ABSTRACT
The paper compares difference between two programming languages which are generally we used in our day to day life to develop many software’s. It shows a quick study about C++ and Python languages difference and how they are different from each other programmatically and theoretically also.

Keywords: C++, Python

INTRODUCTION

C++

C++, as we all know is an extension to C language and was developed by Bjarne Stroustrup at bell labs. C++ is an intermediate level language, as it comprises a confirmation of both high level and low level language features. C++ is a statically typed, free form, metaparadigm, compiled general-purpose language.

C++ is an Object Oriented Programming language but is not purely Object Oriented. Its features like friend and virtual, violate some of the very important OOPS features, rendering this language unworthy of being called completely Object Oriented. It’s a middle level language.

Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python’s elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third party Python modules, programs and tools, and additional documentation.

Difference between C++ and PYTHON

<table>
<thead>
<tr>
<th>C++</th>
<th>PYTHON</th>
</tr>
</thead>
<tbody>
<tr>
<td>C++ is a language designed and developed by Bjarne Stroustrup and C++ got released in 1983.</td>
<td>Python software foundation has released Python in 1991.</td>
</tr>
<tr>
<td>The current version of C++ is C++11.</td>
<td>The current version of Python is Python 3.4.1.</td>
</tr>
<tr>
<td>C++ language type discipline is (a) static and (b) nominative.</td>
<td>Python type discipline is (a) duck, (b) dynamic and (c) strong.</td>
</tr>
<tr>
<td>C++ type safety is unsafe.</td>
<td>Python type safety is safe.</td>
</tr>
<tr>
<td>C++ has influenced (a) Perl, (b) LPC, (c) Lua,</td>
<td>(a) Boo, (b) Cobra, (c) D, (d) F#, (e) Falcon, (f)</td>
</tr>
</tbody>
</table>
## Abhinav International Monthly Refereed Journal of Research In Management & Technology

<table>
<thead>
<tr>
<th>(d) Pike, (e) Ada 95, (f) Java, (g) PHP, (h) D, (i) C99, (j) C#, (k) Falcon and (l) Seed7.</th>
<th>Go, (g) Groovy, (h) JavaScript, (i) Julia, (j) Ruby and (k) Swift are the languages influenced by Python.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename extensions for C++ are (a) .cc, (b) .cpp, (c) .cxx, (d) .cpp, (e) .h, (f) .hh, (g) .hpp, (h) .hxx, and (i) .h++.</td>
<td>Filename extensions for Python are (a) .py, (b) .pyw, (c) .pyc, (d) .pyo and (e) .pyd.</td>
</tr>
<tr>
<td>C++ language paradigms are procedural, object-oriented, functional and generic.</td>
<td>Python paradigms are object-oriented, imperative, functional, procedural and reflective.</td>
</tr>
<tr>
<td>C++ implementations are (a) LLVM Clang, (b) GCC, (c) Microsoft Visual C++ and (d) Intel C++ Compiler.</td>
<td>Python implementations are (a) C Python, (b) Py Py (c) Iron Python and (d) Jython.</td>
</tr>
<tr>
<td>C++ coding requires more lines of codes and hence requires more time for developing.</td>
<td>Python is 5-10 times shorter than equivalent C++ code and hence requires much less time to develop.</td>
</tr>
<tr>
<td>Types of arguments and variables should be declared while using in C++.</td>
<td>Types of arguments and variables need not be declared while using in Python.</td>
</tr>
<tr>
<td>C++ makes use of pointers.</td>
<td>Python does not support pointers.</td>
</tr>
<tr>
<td>Memory management is manual in C++. The standard library tools for memory management are new and delete.</td>
<td>Python supports automatic memory management with reference counting.</td>
</tr>
<tr>
<td>C++ does not have an automatic garbage collector. The conservative garbage collector of Boehm Demers Weiser shall be used as a replacement of garbage collection or we can implement simple garbage collection facility in programs using smart pointers.</td>
<td>Python provides automatic garbage collector.</td>
</tr>
<tr>
<td>C++ uses semicolon as statement terminator and comma as expression separator.</td>
<td>Python is terminated with newline. Expression separation is done with semicolon.</td>
</tr>
<tr>
<td>C++ supports (a) 8-bit integer, (b) 16-bit integer, (c) 32-bit integer, (d) 64-bit integer and (e) word size int (signed and unsigned).</td>
<td>Python does not support (a) 8-bit integer, (b) 16-bit integer, (c) 32-bit integer, (d) 64-bit integer. Python supports word size int (signed).</td>
</tr>
<tr>
<td>C++ supports single precision float, double and long double.</td>
<td>Python does not support single precision float, double and long double instead supports double precision float.</td>
</tr>
<tr>
<td>C++ supports single precision and double precision complex numbers.</td>
<td>Python does not support single precision complex. Python supports double precision complex numbers.</td>
</tr>
<tr>
<td>C++ supports text character, text string, Boolean and enum data types.</td>
<td>Python does not support text character and enum. Python supports text string and Boolean.</td>
</tr>
<tr>
<td>C++ supports one-dimensional fixed size array and one-dimensional dynamic size array.</td>
<td>Python supports one-dimensional dynamic size array.</td>
</tr>
<tr>
<td>In C++, syntax to access the array element is name[index].The array index cannot be negative in C++.</td>
<td>In Python, syntax to access the array element is name [index]. Index can be negative in Python.</td>
</tr>
<tr>
<td>C++ has a form of associative array called std::map.</td>
<td>In Python, the associative arrays are termed as dictionaries.</td>
</tr>
</tbody>
</table>

Imagine you had to write a set of instructions in English which are then to be translated to Italian.

**Writing in C++ is like writing**

If the (integer) time is past (integer) 9 put on some (adjective) work (noun) clothes
While writing in Python is like writing

If the time is past 9 put on some work clothes

In the case of C++ one specifies the "type" of things and so there is less ambiguity about what the instruction means and errors can be "caught" even before the Italian translator even attempts to translate to Italian.

In the case of Python, one doesn't specify these types and the instruction reads a bit more naturally and is shorter in length. The Italian translator kind of has to figure out whether "time" is a noun or an integer or whether "work" is a noun or an adjective.

The example allows one to then understand these differences

In principle

C++ is a "statically typed" language, while Python is a "dynamically typed" language. In one case, the types of the main components of the set of instructions (the computer program) are explicitly declared while in the other case they aren't.

In practice

C++ is "Faster" than Python

C++ is typically a "compiled" language while Python is typically an "interpreted" language. With C++, the instructions are very well understood with all types specified and so they are translated into machine code (they are compiled) and executed unambiguously by the computer ...while with Python a separate computer program has to figure out what the Python instructions really mean [2,3] and how to "interpret" them to produce an output. The overhead of running this extra program versus simply just running the well-defined machine code typically ensures that C++ runs much faster than Python.

Python is more concise than C++

Though this is certainly not an absolute truth, the example also shows one of the reasons why Python code tends to be shorter than that written in C++ (i.e. there is no need to declare types explicitly and this literally saves space in written code). Smaller code size for the same task leads to "rapid prototyping" and this is partly why one would use something like Python when speed of development is important.

Programmatically comparison between C++ and PYTHON

C++ Program to add two numbers

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.   int a, b, c;
6.   cout<< "Enter two numbers to add\n";
7.   cin>> a >> b;
8.   c = a + b;
9.   cout<<"Sum of entered numbers = " << c <<endl;
10.  return 0;
11. }
Python Program To Add Two Numbers

# This program adds two numbers
1. num1 = 1.5
2. num2 = 6.3

# Add two numbers
3. sum = float(num1) + float(num2)

# Display the sum
4. print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))

As we see above two program having same function to add two numbers but as we can see that the lines and programmatically C++ program is lengthier than python.

So as comparison PYTHON uses less amount of space than C++ to store in system.

CONCLUSION

Python is much, much easier to learn. C++ is a very complex language with a ton of features. You can learn the basics of C++ easily enough, but once you get to the more advanced features, you’ll be facing a steep learning curve.

Python is a lot more easily going and flexible. Furthermore, the extensive number of libraries in Python makes life a lot easier.

Python is much easier to learn than C++. The code also is shorter in Python.

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can’t complete in a year. Python shines as a glue language, used to combine components written in C++.

BIBLIOGRAPHY