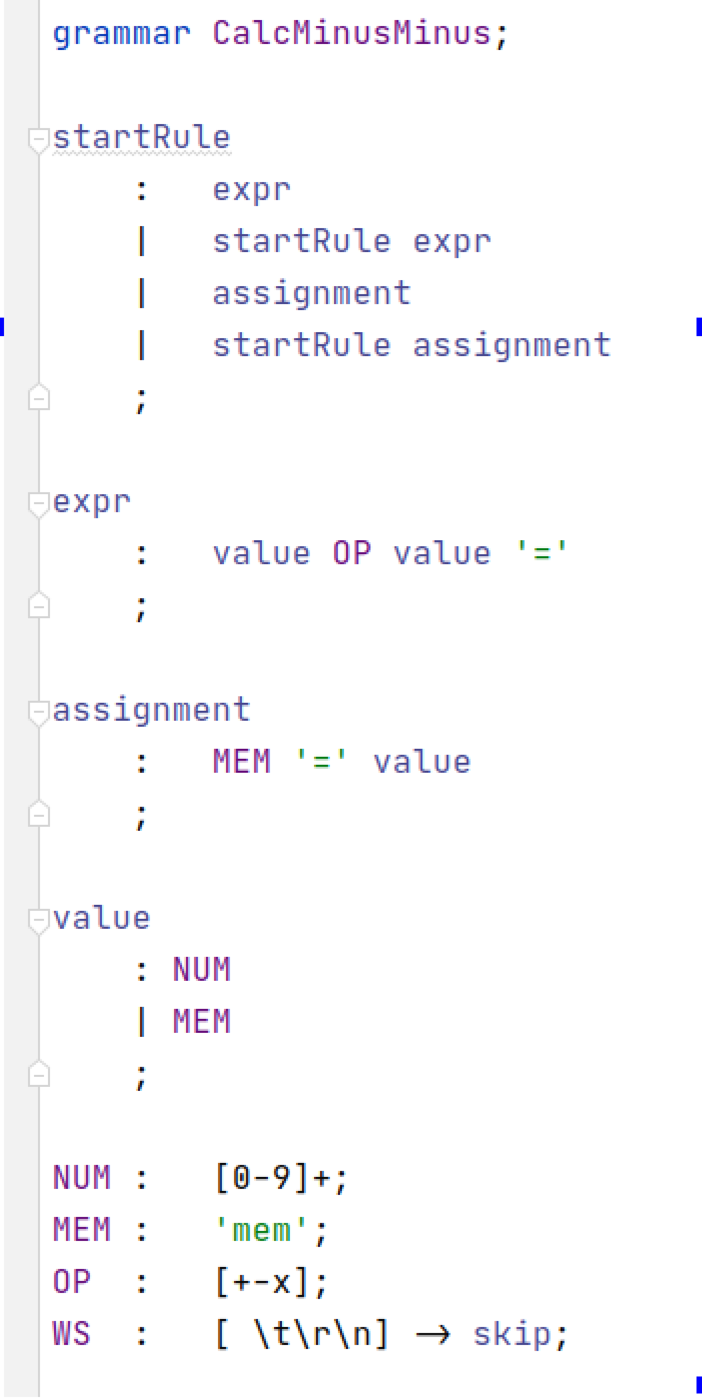
Antlr και Python

Διαλέξεις στο μάθημα: Μεταφραστές ΙΙ
Γιώργος Μανής

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

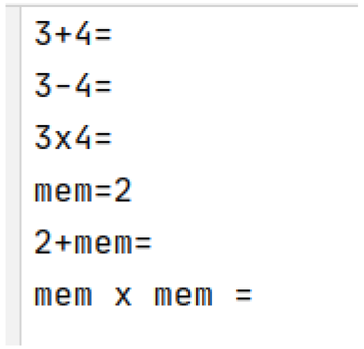
Γραμματική

- Ονομασία γραμματικής: CalcMinusMinus
- Κανόνες συντακτικής ανάλυσης: startRule, expr, assignment, value
- Λεκτικές μονάδες: NUM, MEM, OP, WS

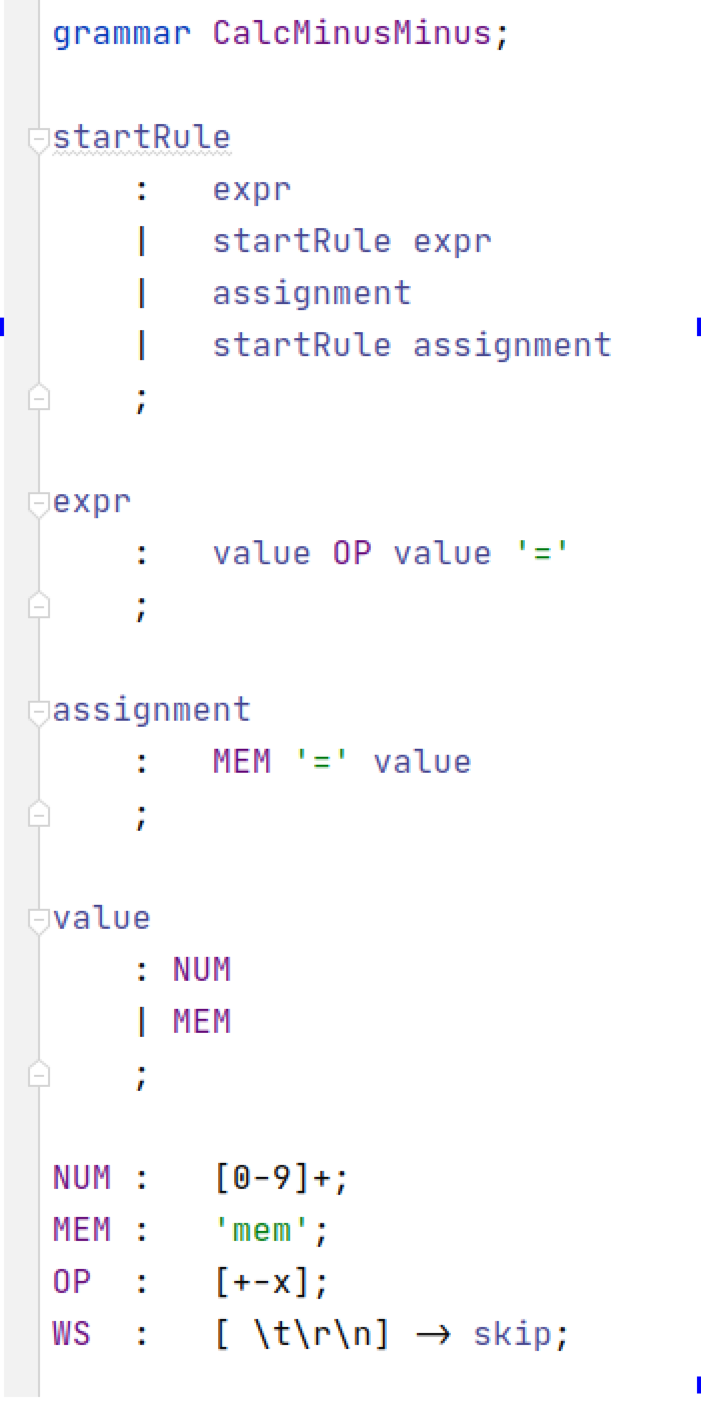


```
grammar CalcMinusMinus;

startRule
      :    expr
      |    startRule expr
      |    assignment
      |    startRule assignment
      ;

expr
      :    value OP value '='
      ;

assignment
      :    MEM '=' value
      ;

value
      : NUM
      | MEM
      ;

NUM :    [0-9]+;
MEM :    'mem';
OP  :    [+-x];
WS  :    [ \t\r\n] -> skip;
```

Γραμματική



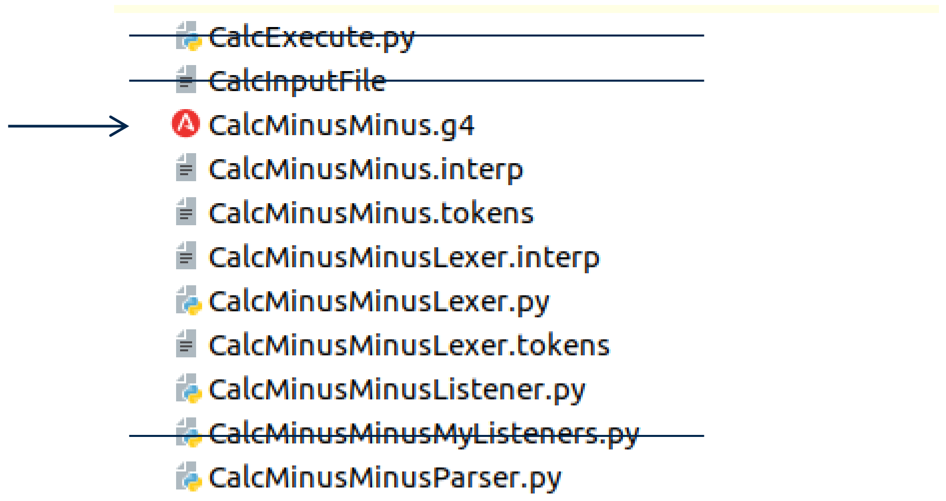
```
3+4=
3-4=
3x4=
mem=2
2+mem=
mem x mem =
```



```
grammar CalcMinusMinus;

startRule
        :   expr
        |   startRule expr
        |   assignment
        |   startRule assignment
        ;

expr
        :   value OP value '='
        ;

assignment
        :   MEM '=' value
        ;

value
        : NUM
        | MEM
        ;

NUM :   [0-9]+;
MEM :   'mem';
OP  :   [+-x];
WS  :   [ \t\r\n] -> skip;
```

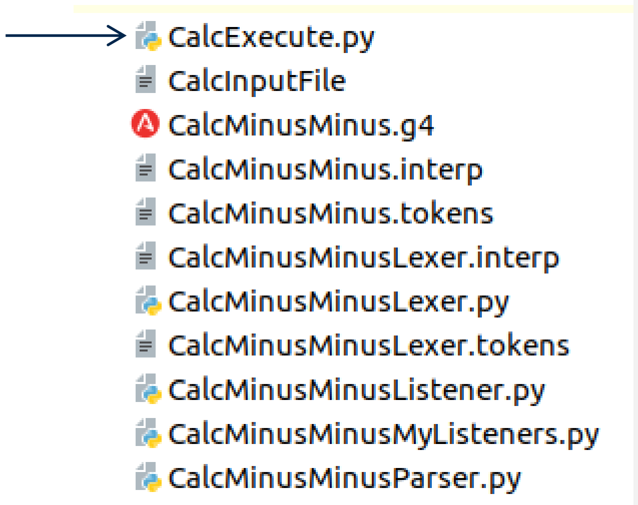
Compilation

- `antlr4 -Dlanguage=Python3 -listener CalcMinusMinus.g4`

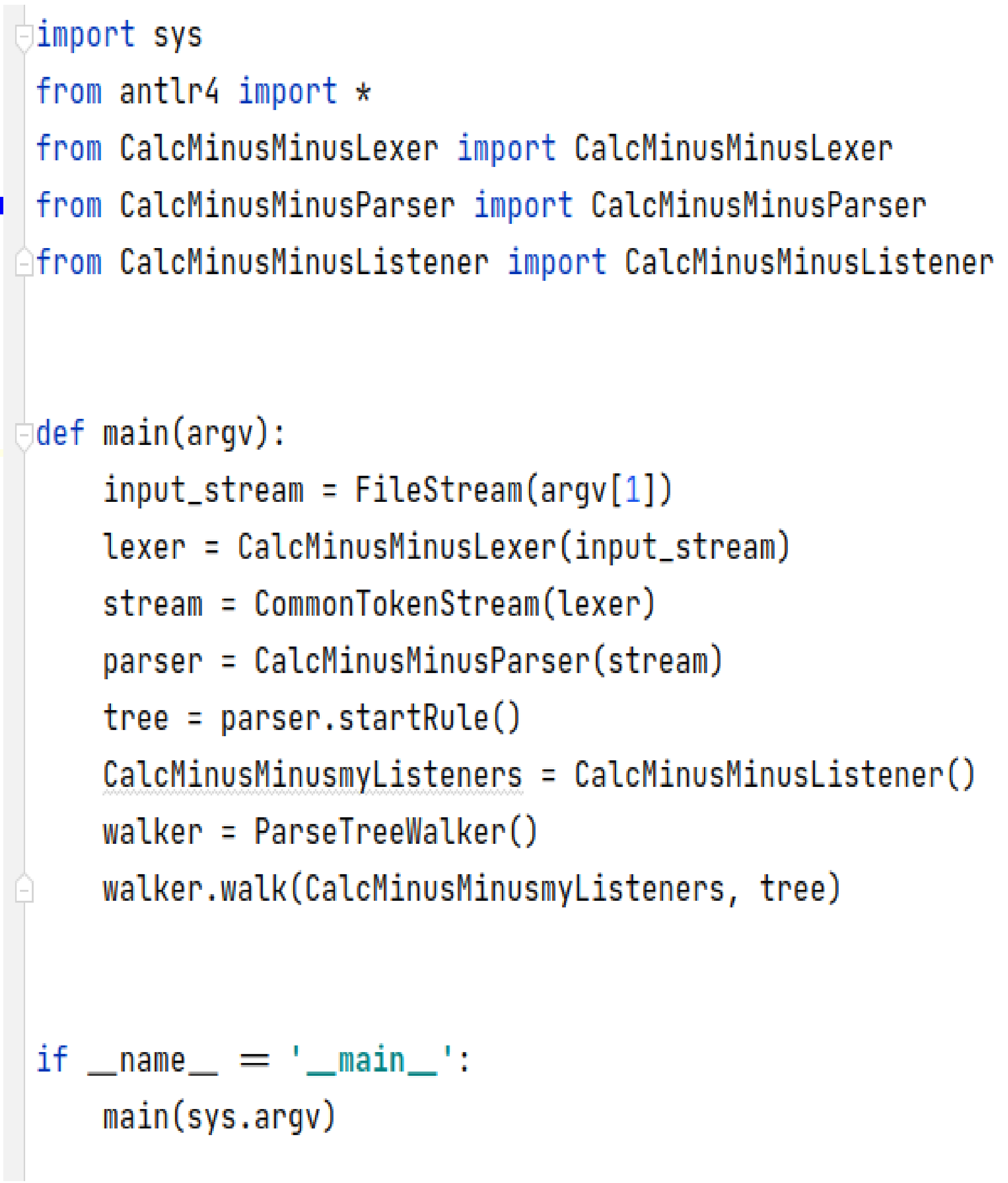


~~CalcExecute.py~~
~~CalcInputFile~~
$\longrightarrow$ CalcMinusMinus.g4
CalcMinusMinus.interp
CalcMinusMinus.tokens
CalcMinusMinusLexer.interp
CalcMinusMinusLexer.py
CalcMinusMinusLexer.tokens
CalcMinusMinusListener.py
~~CalcMinusMinusMyListeners.py~~
CalcMinusMinusParser.py

main.py

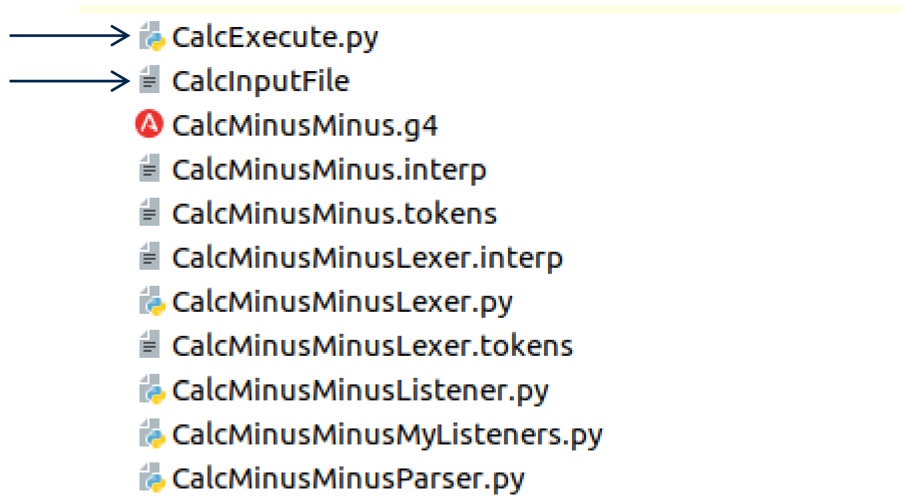


```
CalcExecute.py
CalcInputFile
CalcMinusMinus.g4
CalcMinusMinus.interp
CalcMinusMinus.tokens
CalcMinusMinusLexer.interp
CalcMinusMinusLexer.py
CalcMinusMinusLexer.tokens
CalcMinusMinusListener.py
CalcMinusMinusMyListeners.py
CalcMinusMinusParser.py
```



```python
import sys
from antlr4 import *
from CalcMinusMinusLexer import CalcMinusMinusLexer
from CalcMinusMinusParser import CalcMinusMinusParser
from CalcMinusMinusListener import CalcMinusMinusListener


def main(argv):
    input_stream = FileStream(argv[1])
    lexer = CalcMinusMinusLexer(input_stream)
    stream = CommonTokenStream(lexer)
    parser = CalcMinusMinusParser(stream)
    tree = parser.startRule()
    CalcMinusMinusmyListeners = CalcMinusMinusListener()
    walker = ParseTreeWalker()
    walker.walk(CalcMinusMinusmyListeners, tree)


if __name__ == '__main__':
    main(sys.argv)
```

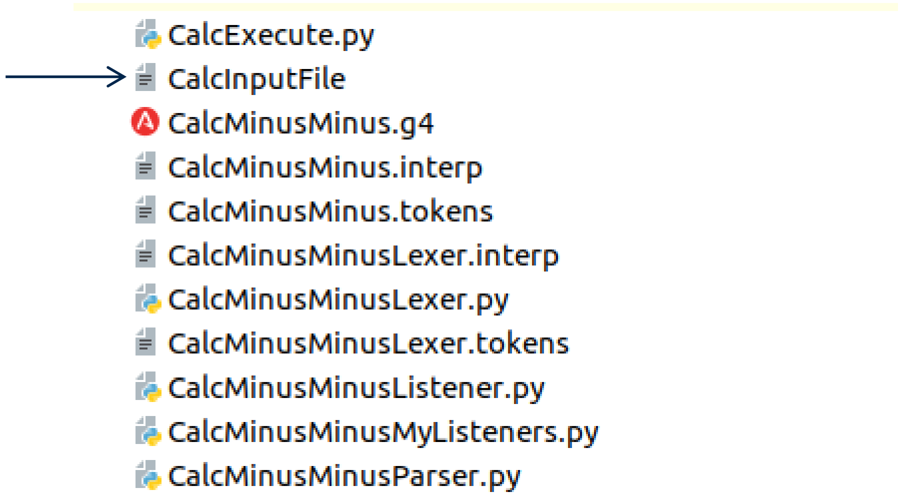
main.py



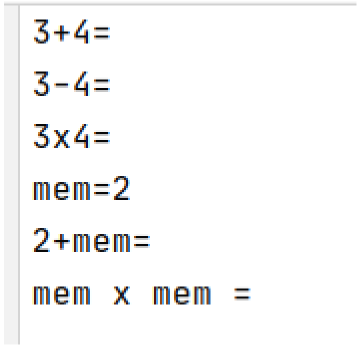
```
CalcExecute.py
CalcInputFile
CalcMinusMinus.g4
CalcMinusMinus.interp
CalcMinusMinus.tokens
CalcMinusMinusLexer.interp
CalcMinusMinusLexer.py
CalcMinusMinusLexer.tokens
CalcMinusMinusListener.py
CalcMinusMinusMyListeners.py
CalcMinusMinusParser.py
```

- `python CalcExecute CalcInputFile`

input file



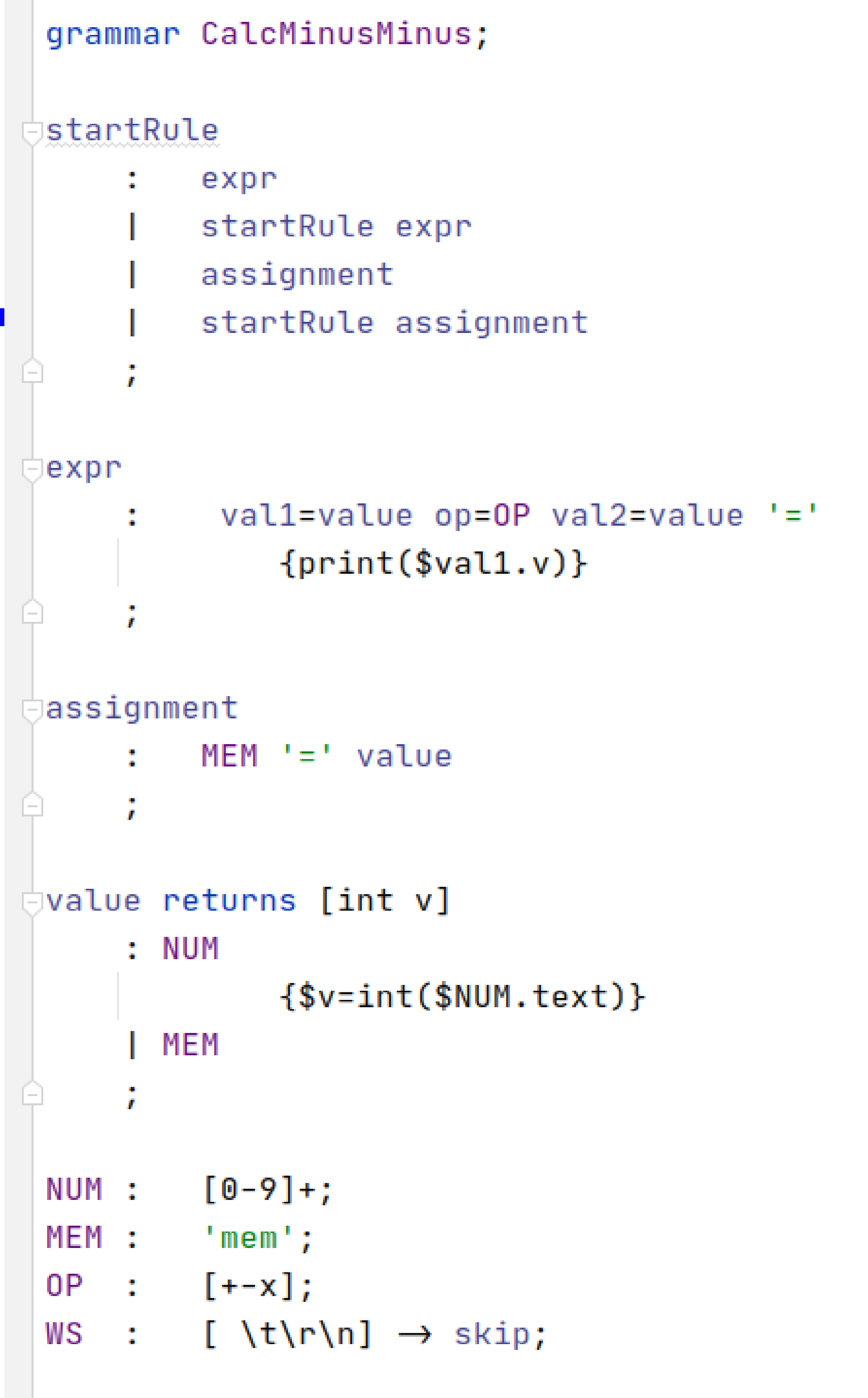
```
CalcExecute.py
CalcInputFile
CalcMinusMinus.g4
CalcMinusMinus.interp
CalcMinusMinus.tokens
CalcMinusMinusLexer.interp
CalcMinusMinusLexer.py
CalcMinusMinusLexer.tokens
CalcMinusMinusListener.py
CalcMinusMinusMyListeners.py
CalcMinusMinusParser.py
```



```
3+4=
3-4=
3x4=
mem=2
2+mem=
mem x mem =
```

actions

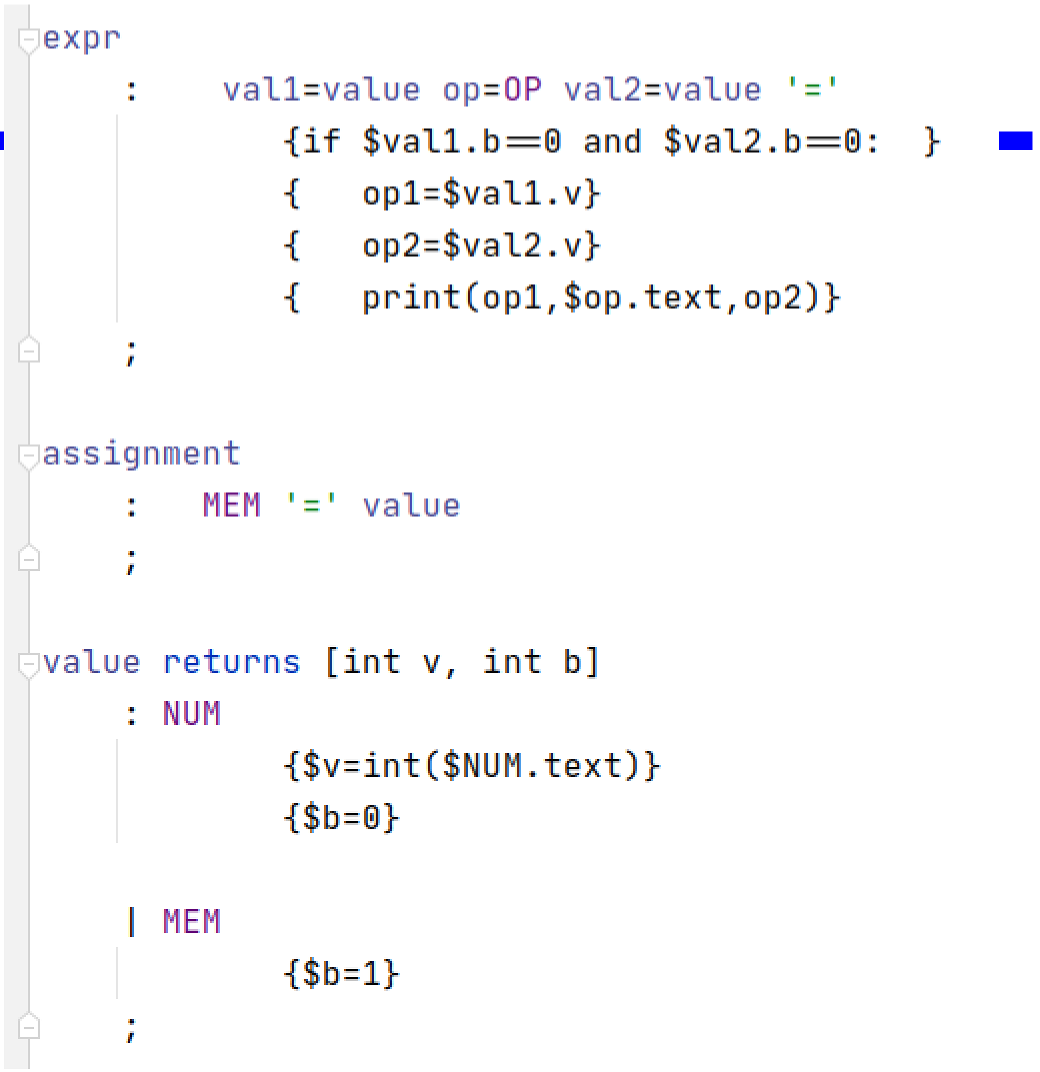
- python κώδικας μέσα σε αγκύλες
- attributes συμβολισμένα με $
- επιστροφή τιμών καθορισμένου τύπου



```
grammar CalcMinusMinus;

startRule
    :   expr
    |   startRule expr
    |   assignment
    |   startRule assignment
    ;

expr
    :    val1=value op=OP val2=value '='
            {print($val1.v)}
    ;

assignment
    :   MEM '=' value
    ;

value returns [int v]
    : NUM
            {$v=int($NUM.text)}
    | MEM
    ;

NUM :   [0-9]+;
MEM :   'mem';
OP  :   [+-x];
WS  :   [ \t\r\n] -> skip;
```

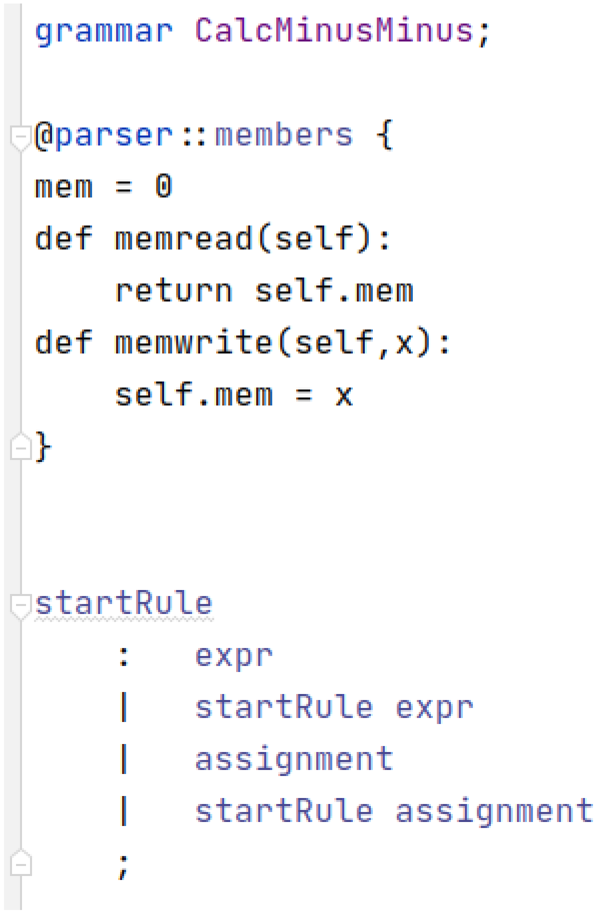
actions

- επιστροφή περισσότερων της μίας τιμής και διαφορετικού τύπου
- στην python η στοίχιση παίζει ρόλο
- Υπάρχει τρόπος να αποφύγουμε τις πολλές αγκύλες

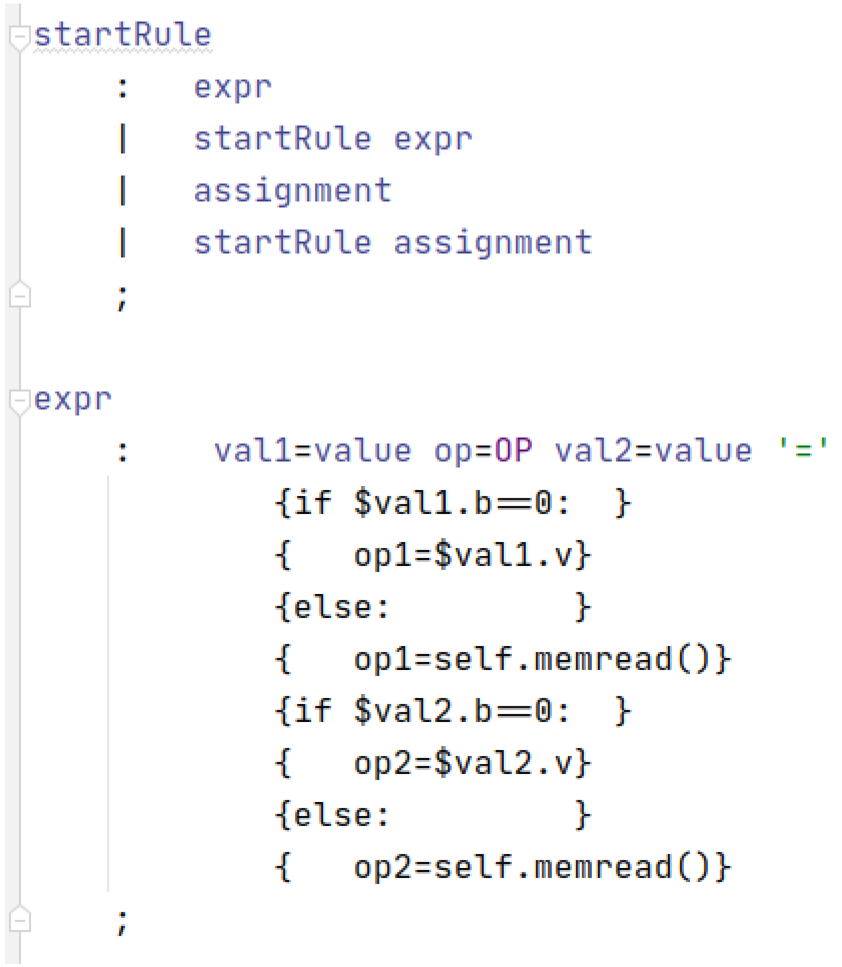


```
expr
    :     val1=value op=OP val2=value '='
              {if $val1.b==0 and $val2.b==0:  }
              {    op1=$val1.v}
              {    op2=$val2.v}
              {    print(op1,$op.text,op2)}
    ;

assignment
    :   MEM '=' value
    ;

value returns [int v, int b]
    : NUM
              {$v=int($NUM.text)}
              {$b=0}

    | MEM
              {$b=1}
    ;
```

members



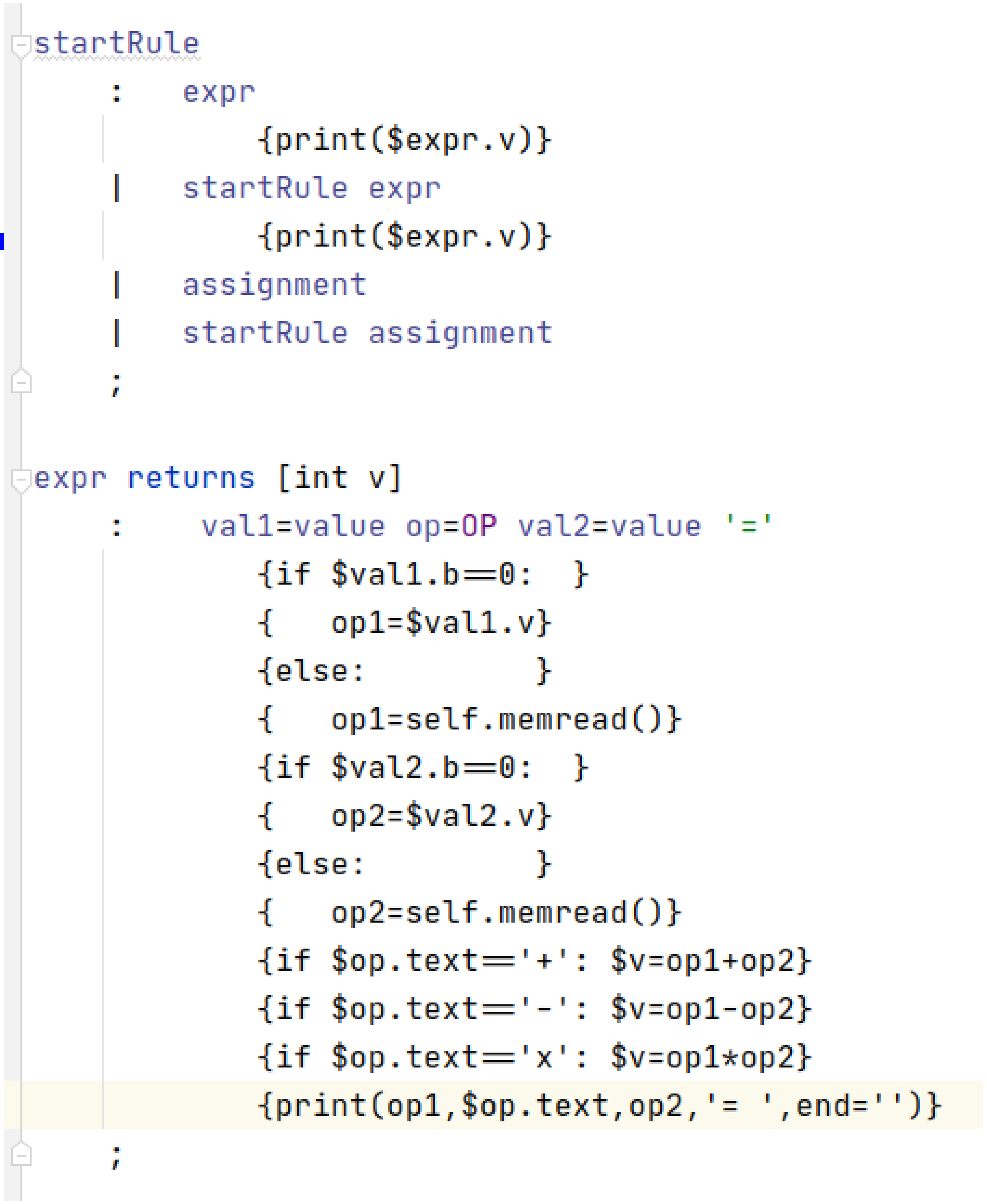
```
grammar CalcMinusMinus;

@parser::members {
mem = 0
def memread(self):
    return self.mem
def memwrite(self,x):
    self.mem = x
}



startRule
    :   expr
    |   startRule expr
    |   assignment
    |   startRule assignment
    ;
```



```
startRule
    :   expr
    |   startRule expr
    |   assignment
    |   startRule assignment
    ;


expr
    :    val1=value op=OP val2=value '='
            {if $val1.b==0:  }
            {   op1=$val1.v}
            {else:          }
            {   op1=self.memread()}
            {if $val2.b==0:  }
            {   op2=$val2.v}
            {else:          }
            {   op2=self.memread()}
    ;
```

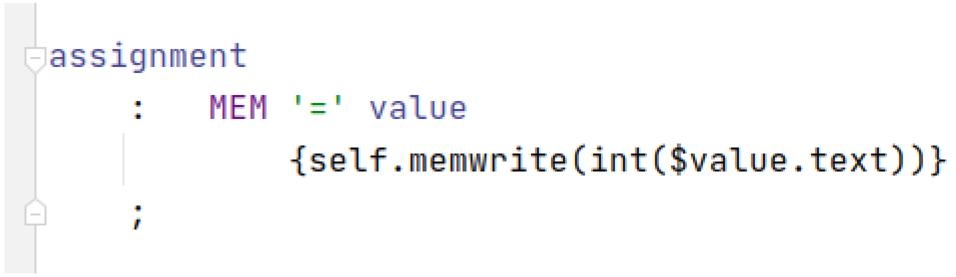
add functionality

- τέλεση πράξεων



```
startRule
        :   expr
                {print($expr.v)}
        |   startRule expr
                {print($expr.v)}
        |   assignment
        |   startRule assignment
        ;

expr returns [int v]
        :    val1=value op=OP val2=value '='
                {if $val1.b==0:  }
                {   op1=$val1.v}
                {else:         }
                {   op1=self.memread()}
                {if $val2.b==0:  }
                {   op2=$val2.v}
                {else:         }
                {   op2=self.memread()}
                {if $op.text=='+': $v=op1+op2}
                {if $op.text=='-': $v=op1-op2}
                {if $op.text=='x': $v=op1*op2}
                {print(op1,$op.text,op2,'= ',end='')}
        ;
```

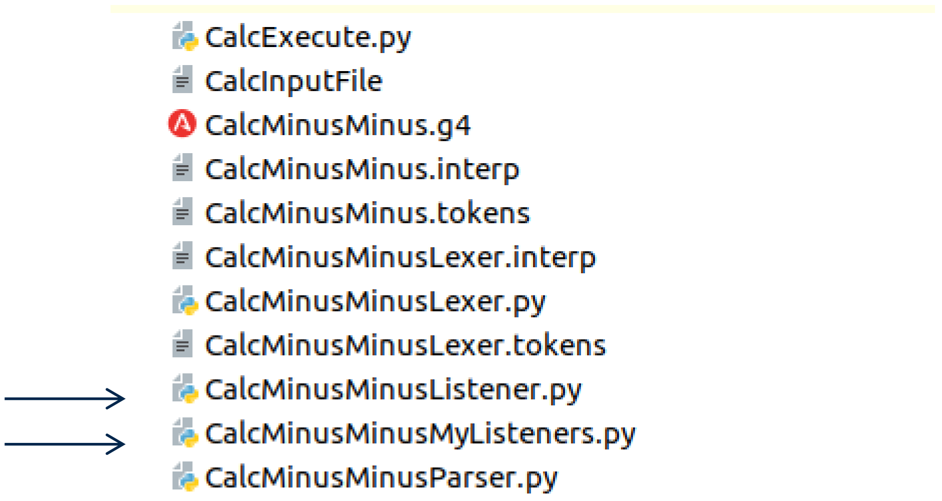
write to memory



```
assignment
    :   MEM '=' value
            {self.memwrite(int($value.text))}
    ;
```

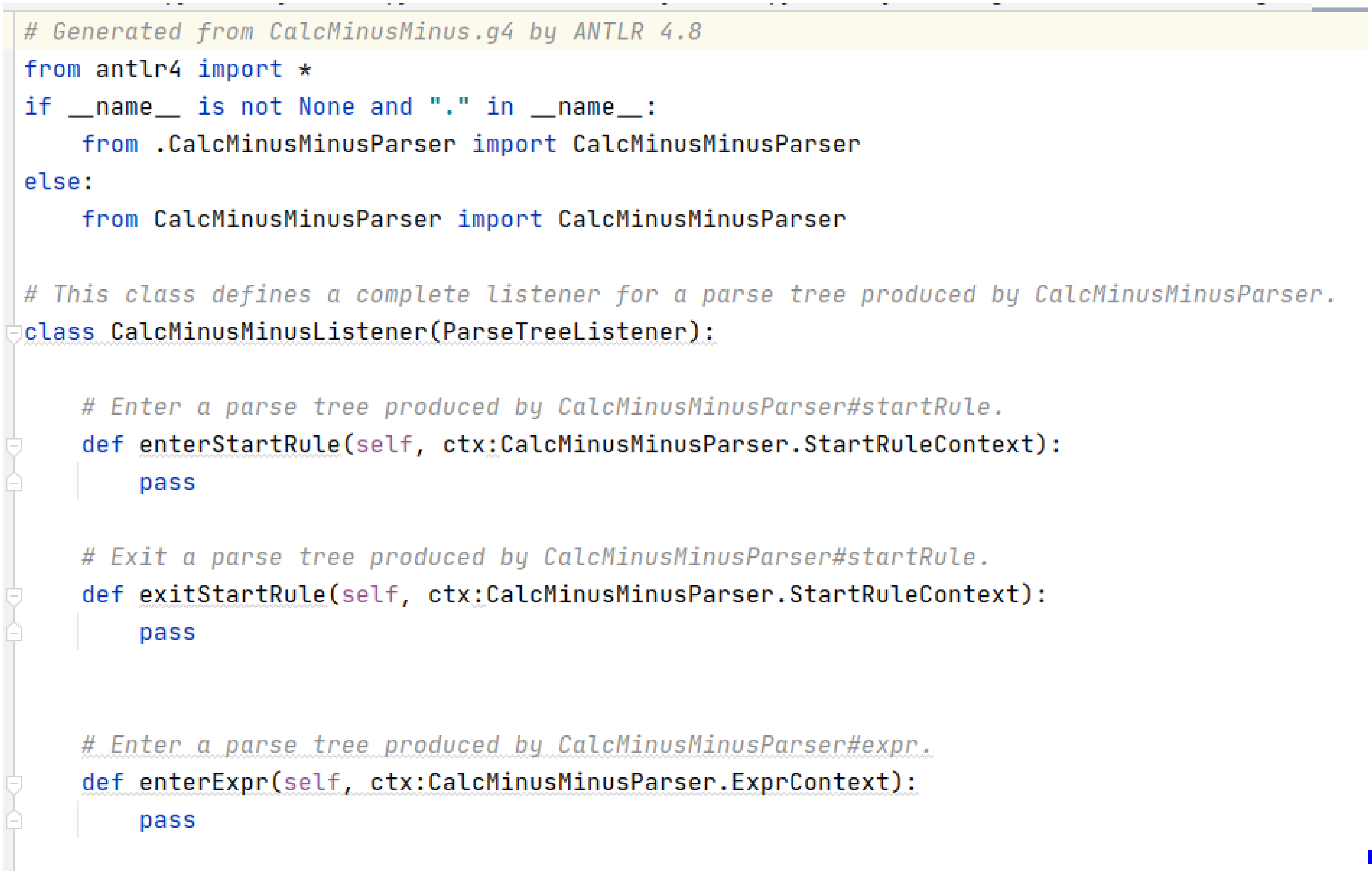
Listeners

- `antlr4 -Dlanguage=Python3 -listener CalcMinusMinus.g4`



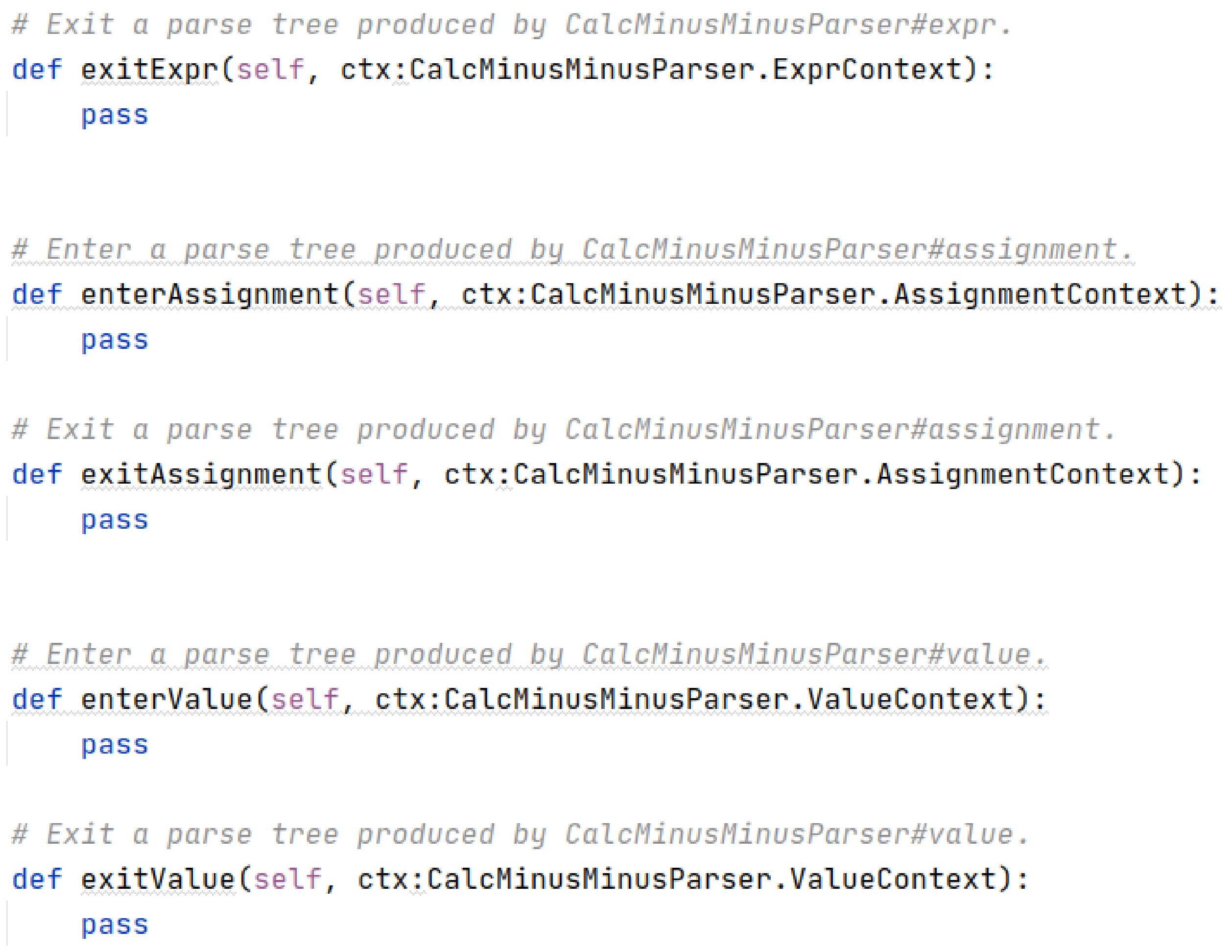
CalcExecute.py
CalcInputFile
CalcMinusMinus.g4
CalcMinusMinus.interp
CalcMinusMinus.tokens
CalcMinusMinusLexer.interp
CalcMinusMinusLexer.py
CalcMinusMinusLexer.tokens
CalcMinusMinusListener.py
CalcMinusMinusMyListeners.py
CalcMinusMinusParser.py

Listener



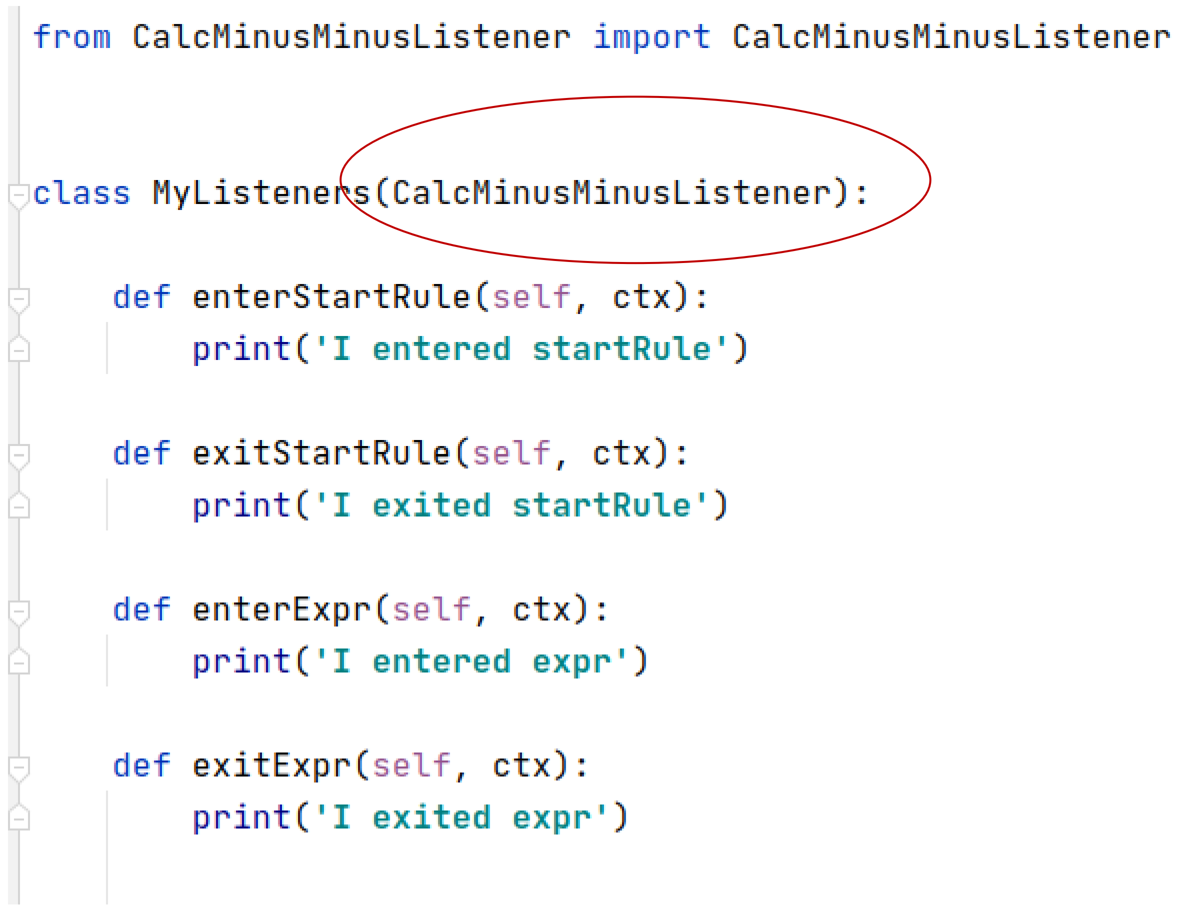
```
# Generated from CalcMinusMinus.g4 by ANTLR 4.8
from antlr4 import *
if __name__ is not None and "." in __name__:
    from .CalcMinusMinusParser import CalcMinusMinusParser
else:
    from CalcMinusMinusParser import CalcMinusMinusParser

# This class defines a complete listener for a parse tree produced by CalcMinusMinusParser.
class CalcMinusMinusListener(ParseTreeListener):

    # Enter a parse tree produced by CalcMinusMinusParser#startRule.
    def enterStartRule(self, ctx:CalcMinusMinusParser.StartRuleContext):
        pass

    # Exit a parse tree produced by CalcMinusMinusParser#startRule.
    def exitStartRule(self, ctx:CalcMinusMinusParser.StartRuleContext):
        pass


    # Enter a parse tree produced by CalcMinusMinusParser#expr.
    def enterExpr(self, ctx:CalcMinusMinusParser.ExprContext):
        pass
```

Listener



```python
# Exit a parse tree produced by CalcMinusMinusParser#expr.
def exitExpr(self, ctx:CalcMinusMinusParser.ExprContext):
    pass


# Enter a parse tree produced by CalcMinusMinusParser#assignment.
def enterAssignment(self, ctx:CalcMinusMinusParser.AssignmentContext):
    pass

# Exit a parse tree produced by CalcMinusMinusParser#assignment.
def exitAssignment(self, ctx:CalcMinusMinusParser.AssignmentContext):
    pass


# Enter a parse tree produced by CalcMinusMinusParser#value.
def enterValue(self, ctx:CalcMinusMinusParser.ValueContext):
    pass

# Exit a parse tree produced by CalcMinusMinusParser#value.
def exitValue(self, ctx:CalcMinusMinusParser.ValueContext):
    pass
```

MyListeners



```
from CalcMinusMinusListener import CalcMinusMinusListener


class MyListeners(CalcMinusMinusListener):

    def enterStartRule(self, ctx):
        print('I entered startRule')

    def exitStartRule(self, ctx):
        print('I exited startRule')

    def enterExpr(self, ctx):
        print('I entered expr')

    def exitExpr(self, ctx):
        print('I exited expr')
```

ευχαριστώ !!!