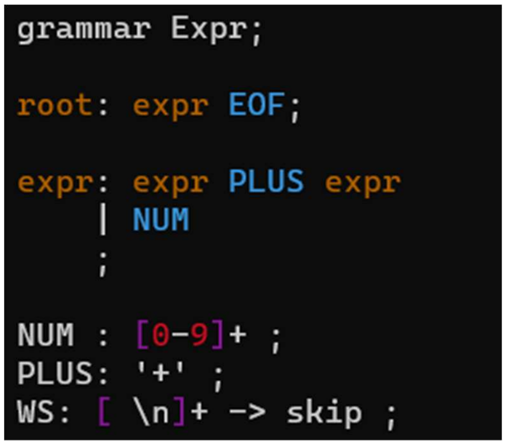
Grammar File (Expr.g)



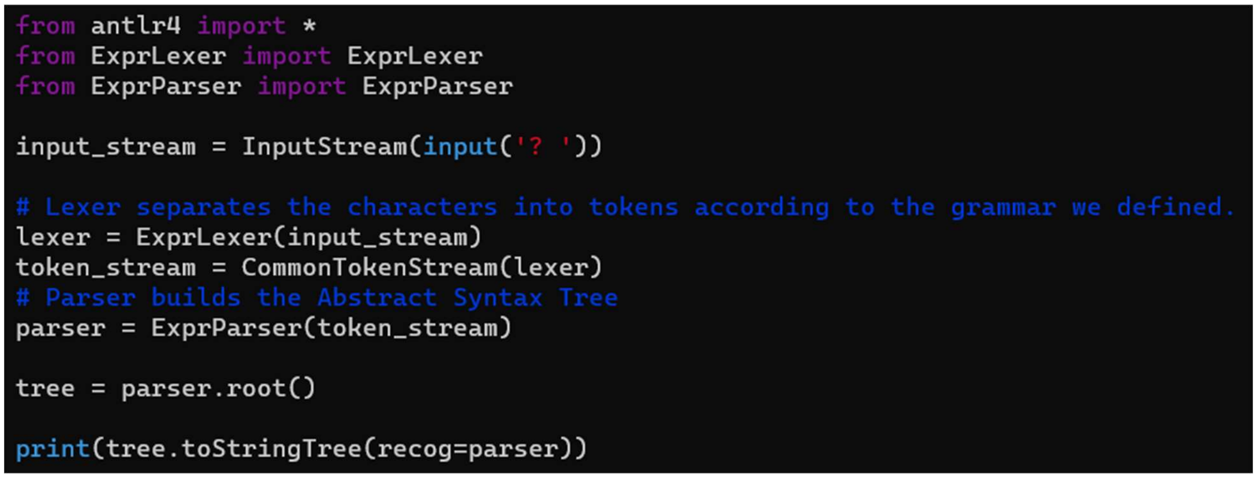
```
grammar Expr;

root: expr EOF;

expr: expr PLUS expr
    | NUM
    ;

NUM : [0-9]+ ;
PLUS: '+' ;
WS: [ \n]+ -> skip ;
```

- The filename MUST match with the name of the grammar.
- *expr*: definition of the grammar for the sum of natural numbers.
- *root*: processes the end file. In our example, this denotes that we only allow one expression.
- *skip*: tells the scanner that the WS token should not reach the parser for constructing the AST.

Compilation in Python 3

- Run the command line: `antlr4 -Dlanguage=Python3 -no-listener Expr.g`
- The output of the execution of the command above will generate
  - ExprLexer.py
  - ExprParser.py
  - ExprLexer.tokens
  - Expr.tokens

Test Script (test.py)



```python
from antlr4 import *
from ExprLexer import ExprLexer
from ExprParser import ExprParser

input_stream = InputStream(input('? '))

# Lexer separates the characters into tokens according to the grammar we defined.
lexer = ExprLexer(input_stream)
token_stream = CommonTokenStream(lexer)
# Parser builds the Abstract Syntax Tree
parser = ExprParser(token_stream)

tree = parser.root()

print(tree.toStringTree(recog=parser))
```