



UNIVERSITY OF NAIROBI

FACULTY OF SCIENCE AND TECHNOLOGY

DEPARTMENT OF COMPUTING AND INFORMATICS

CSC 326: COMPILER CONSTRUCTION

SCANNER 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 SCANNER IN PYTHON

```
import ply.lex as lex
```

```
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())

# Define regular expression rules for tokens

t_ASSIGN = r'='

t_PLUS = r'\+'

t_MINUS = r'\-'

t_MULT = r'\*'

t_DIV = r'\/'

t_LPAREN = r'\('

t_RPAREN = r'\)'

t_LBRACE = r'\{'

t_RBRACE = r'\}'

t_FLOAT = r'\d+\.\d+'

t_INT = r'\d+'

t_EOL = r';'

t_LT = r'<'

t_GT = r'>'

t_LE = r'<='

t_GE = r'>='

t_EQ = r'=='

t_NE = r'!='

# Define rule to check for keywords

def t_ID(t):

    r'[a-zA-Z_][a-zA-Z0-9_]*'

```

```

        t.type = keywords.get(t.value, 'ID')

    return t

# Define ignored characters (whitespace)
t_ignore = ' \t'

# Define new line tracking rule
def t_newline(t):
    r'\n+'

    t.lexer.lineno += len(t.value)

# Define error handling rule
def t_error(t):
    print(f"SyntaxError: Invalid token '{t.value[0]}' at line {t.lineno}")
    t.lexer.skip(1)

# Build lexer
lexer = lex.lex()

# Test lexer with sample input
data = '''x = 42.4;

y = 78;

a = 0.98\n

while: (x > 0) {

    print(x)[[];

    x = x - 1;

}

```

```
...
```

```
# Run the lexer
```

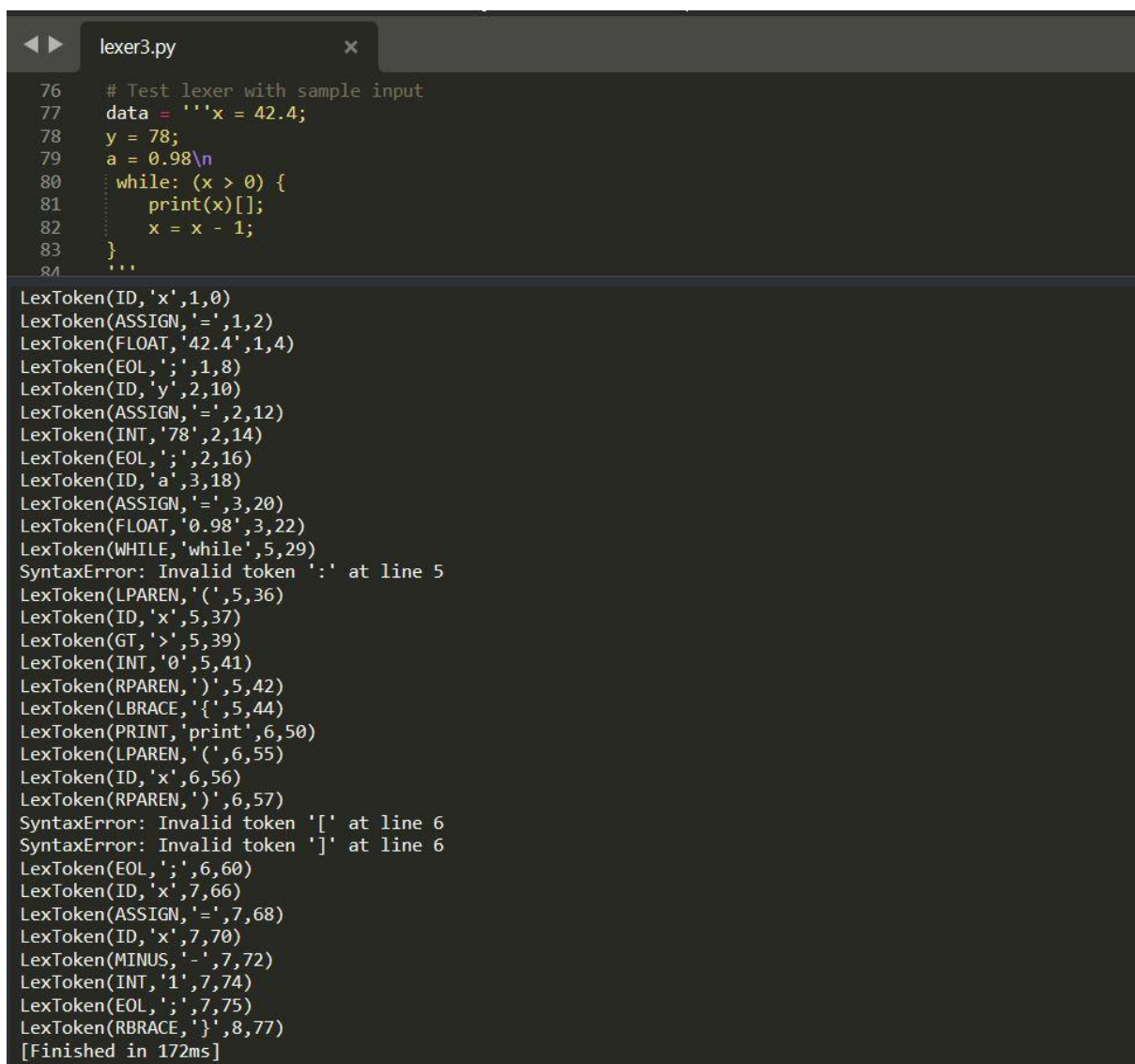
```
lexer.input(data)
```

```
# Tokenize
```

```
for token in lexer:
```

```
    print(token)
```

SCREENSHOT SHOWING TOKENS AND SCANNERS RUN-TIME



```
lexer3.py
76 # Test lexer with sample input
77 data = '''x = 42.4;
78 y = 78;
79 a = 0.98\n
80 while: (x > 0) {
81     print(x>[];
82     x = x - 1;
83 }
84 '''
LexToken(ID, 'x', 1, 0)
LexToken(ASSIGN, '=', 1, 2)
LexToken(FLOAT, '42.4', 1, 4)
LexToken(EOL, ';', 1, 8)
LexToken(ID, 'y', 2, 10)
LexToken(ASSIGN, '=', 2, 12)
LexToken(INT, '78', 2, 14)
LexToken(EOL, ';', 2, 16)
LexToken(ID, 'a', 3, 18)
LexToken(ASSIGN, '=', 3, 20)
LexToken(FLOAT, '0.98', 3, 22)
LexToken(WHILE, 'while', 5, 29)
SyntaxError: Invalid token ':' at line 5
LexToken(LPAREN, '(', 5, 36)
LexToken(ID, 'x', 5, 37)
LexToken(GT, '>', 5, 39)
LexToken(INT, '0', 5, 41)
LexToken(RPAREN, ')', 5, 42)
LexToken(LBRACE, '{', 5, 44)
LexToken(PRINT, 'print', 6, 50)
LexToken(LPAREN, '(', 6, 55)
LexToken(ID, 'x', 6, 56)
LexToken(RPAREN, ')', 6, 57)
SyntaxError: Invalid token '[' at line 6
SyntaxError: Invalid token ']' at line 6
LexToken(EOL, ';', 6, 60)
LexToken(ID, 'x', 7, 66)
LexToken(ASSIGN, '=', 7, 68)
LexToken(ID, 'x', 7, 70)
LexToken(MINUS, '-', 7, 72)
LexToken(INT, '1', 7, 74)
LexToken(EOL, ';', 7, 75)
LexToken(RBRACE, '}', 8, 77)
[Finished in 172ms]
```