MATH INTERPRETER

A Course Project Report - System Programming

Subject Teacher: Prof. Mrs. D. J. Joshi

Group 7

Members:

Sagar Sikchi	65	11810844
Pradunya Maladhari	53	11810567
Shantanu Sontakke	66	11810898
Talib Hussain	68	11811318
Aadarsh Kandewar	27	11810942

CONTENTS

1. What is Math Interpreter?

Math interpreter is a simple interpreter which takes input (mathematical expressions sequentially and generate output in the form of tokens, parse tree and the value of that mathematical expression.

For example:

The input mathematical expression:

57 + 60 * 100 ^ 5

Then, output will be like -

1. Tokens are -
   [NUMBER: 57.0, PLUS, NUMBER: 60.0, MULTIPLY, NUMBER: 100.0, EXPONENT, NUMBER: 5.0]

2. Parse Tree is -
   (57.0+(60.0*(100.0^5.0)))

3. Value evaluated is –
   600000000057.0

This interpreter is made from scratch and it shows that how actually the computer performs mathematical calculations in real cases.

2. Design and Implementation

This interpreter is implemented in several steps. To build the interpreter, the Python programming language is used, since it is easy to understand and has a GUI (graphics user interface) support. The design of an interpreter is as follows –

1. The first is to build a lexer part which can generate the valid tokens.
2. Then the parse tree is generated using parser and generated tokens.
3. After parser, the interpreter is built which evaluate the parsed expression.
4. At last, the file handling is done to generate output in output text file.

- The Lexer

The lexer is the first step in building the interpreter. It is nothing but the lexical analysis of input expression.

In input, what we are looking for is the numbers, operators like plus (+), minus (-), multiplication (*), division (/), modulo (%), exponent (^), left parenthesis and right parenthesis [(,)].

These characters will give the valid tokens and rest all the characters will give invalid tokens. Only valid tokens are needed and hence, invalid tokens are removed.

To build the lexer, we are inbuilt python function like yield, iter(), and next(). Also the inbuilt module like @dataclasses is used to create __repr__() function which gives output string of token in required form.

After lexer, the generated tokens are passed in the form of string to the parser.

- The Parser

The parser is the toughest and, important part in building the interpreter.

The **Bottom-Up Parser (BUP)** is used to generate parse tree. BUP is used since it is easily implemented as **Operator Precedence Parser (OPP).**

OPP is useful in prioritizing the operators like +, -, *, /, %, ^, etc.

The highest precedence is to parenthesis, then for exponent, then next to multiplication, division and modulo and last is to addition and subtraction.

So, the highest priority operator will be at leaf level and least priority operator will be at parent level. Hence, it is bottom-up approach.

- The Interpreter

When parse tree is generated, it is given to interpreter class which will traverse through each node of tree and evaluate it.

Since, tree itself is prioritized, the interpreter will work in correct way and calculate the value.

The value is calculated as the floating-point number. If the parse tree has invalid node, then interpreter will throw calculation error. The errors are handled at each level in the program.

- File Handling

The file handling is also important part since the output generated is stored in form of table in output file. To generate output table, the inbuilt python module prettytable is used which generate a readable tabular output.

The GUI is also used along with python terminal which gives more user friendly experience to user.

See the power point presentation for design and GUI part.

The files created in python are -

1. main.py - Main file to run in the python terminal.
2. gui.py - Creates GUI in python.
3. tokens.py - Creates the token class with token-type and value.
4. lexer.py - The lexer file which creates tokens
5. nodes.py - The nodes of parse tree are defined here.
6. _parse_.py - The parser.
   Note that python has inbuilt parser.py file.
   So, don't use this name.
7. Value.py - The value class of number-node is defined here.
8. Interpreter.py - The evaluation of nodes is done.
   Final value is calculated.
9. Input_file.txt - The input text file.
10. Output_file.txt - The output text file.

3. Power Point Presentation

Math Interpreter

Group 7

PROF. D. J. JOSHI

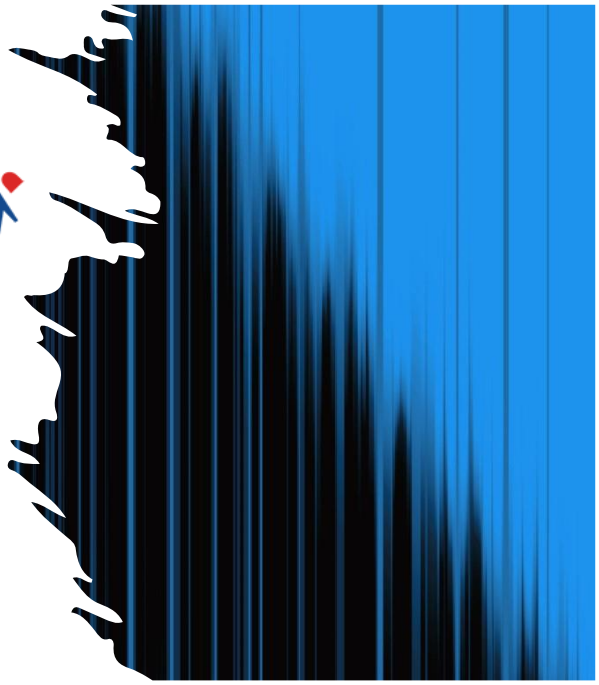
65. SAGAR SIKCHI 11810844

53. PRADUNYA MALADHARI 11811567

68. TALIB HUSSAIN 11811318

66. SHANTANU SONTAKKE 11810898

27. AADARSH KANDEWAR 11810942



Contents

- Implementation
- Tokens
- Parser
- Interpreter
- Input-output Form

Implementation

Language used – python

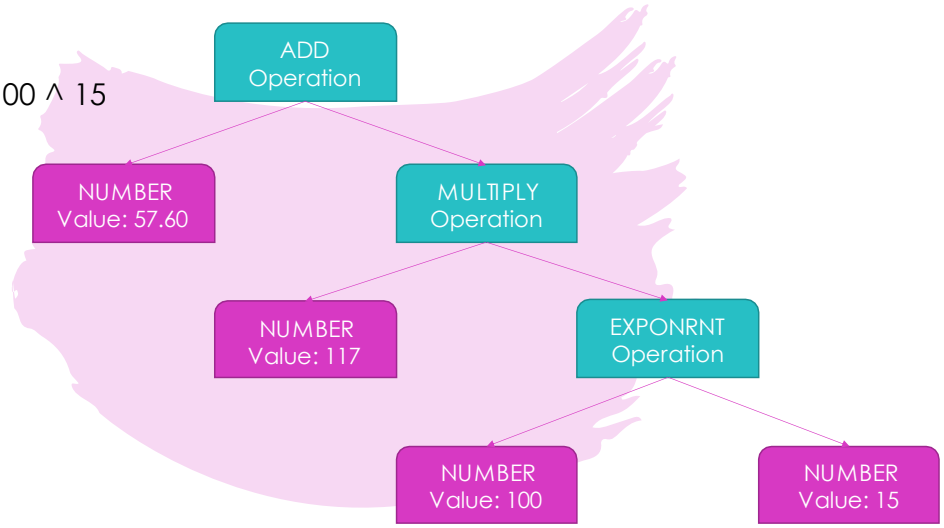
1. Lexer : generate tokens – working on character level
2. Parser : generate parse tree – working on token level
3. Interpreter : evaluate parse tree – working on node level
4. File handling

Tokens

57.60 + 117 * 100 ^ 15

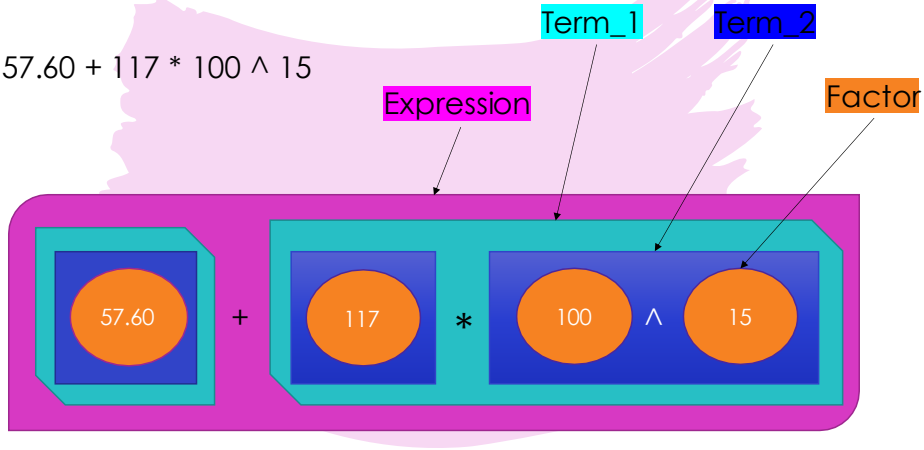
Parser

57.60 + 117 * 100 ^ 15



Grammar Rule

57.60 + 117 * 100 ^ 15



Interpreter

Input expression –

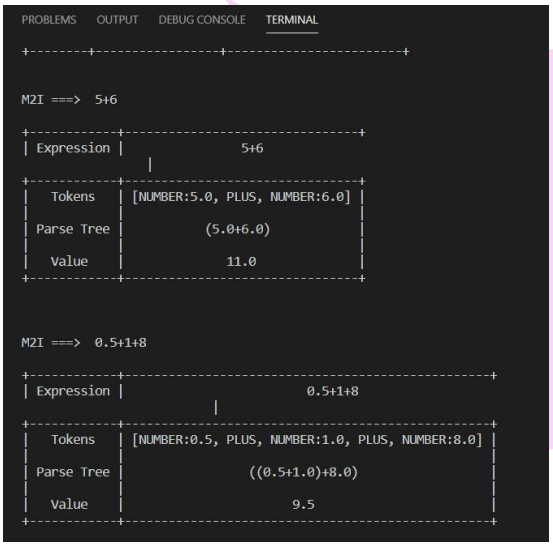
57.60 + 117 * 100 ^ 15

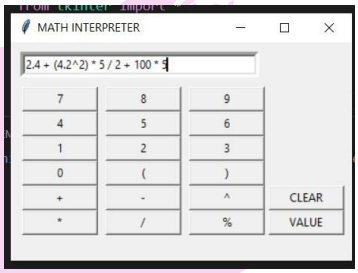
After parsing the expression becomes –

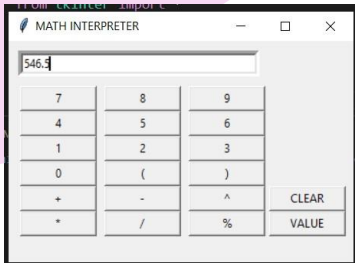
(57.60+(117.0*(100.0^15.0)))

From this, interpreter will able to understand which operation should be performed first.

Input-Output Form







4. References

Python iter() method - GeeksforGeeks

Python | yield Keyword - GeeksforGeeks

Computer Revolution (www.comrevo.com): What are the Different Types of Parsers

Data Classes in Python (dataclasses) (tutorialspoint.com)

(2037) Compiler Design (Complete Playlist) - YouTube