****

**CC ASSIGNMENT # 02**

**LEXICAL ANALYSIS**

**Group Partners:**

|  |  |  |
| --- | --- | --- |
| **NAME** | **SID** | **CID** |
| HAMMAD ABDUL BARI | 10708 | 111707 |
| HAFIZ FARZAN AHMED | 10712 |
| MUHAMMAD BILAL | 10559 |

# What is Lexical Analysis?

Lexical Analysis is the first step in a compiler that breaks down a program into smaller units called tokens. This is done using a technique called Deterministic Finite Automata (DFA). The output is a sequence of tokens that is checked in the next phase to ensure it forms a valid program.

# Tokens:

A token is a group of characters in a program that represents a meaningful unit in the programming language. Tokens can be of different types, such as identifiers, constants, operators, keywords, and punctuation symbols. Identifiers are names given to variables and functions, constants are fixed values that don't change, operators perform actions on data, keywords have a specific meaning and cannot be used as identifiers, and punctuation symbols are used to separate different parts of the program. The job of the compiler's lexical analyzer is to identify these different types of tokens in the program.

# Language:

A formal language is a collection of words made up of symbols from a limited set of characters. To define a formal language, we can use rules that explain how to create the words (such as regular expressions or context-free grammar) or machines that can identify if a given word belongs to the language.

# Python Language:

Python is a popular programming language that is easy to learn and use. It is often used for Rapid Application Development, as well as for scripting or gluing components together. Python's high-level built-in data structures and dynamic typing make it flexible and efficient. It has a simple and readable syntax, which makes it easier to maintain programs. Python supports modules and packages that encourage code reuse and modularity. Python is freely available and can be used on many platforms without charge.

# TCL Language:

Tcl is an easy-to-learn programming language that is powerful and flexible. It uses commands for everything, even programming constructs like assigning values to variables and defining procedures. Tcl can be used in multiple programming styles, including object-oriented, imperative, and functional. It is commonly used for rapid prototyping, scripted applications, GUIs, and testing. Tcl can also be embedded into C applications.

# Code Description:

Documentation:

1) Define a custom exception class TclSyntaxError to handle syntax errors in Tcl code.

2) Define a tokenize function that takes a string of Tcl code and returns a list of tokens. It uses regular expressions to match and extract different types of tokens such as comments, strings, variables, numbers, operators, parentheses, and identifiers. It ignores whitespace and newline characters.

3) Create an empty list tcl\_input to store the user input Tcl code.

4) Prompt the user to enter Tcl code line by line and append each line to tcl\_input until an empty line is entered.

5) Join the lines in tcl\_input using newline characters to form a single string of Tcl code.

6) Create a counter variable counter to keep track of the number of output files.

7) While True:

a. Generate a new output filename file{counter}.csv using the counter variable.

b. Check if a file with the generated filename already exists.

c. If it exists, increment the counter and repeat from step a.

d. If it doesn't exist, try to tokenize the user input Tcl code using the tokenize function.

e. If the tokenization is successful, write the list of tokens to a CSV file with the generated filename.

f. If tokenization fails due to a syntax error, catch the exception and print the error message.

g. Exit the loop.

os.path: This module provides a way of using operating system dependent functionality like reading or writing to the file system. In this code, it is used to check if a file with a certain name already exists.

re: This module provides regular expression matching operations. It is used to tokenize the input string into separate tokens based on certain patterns.

match.lastgroup: This is a method that returns the name of the last matched capturing group in the regular expression.

raise: This is a Python keyword that raises an exception with a specified message if certain conditions are met. In this code, it is used to raise a TclSyntaxError if an unknown token is encountered during tokenization.

try/except: These are Python keywords used for exception handling. In this code, they are used to catch any TclSyntaxError that may be raised during tokenization.

csv: This module provides functionality to read from and write to CSV files. In this code, it is used to write the tokenized output to a CSV file.

tokens: This is a list that holds the tokens generated during tokenization.

writer: This is an instance of the csv.writer class that is used to write the tokenized output to a CSV file.

writerow: This is a method of the csv.writer class that writes a single row to a CSV.

input(): This is a built-in function that allows the user to input a value from the console.

join(): This is a string method that concatenates a sequence of strings into a single string, using the original string as a separator.

append(): This is a list method that adds an item to the end of a list.

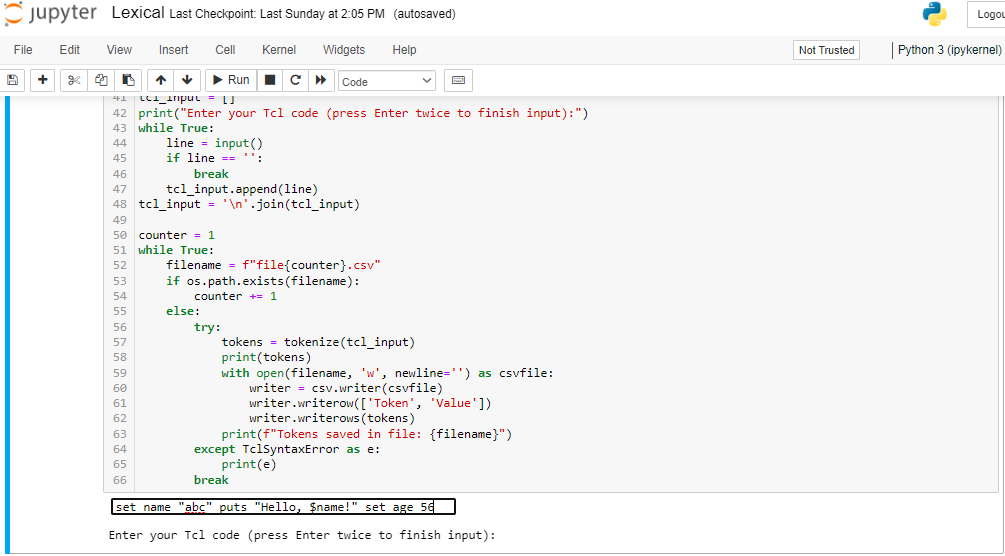
finditer(): This is a method of the re module that returns an iterator yielding match objects over all non-overlapping matches for the regular expression pattern in the string.

group(): This is a method of the match object returned by re.finditer() that returns the string matched by the given group.

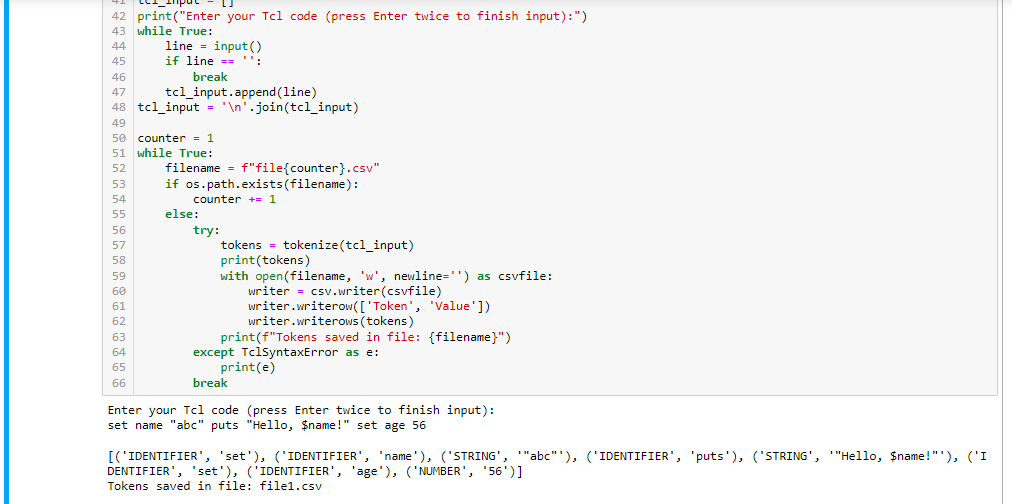
exists(): This is a method of the os.path module that returns True if a file or directory with the given name exists, and False otherwise.

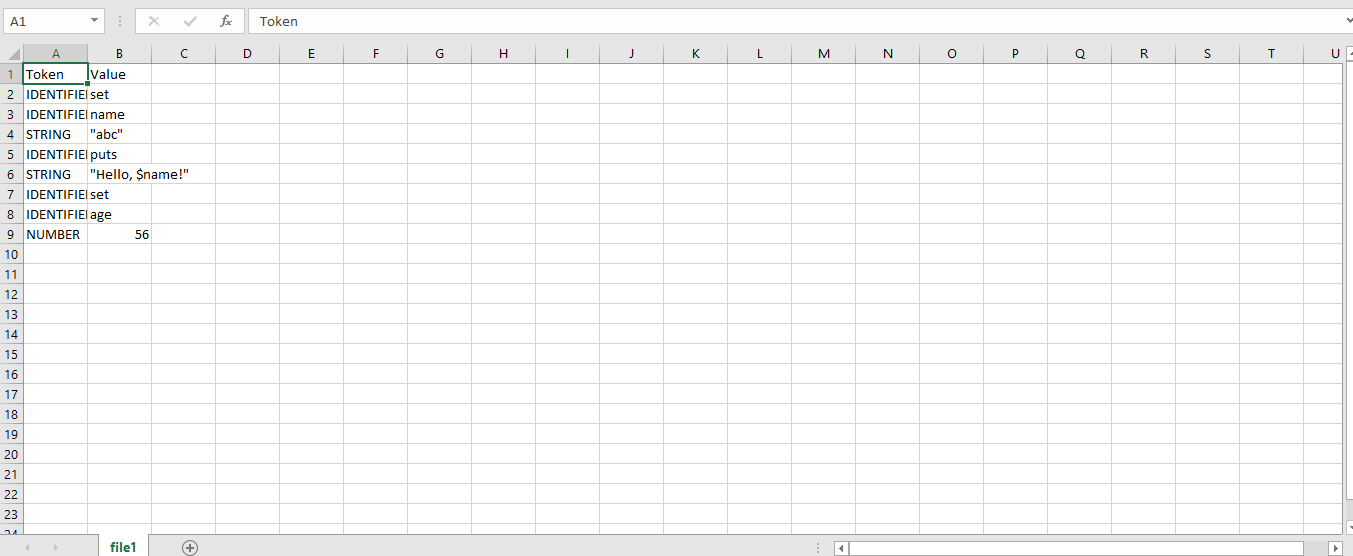
# Test Cases:

## Input:



## Output:





# Code Description:

This code defines a function called tokenize that generates tokens (lexical analysis) for input code in the Tcl programming language. It defines a class TclSyntaxError for syntax errors and uses regular expressions to match the different types of tokens. It then saves the tokens to a CSV file. The user is prompted to input Tcl code, and the tokens are saved to a CSV file named "fileX.csv" where X is a counter. If there is a syntax error, the program prints an error message.

# GitHub Link:

# Conclusion:

The given assignment is about Lexical Analysis, which is the first step in a compiler that breaks down a program into smaller units called tokens. These tokens are then used to form a valid program. The assignment provides information about tokens, formal languages, and Python and Tcl languages. The code description of the assignment explains the use of different Python modules and keywords, and how to handle syntax errors. It also provides the logic of how to tokenize the user input Tcl code and write it to a CSV file. Overall, the assignment covers the basics of Lexical Analysis and how to implement it in Python.

# References:

1) <https://www.geeksforgeeks.org/introduction-of-lexical-analysis/>

2) <https://www.sciencedirect.com/topics/computer-science/formal-language#:~:text=In%20automata%20theory%2C%20a%20formal,accepts%20(recognizes)%20the%20language>.

3) <https://www.python.org/doc/essays/blurb/>