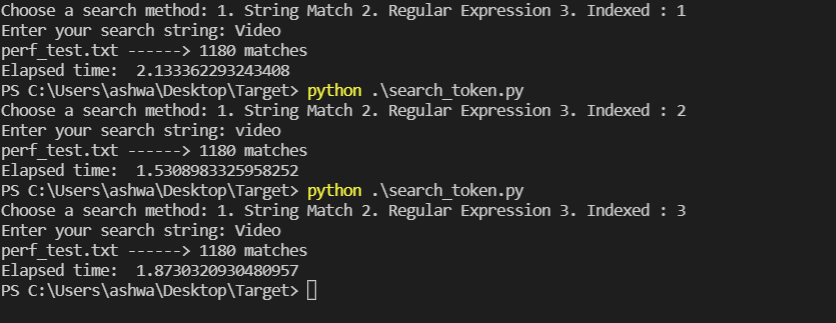
**Run a performance test that does 2M searches with random search terms, and measures execution time. Which approach is fastest? Why?**

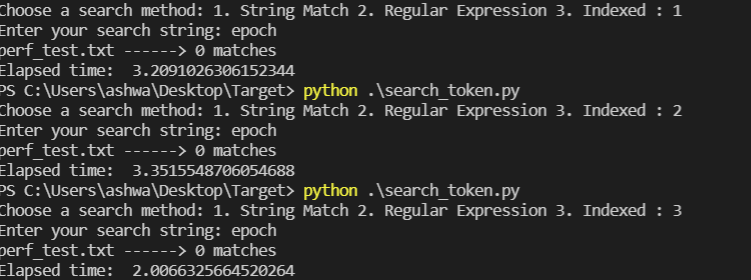


For performance test purposes, I created a document(‘search\_test’) with random words. I search for the same word in the document using the three different methods.

Theoretically, we know that method 3 would be the fastest method, since storing and retrieval of the token results should be O (1). This is because we are using a dictionary to store the words and their occurrence.

Practically, running the search against the index might take longer, since we need to first preprocess the document and store it as an index for each file in the search directory. However, retrieval in this method would be faster than in the other methods.

You can see below the results for a token that does not exist in the file, and the execution time for the different methods:



**Conclusion**: Method 3 would be the fastest since the complexity of storage and retrieval would be O (1).

**Provide some thoughts on what you would do on the software or hardware side to make this program scale to handle massive content and/or very large request volume (5000 requests/second or more).**

For a larger repository or for much larger documents, we can leverage Hadoop MapReduce for processing and generating results with a parallel, distributed algorithm on a cluster. The map procedure performs filtering and sorting and the reduce method would summarize the results.

We can implement sharding which again splits the large document or the document repository into different sub-files. A main server would handle the logging of indexes to determine which shard to search, for a particular search term. Worker threads would be responsible for indexing and handling the different shards and would provide the results to the main thread which would then return this result to the user.

We could use Solr (or ElasticSearch), which is a search platform. We can either preprocess our files, or we can have the Solr index ingest our documents and then perform searches against that index. Solr offers multiple search options including regular expressions and combining multiple searches.

In terms of hardware, we could use a multi-threaded process, so that we can optimize the utilization of our system resources. If we have multiple servers, a main server could also have the details of the memory and CPU performance of each worker server, and create chunks of the data to send to each machine so we have no performance issues.