University of Puerto Rico
Mayagüez Campus
Department of Electrical and Computer Engineering

Is Python Fast or Slow?

Prof. J. Fernando Vega Rivero
ICOM 5015 - 001D

David A. Castillo Martínez
Christian J. Perez Escobales
Ramón J. Rosario Recci
February 28, 2024

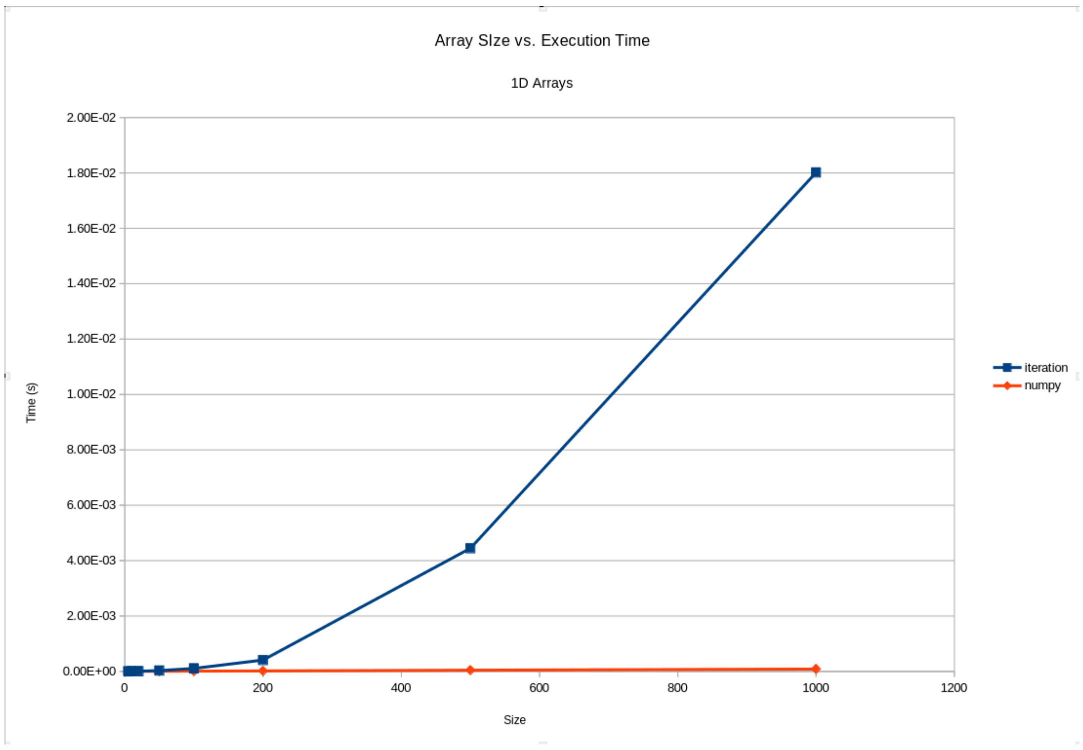
Abstract

This study investigates the efficiency of product operations between native Python arrays and NumPy arrays, hypothesizing that NumPy arrays will outperform Python arrays due to their optimization for numerical operations. Through experimentation with one-dimensional and two-dimensional arrays, it was observed that NumPy arrays indeed exhibit superior execution times, particularly as array size increases. The improved efficiency of NumPy arrays stems from their continuous data storage in memory, facilitating faster operations compared to native Python lists. These results underscore the importance of leveraging specialized libraries to enhance performance and streamline development processes in programming tasks.

The concept of reusable code components or programming libraries traces back to the early days of computing. A programming language is a collection of pre-written code that you can use to perform specific tasks, effectively reducing the amount of code a programmer needs to write by providing reusable functions or classes that can be called upon as needed. [1] Nowadays, most programming languages have available libraries that users can leverage to their utmost potential. In Python, one of the most utilized libraries is known as NumPy, a fundamental package for scientific computing that provides multidimensional array objects and an assortment of routines for fast operation on arrays. [2] Depending on the size of the arrays, the product operation between arrays can take an extensive amount of time. With this assignment we will determine whether product operations are quicker when using native Python arrays or NumPy arrays. The current hypothesis is that NumPy arrays will outperform Python arrays because the library is optimized for numerical operations including the product between arrays, resulting in lower execution times.

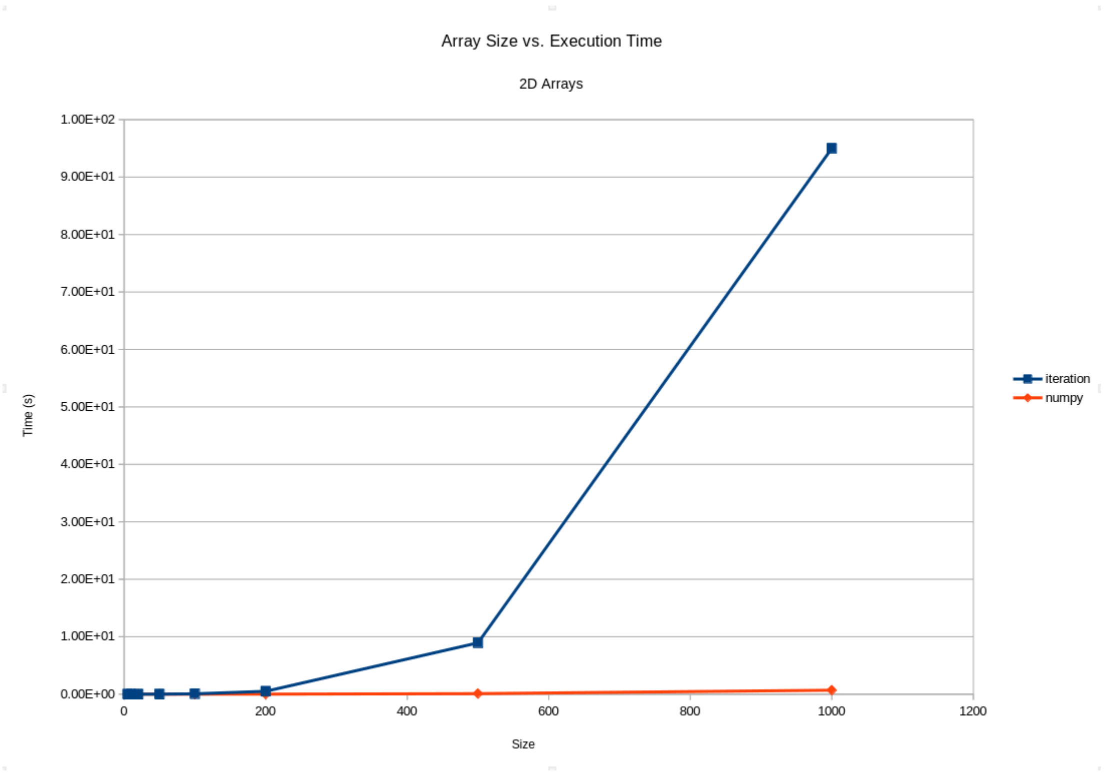
The assignment consists of comparing the execution times of product operations between one-dimensional arrays and two-dimensional arrays when using NumPy arrays and native Python arrays, to determine which type of array is more efficient to use. To determine the execution times when using Python arrays there was some setup to be made, including a function that creates one-dimensional Python arrays, a function that creates two-dimensional Python arrays, a function to calculate the product of 2 one-dimensional Python arrays and finally a function to calculate the product of 2 two-dimensional Python arrays. Using these functions, the execution time of a product operation using Python one-dimensional and two-dimensional arrays was calculated. Additionally, the calculation was performed with a size from 5 to 1000 to cover a variety of cases with different dimensionalities. This same process was repeated using NumPy arrays; however, a function named dot that calculates the product of arrays was used since it is in the NumPy library.

Graph 1 shows the comparison between the times it takes to calculate a product of 2 one-dimensional arrays using iteration in Python and using NumPy. For arrays with a size smaller than 20, the difference in time is negligible; in fact the time it took to perform these calculations were both in the range of microseconds. However for arrays of bigger sizes the time difference becomes more evident. As the size of the array increases, the execution time increases exponentially when using iterations. Meanwhile, when using NumPy arrays the time increases in a linear fashion. For arrays with a size of 1000, the biggest size utilized in this assignment, iteration took around 18 milliseconds, while NumPy took only 86 nanoseconds.



Graph 1: Array size vs. Execution time of one-dimensional arrays

Graph 2 shows a similar behavior to Graph 1 when it comes to calculating the product of 2 two-dimensional arrays. When the arrays had a size smaller than 20, the difference in time between using NumPy and Python arrays is also negligible, taking an approximated execution time in the microseconds range. However, for arrays with a size larger than 20 the difference in time is even more pronounced than when using one-dimensional arrays. While the execution time for both iteration and NumPy increased, the execution time increased exponentially with the size of the array when using Python arrays. On the other hand, the execution time slightly and linearly increases as the array size increases, when using NumPy arrays. For arrays with a size of 1000, iteration took 95 seconds while NumPy took 0.7 seconds, a significant difference between execution times.



Graph 2: Array size vs. Execution time of two-dimensional arrays

After analyzing and comparing the execution times between the Python arrays and NumPy arrays, it was determined that the product operation using NumPy arrays is significantly more efficient than when using Python arrays. This was even more prevalent the larger the size of the arrays as the difference between the executions grows even bigger. Additionally, it was observed that two-dimensional arrays had an overall higher execution time and the exponential difference between the execution times of NumPy and Python arrays was even more prevalent when using two-dimensional arrays. After conducting thorough research into the reasons for this phenomenon, it was found that NumPy arrays store data in memory in a continuous fashion, which improves space utilization and allows for faster operations than native Python lists. [3]

In conclusion, most programming languages these days offer programming libraries that users can leverage to their fullest potential, exemplified by the widespread use of NumPy in Python for scientific computing tasks. This assignment focused on comparing the efficiency of product operations between native Python arrays and NumPy arrays. After experimenting with one-dimensional and two-dimensional arrays, it became evident that NumPy arrays outperform Python arrays in terms of execution time for product operations, particularly as the size of the arrays increases. It was also found that NumPy's efficiency originates from NumPy arrays storing data in a continuous fashion in memory, allowing for faster operations. This outcome perfectly aligns with the established hypothesis that NumPy arrays will outperform Python arrays due to them being optimized for numerical operations like the product between arrays, and result in lower execution times. These findings perfectly highlight the importance of leveraging specialized libraries to enhance performance and streamline development processes when programming.

References:

[1] G. Woke, “The difference between libraries and frameworks,” *Simple Talk*, Mar. 23, 2023. https://www.red-gate.com/simple-talk/development/other-development/the-difference-between-libraries-and-frameworks/ (accessed Feb. 27, 2024)

[2] “What is numpy?#,” What is NumPy? - NumPy v1.26 Manual, https://numpy.org/doc/1.26/user/whatisnumpy.html (accessed Feb. 27, 2024).

[3] P. Qian, “Python Lists Vs. NumPy Arrays: A Deep Dive into Memory Layout and Performance Benefits,” *Data Leads Future*, Jan. 06, 2024. https://www.dataleadsfuture.com/python-lists-vs-numpy-arrays-a-deep-dive-into-memory-layout-and-performance-benefits/ (accessed Feb. 28, 2024)

Task Distribution:

- Ramon J Rosario Recci - Implemented Python code for manual array multiplication and NumPy using its functions, and recorded the code and results for the presentation.
- DAVID A CASTILLO-MARTINEZ - Collaborated on the analysis, interpretation of results, wrote the assignment’s formal report, recorded conclusions and learning slides for the presentation.
- CHRISTIAN J PEREZ-ESCOBALES - Collaborated on the analysis, interpretation, created the presentation, recorded introduction, hypothesis and key concepts slides, and edited the video presentation.