**Marianne Mae T. Tabios**

Role: Lexer.py/Tester/Documentation

Project: LOLCODE Interpreter - CS1332 Group 6

REFLECTION

Working with the lexer.py file provided me with a deeper understanding of how lexical analysis functions as the foundational step in interpreting and compiling programming languages. This script represents a basic lexer that processes a language similar to LOLCODE, highlighting how a stream of source code is broken down into manageable tokens for further analysis.

One of the most valuable insights I gained from studying and working with this code is the importance of structure and precision in language processing. The Lexer class is methodically designed to handle source code line by line, ignoring empty lines and handling strings, numbers, keywords, and identifiers effectively. The handling of quoted strings and the exclusion of commas outside of strings show an attention to detail that is crucial in lexers, as even a minor mistake in tokenizing input could lead to incorrect parsing later.

Another important takeaway is the practical use of object-oriented programming. The Token class encapsulates each token’s type and value, which makes the list of tokens easy to manage and debug. It also provides a flexible way to expand the language by simply adding new keywords or token types.

This project strengthened my understanding of how programming languages are built under the hood, especially in terms of how code is initially broken down into meaningful elements. It also reminded me how essential lexers are not only in compilers but also in interpreters, analyzers, and various other software tools that process code or structured text.

Overall, working with lexer.py gave me a hands-on appreciation of compiler theory concepts and deepened my appreciation for clean, readable, and well-organized code. It encouraged me to think not just as a programmer, but as a language designer even on a small scale.