UNIVERSITY OF WATERLOO
Faculty of Mathematics

A Comparison Between Clojure And Python

Droit Financial Technology
New York, USA

Prepared by
Yu Li
4A Data Science
ID 20603029
December 9th, 2019

Yu Li
601-308 Lester Street
Waterloo, ON
N2L3W6

December 9th

Joceline Zheng
Head of Product
145 West 28th Street
New York, NY

Dear Ms. Zheng,

I am submitting this report, "A Comparison Between Clojure And Python" as my fourth work term report. The information presented in this report is based on my 4A work term in the product team at Droit Financial Technology.

The product team at Droit Financial Technology designs and delivers regulation solutions for clients by transforming law into structured data models and computational representations. The team researches on emerging needs in the regulatory space and collaborates with software engineers to design new product lines. During my time with the team, I have learned to create a Clojure web application, transform regulation logic into ontology-based models using first order logic, test models using in-house technology and performed research on Money Laundering regulations in Europe and position reporting obligations in APAC. This report will focus on comparing Clojure to Python to illustrate their differences and analyze Droit's choice of Clojure as the main programming language for its infrastructure.

I would like to thank the company for giving me the opportunity to gain exposure to the regulation technology industry. I would also like to thank the product team for providing me the tools and guidance to perform the proper analysis at work. I hereby confirm that I received no help, other than what is mentioned above, in writing this report. I also confirm that this report has not been previously submitted for academic credit at this or any other academic institution.

Sincerely,

Yu Li
ID 20603929

Table of Contents

Executive Summary

Finding the right programming language to build a product can be a difficult choice. This report looks into the main programming language behind Droit's infrastructure – Clojure and compares it to one of the popular programming languages in the market – Python.

Both Clojure and Python are general programming languages that can be used for many applications such as web development, numerical analysis or automating processes. Both languages support lazy sequences and duck typing. The differences between the two languages include Clojure being a compiled and functional programming language with a small community and Python being an interpreted and object-oriented programming language with a well-developed community.

Object-oriented programming concepts dominate the market, but this report will show how as a functional programming language, Clojure still manages to support structured programming through polymorphism. At the same time, Clojure's immutability nature makes concurrency easier to implement. On the other hand, Python may not be the best programming language for concurrency or java-based project, but its strong community and highly readable syntax allow fast iterations of development.

Overall, Clojure is a great choice for Droit's backend infrastructure due to its ability to compile on JVM, seamless integration with Java libraries, clean syntax and the ability to implement useful object-oriented concepts while being immutable.

1.0 Introduction

Since the 2008 financial crisis, there has been a major regulatory overhaul in the financial systems. The annual cost on complex, extensive, and ever-changing regulations across different regulators and jurisdictions have increased significantly. To address the growing need for cost-effective solutions to comply with regulatory obligations, Regulation Technology (RegTech) was created.

Droit is one of these RegTech companies that specializes in financial services regulation. Droit currently uses Clojure as the main programming languages for its backend framework, relies on the product team's expertise in developing the regulation models and creates a regulatory decision engine for financial services to use in their business activities.

This report analyzes Droit's choice of Clojure as the main programming language in the production infrastructure and compares it to one of the most popular programming languages in the industry – Python. The report will provide backgrounds on both languages and illustrate their similarities and differences in terms of syntax, design, concurrency capabilities, community and applications. In the end, the report will show that Clojure is a better choice for Droit because of its ability to run on JVM, integration with Java libraries and implementation of polymorphism while being immutable.

2.0 Analysis

2.1 Clojure background

Clojure is a dynamic, functional dialect of the Lisp programming language that can run on the Java and .Net platform. While being a compiled language, Clojure's features are still supported at runtime which makes it easy and flexible to code and debug. As a dialect of Lisp programming language, Clojure exhibits much of Lisp's uniqueness such the ability to evaluate code as a datatype in the language and syntactic abstraction. Clojure also advocates immutability and persistent data structures, which makes it easier to reason the behavior of a program. Another key feature of Clojure programming language is its efficient infrastructure for concurrency and multithreaded programming. Clojure's functional nature, concurrency support and ability to work with Java libraries and platform makes it a practical programming language to solve many industrial problems.

2.2 Python background

Python is an interpreted, general-purpose, object-oriented programming language. Combining Python's many built-in high-level data structures with dynamic binding options makes Python extremely attractive for RAD (Rapid Application Development). Python's syntax was developed with a strong focus on code readability. It has an elegant syntax which allows developers to read and translate code easily and provides them the flexibility in their coding methods. As a result,

this reduces the cost of program maintenance and allows for faster development without significant language and experience barriers in a collaborative environment.

Originally, Python was mainly used for scripting and trivial projects. However, this concept has changed as the Python community grows larger and reliance on Python for larger scaled applications grew. A large group of companies and platforms rely on Python nowadays, such as YouTube, Pinterest, Instagram and the web-oriented transaction system of the New York Stock Exchange[1].

2.3 Comparison between Clojure and Python

Both Clojure and Python are general-purpose, dynamic programming languages that advocate for simple and concise syntax. Being dynamic allows for faster turnaround time when writing code and performing tests as it eliminates the need to compile the code every time. Having a clean syntax can improve readability and reduce the time it takes to understand the code. Both programming languages also support lazy sequences where the main elements of a sequence are not available until they are needed. Lastly, both languages advocate for duck typing which allows a function to support all method signatures and attributes of the input object at run time. Despite having a similar purpose, there are many differences between the two programming languages in the context of compilation, design, concurrency capabilities and community.

[1] https://www.codeconquest.com/tutorials/python/?cv=1

2.3.1 Compiled vs Interpreted

Clojure is a compiled programming language that runs on the JVM and Python is an interpreted programming language. Compiled languages are converted directly into machine code that the processor can execute directly. As a result, compiled language tends to be faster and more efficient to execute than interpreted language. The disadvantage of being a compiled programming language is the need to wait for the entire program to compile first before testing. However, Clojure eliminates this problem by providing the ability to invoke the compiler at run-time if needed.

Python has an interpreter that will run the program on a line by lines basis, and this provides a lot of flexibility in coding. Interpreted language used to be significantly slower than compiled languages, but with the development of just-in-time compilation, interpreted programs have been able to execute much faster than before.

For Droit, choosing Clojure is a good choice in this aspect as the company was primarily a Java shop before. Clojure integrates with java libraries and compiles to the JVM platform seamlessly. This helps the company eliminate the need to restructure its product completely while eliminating Java's verbose and mutable nature.

2.3.2 Functional vs Object Oriented

By nature, Clojure is a functional programming language while Python is an object-oriented programming (OOP) language. OOP is widely adopted in the industry for its intuitive approach

to model objects, inheritance capability, encapsulation and polymorphism. In general, OOP uses the concept of classes and objects to organize data and code for better reusability and templating. Each class serves as a template for its children instances, which can have access to the parent class' methods and attributes. This idea is easy to conceptualize as humans tend to structure things in a similar manner. The concept of inheritance in OOP allows for reusability of methods in the parent class which can save more time without repeating the same code. OOP also provides encapsulation capabilities which hides implementation details and only allow interactions with well-behaved interfaces. Lastly, OOP supports polymorphism, where inherited classes can have different behaviors for the same parent method. This allows for developers to create a general class with certain behaviors yet also make customized changes to subclasses. The major problem with OOP is when classes becomes deeply nested and difficult to understand.

Functional programming is based on solving pure computational and mathematical problems. In Clojure, there is a separation between the data and the behavior of the program. The object-oriented approach of creating new data for each situation is disposed in Clojure, but polymorphism is still supported in Clojure through a multimethod system that uses dispatching methods. Derivation in Clojure can also be achieved either through Java inheritance or its own hierarchy structure. As Droit's regulatory models are ontology-based and logic-based rather than object-based, Clojure would be the right choice for creating the company's main product. Functional programming also eliminates the problem of race conditioning. This can happen when multiple functions need the same resource. An example would be if a show function depends on the date from a retrieve data function. If the show function runs faster than the

retrieve data function, this could cause an error. In Clojure, resources are not shared in this manner so such problems would not exist.

2.3.2 Immutability vs Mutability

Immutability is the default in Clojure while Python supports mutable data structures. Mutability can make it difficult to understand programs, as objects passed in a different method could be changed under the cover. With immutability, data structures will not be changed, and this makes it easier for developers to write, use and reason about code.

Immutable objects are also thread-safe and great for solving concurrency problems. Concurrency is the occurrence of multiple events within overlapping time frame, but not simultaneously. Concurrent programming takes advantage of how multiple threads can make progress on a task without waiting for others to complete. [2] If an object cannot be changed, it can be shared among different concurrent threads without being corrupted by them accessing it concurrently. On the other hand, most programming language need to rely on locks to achieve thread synchronization, which can be difficult to implement. The real disadvantage to immutability is the necessity to create a new object every time one needs to make a modification. This could also potentially cause more frequent garbage collections.

[2] https://www.computerhope.com/jargon/c/concurrent-computing.htm

At Droit, many lightweight tools such as futures, delays and promises are used in the backend to achieve concurrent programming. Futures are used to define a task and place it in another thread without requiring the result right away. Delays allow one to define a task without having to execute it or require the result immediately. Promises are made to express expected results without defining it or knowing when the task should be run. These small tools assist with managing concurrency in Droit's applications.

2.3.4 Community

Support for a language is important in terms of finding answers on Stack Overflow, reading up articles for new ideas and finding available libraries. Python has a larger community and greater presence in the industry than Clojure. GitHub statistics shows that Python has 25.3k stars and 10.5k forks, while Clojure has 7.85k stars and 1.25k forks. Python has a broader approval, being mentioned in 2826 company stacks and 3632 developers' stacks; compared to Clojure, which is listed in 95 company stacks and 80 developer stacks. Python is well praised for its resourceful ecosystem with plenty of existing libraries in web development, scripting, data science, artificial intelligence and much more. While Clojure also has a strong community, it is more niche and there are less documents available compared to Python. This may cause problem for Droit in terms of hiring experienced Clojure programmers in the industry because the community is small. Droit might also expect more time for new programmers to pick up the language as it is more difficult to learn than a mature language like Python.

2.0 Conclusion

Python and Clojure each have their own benefits and setbacks when it comes to programming. Both are dynamic, general programming languages that can be used for building web applications, data science and automating processes. They are both lazy and support duck typing. In terms of differences, Clojure is compiled on JVM, while Python is an interpreted language. This makes Clojure a better choice for Droit as it can integrate seamlessly with the Java platform. Clojure is also functional and immutable while Python is object oriented and mutable. Since Droit's product is mainly functional by nature, Clojure is a better choice in this regard. Clojure does have a smaller community than Python, which makes the language more difficult to learn. However, Clojure's Java interoperability exposes the language to Java libraries' mature community. So an experienced Java programmer might have an easier time picking up this difficult language. Overall, Clojure is a better language than Python for Droit as it is functional by nature, runs on the JVM, have a clean syntax and can be used for many applications.

References

Amarasinghe, Saman, et al. “Mutability & Immutability.” *Reading 9: Mutability & Immutability*. Retrieved November 28th, 2019 from web.mit.edu/6.005/www/fa15/classes/09-immutability/.

Code Conquest. “Code Conquest.” *Code Conquest*. Retrieved November 28th, 2019 from www.codeconquest.com/tutorials/python/?cv=1.

FreeCodeCamp. “Compiled Versus Interpreted Languages.” *FreeCodeCamp Guide*. Retrieved November 28th, 2019 from guide.freecodecamp.org/computer-science/compiled-versus-interpreted-languages/.

Molitor, Steve. “Immutability in Clojure - Part 1, Programming Without Variables.” *Object Computing, Inc.*, 2011, objectcomputing.com/resources/publications/sett/march-2011-immutability-in-clojure-part-1-programming-without-variables.

“What's the Difference between a Compiled and an Interpreted Language?” *Programmer and Software Interview Questions and Answers*. Retrieved November 28th, 2019 from www.programmerinterview.com/general-miscellaneous/whats-the-difference-between-a-compiled-and-an-interpreted-language/.