



**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIA DO  
CEARÁ**

**IFCE CAMPUS ARACATI  
CIÊNCIAS DA COMPUTAÇÃO**

**MARCOS VINICIUS BARBOSA DOS SANTOS**

**TRABALHO DE COMPILADORES: ANALISADOR LÉXICO E SINTÁNTICO**

**ARACATI-CE**

**18/02/2025**

# Relatório do Analisador Léxico e Sintático

## 1. O que foi feito no trabalho

Foi desenvolvido um analisador léxico e sintático utilizando a biblioteca PLY (Python Lex-Yacc). O trabalho foi dividido em duas etapas principais:

- Analisador Léxico: Responsável por identificar tokens, como palavras-chave (int, float, if, else), operadores (+, -, \*, /, >), números e identificadores de variáveis.
- Analisador Sintático: Implementado para verificar se a estrutura das expressões e declarações está correta, garantindo que as regras de Mini Lang sejam seguidas.

## 2. Como o programa funciona

O programa utiliza expressões regulares para identificar tokens no código de entrada e, em seguida, aplica regras gramaticais para validar a sintaxe.

### 2.1. Fluxo de Execução

O código Mini Lang é fornecido como entrada.

O analisador léxico varre o código e transforma cada elemento em um token.

O analisador sintático processa os tokens e verifica se as regras da linguagem são seguidas.

Caso haja erro, uma mensagem é exibida.

## 3. Exemplos de entradas e saídas

### 3.1. Entrada:

```
int x = 10;  
y = x + 2;  
if (x > 0) { }
```

### 3.2.Saída:

=== Analisador Léxico ===

LexToken (INT,'int',1,0)

LexToken (ID,'x',1,4)

LexToken (EQUALS,'=',1,6)

LexToken (NUMBER,10,1,8)

LexToken (SEMICOLON,';',1,10)

...

=== Analisador Sintático ===

Declaração de variável:  $x = 10$

Expressão:  $y = x + 2$

Condição:  $\text{if } (x > 0)$

## 4. Dificuldades enfrentadas

Durante o desenvolvimento, algumas dificuldades surgiram, entre elas:

**Erro na identificação de operadores:** Inicialmente, a expressão  $y = x + 2$  era interpretada incorretamente como  $y = x = 2$ . Esse problema foi corrigido ajustando a regra de impressão na análise sintática.

**Ignorar espaços e quebras de linha:** O tratamento correto desses caracteres foi necessário para evitar falhas na tokenização.

**Definição de precedência na gramática:** Algumas ambiguidades precisaram ser resolvidas para evitar conflitos durante a análise sintática.

## 5. E quais são as próximas etapas do compilador.

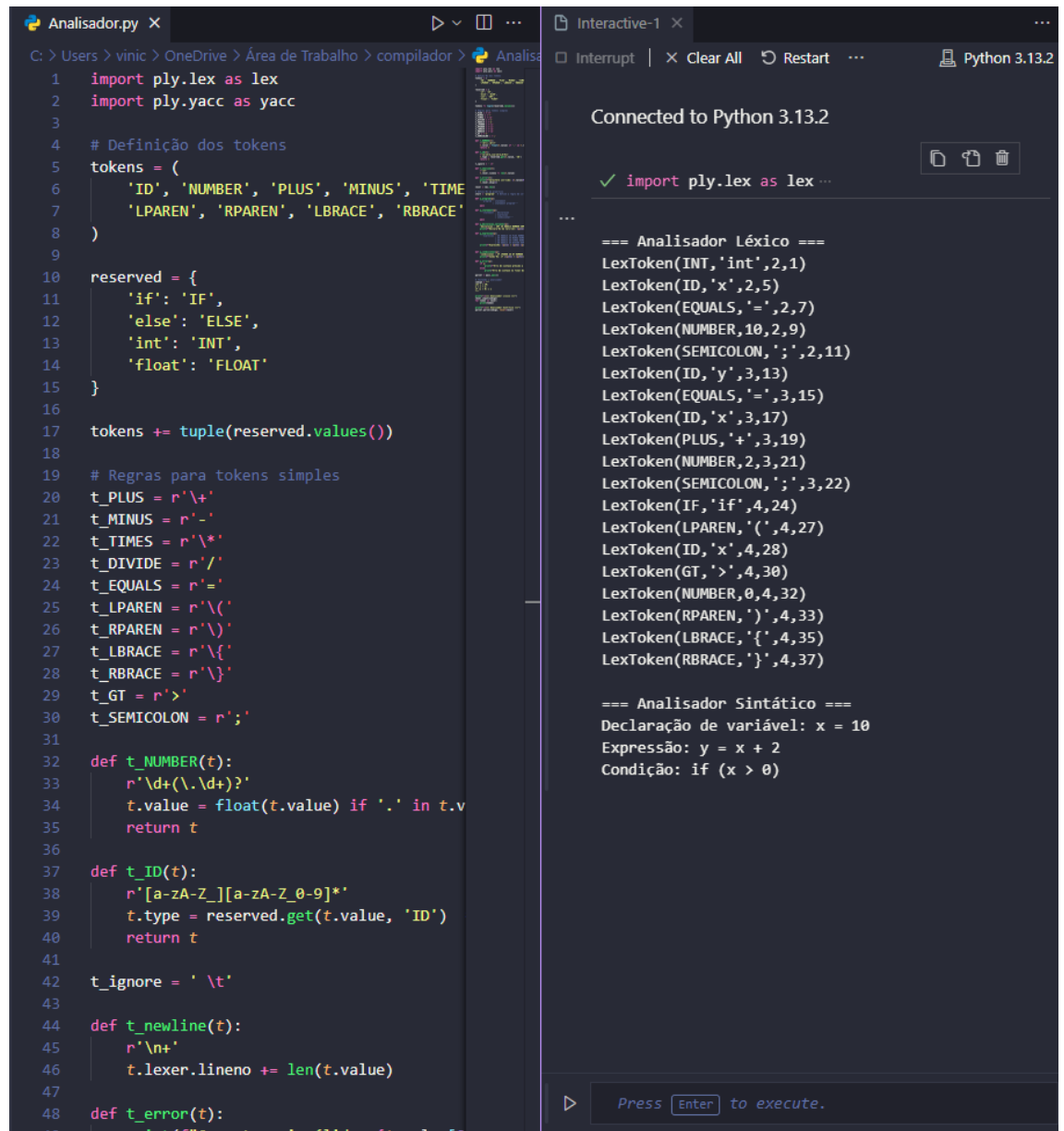
O próximo passo do projeto é expandir o compilador para incluir:

**Suporte a mais operadores e estruturas,** como while, for e funções.

**Geração de código intermediário,** permitindo a execução do código em uma máquina virtual.

**Otimização de código,** para melhorar a eficiência das expressões.

## Print da saída do Analisador Léxico e Sintático



The image shows a Python IDE with two panels. The left panel displays the source code for 'Analisador.py', and the right panel shows the interactive output of the script.

**Left Panel: Analisador.py**

```
1 import ply.lex as lex
2 import ply.yacc as yacc
3
4 # Definição dos tokens
5 tokens = (
6     'ID', 'NUMBER', 'PLUS', 'MINUS', 'TIMES', 'DIVIDE',
7     'LPAREN', 'RPAREN', 'LBRACE', 'RBRACE'
8 )
9
10 reserved = {
11     'if': 'IF',
12     'else': 'ELSE',
13     'int': 'INT',
14     'float': 'FLOAT'
15 }
16
17 tokens += tuple(reserved.values())
18
19 # Regras para tokens simples
20 t_PLUS = r'\+'
21 t_MINUS = r'\-'
22 t_TIMES = r'\*'
23 t_DIVIDE = r'\/'
24 t_EQUALS = r'\='
25 t_LPAREN = r'\('
26 t_RPAREN = r'\)'
27 t_LBRACE = r'\{'
28 t_RBRACE = r'\}'
29 t_GT = r'\>'
30 t_SEMICOLON = r';'
31
32 def t_NUMBER(t):
33     r'\d+(\.\d+)?'
34     t.value = float(t.value) if '.' in t.value else int(t.value)
35     return t
36
37 def t_ID(t):
38     r'[a-zA-Z_][a-zA-Z_0-9]*'
39     t.type = reserved.get(t.value, 'ID')
40     return t
41
42 t_ignore = '\t'
43
44 def t_newline(t):
45     r'\n+'
46     t.lexer.lineno += len(t.value)
47
48 def t_error(t):
49     print("Erro: %s" % t.value[0])
50     return None
```

**Right Panel: Interactive-1**

Connected to Python 3.13.2

```
✓ import ply.lex as lex ...
...
=== Analisador Léxico ===
LexToken(INT,'int',2,1)
LexToken(ID,'x',2,5)
LexToken(EQUALS,'=',2,7)
LexToken(NUMBER,10,2,9)
LexToken(SEMICOLON,';',2,11)
LexToken(ID,'y',3,13)
LexToken(EQUALS,'=',3,15)
LexToken(ID,'x',3,17)
LexToken(PLUS,'+',3,19)
LexToken(NUMBER,2,3,21)
LexToken(SEMICOLON,';',3,22)
LexToken(IF,'if',4,24)
LexToken(LPAREN,'(',4,27)
LexToken(ID,'x',4,28)
LexToken(GT,'>',4,30)
LexToken(NUMBER,0,4,32)
LexToken(RPAREN,')',4,33)
LexToken(LBRACE,'{',4,35)
LexToken(RBRACE,'}',4,37)

=== Analisador Sintático ===
Declaração de variável: x = 10
Expressão: y = x + 2
Condição: if (x > 0)
```

Press Enter to execute.