CS420 – Homework 1: Lexical Scanner for Eck

The task for this homework is to write a lexical scanner for the Eck programming language. If you wish, you may work on this homework with one other student from the class; be sure to list both students' names in all deliverables. As part of this assignment, you are specifically forbidden from:

- Using a Python function to directly convert a multi-digit string representation of an integer to its integer value.
- Using modules other than those already imported in the supplied *scanner.py* file.
- Using a tool that automatically generates any part of your scanner. In other words, you need to program your scanner from scratch.
- Using an AI to generate all or part of your code.

Supporting Materials

The assignment on Moodle includes a zip file containing the following files:

- *Eck Lexical Description.pdf*: A detailed description of the lexical elements your scanner must recognize.
- *lexeme.py*: A module containing two classes – `Lexeme`, which is an enumerated type for the different lexeme categories; and `LexemeMap`, which contains dictionaries for mapping keywords and symbols to their associated Lexeme category. You must use the constants declared in this file; you cannot roll your own representation. Do not modify the contents of this file!
- *scanner.py*: A module template for your scanner that contains three things of importance:
  - The complete class definition for `ScannerError`, which is a custom error class you should use for raising scanning errors.
  - An incomplete definition of the `Scanner` class, showing the signatures of the two methods you must include in your implementation, `__init__` and `nextLexeme`. You can include as many other methods and class definitions as you see fit to accomplish the task.
  - In the testing section of the module (i.e., the code after the `if __name__ == "__main__"` statement near the bottom of the file) is a script for testing your scanner on the set of Eck files described below. I will be using this script to test your software.
- *ScannerTests*: A folder containing ".eck" files for testing your scanner. The correct results that should be produced by the testing script are found in the corresponding ".answer" files. Note that not all of the ".eck" files are syntactically correct examples of the Eck programming language; they were written specifically to provide a robust test of the scanner.

Deliverables

You need to flesh out the implementation of the `Scanner` class. In particular, the method `nextLexeme`() should return a token pair (*category*, *value*) where *category* is an enumerated constant of the `Lexeme` class, and *value* is the value (if any) associated with the lexeme. The values associated with the lexeme categories are:

Category	***Value***
`Lexeme.INTEGER_CONST`	Integer value of the constant
`Lexeme.STRING_CONST`	String value of the constant
`Lexeme.IDENTIFIER`	String representation of identifier
Keywords	None
Symbols	None

You should look at the provided ".answer" files for a better idea of what should be returned. In particular, be sure to use the exact same wording for error messages as you see in the files *Error1.answer*, *Error2.answer*, *Error3.answer,* and *Error4.answer*.

You should upload to Moodle your *scanner.py* module and any other modules you may have written as part of this homework. You do not need to upload the *lexeme.py* module.

Grading Criteria

5%	In a comment at the top of each file, you include your name (and your partner's, if any), the course number, the assignment number, and the Eckerd honor pledge.
10%	Each of your modules, classes, methods, and functions are appropriately documented.
10%	Meaningful names are used for each of your modules, classes, methods, functions, and variables.
10%	Your modules are clean, not containing dead code, debugging print statements, or other cruft.
20%	All provided test files are correctly scanned.
45%	Quality and correctness of software design