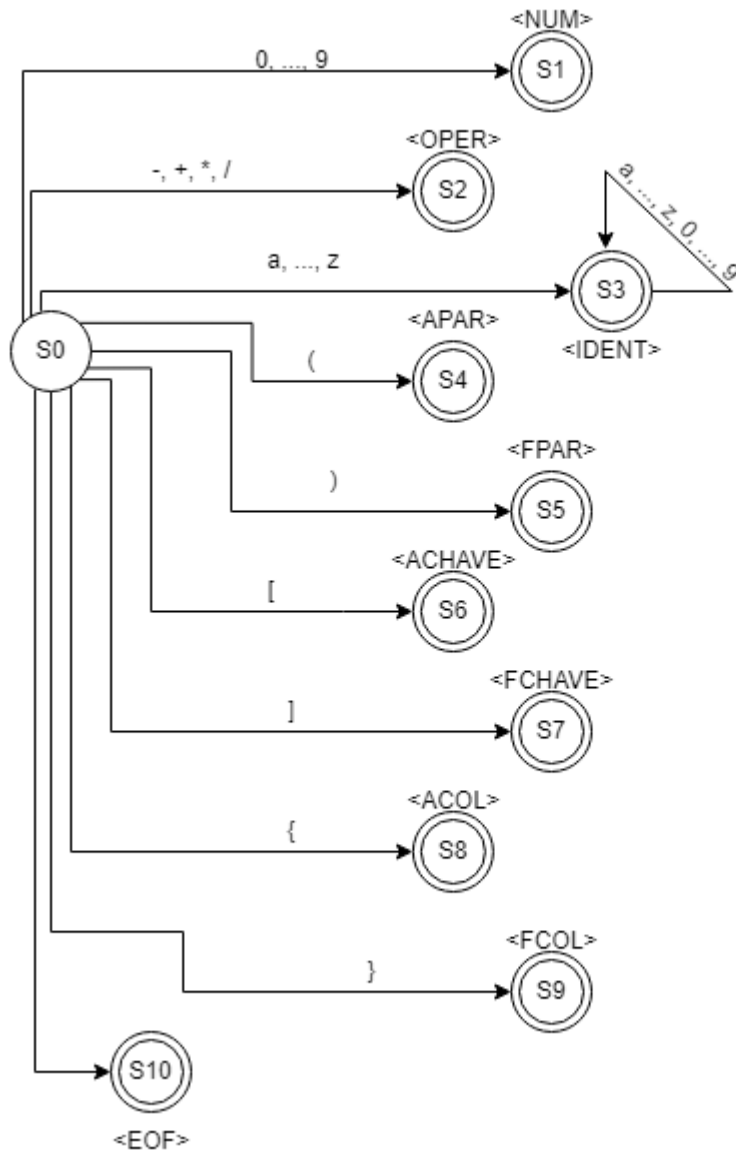


Disciplina: Linguagens Formais e Autômatos	Semestre: 2022/1
Prof. Faimison Rodrigues Porto	DCEL / CEUNES / UFES
Alunos: Paulo Cesar Viana de Albuquerque e Rafael Merlo Mendes	Matrícula: 2018204963 2016204382

Parte 1.

Expressões regulares para os tokens:

Token	ER	Exemplos de lexemas
<NUM>	$(0 ... 9)^+$	0, 31, 310, 0013
<OPER>	$+, -, *, /$	
<IDENT>	$(a ... z) \cdot (a ... z 0 ... 9)$	x, a, z
<APAR>	$($	
<FPAR>	$)$	
<ACHAVE>	$\{$	
<FCHAVE>	$\}$	
<ACOL>	$[$	
<FCOL>	$]$	
<EOF>	$\backslash t, \backslash n \backslash r, \text{ espaço}$	



Testes do analisador léxico - implementação da máquina de Moore:

Teste 1:

{(1+1)}+a-2

Console de saída:

<ACHAVE><APAR><NUM><OPER><NUM><FPAR><FCHAVE><OPER><IDENT><OPER><NUM>

Teste 2:

2+a/3

Console de saída:

<NUM><OPER><IDENT><OPER><NUM>

Teste 3:

$[(2+a/3)/(x*2)]$

Console de saída:

<ACOL><APAR><NUM><OPER><IDENT><OPER><NUM><FPAR><OPER><APAR>
><IDENT><OPER><NUM><FPAR><FCOL>

```
AnalizadorLexico.py - C:\Users\rafae\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help
# -*- coding: utf-8 -*-

import sys
from abc import ABC, abstractmethod
class Analizador(ABC):

    def __init__(self):

        self.NOME_DEFAULT_ARQUIVO_ENTRADA = 'entrada.txt'
        self.tokens = {

            'NUM': ['0','1','2','3','4','5','6','7','8','9'],
            'APAR': '(',
            'FPAR': ')',
            'ACHAVE': '{',
            'FCHAVE': '}',
            'ACOL': '[',
            'FCOL': ']',
            'OPER': ['+', '-', '*', '/'],
            'IDENT': ['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','x','y','z']
        }

    @abstractmethod
    def Analizador(self, _nomeArquivoEntrada):
        self.nomeArquivoEntrada = self._nomeArquivoEntrada

    def Analizador(self):
        self.nomeArquivoEntrada = self.NOME_DEFAULT_ARQUIVO_ENTRADA

class AnalizadorLexico(Analizador):

    def __init__(self, _nomeArquivoEntrada):
        super().__init__()
        self.proxCaractere = ''
        self.linha = 1
        self.posicao = -1
        self.tokenReconhecido = ''
        self.s = []
        self.entrada = ''
        try:
```

Ln: 18 Col: 0

```
AnalizadorLexico.py - C:\Users\rafael\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help

    try:
        self.file = open(_nomeArquivoEntrada, "r")
        self.entrada = self.file.read()
        self.file.close()
        self.leProxCaractere()

    except:
        print("Erro na leitura do arquivo" + _nomeArquivoEntrada)

def leProxCaractere(self):

    try:

        self.posicao += 1
        self.proxCaractere = self.entrada[self.posicao]

    |
    except IndexError:

        self.proxCaractere = '<EXIT>'

def proxCaractereIs(self, s):

    if self.proxCaractere in s:
        return True
    else:
        return False

class MyAnalizadorLexico(AnalizadorLexico):

    def __init__(self, _nomeArquivoEntrada):
        super().__init__(_nomeArquivoEntrada)

    def s0(self):

        if(self.proxCaractereIs(self.tokens['NUM'])):
            self.leProxCaractere()
            self.s1()

Ln: 57 Col: 12
```

```
AnalizadorLexico.py - C:\Users\rafael\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help

    elif(self.proxCaractereIs(self.tokens['APAR'])):
        self.leProxCaractere()
        self.s2()

    elif(self.proxCaractereIs(self.tokens['FPAR'])):
        self.leProxCaractere()
        self.s3()

    elif(self.proxCaractereIs(self.tokens['ACHAVE'])):
        self.leProxCaractere()
        self.s4()

    elif(self.proxCaractereIs(self.tokens['FCHAVE'])):
        self.leProxCaractere()
        self.s5()

    elif(self.proxCaractereIs(self.tokens['ACOL'])):
        self.leProxCaractere()
        self.s6()

    elif(self.proxCaractereIs(self.tokens['FCOL'])):
        self.leProxCaractere()
        self.s7()

    elif(self.proxCaractereIs(self.tokens['OPER'])):
        self.leProxCaractere()
        self.s8()

    elif(self.proxCaractereIs(self.tokens['IDENT'])):
        self.leProxCaractere()
        self.s9()

    else:

        if(self.proxCaractere == '<EXIT>'):
            sys.exit()

        print('\nErro léxico: caractere encontrado:' + self.proxCaractere)
        print('era(m) esperados(s):')

Ln: 87 Col: 8
```

```
AnalizadorLexico.py - C:\Users\rafael\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help

    print('era(m) esperados(s):')
    print(self.tokens)

    self.leProxCaractere()

    self.s0()

def s1(self):

    self.tokenReconhecido = '<NUM>'

    if(self.proxCaractereIs(self.tokens['NUM'])):
        self.leProxCaractere()
        self.s1()

def s2(self):

    self.tokenReconhecido = '<APAR>'

    if(self.proxCaractereIs(self.tokens['APAR'])):

        self.leProxCaractere()
        self.s2()

def s3(self):

    self.tokenReconhecido = '<FPAR>'

    if(self.proxCaractereIs(self.tokens['FPAR'])):

        self.leProxCaractere()
        self.s3()

def s4(self):

    self.tokenReconhecido = '<ACHAVE>'

    if(self.proxCaractereIs(self.tokens['ACHAVE'])):

        self.leProxCaractere()

Ln: 92 Col: 48
```

```
AnalizadorLexico.py - C:\Users\rafael\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help

        self.leProxCaractere()
        self.s4()

def s5(self):

    self.tokenReconhecido = '<FCHAVE>'

    if(self.proxCaractereIs(self.tokens['FCHAVE'])):

        self.leProxCaractere()
        self.s5()

def s6(self):

    self.tokenReconhecido = '<ACOL>'

    if(self.proxCaractereIs(self.tokens['ACOL'])):

        self.leProxCaractere()
        self.s6()

def s7(self):

    self.tokenReconhecido = '<FCOL>'

    if(self.proxCaractereIs(self.tokens['FCOL'])):

        self.leProxCaractere()
        self.s7()

def s8(self):

    self.tokenReconhecido = '<OPER>'

    if(self.proxCaractereIs(self.tokens['OPER'])):
        self.leProxCaractere()
        self.s8()

def s9(self):

Ln: 137 Col: 54
```

```
AnalizadorLexico.py - C:\Users\rafael\Downloads\AnalizadorLexico.py (3.10.5)
File Edit Format Run Options Window Help

def s7(self):
    self.tokenReconhecido = '<FCOL>'

    if(self.proxCaractereIs(self.tokens['FCOL'])):
        self.leProxCaractere()
        self.s7()

def s8(self):
    self.tokenReconhecido = '<OPER>'

    if(self.proxCaractereIs(self.tokens['OPER'])):
        self.leProxCaractere()
        self.s8()

def s9(self):
    self.tokenReconhecido = '<IDENT>'

    if(self.proxCaractereIs(self.tokens['IDENT'])):
        self.leProxCaractere()
        self.s9()

#Main
A = MyAnalizadorLexico('file.txt')

while True:
    A.s0()
    print(A.tokenReconhecido, end='')

    if A.proxCaractere == '<EXIT>':
        break

#file.close()

Ln: 177 Col: 12
```

Parte 2.

```
E -> <NUM> R | <IDENT> R | (E1) R | [E2] R | {E3} R
File Edit Selection Find View Goto Tools Project Preferences Help

E -> <NUM> R | <IDENT> R | (E1) R | [E2] R | {E3} R
1 E -> <NUM> R | <IDENT> R | (E1) R | [E2] R | {E3} R
2 E1 -> <NUM> R1 | <IDENT> R | (E1) R1
3 E2 -> <NUM> R2 | <IDENT> R2 | [E2] R2 | (E1) R3
4 E3 -> <NUM> R3 | <IDENT> R3 | {E3} R3 | [E2] R3 | (E1) R3
5 OPER -> + | - | * | /
6 NUM -> (0NUM | ... | 9NUM)
7 IDENT -> (aIDENT | ... | zIDENT) . (aIDENT | ... | zIDENT | 0IDENT |
... | 9IDENT)
```

```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

# -*- coding: utf-8 -*-

import sys
from abc import ABC, abstractmethod
class Analizador(ABC):

    def __init__(self):

        self.NOME_DEFAULT_ARQUIVO_ENTRADA = 'entrada.txt'
        self.tokens = {

            'INT': ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '<INT>'],
            'APAR': ['(', '<APAR>'],
            'FPAR': [')', '<FPAR>'],
            'ACHAVE': ['{', '<ACHAVE>'],
            'FCHAVE': ['}', '<FCHAVE>'],
            'ACOL': ['[', '<ACOL>'],
            'FCOL': [']', '<FCOL>'],
            'OPER': ['+', '-', '*', '/', '<OPER>'],
            'IDENT': ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
            'EOF': ''

        ]

    @abstractmethod
    def Analizador(self, _nomeArquivoEntrada):
        self.nomeArquivoEntrada = self._nomeArquivoEntrada

    def Analizador(self):
        self.nomeArquivoEntrada = self.NOME_DEFAULT_ARQUIVO_ENTRADA

class AnalizadorLexico(Analizador):

    def __init__(self, _nomeArquivoEntrada):
        super().__init__()
        self.proxCaractere = ''
        self.linha = 1
        self.posicao = -1
        self.tokenReconhecido = ''
        self.s = []

Ln: 25 Col: 0
```

Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)

File Edit Format Run Options Window Help

```
self.tokenReconhecido = ''
self.s = []
self.entrada = ''
try:
    self.file = open(_nomeArquivoEntrada, "r")
    self.entrada = self.file.read()
    self.file.close()
    self.leProxCaractere()

except:
    print("Erro na leitura do arquivo" + _nomeArquivoEntrada)

def leProxCaractere(self):

    try:

        self.posicao += 1
        self.proxCaractere = self.entrada[self.posicao]

    except IndexError:

        self.proxCaractere = '<EXIT>'

def proxCaractereIs(self, s):

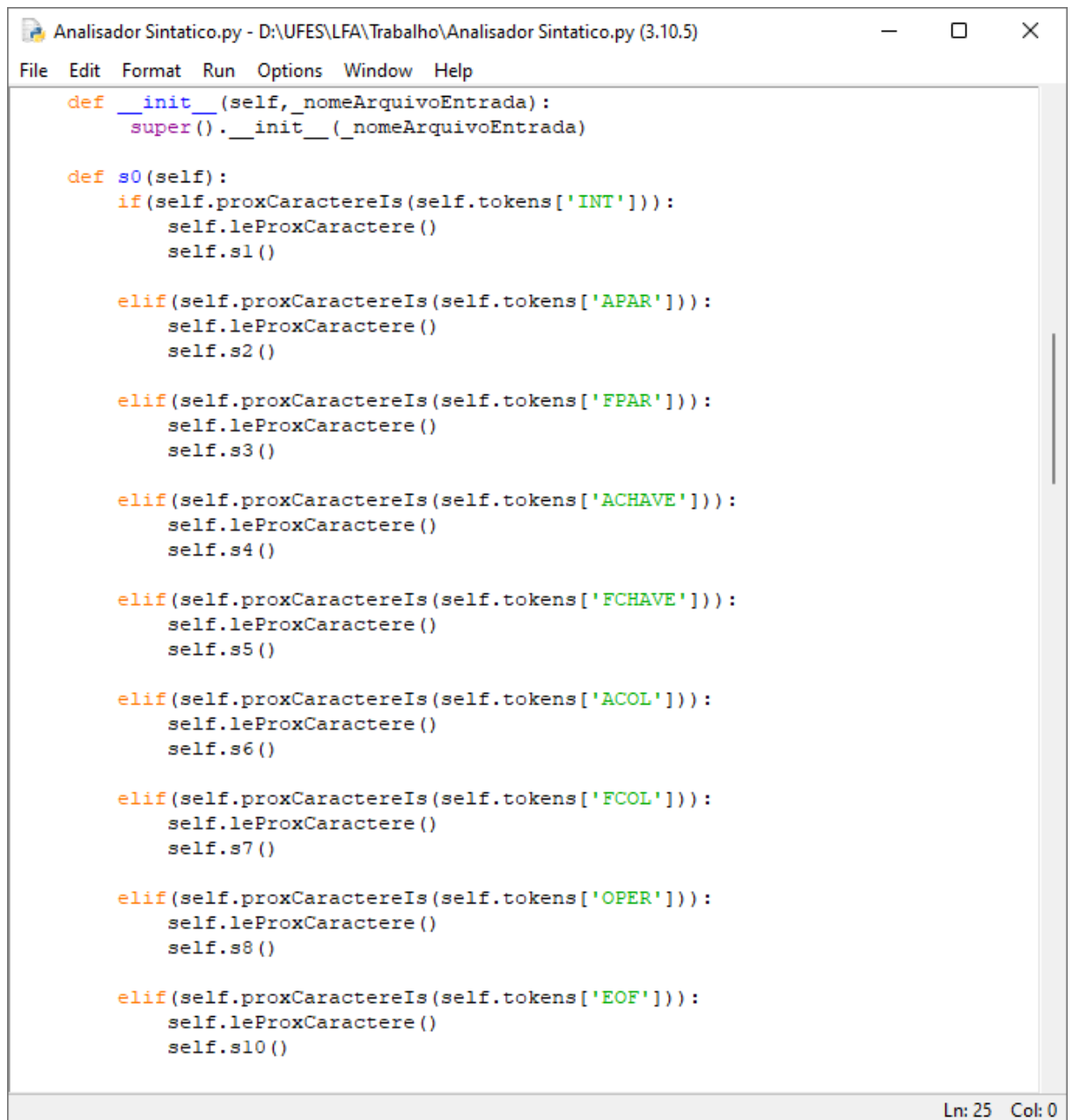
    if self.proxCaractere in s:
        return True
    else:
        return False

class MyAnalizadorLexico(AnalizadorLexico):

    def __init__(self, _nomeArquivoEntrada):
        super().__init__(_nomeArquivoEntrada)

    def s0(self):
        if(self.proxCaractereIs(self.tokens['INT'])):
            self.leProxCaractere()
```

Ln: 25 Col: 0



```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

def __init__(self, _nomeArquivoEntrada):
    super().__init__(_nomeArquivoEntrada)

def s0(self):
    if(self.proxCaractereIs(self.tokens['INT'])):
        self.leProxCaractere()
        self.s1()

    elif(self.proxCaractereIs(self.tokens['APAR'])):
        self.leProxCaractere()
        self.s2()

    elif(self.proxCaractereIs(self.tokens['FPAR'])):
        self.leProxCaractere()
        self.s3()

    elif(self.proxCaractereIs(self.tokens['ACHAVE'])):
        self.leProxCaractere()
        self.s4()

    elif(self.proxCaractereIs(self.tokens['FCHAVE'])):
        self.leProxCaractere()
        self.s5()

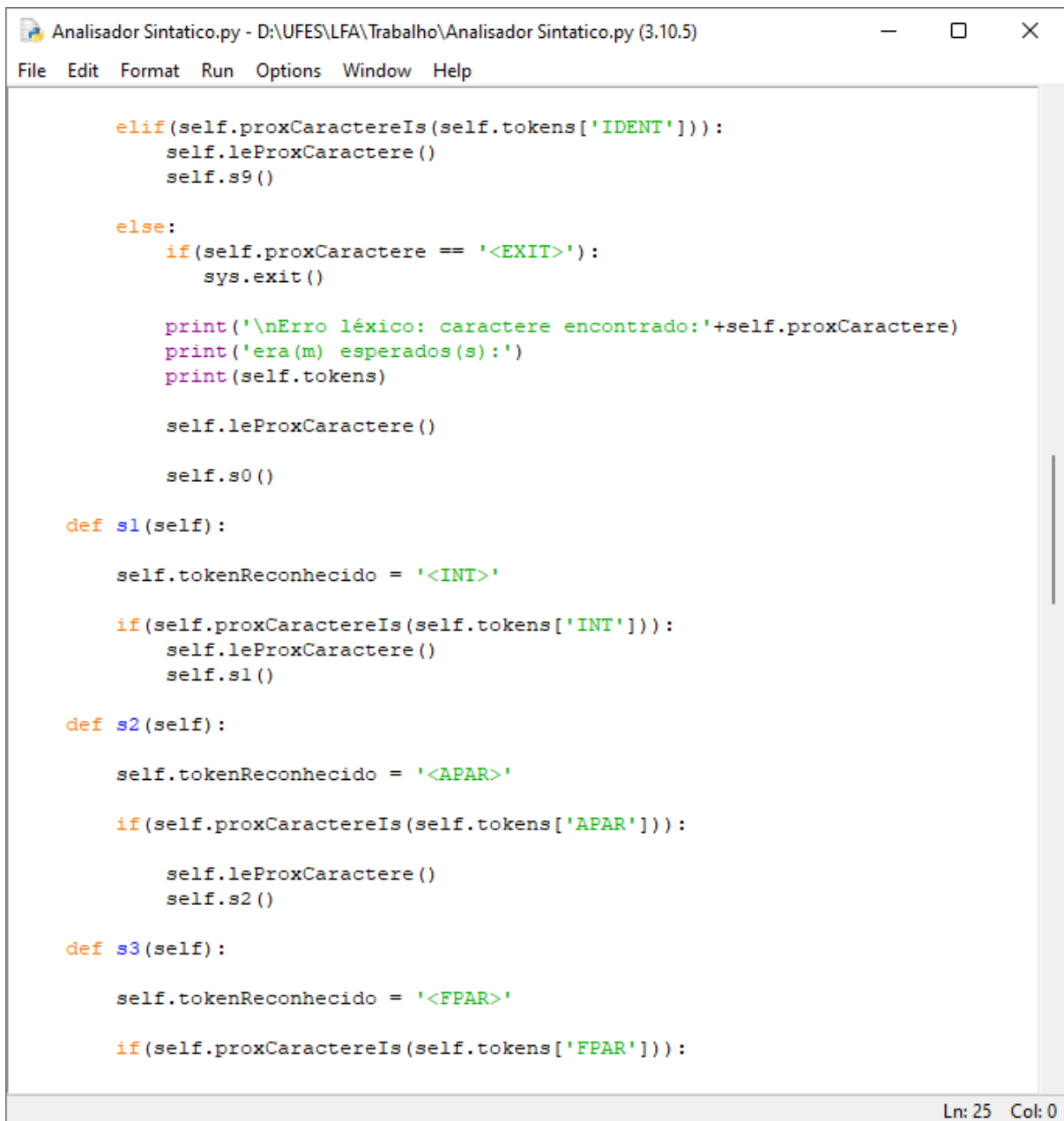
    elif(self.proxCaractereIs(self.tokens['ACOL'])):
        self.leProxCaractere()
        self.s6()

    elif(self.proxCaractereIs(self.tokens['FCOL'])):
        self.leProxCaractere()
        self.s7()

    elif(self.proxCaractereIs(self.tokens['OPER'])):
        self.leProxCaractere()
        self.s8()

    elif(self.proxCaractereIs(self.tokens['EOF'])):
        self.leProxCaractere()
        self.s10()

Ln: 25 Col: 0
```



```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

    elif(self.proxCaractereIs(self.tokens['IDENT'])):
        self.leProxCaractere()
        self.s9()

    else:
        if(self.proxCaractere == '<EXIT>'):
            sys.exit()

        print('\nErro léxico: caractere encontrado:'+self.proxCaractere)
        print('era(m) esperados(s):')
        print(self.tokens)

        self.leProxCaractere()

        self.s0()

def s1(self):

    self.tokenReconhecido = '<INT>'

    if(self.proxCaractereIs(self.tokens['INT'])):
        self.leProxCaractere()
        self.s1()

def s2(self):

    self.tokenReconhecido = '<APAR>'

    if(self.proxCaractereIs(self.tokens['APAR'])):

        self.leProxCaractere()
        self.s2()

def s3(self):

    self.tokenReconhecido = '<FPAR>'

    if(self.proxCaractereIs(self.tokens['FPAR'])):
```

Ln: 25 Col: 0



The image shows a screenshot of a Python IDE window titled "Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main editor area contains Python code for a syntactic analyzer. The code defines methods `s3`, `s4`, `s5`, `s6`, and `s7` within a class. Each method `s4` through `s7` sets a `tokenReconhecido` attribute and checks if the next character in the `tokens` list matches a specific token (`<ACHAVE>`, `<FCHAVE>`, `<ACOL>`, and `<FCOL>` respectively). If the match is successful, it calls `leProxCaractere()` and then the corresponding `s` method (`s4` through `s7`). The status bar at the bottom right indicates "Ln: 25 Col: 0".

```
        self.leProxCaractere()
        self.s3()

    def s4(self):

        self.tokenReconhecido = '<ACHAVE>'

        if(self.proxCaractereIs(self.tokens['ACHAVE'])):

            self.leProxCaractere()
            self.s4()

    def s5(self):

        self.tokenReconhecido = '<FCHAVE>'

        if(self.proxCaractereIs(self.tokens['FCHAVE'])):

            self.leProxCaractere()
            self.s5()

    def s6(self):

        self.tokenReconhecido = '<ACOL>'

        if(self.proxCaractereIs(self.tokens['ACOL'])):

            self.leProxCaractere()
            self.s6()

    def s7(self):

        self.tokenReconhecido = '<FCOL>'

        if(self.proxCaractereIs(self.tokens['FCOL'])):

            self.leProxCaractere()
            self.s7()
```

Ln: 25 Col: 0

```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

def s8(self):
    self.tokenReconhecido = '<OPER>'

    if(self.proxCaractereIs(self.tokens['OPER'])):
        self.leProxCaractere()
        self.s8()

def s9(self):
    self.tokenReconhecido = '<IDENT>'

    if(self.proxCaractereIs(self.tokens['IDENT'])):
        self.leProxCaractere()
        self.s9()

def s10(self):
    self.tokenReconhecido = ''

class AnalizadorSintatico(Analizador):

    def __init__(self, _nomeArquivoEntrada):
        self.t = []
        self.A = MyAnalizadorLexico(_nomeArquivoEntrada)
        super().__init__()
        self.leProxToken()

    def leProxToken(self):
        self.A.s0()

    def reconhece(self, t):

        if self.A.tokenReconhecido in t:
            self.leProxToken()
        else:
            print('\nErro Sintático: token encontrado: ' + self.A.tokenReconhecido)

Ln: 25 Col: 0
```

```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

    else:
        print('\nErro Sintático: token encontrado:' + self.A.tokenReconhecido)
        print('era(m) esperados(s):' , end = '')
        print(t)

    def proxTokenIs(self, t):

        if self.A.tokenReconhecido in t:
            return True
        else:
            return False

class MyAnalizadorSintatico(AnalizadorSintatico):
    def __init__(self, _nomeArquivoEntrada):

        self.A = AnalizadorSintatico(_nomeArquivoEntrada)

        super().__init__(_nomeArquivoEntrada)
    def inicio(self):
        self.corpo()

    def corpo(self):

        if (self.proxTokenIs(self.A.tokens['EOF'])):
            print('\nAnálise Sintática: Concluída')
        else:
            print('\nErro Sintático: token encontrado:' + self.A.tokenReconhecido)
            print('era(m) esperados(s): ',end='')

    def exp(self):

        if (self.proxTokenIs(self.tokens['INT'])):
            self.leProxToken()
            if (self.proxTokenIs(self.tokens['OPER'])):
                self.leProxToken()
                self.exp()
            elif (self.proxTokenIs(self.tokens['INT'])):
                self.leProxToken()
```

Ln: 25 Col: 0

```
Analizador Sintatico.py - D:\UFES\LFA\Trabalho\Analizador Sintatico.py (3.10.5)
File Edit Format Run Options Window Help

    self.leProxToken()
    if (self.proxTokenIs(self.tokens['OPER'])):
        self.leProxToken()
        self.exp()
    elif (self.proxTokenIs(self.tokens['INT'])):
        self.leProxToken()
    elif (self.proxTokenIs(self.tokens['VAZIO'])):
        self.leProxToken()

    else:
        print('\nErro Sintático: token encontrado:' + self.A.tokenReconhecido)
        print('era(m) esperados(s):', end = '')
        print( self.t)

def bloco(self):
    self.exp()

def cmdSwitchCase(self):
    self.reconhece(self.tokens['INT'])
    self.bloco()

#Main
A = MyAnalizadorLexico('file.txt')

while True:
    A.s0()
    print(A.tokenReconhecido, end='')

    if A.proxCaractere == '<EXIT>':
        break

tst = MyAnalizadorSintatico('file.txt')
tst.inicio()
print('Análise concluída com sucesso')
```

Ln: 25 Col: 0