



**UNIVERSITY OF NAIROBI**

**FACULTY OF SCIENCE AND TECHNOLOGY**

**DEPARTMENT OF COMPUTING AND INFORMATICS**

**CSC 326: COMPILER CONSTRUCTION**

## **PARSER FOR OUR MINI LANGUAGE USING PLY TOOL**

Done by Group 17:

P15/137032/2019

P15/1894/2020

P15/1915/2020

Hamud Abdulrahman Abeid

Emmanuel Mbithi Muthiani

Mugo Staicy Nelima Muthoni

## SOURCE CODE FOR THE PARSER IN PYTHON

```
import ply.yacc as yacc

# Get the token list from the lexer

from lexer3 import tokens

# Define the grammar rules

def p_statement_assign(p):

    'statement : ID ASSIGN expression EOL'

    p[0] = ('ASSIGN', p[3], "to", p[1])

def p_expression_binop(p):

    '''expression : expression PLUS expression
                  | expression MINUS expression
                  | expression MULT expression
                  | expression DIV expression
                  | expression LT expression
                  | expression GT expression
                  | expression LE expression
                  | expression GE expression
                  | expression EQ expression
                  | expression NE expression'''

    p[0] = (p[1], p[2], p[3])

def p_expression_group(p):

    'expression : LPAREN expression RPAREN'

    p[0] = p[2]

def p_expression_number(p):

    '''expression : INT
                  | FLOAT
```

```

        | ID'''

p[0] = p[1]

def p_statement_if(p):
    'statement : IF expression LBRACE statements RBRACE'
    p[0] = ('IF', p[2], p[4])

def p_statement_else(p):
    'statement : IF expression LBRACE statements RBRACE ELSE LBRACE statements
    RBRACE'
    p[0] = ('ELSE', p[2], p[4], p[8])

def p_statement_while(p):
    'statement : WHILE expression LBRACE statements RBRACE'
    p[0] = ('WHILE', p[2], p[4])

def p_statement_print(p):
    'statement : PRINT expression EOL'
    p[0] = ('PRINT', p[2], p[3])

def p_statements(p):
    '''statements : statement
        | statements statement'''
    if len(p) == 2:
        p[0] = [p[1]]
    else:
        p[1].append(p[2])
        p[0] = p[1]

def p_error(p):
    print(f"SyntaxError: Incorrect syntax at line {p.lineno-1}")

```

```
def p_empty(p):  
    'empty :'  
    pass  
  
# Build parser  
parser = yacc.yacc()  
  
# Test parser with sample input  
data = '''  
    if (3+2) {  
        print a ;  
        a = 2 ;  
    }  
'''  
  
# Parse the input  
result = parser.parse(data)  
  
# Print the result  
print(result)
```

## TOKEN LIST PROVIDED BY OUR SCANNER

```
keywords = {  
    'if'      : 'IF',  
    'else'    : 'ELSE',  
    'while'   : 'WHILE',  
    'print'   : 'PRINT'  
}  
  
# Define tokens  
tokens = [  
    'ASSIGN',  
    'PLUS',  
    'MINUS',  
    'MULT',  
    'DIV',  
    'LPAREN',  
    'RPAREN',  
    'LBRACE',  
    'RBRACE',  
    'ID',  
    'INT',  
    'FLOAT',  
    'EOL',  
    'LT',  
    'GT',  
    'LE',  
    'GE',  
    'EQ',  
    'NE'  
] + list(keywords.values())
```

## SCREENSHOT SHOWING PARSER OUTPUT AND RUNTIME

```
C:\Users\Abdulrahman Hamud\pythonPrograms\parser3.py (Data Entry Automation) - Sublime Text (UNREGI
File Edit Selection Find View Goto Tools Project Preferences Help
lexer3.py x parser3.py x parser4.py x
57     p[0] = p[1]
58
59 def p_error(p):
60     print(f"SyntaxError: Incorrect syntax at line {p.lineno-1}")
61
62 def p_empty(p):
63     'empty :'
64     pass
65
66 # Build parser
67 parser = yacc.yacc()
68
69 # Test parser with sample input
70 data = '''
71     if (3+2) {
72         print a ;
73         a = 2 ;
74     }
75 '''
76
77 # Parse the input
78 result = parser.parse(data)
79
80 # Print the result
81 print(result)
82
83
('IF', ('3', '+', '2'), [('PRINT', 'a', ';'), ('ASSIGN', '2', 'to', 'a')])
[Finished in 172ms]
```