Java vs. Python

Java and Python are two of the hottest programming languages in the market right now because of their versatility, efficiency, and automation capabilities. Both languages have their merits and their flaws, but the main difference is that Java is statically typed and Python is dynamically typed.

They have similarities, as they both adopt the “everything is an object” design, have great cross-platform support, and use immutable strings and deep standard libraries. However, they have plenty of differences that steer some coders towards Java and others towards Python. Java has always had a single large corporate sponsor, while Python is more distributed.

A quick comparison:

1. Java is more efficient when it comes to performance speed thanks to its optimizations and virtual machine execution.
2. You can add Python implementations without this restriction, but they can negatively affect portability assumptions within the Python code. Therefore, when it comes to sheer speed performance, Java has the edge.
3. Python is more effective when it comes to adapting legacy systems. The language is more suited to make changes to an existing legacy system. Python can make gradual shifts instead of fully rewriting and revamping the system like Java does.
4. Python’s dynamic system relies on brevity, fluidity, and experimentation in code, giving it more versatility than Java’s rigid style. Python has also been adapting to automated testing in modern development, but this happens more frequently in integration instead of unit testing.

While Java churns out higher performance speed, Python is more suited to evolve legacy systems. When it comes to practical agility, Java is a more proven option, while Python has more flexibility for experimentation.

Python has surpassed Java in terms of the most popular language. There are several reasons for this shift, including that Python is a general-purpose language, which means that it can be used to build almost anything. The language is great for backend web development, data analysis, artificial intelligence and scientific computing. In addition to its professional uses, it can also be used to create games, productivity tools, and desktop apps.

There’s also Kivy, a Python tool that makes it easier to write mobile applications. The language moves away from traditional web technologies, making it an exciting option for the future. With the language, you could talk to telco gear through a custom C extension.

Python has the slight edge over Java as far as the future goes, but neither language is perfect and Java adopters will continue to try to perfect the language moving forward.

**Head-to-head comparison between Java and Python:**

|  |  |  |
| --- | --- | --- |
| **Parameters** | **JAVA** | **PYTHON** |
| **Code** | Longer line of code as compared to Python public class HelloWorld{public static void main (String [] args) {System.out.println("Hello World"); | print("Hello World") |
| **Syntax** | At the end of statement if you miss semicolon it throws an error. In java you must define particular block using curly braces without it code won't work. | Statements do not need a semicolon at the end. In python you do not see a sight of curly braces but indentation is mandatory. Indentation improves readability of codes. |
| **Dynamic** | Must declare type of data. | You don't need to declare type of data. |
| **Speed** | Java is faster | It is slower because python is interpreter and also it determines the type of data at run time. |
| **Portability** | Due to high popularity of java, JVM is available almost everywhere. | Python is also portable but in front of java python is not popular. |
| **Databases** | JDBC is very popular and widely used to connect to databases. | Python's database access layers are weaker than Java's JDBC. |
| **Easy to Use** | Java is not easy to use as compared to python because there is no dynamic programming concept and codes are longer. | Codes are shorter than Java. Also, as it follows dynamic programming, it is not only easy to use but alos easy to understand because of indentation. |
| **Practical Dexterity** | Java enjoys more undeviating refactoring support than python thanks to its static type system and universality of IDE's in development. | Python has always had an existence in the talent space and has popularity for many reasons including Data Science and DevOps movement. |
| **Legacy** | Java's history in the enterprise and its slightly more verbose coding style mean that Java legacy systems are typically larger and more numerous than python's | Python has less legacy problem so organization find difficulty for script to copy and paste codes. |

# Code Standards in Python

PEP-8, Python's style guide helps to write and maintain codes in efficient ways. The following are the things that we follow while writing standard codes in python:

1. Indentation: When programming in Python, indentation is something that you will definitely use. However, you should be careful with it, as it can lead to syntax errors. The recommendation is therefore to use 4 spaces for indentation.
2. Maximum Line Length: it's good to aim for a line length of 79 characters in your Python code.
3. Blank Lines: In Python scripts, top-level function and classes are separated by two blank lines. Method definitions inside classes should be separated by one blank line.
4. Whitespaces in Expressions and Statements: There are certain scenarios where we should avoid whitespaces in expressions and statements.
5. Source File Encoding: To use bits to represent anything at all besides bits, you need a set of rules. You need to convert a sequence of bits into something like letters, numbers and pictures using an encoding scheme or encoding. Examples of encoding schemes are ASCII, UTF-8, etc.
6. Imports: Importing libraries and/or modules is something that you'll often do when you're working with Python for data science. As you might already know, you should always import libraries at the start of your script.
7. Absolute and Relative Imports: Next, it's good to know the difference between absolute and relative imports. In general, absolute imports are preferred in Python, as it adds up more readability. However, as your application becomes more complex, you can go on using the relative imports also. Implicit relative imports should never be used and have been removed in Python 3.

For more details, please visit: <https://www.python.org/dev/peps/pep-0008/>