Analizador Léxico

PLY (Python Lex-Yacc)

- Es una implementación de las herramientas de análisis gramatical ("parsing") lex y yacc en Python.

- La versión original de PLY se desarrolló en 2001 para usarse en una clase de introducción a los compiladores dónde los estudiantes usaban esta herramienta para construir un compilador para un lenguaje simple similar a Pascal. Debido a esto, ofrece una revisión de errores extensiva.

- Se compone de dos módulos separados:
  - Lex.py se usa para convertir el texto de entrada en una colección de tokens especificadas por una colección de reglas de expresiones regulares.
  - Yacc.py se utiliza para reconocer la sintaxis del lenguaje que se ha especificado en el formulario.

Ejemplo de ply.lex

```
import lex as lex

filename = sys.argv[1]
file_handle = open (filename, "r")
file_contents = file_handle.read()


# List of token names.   This is always required
tokens = (
   'NUMBER',
   'PLUS',
)

# Regular expression rules for simple tokens
t_PLUS    = r'\+'

# A regular expression rule with some action code
def t_NUMBER(t):
    r'[0-9]+'
    t.value = int(t.value)     # this is the lexeme
    return t

# Define a rule so we can track line numbers
def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value) # we have to do this for PA2

# A string containing ignored characters (spaces and tabs)
"main.py" 43L, 924C written

[1]+  Stopped                 vim main.py

weimer@inochi ~/Desktop/ldi
$ ls
```

Ejemplo ply.yacc

```
import ply.yacc as yacc
import mylexer              # Import lexer information
tokens = mylexer.tokens     # Need token list

def p_assign(p):
    '''assign : NAME EQUALS expr'''

def p_expr(p):
    '''expr : expr PLUS term
            | expr MINUS term
            | term'''
def p_term(p):
    '''term : term TIMES factor
            | term DIVIDE factor
            | factor'''
def p_factor(p):
    '''factor : NUMBER'''

yacc.yacc()             # Build the parser
```

grammar rules encoded as functions with names p_*rulename*

Note: Name doesn't matter as long as it starts with p_

Como se observa en los ejemplos anteriores, un analizador léxico utilizando PLY se compone a grandes rasgos de 4 partes:

1. Importar PLY y las librerías necesarias.
2. Definir lista de tokens.
3. Definir expresiones regulares y sus reglas para cada token. Si es necesario realizar algún tipo de acción, se puede especificar una regla de token como una función.
4. Código para la prueba del archivo fuente a analizar.

Código implementado. Parte 1.

```
3    import ply.lex as lex
4    import argparse
5    import sys
6
7    columna = 0
```

Código implementado. Parte 2.

```
8   # list of tokens
9   tokens = (
10
11          # Reserverd words
12          'KEY_ELSE',
13          'KEY_IF',
14          'INT',
15          'FLOAT',
16          'CHAR',
17          'RETURN',
18          'VOID',
19          'MAIN',
20          'KEY_WHILE',
21          'KEY_FOR',
22
```

```
23          # Symbols
24          'PLUS',
25          'MINUS',
26          'TIMES',
27          'DIVIDE',
28          'MENOR_QUE',
29          'MENOR_QUE_IGUAL',
30          'MAYOR_QUE',
31          'MAYOR_QUE_IGUAL',
32          'IGUAL',
33          'DIGUAL',
34          'DISTINT',
35          'PUNTOCOMA',
36          'COMMA',
37          'LPAREN',
38          'RPAREN',
39          'LBRACKET',
40          'RBRACKET',
41          'LBLOCK',
42          'RBLOCK',
```

```
44          # Others
45          'IDENTIFIER',
46          'IDENTIFIER_1',
47          'NUMBER',
48
49          #libs
50          'LIB',
51          'DEFINE',
52
53          #Literarl
54          'LITERAL',
55          'LITERAL_2',
56
57          #WhiteSpace
58          'WHITESPACE',
59          'WHITESPACE_T',
60          'WHITESPACE_N',
61
```

```
62            # Regular expressions
63            'PLUS',
64            'MINUS',
65            'TIMES',
66            'DIVIDE',
67            'IGUAL',
68            'MENOR_QUE',
69            'MAYOR_QUE',
70            'PUNTOCOMA',
71            'COMMA',
72            'LPAREN',
73            'RPAREN',
74            'LBRACKET',
75            'RBRACKET',
76            'LBLOCK',
77            'RBLOCK',
78
79    )
80
```

Código implementado. Parte 3.

```
81  # Regular expressions rules for a simple tokens
82  def t_PLUS(t):
83          r'\+'
84          global columna
85          columna += 1
86          return t
87
88  def t_MINUS(t):
89          r'-'
90          global columna
91          columna += 1
92          return t
93
94  def t_TIMES(t):
95          r'\*'
96          global columna
97          columna += 1
98          return t
```

```
100   def t_DIVIDE(t):
101           r'/'
102           global columna
103           columna += 1
104           return t
105
106   def t_IGUAL(t):
107           r'='
108           global columna
109           columna += 1
110           return t
111
112   def t_MENOR_QUE(t):
113           r'<'
114           global columna
115           columna += 1
116           return t
117
118   def t_MAYOR_QUE(t):
119           r'>'
120           global columna
121           columna += 1
122           return t
123
```

```
124   def t_PUNTOCOMA(t):
125           r';'
126           global columna
127           columna += 1
128           return t
129
130   def t_COMMA(t):
131           r','
132           global columna
133           columna += 1
134           return t
135
136   def t_LPAREN(t):
137           r'\('
138           global columna
139           columna += 1
140           return t
141
142   def t_RPAREN(t):
143           r'\)'
144           global columna
145           columna += 1
146           return t
147
```

```
148   def t_LBRACKET(t):
149           r'\['
150           global columna
151           columna += 1
152           return t
153
154   def t_RBRACKET(t):
155           r'\]'
156           global columna
157           columna += 1
158           return t
159
160   def t_LBLOCK(t):
161           r'{'
162           global columna
163           columna += 1
164           return t
165
166   def t_RBLOCK(t):
167           r'}'
168           global columna
169           columna += 1
170           return t
171
```

```
188   def t_LIB(t):
189           r'\#include'
190           global columna
191           columna += 1
192           return t
193
194   def t_DEFINE(t):
195           r'\#define'
196           global columna
197           columna += 1
198           return t
199
200   def t_VOID(t):
201           r'void'
202           global columna
203           columna += 1
204           return t
205
206   def t_MAIN(t):
207           r'main'
208           global columna
209           columna += 1
210           return t
```

```
212   def t_INT(t):
213           r'int'
214           global columna
215           columna += 1
216           return t
217
218   def t_FLOAT(t):
219           r'float'
220           global columna
221           columna += 1
222           return t
223
224   def t_CHAR(t):
225           r'char'
226           global columna
227           columna += 1
228           return t
229
230   def t_KEY_IF(t):
231           r'if'
232           global columna
233           columna += 1
234           return t
```

```
236   def t_KEY_ELSE(t):
237           r'else'
238           global columna
239           columna += 1
240           return t
241
242   def t_RETURN(t):
243           r'return'
244           global columna
245           columna += 1
246           return t
247
248   def t_KEY_WHILE(t):
249           r'while'
250           global columna
251           columna += 1
252           return t
253
254   def t_KEY_FOR(t):
255           r'for'
256           global columna
257           columna += 1
258           return t
259
```

```
260   def t_NUMBER(t):
261           r'\d+'
262           t.value = int(t.value)
263           global columna
264           columna += 1
265           return t
266
267   def t_IDENTIFIER_1(t):
268           r'\w+(\.\w)*'
269           global columna
270           columna += 1
271           return t
272
273   def t_IDENTIFIER(t):
274           r'\w+(_\d\w)*'
275           global columna
276           columna += 1
277           return t
278
279   def t_LITERAL(t):
280           r'\"(.)*?\"'
281           t.lexer.lineno += t.value.count('\n')
282           global columna
283           columna += 1
284           return t
285
```

```
286   def t_LITERAL_2(t):
287           r'\'(.)*?\''
288           t.lexer.lineno += t.value.count('\n')
289           global columna
290           columna += 1
291           return t
292
293   def t_MENOR_QUE_IGUAL(t):
294           r'<='
295           global columna
296           columna += 1
297           return t
298
299   def t_MAYOR_QUE_IGUAL(t):
300           r'>='
301           global columna
302           columna += 1
303           return t
304
305   def t_DIGUAL(t):
306           r'=='
307           global columna
308           columna += 1
309           return t
310
```

```
311  def t_DISTINT(t):
312          r'!='
313          global columna
314          columna += 1
315          return t
316
317  def t_WHITESPACE(t):
318          r'\ '
319  #       t.lexer.lineno += len(t.value)
320          global columna
321          columna += 1
322          return t
323
324  def t_WHITESPACE_N(t):
325          r'\n+'
326          t.lexer.lineno += len(t.value)
327          global columna
328          columna += 1
329          return t
330
331  def t_WHITESPACE_T(t):
332          r'\t+'
333  #       t.lexer.lineno += len(t.value)
334          global columna
335          columna += 1
336          return t
337
```

```
340  def t_comments(t):
341          r'/\*(.|\n)*?\*/'
342          t.lexer.lineno += t.value.count('\n')
343
344  def t_comments_C99(t):
345          r'//(.)*?\n'
346          t.lexer.lineno += 1
347
348  def t_error(t):
349          print "Lexical error: " + str(t.value[0])
350          t.lexer.skip(1)
351
352  def test(data, lexer):
353          lexer.input(data)
354          while True:
355                  tok = lexer.token()
356                  if not tok:
357                          break
358                  #print(tok)
359                  if      tok.value=="\n":
360                          string=str("\\n")
361                  else:
362                          string=str(tok.value)
363                  print "Lex (" + str(tok.type) + ", " + str(string) + "," + str(tok.lineno) + "," + str(columna) +"," + str(tok.lexpos) + ")"
364
```

Código implementado. Parte 4.

```
366  lexer = lex.lex()
367
368  # Test
369  if __name__ == '__main__':
385          if (len(sys.argv) > 1):
386                  fin = sys.argv[1]
387          else:
388                  fin = 'Matriz_File.c'
389
390          print "Nombre del Archivo: " + fin + "\n"
391          f = open(fin, 'r')
392          lineas = f.readlines()
393          for linea in lineas:
394                  lexer.input(linea)
395                  test(linea, lexer)
396                  columna=0
397                  print "----------------------------------------------------------------------------"
```

Resultados

```
Nombre del Archivo: Matriz_File.c

Lex (LIB, #include,1,1,0)
Lex (WHITESPACE,  ,1,2,8)
Lex (MENOR_QUE, <,1,3,9)
Lex (IDENTIFIER_1, stdio.h,1,4,10)
Lex (MAYOR_QUE, >,1,5,17)
Lex (WHITESPACE_N, \n,1,6,18)
-------------------------------------------------------------------------
Lex (LIB, #include,2,1,0)
Lex (WHITESPACE,  ,2,2,8)
Lex (MENOR_QUE, <,2,3,9)
Lex (IDENTIFIER_1, string.h,2,4,10)
Lex (MAYOR_QUE, >,2,5,18)
Lex (WHITESPACE_N, \n,2,6,19)
-------------------------------------------------------------------------
Lex (LIB, #include,3,1,0)
Lex (WHITESPACE,  ,3,2,8)
Lex (MENOR_QUE, <,3,3,9)
Lex (IDENTIFIER_1, stdlib.h,3,4,10)
Lex (MAYOR_QUE, >,3,5,18)
Lex (WHITESPACE_N, \n,3,6,19)
-------------------------------------------------------------------------
Lex (WHITESPACE_N, \n,4,1,0)
-------------------------------------------------------------------------
```

```
Lex (DEFINE, #define,5,1,0)
Lex (WHITESPACE,  ,5,2,7)
Lex (IDENTIFIER_1, FILAS_MATRIZ_A,5,3,8)
Lex (WHITESPACE,  ,5,4,22)
Lex (NUMBER, 3,5,5,23)
Lex (WHITESPACE_N, \n,5,6,24)
-----------------------------------------------------------------------
Lex (DEFINE, #define,6,1,0)
Lex (WHITESPACE,  ,6,2,7)
Lex (IDENTIFIER_1, COLUMNAS_MATRIZ_A,6,3,8)
Lex (WHITESPACE,  ,6,4,25)
Lex (NUMBER, 3,6,5,26)
Lex (WHITESPACE_N, \n,6,6,27)
-----------------------------------------------------------------------
Lex (DEFINE, #define,7,1,0)
Lex (WHITESPACE,  ,7,2,7)
Lex (IDENTIFIER_1, FILAS_MATRIZ_B,7,3,8)
Lex (WHITESPACE,  ,7,4,22)
Lex (NUMBER, 3,7,5,23)
Lex (WHITESPACE_N, \n,7,6,24)
-----------------------------------------------------------------------
Lex (DEFINE, #define,8,1,0)
Lex (WHITESPACE,  ,8,2,7)
Lex (IDENTIFIER_1, COLUMNAS_MATRIZ_B,8,3,8)
Lex (WHITESPACE,  ,8,4,25)
Lex (NUMBER, 3,8,5,26)
Lex (WHITESPACE_N, \n,8,6,27)
```

```
Lex (DEFINE, #define,5,1,0)
Lex (WHITESPACE,  ,5,2,7)
Lex (IDENTIFIER_1, FILAS_MATRIZ_A,5,3,8)
Lex (WHITESPACE,  ,5,4,22)
Lex (NUMBER, 3,5,5,23)
Lex (WHITESPACE_N, \n,5,6,24)
--------------------------------------------------------------------------------
Lex (DEFINE, #define,6,1,0)
Lex (WHITESPACE,  ,6,2,7)
Lex (IDENTIFIER_1, COLUMNAS_MATRIZ_A,6,3,8)
Lex (WHITESPACE,  ,6,4,25)
Lex (NUMBER, 3,6,5,26)
Lex (WHITESPACE_N, \n,6,6,27)
--------------------------------------------------------------------------------
Lex (DEFINE, #define,7,1,0)
Lex (WHITESPACE,  ,7,2,7)
Lex (IDENTIFIER_1, FILAS_MATRIZ_B,7,3,8)
Lex (WHITESPACE,  ,7,4,22)
Lex (NUMBER, 3,7,5,23)
Lex (WHITESPACE_N, \n,7,6,24)
--------------------------------------------------------------------------------
Lex (DEFINE, #define,8,1,0)
Lex (WHITESPACE,  ,8,2,7)
Lex (IDENTIFIER_1, COLUMNAS_MATRIZ_B,8,3,8)
Lex (WHITESPACE,  ,8,4,25)
Lex (NUMBER, 3,8,5,26)
Lex (WHITESPACE_N, \n,8,6,27)
--------------------------------------------------------------------------------
Lex (DEFINE, #define,9,1,0)
Lex (WHITESPACE,  ,9,2,7)
Lex (IDENTIFIER_1, FILAS_MATRIZ_C,9,3,8)
Lex (WHITESPACE,  ,9,4,22)
Lex (NUMBER, 3,9,5,23)
Lex (WHITESPACE_N, \n,9,6,24)
```

```
Lex (WHITESPACE,  ,16,1,0)
Lex (WHITESPACE,  ,16,2,1)
Lex (WHITESPACE,  ,16,3,2)
Lex (WHITESPACE,  ,16,4,3)
Lex (FLOAT, float,16,5,4)
Lex (WHITESPACE,  ,16,6,9)
Lex (IDENTIFIER_1, pi,16,7,10)
Lex (WHITESPACE,  ,16,8,12)
Lex (IGUAL, =,16,9,13)
Lex (WHITESPACE,  ,16,10,14)
Lex (NUMBER, 3,16,11,15)
Lexical error: .
Lex (NUMBER, 1416,16,12,17)
Lex (WHITESPACE_N, \n,16,13,21)
--------------------------------------------------------------------------------
Lex (WHITESPACE,  ,17,1,0)
Lex (WHITESPACE,  ,17,2,1)
Lex (WHITESPACE,  ,17,3,2)
Lex (WHITESPACE,  ,17,4,3)
Lex (CHAR, char,17,5,4)
Lex (WHITESPACE,  ,17,6,8)
Lex (IDENTIFIER_1, linea,17,7,9)
Lex (LBRACKET, [,17,8,14)
Lex (NUMBER, 1024,17,9,15)
Lex (RBRACKET, ],17,10,19)
Lex (PUNTOCOMA, ;,17,11,20)
Lex (WHITESPACE_N, \n,17,12,21)
--------------------------------------------------------------------------------
Lex (WHITESPACE,  ,18,1,0)
Lex (WHITESPACE,  ,18,2,1)
Lex (WHITESPACE,  ,18,3,2)
Lex (WHITESPACE,  ,18,4,3)
Lex (IDENTIFIER_1, FILE,18,5,4)
Lex (WHITESPACE,  ,18,6,8)
Lex (TIMES, *,18,7,9)
Lex (IDENTIFIER_1, fich,18,8,10)
Lex (PUNTOCOMA, ;,18,9,14)
Lex (WHITESPACE_N, \n,18,10,15)
--------------------------------------------------------------------------------
Lex (WHITESPACE,  ,19,1,0)
Lex (WHITESPACE,  ,19,2,1)
Lex (WHITESPACE,  ,19,3,2)
Lex (WHITESPACE,  ,19,4,3)
Lex (FLOAT, float,19,5,4)
Lex (WHITESPACE,  ,19,6,9)
Lex (IDENTIFIER_1, matrizA,19,7,10)
Lex (LBRACKET, [,19,8,17)
Lex (IDENTIFIER_1, FILAS_MATRIZ_A,19,9,18)
Lex (RBRACKET, ],19,10,32)
Lex (LBRACKET, [,19,11,33)
Lex (IDENTIFIER_1, COLUMNAS_MATRIZ_A,19,12,34)
Lex (RBRACKET, ],19,13,51)
Lex (PUNTOCOMA, ;,19,14,52)
Lex (WHITESPACE_N, \n,19,15,53)
--------------------------------------------------------------------------------
```

```
Lex (WHITESPACE,  ,30,1,0)
Lex (WHITESPACE,  ,30,2,1)
Lex (WHITESPACE,  ,30,3,2)
Lex (WHITESPACE,  ,30,4,3)
Lex (INT, int,30,5,4)
Lex (WHITESPACE,  ,30,6,7)
Lex (IDENTIFIER_1, c,30,7,8)
Lex (WHITESPACE,  ,30,8,9)
Lex (IGUAL, =,30,9,10)
Lex (WHITESPACE,  ,30,10,11)
Lex (NUMBER, 0,30,11,12)
Lex (PUNTOCOMA, ;,30,12,13)
Lex (WHITESPACE_N, \n,30,13,14)
--------------------------------------------------------------------------------
Lex (WHITESPACE,  ,31,1,0)
Lex (WHITESPACE,  ,31,2,1)
Lex (WHITESPACE,  ,31,3,2)
Lex (WHITESPACE,  ,31,4,3)
Lex (KEY_WHILE, while,31,5,4)
Lex (LPAREN, (,31,6,9)
Lex (IDENTIFIER_1, fgets,31,7,10)
Lex (LPAREN, (,31,8,15)
Lex (IDENTIFIER_1, linea,31,9,16)
Lex (COMMA, ,,31,10,21)
Lex (WHITESPACE,  ,31,11,22)
Lex (NUMBER, 1024,31,12,23)
Lex (COMMA, ,,31,13,27)
Lex (WHITESPACE,  ,31,14,28)
Lex (LPAREN, (,31,15,29)
Lex (IDENTIFIER_1, FILE,31,16,30)
Lex (TIMES, *,31,17,34)
Lex (RPAREN, ),31,18,35)
Lex (WHITESPACE,  ,31,19,36)
Lex (IDENTIFIER_1, fich,31,20,37)
Lex (RPAREN, ),31,21,41)
Lex (RPAREN, ),31,22,42)
Lex (WHITESPACE,  ,31,23,43)
Lex (LBLOCK, {,31,24,44)
Lex (WHITESPACE_N, \n,31,25,45)
--------------------------------------------------------------------------------
```

- NOTA: El resto de las imágenes de los resultados junto con el código fuente se encuentra en el repositorio de github:

- https://github.com/fjlic/Compiladores_Librerias/blob/Tarea3