Mini Compiler - Python Code

```
# MINI COMPILER - Compiler Construction

TOKENS = {
    '=': 'ASSIGN',
    '+': 'PLUS',
    '-': 'MINUS',
    '*': 'MULT',
    '/': 'DIV'
}

KEYWORDS = ['print']

def lexical_analyzer(code):
    tokens = []
    i = 0
    while i < len(code):
        if code[i].isspace():
            i += 1
            continue
        if code[i].isalpha():
            word = ''
            while i < len(code) and code[i].isalnum():
                word += code[i]
                i += 1
            if word in KEYWORDS:
                tokens.append(('KEYWORD', word))
            else:
                tokens.append(('IDENTIFIER', word))
        elif code[i].isdigit():
            num = ''
            while i < len(code) and code[i].isdigit():
                num += code[i]
                i += 1
            tokens.append(('NUMBER', int(num)))
        elif code[i] in TOKENS:
            tokens.append((TOKENS[code[i]], code[i]))
            i += 1
        else:
            raise Exception("Lexical Error")
    return tokens

def execute(tokens):
    variables = {}
    i = 0
    while i < len(tokens):
        if tokens[i][0] == 'KEYWORD':
            print(variables[tokens[i+1][1]])
            i += 2
        else:
            var = tokens[i][1]
            val = evaluate(tokens[i+2:i+5], variables)
            variables[var] = val
            i += 5

def evaluate(exp, variables):
    left = exp[0]
    op = exp[1][0]
    right = exp[2]
    l = left[1] if left[0]=='NUMBER' else variables[left[1]]
    r = right[1] if right[0]=='NUMBER' else variables[right[1]]
    if op=='PLUS': return l+r
    if op=='MINUS': return l-r
    if op=='MULT': return l*r
    if op=='DIV': return l//r

code = '''
a = 5 + 3
b = a * 2
print b
'''
tokens = lexical_analyzer(code)
execute(tokens)
```