```
LETTER = 0

DIGIT = 1

UNKNOWN = 99

EOF = -1


INT_LIT = 10

IDENT = 11

ASSIGN_OP = 20

ADD_OP = 21

SUB_OP = 22

MULT_OP = 23

DIV_OP = 24

LEFT_PAREN = 25

RIGHT_PAREN = 26


class Lexer:

  def __init__(self, input_string):

    self.input = input_string

    self.index = 0

    self.char_class = None

    self.lexeme = ''

    self.next_char = ''

    self.token = None

    self.next_token = None

    self.get_char()
```

```
def add_char(self):
    self.lexeme += self.next_char

def get_char(self):
    if self.index < len(self.input):
        self.next_char = self.input[self.index]
        if self.next_char.isalpha():
            self.char_class = LETTER
        elif self.next_char.isdigit():
            self.char_class = DIGIT
        else:
            self.char_class = UNKNOWN
        self.index += 1
    else:
        self.char_class = EOF
        self.next_char = ''

def get_non_blank(self):
    while self.next_char.isspace():
        self.get_char()

def lookup(self, ch):
    symbol_tokens = {
        '(': LEFT_PAREN,
        ')': RIGHT_PAREN,
        '+': ADD_OP,
        '-': SUB_OP,
        '*': MULT_OP,
        '/': DIV_OP,
        '=': ASSIGN_OP
    }
    self.add_char()
    self.next_token = symbol_tokens.get(ch, EOF)

def lex(self):
    self.lexeme = ''
    self.get_non_blank()
    if self.char_class == LETTER:
        self.add_char()
        self.get_char()
        while self.char_class in [LETTER, DIGIT]:
            self.add_char()
            self.get_char()
        self.next_token = IDENT
    elif self.char_class == DIGIT:
        self.add_char()
        self.get_char()
        while self.char_class == DIGIT:
            self.add_char()
            self.get_char()
        self.next_token = INT_LIT
    elif self.char_class == UNKNOWN:
            self.lookup(self.next_char)
            self.get_char()
        elif self.char_class == EOF:
            self.next_token = EOF
            self.lexeme = 'EOF'
        print(f"Next token is: {self.next_token}, Next lexeme is '{self.lexeme}'")
        return self.next_token


if __name__ == "__main__":
    input_expr = input(">>> ")
    lexer = Lexer(input_expr)
    while lexer.next_token != EOF:
        lexer.lex()
```