INSTITUTO TECNOLÓGICO DE MORELIA

División De Estudios Profesionales

Departamento De Sistemas Y Computación

LENGUAJES Y AUTÓMATAS II

Unidad I: Análisis semántico.

PRÁCTICA 2: ANALIZADOR LÉXICO DEL LENGUAJE PYTHON

PROFESOR

BALLESTEROS LEÓN JAVIER

ALUMNOS

Ascencio Flores Ricardo - 15120269

Gil Jacome Arturo - 14121426

Martínez Aguilar Diego Ulises - 15121185

MORELIA, MICHOACÁN, MÉXICO

Jueves, 31 de enero del 2019

Tabla de contenido

INTRODUCCIÓN

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, *http://www.python.org/*, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables.

Figura I1: Logo Python.

Modo interactivo: el intérprete de Python estándar incluye un modo interactivo en el cual se escriben las instrucciones en una especie de intérprete de comandos: las expresiones pueden ser introducidas una a una, pudiendo verse el resultado de su evaluación inmediatamente, lo que da la posibilidad de probar porciones de código en el modo interactivo antes de integrarlo como parte de un programa. Esto resulta útil tanto para las personas que se están familiarizando con el lenguaje como para los programadores más avanzados.

PRÁCTICA NO.1

CREAR UN ANALIZADOR LÉXICO PARA EL LENGUAJE DE PYTHON.

DESARROLLO DE LA PRÁCTICA

Practica 2.jj | Prueba.txt | Comandos.txt

```
options { Ignore_Case = false; }

PARSER_BEGIN (Practica2)
public class Practica2 {
  //Ejecución del analizador
  public static void main (String[] argum) throws ParseException {
    //Inicialización del analizador
    Practica2 anLexSint = new Practica2 (System.in);
    anLexSint.expression();
    System.out.println("Analisis terminado");
  }
}

PARSER_END (Practica2)

```

Figura 1: Se crea un archivo nuevo con extensión **.jj** y se establece la estructura necesaria para la ejecución e inicialización del analizador.

```
//Tokens estáticos
TOKEN:
{
  <RESERVADA: ("and" | "as" | "assert" | "async" |"await" |"break" |"class" |"continue" |"def" |"del" |"elif" |"else" |"except" |"exec" |
    "False" |"finally" |"for" |"from" |"global" |"if" | "object" | "self" |"import" |"in" |"is" | "int" |"lambda" |"None" |"nonlocal" |
    "not" |"or" |"pass" |"print" | "raise" |"return" |"True" |"try" |"while" |"with" |"yield") > {System.out.println("Palabra reservada");} |
  <TIPOS: "int" | "float" | "complex" | "bool" | "str" | "type" | "format" > {System.out.println("Nombre de funcion");} |
  <CADENA: ("\""(~["\r"])*"\"" | "\'"(~["\r"])*"\'")> {System.out.println("Cadena");}|
  <BINARIO: "0b"(["0"-"1"])+> {System.out.println("Binario");}|
  <CONSTRUCTOR: "__init__"> {System.out.println("Definicion de constructor");}|
  <OCTAL: "0o"(["0"-"7"])+> {System.out.println("Octal");} |
  <HEX: "0x"(<NUM>| ["A"-"F"]| ["a"-"f"])+> {System.out.println("Hexadecimal");} |
  <SALTO:"\n"> {System.out.println("Salto de linea");} |
  <COMENTARIO: "#"(~["\r", "\n"])*> {System.out.println("Comentario");}|
  <NUM:  (["0"-"9"])+> {System.out.println("Numero");} |
  <ID: (<LETRAMIN>|<LETRAMAY>)(<LETRAMIN>|<LETRAMAY>|["0"-"9"]|"_"){0,63}> {System.out.println("Identificador");} |
  <LETRAMIN: ["a"-"z"]> {System.out.println("Letra Minuscula");} |
  <LETRAMAY: ["A"-"Z"]> {System.out.println("Letra Mayuscula");} |
  <GUION: "_"> {System.out.println("Guion bajo");} |
  <ABRIRPARENTESIS:"("> {System.out.println("Abrir Parentesis");} |
  <CERRARPARENTESIS:")"> {System.out.println("Cerrar Parentesis");} |
  <ABRIRLLAVE:"{"> {System.out.println("Abrir Llave");} |
  <CERRARLLAVE:"}"> {System.out.println("Cerrar Llave");} |
  <COMPARACION: "=="> {System.out.println("Operador comparacion");} |
  <ASIGNACION: "="> {System.out.println("Operador asignacion");} |
  <OPSUMA: "+"> {System.out.println("Operador Suma");} |
  <OPRESTA: "-"> {System.out.println("Operador Resta");} |
  <OPPOTENCIA:"**"> {System.out.println("Operador Potencia");} |
  <OPMULTIPLICACION:"*"> {System.out.println("Operador multiplicacion");} |
  <NEGACION:"!"> {System.out.println("Operador negacion");} |
  <OPLOGAND:"&"> {System.out.println("Operador logico AND");} |
  <OPLOGOR:"|"> {System.out.println("Operador logico OR");} |
  <OPLOGXOR:"^"> {System.out.println("Operador logico XOR");} |
  <OPDIVISION:"/"> {System.out.println("Operador Division");} |
  <DP: ":"> {System.out.println("Dos Puntos");} |
  <PUNTO: "."> {System.out.println("Punto");} |
  <OPMODULO: "%"> {System.out.println("Operador modulo");} |
  <COMA: ","> {System.out.println("Coma");} |
  <MOVDER: ">>"> {System.out.println("Mover bits a la derecha");} |
  <MOVIZQ: "<<"> {System.out.println("Mover bits a la izquierda");} |
  <MAYORQUE: ">"> {System.out.println("Mayor que");} |
  <MENORQUE: "<"> {System.out.println("Menor que");} |
  <TABULADOR:"\t"> {System.out.println("Tabulador");}
```

Figura 2: Se establecen los tokens estáticos que nos permitirán reconocer los los lexemas establecidos.

```
61  }
62
63  //Estructuras y caracteres de escape
64  SKIP:
65  { " " | "\r"}
66
67  //Unidad principal
68  void expression() : {}
69  {
70    (<CONSTRUCTOR>|<BINARIO>|<OCTAL>|<HEX>|<TIPOS>|<OPLOGAND>|<OPLOGOR>|<OPLOGXOR>|<DP>|<MAYORQUE>|<MENORQUE>|
71      <PUNTO>|<OPMODULO>|<COMA>|<RESERVADA>|<COMENTARIO>|<ID>|<NUM>|<LETRAMIN>|<LETRAMAY>|<ABRIRPARENTESIS>|
72      <CERRARPARENTESIS>|<ABRIRLLAVE>|<CERRARLLAVE>|<OPSUMA>|<OPRESTA>|<OPPOTENCIA>|<NEGACION>|<OPMULTIPLICACION>|
73      <OPDIVISION>|<GUION>|<SALTO>|<TABULADOR>|<COMPARACION>|<ASIGNACION>|<CADENA>)+ <EOF>
74  }
75
```

Figura 3: Se establecen las estructuras de escape, así como también se indica la unidad principal como aquella que va a aceptar la gramática establecida.

```
C:\Users\DiegoUlises\Documents\8vo Semestre\Automatas II\Unidad 1\Practica 2>javacc "Practica 2.jj"
Java Compiler Compiler Version 5.0 (Parser Generator)
(type "javacc" with no arguments for help)
Reading from file Practica 2.jj . . .
File "TokenMgrError.java" is being rebuilt.
File "ParseException.java" is being rebuilt.
File "Token.java" is being rebuilt.
File "SimpleCharStream.java" is being rebuilt.
Parser generated successfully.

C:\Users\DiegoUlises\Documents\8vo Semestre\Automatas II\Unidad 1\Practica 2>javac *.java

C:\Users\DiegoUlises\Documents\8vo Semestre\Automatas II\Unidad 1\Practica 2>
```

Figura 4: Para compilar este fichero, se debe hacer con "javacc" y posteriormente con "javac".

Practica 2.jj | Prueba.txt | Comandos.txt

```
1   # Python program to check if the input number is odd or even.
2   # A number is even if division by 2 give a remainder of 0.
3   # If remainder is 1, it is odd number.
4
5   num = int(input("Enter a number: "))
6   if (num % 2) == 0:
7      print("{0} is Even".format(num))
8   else:
9      print("{0} is Odd".format(num))
10
11
12  class InputOutString(object):
13      def __init__(self):
14          self.s = ""
15
16      def getString(self):
17          self.s = raw_input()
18
19      def printString(self):
20          print self.s.upper()
21
22  strObj = InputOutString()
23  strObj.getString()
24  strObj.printString()
```

```
# Python 3: Fibonacci series up to n
def fib(n):
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
fib(1000)

### Introducir un nuumero por teclado y decir si es par o impar
h = input('Introduzca un numero: ')
if h%2 == 0:
    print 'Este numero es par'
else:
    print 'Este numero es impar'

##Introducir los valores A, B y C:
###Si A/B>30, calcular e imprimir A/C * B^3
###Si A/B<=30, calcular e imprimir 2^2+4^2+6^2+...+30^2
a = input('Primer valor: ')
b = input('Segundo valor: ')
c = input('Tercer valor: ')
n = 2
suma = 0
sumas = 0
if a/b > 30:
    suma = a/c*b**3
    print suma
if a/b <= 30:
    while n <= 30:
        sumas += n**2
        n += 2
    print sumas
```

Figura 5: En un archivo de prueba se crea con extensión **.txt** el cual será evaluado por el analizador léxico antes creado, tal archivo contiene ejemplos en desorden de cada uno de los lexemas.

```
C:\Users\DiegoUlises\Documents\8vo Semestre\Automatas II\Unidad 1\Practica 2>java Practica2 < Prueba.txt
Comentario
Salto de linea
Comentario
Salto de linea
Comentario
Salto de linea
Salto de linea
Identificador
Operador asignacion
Palabra reservada
Abrir Parentesis
Identificador
Abrir Parentesis
Cadena
Cerrar Parentesis
Cerrar Parentesis
Salto de linea
Palabra reservada
Abrir Parentesis
Identificador
Operador modulo
Numero
Cerrar Parentesis
Operador comparacion
Numero
Dos Puntos
Salto de linea
Palabra reservada
Abrir Parentesis
Cadena
Punto
Nombre de funcion
Abrir Parentesis
Identificador
Cerrar Parentesis
Cerrar Parentesis
Salto de linea
Palabra reservada
Dos Puntos
Salto de linea
Palabra reservada
Abrir Parentesis
```

Figura 6: Para ejecutar el programa y dentro del símbolo del sistema con los comandos de **java Practica1 < prueba.txt** con el cual comprobamos el funcionamiento del analizador léxico, no trata de comprobar las estructuras del lenguaje ni compilará el fichero en código binario.

CONCLUSIONES

El analizador léxico es una parte fundamental de un compilador, ya que se dedica a verificar que todas las palabras y símbolos pertenecen al lenguaje cuyo código se está tratando de compilar. En el caso de Python hay ciertas características que lo hacen diferente a la gran mayoría de lenguajes de programación, por ejemplo, la carencia del **;** además de que no hay necesidad de declarar los tipos de datos de las variables a utilizar, por lo que el análisis léxico para éste se vuelve de cierta forma mas simple al tener menos elementos a reconocer.

REFERENCIAS

[1] https://www.python.org/

[2] https://www.comoprogramar.org/ejemplos-de-python-codigo/

[3] http://www.pythondiario.com/p/ejercicios-de-programacion-python.html