Language Report for Python and Java

Xinliang Lu 0822760

January 19, 2023

Abstract

In this report, I choose to analyze Python and Java, by first giving their history, then introducing detailed comparison and discussion, and using the application examples of them as the ending. Specifically, in the comparison section, multiple features, including: laziness, error handling, memory management, meta-programming, scoping, and points, are demonstrated and further information around these features is given to better support the findings.

1 Introduction

In this language report, I compare the differences between Python (all discussions around Python are based on version 3.x) and Java, which are both modern general-purpose languages and are currently being widely used by various groups of programmers. In this section, a brief introduction is given to these two selected languages, including their history, main features, and the reason why they are so popular.

1.1 Python

Python is a popular, interpreted, object-oriented, high-level, dynamically semantic, and general-purpose programming language, and many Python-powered gadgets are used by a wide range of consumers on daily basis. On February 20, 1991, Guido van Rossum published the first version of Python which takes its name from the Monty Python's Flying Circus television comedy sketch series, even if many people may only think of the python as a huge snake.

The main feature and advantage of Python would be easy to use. In particular, despite being a relatively advanced programming language, Python is simple to learn, since coding may be learned by anyone in a matter of hours or days. It can take some more time to fully understand all of Python's complex ideas, packages, and modules. However, compared to other well-known languages like C, C++, and Java, understanding the fundamentals of Python syntax is quite simple. Another reason for making Python user-friendly is the wide range of standard libraries it offers to its users. This means the developers do not need to rewrite some functions from scratch, saving both time and effort on the developer's end. In the end, All these factors make Python one of the most popular languages in the world.

1.2 Java

Java is an object-oriented, network-centric, multi-platform language that may be used as a platform by itself. It is a quick, safe, and dependable programming language that can be used

to create everything from mobile apps and business software to big data applications and server-side technologies. In 1995, Java was first released by Sun Microsystems, its capacity for interactive features and multimedia demonstrated that the Web was a particularly suitable application for it.

The biggest pros of Java would be its platform-independent feature, because of the Java Virtual Machine, programmers who created their software on one platform may run it on any platform or operating system, independent of the underlying configuration. With this and other fancy features, such as object-oriented, portable, interpreted, and multi-threaded, make Java welcomed by programmers.

2 Comparison and Discussion

The main differences between these two languages in the chosen aspect are shown in Table.1, further discussion is listed in the following subsections.

Table 1: Comparison of Python and Java.

	Python	**Java**
Support for Strictness and Laziness	Partial support including logical expression operators, generator expressions/functions, and "map()" function.	Not support, but there is a workaround.
Error and Exception Handling	Error and exception are the same.	A clear line between these two classes.
Memory Management	Variables refer to a memory space, garbage collection is based on reference count and happens without delay.	Allocating new space is similar, but GC uses concurrent mark sweep.
Support for Metaprogramming	Supported by decorators and meta-classes.	Only support reflection.
Module System and Scoping	Modules are allowed to be written in Python and C, then import certain module. Scope is stored in the namespace.	Similar to Python, modular is also the main feature, although dependency is manually handled. Scoping is also similar.
References and Pointers	Supports references, but not pointers.	Same as Python.

2.1 Strictness and Laziness

In Python, lazy evaluation is already offered in version 3.x, among all built-in functions, logical expression operators[4], generator expressions/functions, and "map()" function are

lazy[7]. The reason behind this design is quite straightforward, since generating large amount of contents would take up lots of memory space and execution time, the values generated by these functions are only evaluated when they are called. And it is the same thing for logical expressions/operators, since the operators don't need to evaluate all conditions to choose the returning expression. In addition, Python is a flexible programming language, although functions defined in a class are not lazy (all expressions will be evaluated at once), we can use "@property" decorator to create some laziness features

Unlike Python, Java eagerly evaluates arguments of functions or methods[5], in Listing.1 we define a computational function and call it twice by passing two different arguments to it, and in the output console we can observe it runs twice, but actually, since we use "and" operator, in lazy evaluation, it should only be evaluated once and return "false" then exit. However, we can use "Supplier" interface to implement the lazy version of functions, like Listing.2, in the code, combination of "&&" operator and "Supplier" interface behaves short-circuiting which means the second argument isn't evaluated when the first one is false. Thus, despite Java takes eager mode in evaluating expressions, we can still use a workaround to bypass this feature.

Listing 1: Eager evaluation in Java.

```
static boolean isContain(String str) {
    System.out.println("executing...");
    // expensive computation here
    return str.contains("aa");}
static String eagerMatch(boolean b1, boolean b2) {
    return b1 && b2 ? "match" : "incompatible!";}
public static void main(String [] args) {
    System.out.print(eagerMatch(isContain("bb"), isContain("aa"
    ↪ )));}
```

Listing 2: Lazy evaluation in Java.

```
static String lazyMatch(Supplier a, Supplier b) {
    return a.get() && b.get() ? "match" : "incompatible!";}
public static void main(String [] args) {
    System.out.println(lazyMatch(() -> isContain("bb"), () ->
    ↪ isContain("aa")));}
```

2.2 Error and Exception

For handling exceptions in Python, it provides "try" and "except" code blocks for developers to catch and do something with errors. And in the design of Python, the line between error and exception is less clear, since both error and exception can remain unchecked, the user decides whether to do something with the error/exception or not.

While in Java, there's a strict distinction between the two classes, Error and Exception. From the perspective of the compiler, an exception is needed to be fixed in the source file,

otherwise it won't be compiled. But an error in the program is not necessarily to be caught.

2.3 Memory Management

In Python, developers don't need to worry about memory management in most cases, since the interpreter handles this automatically, including allocation and freeing memory. Specifically, in the lower level of Python, all the basic features of it are normally implemented by C language[8], which means the memory management mechanism is reached by most of "malloc" and "free" function in C. In detail, like objects in Python, each individual consists of value, type, and reference count, when creating a new object, CPython will allocate an actual memory space for it and this new space will be pointed by the variable in Python. For garbage collection, when the reference count of an object is zero, it will be removed from the memory and also trigger the deletion and garbage collection for other objects which are referenced to the deleted one. However, keeping scanning for reference counting number usually takes extra computation and may slow down the overall performance, while the pros would be responsive deletion.

For Java, allocating memory space for new objects is quite similar to Python, but in garbage collection, instead of using reference counting mechanism, Java performs a logical method called concurrent mark sweep[12] which scans all reachable code and marks other unreachable parts including objects as dead, then the interpreter will move dead objects into a specific memory area and trigger the deletion for these objects at one time. Although concurrent mark sweep is more logical and sometimes reliable, extra memory space will be used during moving dead objects.

2.4 Meta-programming

In Python, decorators and meta-classes allow users to perform meta-programming[9]. By using decorators, new functional features can be added in the existing function without changing the original structure and code. Meta-classes, on the other hand, define the behavior of ordinary classes and its instance, which means it can add or remove a functional feature or field for a class.

In common sense, because Java is a static and verbose language, maybe meta-programming features are hard to implement, but that is not true. Reflection in Java enables us to reuse class-independent code blocks[13], also it enables internal program properties to be examined or "introspected" upon by a Java program while it is running. For instance, a Java class may get the names of each of its members and show them.

2.5 Module and Scoping

Module for Python can be written in Python, or in C then dynamically loaded when running the program[2], for built-in modules they are contained in the interpreter. Among them, "import" will work if certain modules are needed and dependency will be checked and handled by the interpreter, which makes building programs in modules easier. The terms "scope" and "namespace" are closely connected in the Python programming language[11]. A Python scope determines where a name appears in the program. In the implementation of Python scopes,

names are mapped to objects using dictionaries that have a term – Namespaces. A module's namespace is also kept in the dict property of the module.

For Java, packing a Java application or Java API as a distinct Java module using the packaging process is known as a Java module. A modular JAR file is how a Java module is packaged, which has the option to declare which Java packages should be available to other Java modules that utilize it[6]. It must also list the Java modules it needs in order to function well. As for scoping, it defines where a specific variable or function is visible in Java[10], to be specific, defined or passed variables are only visible in methods or certain code blocks, while class/object variables exist until the containing object is deleted.

2.6 References and Pointers

In general, pointers do not exist in Python[3], since they encourage implicit changes rather than explicit ones, which may lead to terribly unsafe operations. So the variables and objects passed to functions are by reference. In Java, pointers are only present as a reference implementation feature[1]. A duplicate of the reference, referring to the same object, is copied to the stack of a called function.

3 Application

As we show in section 2, Python is more suitable for applications that are not computationally expensive, and value portability and swift development, since Python is slower in garbage collection but has a better solution in supporting laziness, error/exception handling, meta-programming, and provides more easy-to-use dependency resolving for modules. All these features make it developer-friendly and platform-independent but not a high-performance language. While Java, is a statically typed and half-compiled language, combined with a more efficient GC algorithm, which means compared with Python, Java is more suitable for computationally critical tasks, like running heavy web apps.

4 Conclusion

In this report, I compare Python with Java in aspects of laziness, error handling, memory management, meta-programming, scoping, and points, then discuss these differences in detail, and give some examples of various suitable applications for these two languages respectively.

References

[1] C/c++ pointers vs java references. https://www.geeksforgeeks.org/is-there-any-concept-of-pointers-in-java/. Accessed: 2023-1-16.

[2] Chapter 10. modules and scoping rules. https://livebook.manning.com/book/the-quick-python-book-second-edition/chapter-10/14. Accessed: 2023-1-16.

[3] Do we have pointers in python like other programming languages. https://www.edureka.co/community/40869/we-have-pointers-in-python-like-other-programming-languages. Accessed: 2023-1-16.

[4] Does python support short-circuiting? https://stackoverflow.com/questions/2580136/does-python-support-short-circuiting. Accessed: 2023-1-16.

[5] Java 8 - lazy argument evaluation. https://www.rapid7.com/blog/post/2017/01/13/java-8-lazy-argument-evaluation/. Accessed: 2023-1-16.

[6] Java modules. https://jenkov.com/tutorials/java/modules.html. Accessed: 2023-1-16.

[7] Lazy evaluation in python. https://stackoverflow.com/questions/20535342/lazy-evaluation-in-python. Accessed: 2023-1-16.

[8] Memory management in python. https://www.honeybadger.io/blog/memory-management-in-python/. Accessed: 2023-1-16.

[9] Meta-programming in python. https://betterprogramming.pub/meta-programming-in-python-7fb94c8c7152. Accessed: 2023-1-16.

[10] Scope in java. https://www.codecademy.com/article/variable-scope-in-java. Accessed: 2023-1-16.

[11] Scope of variable in python. https://www.scaler.com/topics/python/python-functions-scope/. Accessed: 2023-1-16.

[12] What are the differences between python and java memory management? https://www.quora.com/What-are-the-differences-between-Python-and-Java-memory-management. Accessed: 2023-1-16.

[13] Weiyu Miao and Jeremy Siek. Compile-time reflection and metaprogramming for java. In *Proceedings of the ACM SIGPLAN 2014 Workshop on Partial Evaluation and Program Manipulation*, pages 27–37, 2014.