artificialcorner.com

Mojo: The Programming Language for AI That Is Up To 35000x Faster Than Python

The PyCoach

4–5 minutes

Introducing Mojo — the new programming language for AI developers.

Image created with Midjourney

A new programming language for AI developers was just released: Mojo.

I know what you might be thinking — a new programming language to learn from scratch … Well, I have good news, Mojo is designed as a superset of Python, so if you already know Python learning Mojo shouldn't be hard.

But that's not all. Mojo combines the usability of Python with the performance of C obtaining a speed that is up to 35000x faster than Python.

If you're into AI and already know Python, Mojo is definitely worth a try. Here's everything you need to know about Mojo.

Why do we need Mojo if we already have Python?

Python's simplicity and versatility made it the language of choice in fields such as data science, machine learning, and AI. It has tons of packages that are very useful for anyone working with data, but for libraries that require great performance, Python only acts as a glue layer and low-level bindings to C, C++, and other languages with better performance.

This enabled the development of libraries such as numpy and TensorFlow. However, this comes with a drawback: building these libraries is very complicated, it requires a low-level understanding of the internals of CPython, requires knowledge of C/C++, etc.

According to the Mojo doc, the issues brought by Python go deeper and particularly impact the AI field.

Python alone isn't able to address all the issues that applied AI systems need and that's how Mojo was born. Mojo is a

programming language that combines the usability of Python with the performance of C.

The best of both worlds!

But Mojo isn't a random project that emerged out of nowhere. In fact, Mojo comes from a company named Modular, co-founded by Chris Lattner, the same guy who created the Swift programming language and LLVM.

That's why I think this project is worth paying attention to. Now let's see some of Mojo's best features.

Features of Mojo

Mojo comes with many interesting features out of the box. Here are some of them.

1. Mojo is designed as a superset of Python

Mojo aims to be fully compatible with the Python ecosystem.

This means that you could easily work with Mojo if you're a Python programmer because both programming languages have many functions, features, and libraries in common.

Libraries such as numpy, pandas, and matplotlib are also available in Mojo. Here's how you'd make a plot with matplotlib using Mojo.

```
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())
```

Screenshot: Mojo

That said, Mojo is still in a very early stage, so it still misses many features of Python (for example it doesn't support classes yet).

Hopefully, in future updates Mojo will be fully compatible with Python.

2. Strong type checking

Mojo leverage types for better performance and error checking.

```
def sort(v: ArraySlice[Int]):
  for i in range(len(v)):
    for j in range(len(v) - i - 1):
      if v[j] > v[j + 1]:
        swap(v[j], v[j + 1])
```

Screenshot: Mojo

Although you can still use flexible types like with Python, Mojo lets you use strict type-checking. This can make your code more predictable, manageable, and secure.

3. Memory ownership and borrowing checker

Mojo supports a `owned` argument convention that is used for functions that want to take exclusive ownership over a value.

```
def reorder_and_process(owned x: HugeArray):
  sort(x)       # Update in place

  give_away(x^) # Transfer ownership

  print(x[0])   # Error: 'x' moved away!
```

Screenshot: Mojo

This will help you take advantage of memory safety without the rough edges.

4. Auto-tuning

Mojo has built-in autotuning that helps automatically find the best values for your parameters to take advantage of target hardware.

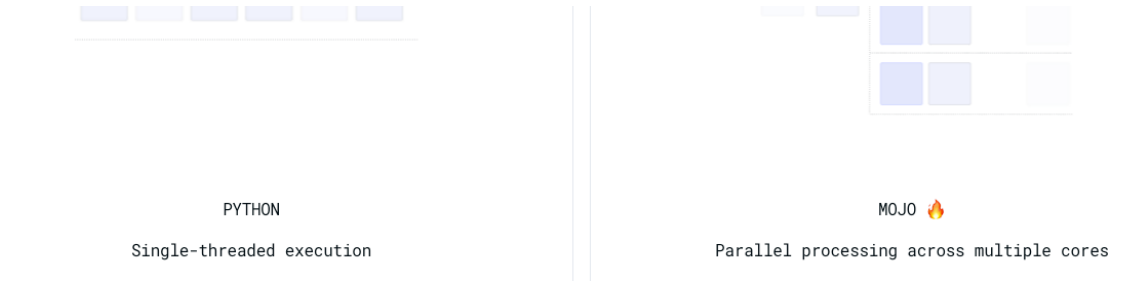
```
def exp_buffer[dt: DType](data: ArraySlice[dt]):

  # Search for the best vector length
  alias vector_len = autotune(1, 4, 8, 16, 32)

  # Use it as the vectorization length
  vectorize[exp[dt, vector_len]](data)
```

Screenshot: Mojo

5. Mojo leverages MLIR

By using the full power of Multi-Level Intermediate Representation (MLIR), Mojo developers can take advantage of vectors, threads, and AI hardware units.

This helps Mojo achieve great performance because, unlike Python which works with single-threaded execution, Mojo can work with parallel processing across multiple cores.



Screenshot: Mojo

That’s one of the reasons why Mojo it’s 35000x faster than Python.

LANGUAGES	TIME (S) *	SPEEDUP VS PYTHON
PYTHON 3.10.9	1027 s	1x
PYPY	46.1 s	22x
SCALAR C++	0.20 s	5000x
MOJO 🔥	0.03 s	35000x

Screenshot: Mojo

How to start using Mojo

Mojo is still a work in progress, but you can try it today on the JupyterHub-based Playground. To try Mojo go to this website to register and don’t forget to check the Mojo box in the “Modular Product Interest” section.

Happy coding!