Python programing

Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming. It was created by Guido van Rossum, and first released on February 20, 1991.

Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called Monty Python's Flying Circus.

One of the amazing features of Python is the fact that it is actually one person's work. Usually, new programming languages are developed and published by large companies employing lots of professionals, and due to copyright rules, it is very hard to name any of the people involved in the project. Python is an exception.

About 20 years later, it is clear that all these intentions have been fulfilled. Some sources say that Python is the third-most popular programming language in the world, while others claim it's the fifth.

- it's **easy to learn** – the time needed to learn Python is shorter than for many other languages; this means that it's possible to start the actual programming faster;
- it's **easy to teach** – the teaching workload is smaller than that needed by other languages; this means that the teacher can put more emphasis on general (language-independent) programming techniques, not wasting energy on exotic tricks, strange exceptions and incomprehensible rules;
- it's **easy to use** for writing new software – it's often possible to write code faster when using Python;
- it's **easy to understand** – it's also often easier to understand someone else's code faster if it is written in Python;
- It's **easy to obtain**, install and deploy – Python is free, open and multiplatform; not all languages can boast that.

C++ Programing

C++, high-level computer programming language. Developed by Bjarne Stroustrup of Bell Laboratories in the early 1980s, it is based on the traditional C language but with added object-oriented programming and other capabilities. C++, along with Java, has become popular for developing commercial software packages that incorporate multiple interrelated applications. C++ is considered one of the fastest languages and is very close to low-level languages, thus allowing complete control over memory allocation and management. This very feature and its many other capabilities also make it one of the most difficult languages to learn and handle on a large scale.

C++ allows the programmer to define their own function.
A user-defined function groups code to perform a specific task and that group of code is given a name (identifier).
When the function is invoked from any part of the program, it all executes the codes defined in the body of the function.

- C++ is closer to the hardware. Therefore, C++ produces most of the embedded systems around. By embedded systems, we mean smart watches, medical machines, IoT sensors, etc.
- C++ plays a part in the development of applications such as servers and microcontroller programs.
- C++ is the leading language for 3D, multiplayer, or other types of game development. It is powerful enough to create such elaborate games as Counterstrike, Doom, and Red Dead Redemption. For instance, even the framework Unity is written in C++ even though its users apply C#.

Python Vs. C++ programing

Python is dynamically typed. **C++** is statically typed. **Python** leads to one conclusion: **Python** is **better** for beginners in terms of its easy-to-read code and simple syntax. Additionally, **Python** is a good option for web development (backend), while **C++** is not very popular in web development of any kind.

Python is better for beginners in terms of its easy-to-read code and simple syntax. Additionally, Python is a good option for web development (backend), while C++ is not very popular in web development of any kind.
Python is also a leading language for data analysis and machine learning. While it is possible to use C++ for machine learning purposes as well, it is not a good option. In terms of simplicity, Python is much easier to use and has a great support system when it comes to AI and ML frameworks.

- C++ code needs curly brackets and semicolons to work. Python offers a *more friendly approach* as it abandons such programming rules. It mainly depends on the indentation of code. This feature refers to the fact that each level of indentation creates the structure of code.
- Instead of using a semicolon, Python treats the end of the line as the end of the statement. If you need your statement to continue for several lines, you should use the backslash (\) sign. In C++, you need to use a semicolon to indicate the end of the statement.
- Boolean expressions are different in Python and C++. C++ returns either false or true based on numeric values. For instance, everything labeled as 0 is false, and other numeric values are true. Python has other possibilities as well. For example, none and false constants are false, just as empty sequences or collections.
- Variables in C++ need to have a type such as a float or an int because this language is statically-typed. Dynamic typing is a feature of Python, meaning that you do not need to indicate the type

of the object. Python offers a lot of flexibility, which can lead to using variables in not appropriate contexts.

- Single and multiple inheritance works in both Python in C++.
- One important aspect of Python vs C++ is memory management. Python does not let you handle memory directly. Instead, it offers automatic memory management, referred to as a garbage collector. C++ does not have such a feature, and all memory management happens manually.
- Python dictionary vs. C++ map refers to a simple difference between the terminologies of these languages. In C++, a map is a container storing values indexed by a key. A dictionary in Python is the same, but more flexible. Why? Because the keys and values do not have to be the same type.

One good thing is that learning Python for C++ programmers should be quite easy. Python follows simple conventions that are not too difficult to master in a considerably short time.

However, when it comes to Python vs C++, learning C++ for Python programmers is different since C++ is more complex, requires more contemplation and research. If you're brave enough to learn C++, I also have an option for you.