




DFA Multithreading and Multiprocessing Comparison:




This project checks how much time a DFA program takes to run when we use different numbers of threads and processes.

We tested it with 5, 10, and 15 threads and processes.

Multithreading Results:

```
 Threads: 5 | Time taken: 0.0210 seconds  
 Threads: 10 | Time taken: 0.0407 seconds  
 Threads: 15 | Time taken: 0.0590 seconds
```

Multiprocessing Results:

```
 Processes: 5 | Time taken: 1.1638 seconds  
 Processes: 10 | Time taken: 1.3755 seconds  
 Processes: 15 | Time taken: 1.8436 seconds
```

Comparison Summary:

Multithreading is much faster than Multiprocessing.- DFA code is not heavy, so threads work faster because they share the same memory.

Processes take more time because each process runs separately and uses more memory.

Conclusion:

In this project, we tested how long the DFA program takes to run with different numbers of threads and processes — 5, 10, and 15 each.

The results showed that **multithreading** was much faster in all cases. For example, with 5 threads the program finished very quickly, and even with 15 threads it was still faster than multiprocessing.

On the other hand, **multiprocessing** took more time to complete because each process runs separately and uses more system memory.

This means that for our DFA project, where the tasks are not very heavy, **multithreading** works better and saves more time, while **multiprocessing** is slower and less efficient for this type of work.