2} |
| Set | Unordered collections of unique values within curly braces | exampleSet = {1,2} |

**c) Special Literal:** 'None' is a special literal in python to signify a particular field has not been created. (e.g., var = None)

**SPECIFIC TOKENS IN C**

**Constants:** Constants are values that will remain the same once it is assigned throughout the program. There are two ways of defining a constant value in C.

Using const keyword → const var = 100;

Using #define pre-processor → #define var 100;

**Strings:** Strings are represented as an array of characters having a null character at the end of it in C. Strings in c are enclosed with double quotes (e.g., char str[] = "abc";)

## SEMANTICS

Semantics can be defined as the rules that how a computer should take steps during the execution of a program written in a specific programming language. This can be shown as the connection between input and output of a program. C and Python have a lot of differences in semantics.

### TYPING SYSTEM

C is statistically typed, which means you need to declare variables with their data types. This ensures type-related errors do not occur. However, Python is dynamically typed. It is not required to declare the data type of a variable, providing programmers flexibility, but can lead to certain problems.
(e.g., (In C) int x = 5; (In Python) x = 5)

### CONTROLS

C relies on manual memory management and provides programmers direct access to hardware and memory using pointers and memory allocation. They are handled by functions like ‘malloc’ and ‘free’. This feature may lead to memory leaks, so one should be very careful. That makes C suitable for embedded systems and programs that are performance is prioritized. On the other hand, Python lacks low-level features, meaning it let programmers take control over their memory and hardware as much as C, but this feature is considered more user-friendly. This difference in control of memory and hardware makes C a low-level and Python a high-level programming language.

## OTHERS

### PARADIGM

Both C and Python are in the imperative programming paradigm. C is a procedural programming language that contains a sequence of algorithmic steps to be executed to accomplish tasks. It is also called the top-down approach. The program is divided into small parts called functions which makes it a structured programming language. It has advantages like speed and efficiency. On the other hand, Python is a multi-paradigm language that supports three paradigms.

**i) Object Oriented:** In this paradigm, the program revolves around classes and objects. Objects are instances of classes that have fields and methods. It has concepts like encapsulation and inheritance. It has advantages like code reusability and data hiding.

**ii) Procedural:** Like the C, python supports procedural programming.

**iii) Functional:** It’s a paradigm where everything revolves around pure mathematical functions and avoids mutable data. It has advantages of being understood easily but it performs low performance.

## LEARNING CURVE

The Learning curve is a very important topic for beginners when learning a new programming language. In terms of syntax simplicity, Python is known for its syntax that is designed to be easy to read and write that uses indentation. C, on the other hand, can be harder for beginners due to its low-level features. Another difference is what makes C harder to write compared to Python is its typing system, which you must declare data type of each variable. Since python uses a dynamic typing system it will be easier to get started learning python for beginners. Another topic is memory management, that makes C language harder compared to Python. Since C requires manual memory management it will be harder for beginners to master C. On the other hand, programmers do not have to think about memory management in Python due to its automatic garbage collection. One more thing to mention is that since today's one of the most popular programming languages is Python, newbies can access more resources and tutorials on the web compared to C. In summary, learning curve for C can be harder for beginners for its low-level features and typing system compared to Python, which is designed to be easy to read and write.

## AVAILABILITY

**Standard Library:** C comes with a standard library that provides a bunch of functions for common tasks, which saves a lot of time during development. Python also comes with a comprehensive standard library that provides a lot of modules for a wide range of tasks, reducing the time for development and the need for external libraries.

**Portability:** C is a portable programming language, but occasionally it may require some changes. The code might be written in a platform-specific way. Compilers for C can change their implementations, which can cause adjustments for different platforms. Python is also a highly portable programming language, and it can be run on any platform with a compatible Python interpreter. But sometimes it may have platform dependencies that can reduce its portability.

**Built-in Error Handling:** C does not support a built-in exception handling mechanism. It is usually done using functions like 'perror()', 'ferror()', 'strerror()'. On contrast, Python provides a strong exception handling mechanism that allows programmers to handle errors in the program. It is handled in code blocks of 'try:', 'except', 'else','finally'.

## EFFICIENCY

In terms of execution speed, C is generally a faster programming language than Python. Because C is a compiled language. Meaning it must be compiled into machine code before it is executed. On the other hand, Python is an interpreted language, which means the code is executed line by line by the interpreter. Because of this difference, Python generally cannot match the raw speed of C.

Memory usage is also an important topic to consider for efficiency. Since Python uses automatic memory management, it leads to higher memory usage in general compared to C, which uses manual memory management so that programmers can optimize memory usage efficiently.

There are topics like libraries and portability that can be covered in efficiency, but both languages have vast libraries, and they are highly portable. In terms of development speed, since C requires manual memory control and has a bit more complex speed than Python, it may take more time to write C code than Python. So, Python is generally easier to read and write but performs worse than C.

## CONCLUSION

In conclusion, I have compared and contrasted C and Python in terms of syntax, semantics, and other important topics. These differences are crucial for understanding learning how a programming language works and what distinguishes one programming language from another. Both languages have their advantages and disadvantages in the topics above. Ultimately, the choice between these two languages depends on the goal and the requirements of the project.