Lexer/Parser for simple calculator

Theory of Computing Phase 2

Supervised by

Dr. Zeinab Abdelhalim

Eng. Ahmed Gamal

Bahaa Shamoon Atia **Ahmed Aboelkassem** **Mouhab Abdelaziz**

202001572 **211001202** **202000412**

Ahmed AlWatany

202000689

Table of Contents:

1. **What is Lexer analysis / Parser analysis**
   - **1.1 Lexer analysis**
   - **1.2 Parser analysis**
2. **Logic Behind Lexer analysis / Parser analysis**
3. **Code**
   - **3.1. Inputs Format**
   - **3.2. Outputs Format**
   - **3.3 inside mechanism**
4. **used libraries & References**

1. What is Lexer analysis / Parser analysis

1.1 Lexer analysis

The lexer conducts lexical analysis, wherein the input string is transformed into a list of tokens. Each token possesses its unique significance and may carry a value. Initially, we delineate all the tokens required for the process.

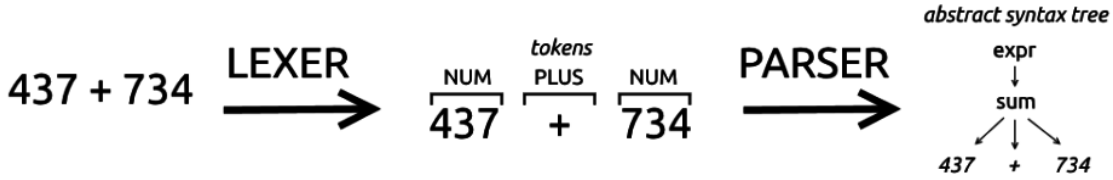
Lexical analysis involves breaking down the input stream into tokens. In the case of a calculator, tokens may include numbers, operators, parentheses, and other symbols. The lexer scans the input character by character, identifying and categorizing each segment into respective token types. Techniques such as regular expressions or finite automata are commonly employed for efficient tokenization.

1.2 Parser

Parsing is the process of analyzing the structure of tokens to determine their syntactic validity according to predefined grammar rules. In a calculator, parsing involves constructing a parse tree that represents the hierarchical relationship between the tokens. Popular parsing techniques include recursive descent parsing, operator-precedence parsing, and parser combinators. These techniques vary in complexity and efficiency, depending on the grammar of the language being parsed.

2. Logic Behind Lexer analysis / Parser analysis

Lexer Analysis and parsing are Two consecutive operations first we tokenize the input then we parse this token.



The lexer scans the text and finds 4, 3, 7 and then a space, The job of the lexer is to recognize that the characters 437 constitute one token of type NUM. The parser will typically combine the tokens produced by the lexer and group them.

3. Code

The code was written in Python

3.1 input Format

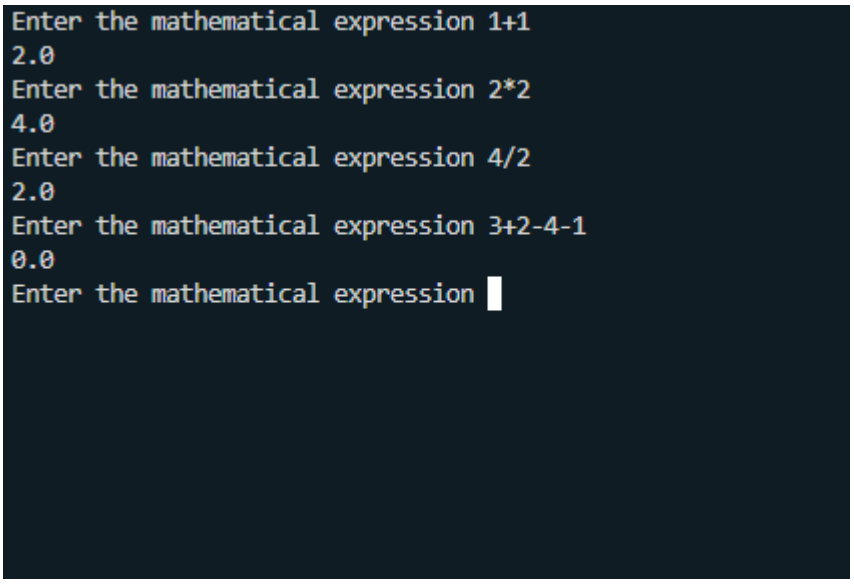
The program will be interactive the user will write the mathematical expression.

for example:

- 1+1, 2*2, 4/2 …etc.

3.2 Output Format

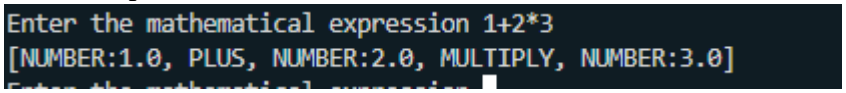
The output will be the result of the mathematical operation after each user input.



3.3 Inside Mechanism:

The code is divided into 3 main parts (lexes analysis – parsing – Interpreter).

At the first we take the input from the user and translate it into tokens



As we can see the token is divided int 2 parts type and values, the value may be none as shown for the operational types like plus, multiply…etc.

After tokenizing we start parsing and building the tree and make sure to write the tree in the right mathematical operational order like multiplication before addition or prances first. So, to do that we follow this Grammar Rules

Where the Factor is the single number and term is looking for multiplication and division then Expression is looking for the addition and subtraction.

```
Enter the mathematical expression ((123-1)*2-1)
(((123.0-1.0)*2.0)-1.0)
```

Finaly, we travers the tree and execute the appropriate code using Interpreter code.

Used libraries.

We didn't use any external libraries we just imported Datacalss from the standard library of python.

References

1. https://www.javacodegeeks.com/2017/09/guide-parsing-algorithms-terminology.html
2. https://www.geeksforgeeks.org/parse-tree-in-compiler-design/