Module 4: Portfolio Milestone

Read. Double. Write.

Overview

We compared three Python methods against each other and against a Bash method. Each with the task of processing a file(file1.txt) containing 1,000,000 random integers. The processing goal was to double each number, read into memory, and capture the execution times for all of the approaches.

Outputs

```
➜ portfolio_milestone git:(main) ✗ ./double_numbers.sh
Doubled numbers have been saved to newfile1.txt.
Script execution time: .370118553 seconds.
```

Bash Output

```
➜ portfolio_milestone git:(main) ✗ python processor.py
Method read_all_into_memory() execution time: 0.4281 seconds.
Method read_row_by_row() execution time: 0.3796 seconds.
Method halve_file_into_memory() execution time: 0.3223 seconds.
```

Python Output

Questions

How do they compare to each other, and to that of the double_numbers.sh script?

All four of the Python and Bash methods ran with similar execution times, at about one third of a second. However, the Python method to halve the input file and read each into memory ran the quickest. Whereas, the method to read the file line by line and read into memory ran the slowest.

Were there any surprises for you, or did the results match your expectations?

I assumed the Bash script would run quicker than the Python Script. Which was confirmed.

However, I was surprised with how the Bash script ran about three times faster than the Python script. The Bash script has proved to be more efficient for this task.