

MOJO: A Programming Review

Monika Gaur¹, Jyoti Kataria², Sahil Gupta³

¹Professor, Anangpuria School of Management and Technology

²Assistant Professor, Anangpuria School of Management and Technology

³Student Btech IT, Anangpuria School of Management and Technology

Abstract

MOJO combines the serviceability of python with its easy and reliable syntax with the high performance of low-level languages like C++ and JAVA. In this paper, we are describing the working features and the help we can get from mojo to build more optimized, reliable and faster algorithms to build systems in the software engineering industry which will help developers to reduce the workload from the hardware components of user's device and to increases the user experience.

Keywords

LLVM, MLIR, borrow checker

1. Introduction

In the past few years, high level technologies like Artificial Intelligence, Machine Learning, Deep Learning had taken a drastic boom in the IT industry. To fulfill the requirements of these type of high-level technologies, we need top-notch quality of hardware components like CPUs and GPUs with that we also need an optimized, stable and a faster programming language to develop the algorithms to integrate the software components with the hardware components.

Identifying these problems and requirements, in 2022 Modular company was founded by Chris Lattner who was the original architect of swift programming language and Tim Davis, a former google employee started to design a new programming language named **MOJO**.

Due to its compatibility with python libraries, it becomes a robust language for developing Artificial Intelligence and Machine Learning with a great memory management system like C++ and Rust with the help of pointers and also offers to watch low-level details. Its features like tiling optimization tool, autotune module, ownership and borrowing system makes it a very advanced language with easy syntax like python and features of different types of languages like C++, Rust and python combined with the speed like low-level languages to develop highly scalable and stable Artificial Intelligence and Machine Learning systems.

Mojo offers its native libraries for different type of developments which are purely written in mojo for example Basalt, Lightbug HTTP, Endia, LLM.mojo etc. which are managed by the Modular company itself. These native libraries are more optimized and beneficial to use in the development because these are built under the MOJO ecosystem and they will perform better if one writes an algorithm in these libraries because it will use the full potential of the MOJO language.

2. *Features*

- To develop frontend and backend for any instruction set architecture, MOJO uses Low Level Virtual Machine (LLVM) as a set of compiler and toolchain.
- To make optimal use of GPUs, DPUs, TPUs, AI ASICS, FPGAS, and Quantum Computing, it uses Multi Level Intermediate Representation (MLIR).
- It uses inferred static typing.
- It is an Object-Oriented Programming language.
- It is not source-compatible with python 3, only providing a subset of python's syntax.
- MOJO 'def' functions use value semantics by default (functions receive a copy of all arguments and any modifications are not visible outside the function), on the other hand Python functions use reference semantics (functions receive a reference on their arguments and any modification of a mutable arguments inside the function is visible outside).
- It has a borrow checker which allows a function or method to temporarily borrow a reference to a value owned by another part of program, this feature is influenced by Rust programming language.

3. *Utilization of MOJO*

Developers can use MOJO to build more optimized, controlled, stable and more faster algorithms to build applications that predict customer behavior, detect fraud, make data-driven decisions and easily implement the benefits of Artificial Intelligence and Machine Learning.

As we discussed above that MOJO was primarily made for the development of Machine Learning and Artificial Intelligence with the speed like low level languages and syntax like high level languages. It can easily calculate large datasets to train the AI models within less time because of its fast speed. The algorithm that has been written to make these types of systems are also very easy to understand for the developers who are going to make changes in the algorithm in future for the development of the models.

4. *Speed of MOJO*

MOJO developers have the advantage to use vectors, threads, AI hardware units in a more efficient manner because it uses the MLIR technology. In python, there is a single thread execution at a time whereas MOJO takes advantage over python where it has parallel processing across multiple cores which we can clearly see in *Figure 1*.

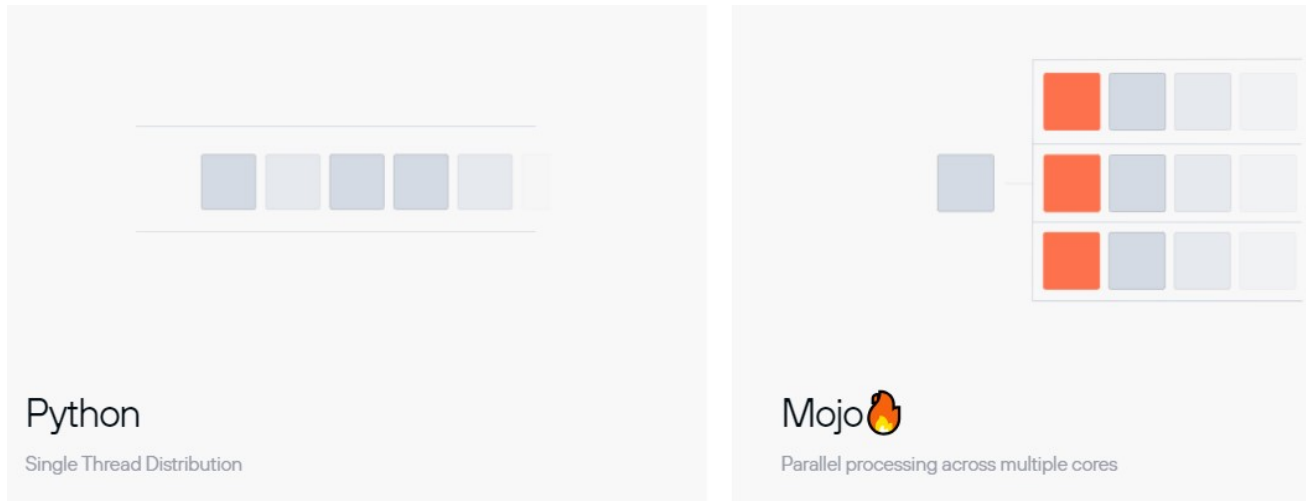


Figure 1 (credit: <https://www.modular.com/mojo>)

PARALLELIZATION


Here is a benchmarks figure (*Figure 2*) of the speed comparison between python, NumPy, SCALAR C++, and MOJO.

LANGUAGES	TIME (S) *	SPEEDUP VS PYTHON
PYTHON 3.10.9	970 s	1x
NUMPY	171 s	6x
SCALAR C++	0.11 s	9000x
MOJO 	0.0142 s	68000x

Figure 2 (credit: <https://shriramsivanandhan.medium.com/mojo-programming-language-68000x-faster-than-python-programming-in-mojo-part-ii-d162740a2f67>)

If we consider these benchmarks score vs python language then we can clearly observe that how fast is MOJO language in comparison to other languages due to its built structure in LLVM and MLIR to interact with GPUs, CPUs, TPUs and other hardware components with its very easy syntax.

5. *MOJO File Extension*

MOJO files use the .mojo or  extensions.

6. *MOJO Code Examples*

- Firstly, creating a program to print “Hello, world!”

Create a file name for example hello. mojo and add the following code (*in Figure 3*):

```
fn main():  
    print("Hello, world!")
```

Figure 3 (credit: <https://docs.modular.com/mojo/manual/get-started>)

- In Mojo programming language, functions can be declared using both fn (can be used for performant functions) or def (can be used for Python compatibility).

Here is a basic example of arithmetic operations in Mojo with help of a def function (*Figure 4*):

```
def subs(x, y):  
    """A pythonic subtraction."""  
    res = x - y  
    return res
```

and with a fn function:

```
fn add(x: Int, y: Int) -> Int:  
    """A rustacean addition."""  
    let res: Int = x + y  
    return res
```

The manner in which Mojo employs `var` and `let` for mutable and immutable variable declarations respectively mirrors the syntax found in Swift. In Swift, `var` is used for mutable variables, while `let` is designated for constants or immutable variables.^[15]

Variable declaration and usage in Mojo:

```
fn main():  
    let x = 1  
  
    let y: Int  
    y = 1  
  
    var z = 0  
    z += 1
```

Figure 4 (credit: [https://en.wikipedia.org/wiki/Mojo_\(programming_language\)](https://en.wikipedia.org/wiki/Mojo_(programming_language)))

- There is also VS Code Extension for MOJO available on the internet provided by the Modular company itself. It is an optional choice for the MOJO developers to download it from the provided link.

Here is the google chrome link for installing the VS Code Extension:
<https://docs.modular.com/mojo/manual/get-started>

7. Access the Python Ecosystem

As we discussed above, MOJO offers the subset of python's syntax and some libraries of the python ecosystem to rapidly increase the speed and quality of development. Seamlessly intermix arbitrary libraries like NumPy and Matplotlib and your custom code with MOJO.

Here is a Matplotlib code example to understand the correlation in syntax of both programming languages Python and MOJO in *Figure 5*:

```
def make_plot(m: Matrix):
    plt = Python.import_module("matplotlib.pyplot")
    fig = plt.figure(1, [10, 10 * yn // xn], 64)
    ax = fig.add_axes([0.0, 0.0, 1.0, 1.0], False, 1)
    plt.imshow(image)
    plt.show()

make_plot(compute_mandelbrot())
```

Figure 5 MOJO Matplotlib example (Source: <https://xpertlab.com/mojo-the-programming-language-for-ai-that-is-up-to-35000x-faster-than-python/>)

8. Benefits of MOJO over python

Some benefits of using MOJO over Python in the future for developing AI and ML:

- MOJO is 68000x faster than Python.
- It has better integrity with GPUs, TPUs and other AI hardware components because of the MLIR technology used in the built structure of MOJO.
- Its syntax as easy as high-level language like Python and fast as low-level language like C.
- It has borrowed checker influenced from Rust to make it easy for the MOJO developers to work within different functions.
- Mojo supports both Just-In-Time (JIT) and Ahead-Of-Time (AOT) compilation. Mojo compiler applies advanced optimization and GPU and TPU code generation.
- Mojo gives full control over memory management, layout and low-level details.
- It provides manual memory management system with the help of pointers that is similarly used in C++ and Rust.
- By combining dynamic and system language capabilities, it follows unified programming model which makes it scalable for various use cases based on accelerators.
- It has built in tiling optimization tool that improves cache locality which divides computations into smaller tiles that fit into fast cache memory.
- The autotune module offers interfaces for adaptive compilation which find the best parameters for the target hardware by automatically tuning the code.
- It uses the ownership and borrowing system to manage memory, negating the need for garbage collector which keeps consistent runtime performance.
- It offers a frameworks like “Basalt” which is a standalone Machine Learning framework, “Endia” which is a dynamic Array library for scientific computing, “Lightbug HTTP” which is a HTTP web framework which is a purely written in mojo and many more open-source projects which are managed by the Modular company.

9. *Conclusion and Future Work*

MOJO has a bright future as a programming language in developing AI and ML with the ease of learning it and its syntax with the performance of low-level languages.

Modular company open sourced the MOJO standard library and the whole modular team is working on making it a better language and make it available for everyone so that the development of Artificial Intelligence and Machine Learning can take a rapid growth in the future.

A lot of work is remaining on this language but Modular team is currently working on it and in future, it will be one of the best languages to be used for the development of Artificial Intelligence and Machine Learning.

Now, the question comes is will MOJO replace python? The answer to this question is that there is not a certain answer available with the current level of mojo. Python has a very large community whereas it is a new programming language that's why it will take years for mojo to get to the level of python. There are developed libraries available for the python developers and mojo only offers few libraries for the developers but, due to the speed that mojo offers in the development of Artificial Intelligence, the mojo has an advantage where the developers need speed in their system.

Mojo have a great potential in field of Artificial Intelligence and Machine Learning development. We need to work on the language to make it better by making a strong community for mojo developers and making it compatible with other python libraries.

11. *References*

1. Wikipedia – for providing the relevant information for the technology and the developing team
2. Modular's official info. – for providing all the benchmarks and examples of codes in the MOJO programming language.
3. <https://xpertlab.com/mojo-the-programming-language-for-ai-that-is-up-to-35000x-faster-than-python/>
4. [https://en.wikipedia.org/wiki/Mojo_\(programming_language\)](https://en.wikipedia.org/wiki/Mojo_(programming_language))
5. <https://docs.modular.com/mojo/manual/get-started>
6. <https://shriramsivanandhan.medium.com/mojo-programming-language-68000x-faster-than-python-programming-in-mojo-part-ii-d162740a2f67>
7. <https://www.modular.com/mojo>