Analizador Léxico, Sintáctico Y Semántico en Python.

Gramatica.

- programa ⇨ declaraciones
- declaraciones ⇨ declaraciones declaracion | declaracion
- declaracion ⇨ IDENTIFICADOR = expresion
- expresion ⇨ expresion + termino | expresion - termino | termino
- termino ⇨ termino * factor | termino / factor | factor
- factor ⇨ ENTERO | IDENTIFICADOR | (expresion)

Códico Fuente.

https://github.com/Gilberto-Guzman/Analizador_Lexico_Sintactico_Y_Semantico/blob/master/analizador.py

Descripción.

Este es un pequeño programa que implementa un analizador léxico, un analizador sintáctico y un analizador semántico. Su objetivo es analizar una serie de declaraciones de asignación de variables en un lenguaje de programación muy simple.

El lenguaje de programación que se está analizando consiste únicamente en declaraciones de asignación de variables, con valores enteros, y operaciones aritméticas básicas (suma, resta, multiplicación y división).

La sintaxis del lenguaje es muy simple: las declaraciones comienzan con un identificador (que puede ser una sola letra o una cadena de letras) seguido de un signo igual (=) y un valor entero. El programa puede contener comentarios en cualquier lugar del código, precedidos por el carácter #.

El programa consta de tres clases:

La clase Token representa los tokens que se generan en el análisis léxico. Cada token tiene un tipo (ENTERO, IDENTIFICADOR, +, -, *, /, =) y un valor (para los tokens ENTERO e IDENTIFICADOR). La clase Lexer implementa el analizador léxico. Se encarga de recorrer el código fuente y generar una lista de tokens. La clase Analizador implementa el analizador sintáctico. Se encarga de recorrer la lista de tokens y comprobar que se corresponden con la sintaxis del lenguaje. También se encarga de generar una estructura de árbol que representa la expresión aritmética. La clase AnalizadorSemantico implementa el analizador semántico. Se encarga de comprobar que todas las variables que se utilizan en el programa han sido declaradas previamente. El programa en sí mismo es bastante simple: después de definir dos variables booleanas Test_Analizador_Sintactico y Test_Analizador_Semantico (inicialmente ambas a True), define una cadena de código fuente que contiene dos declaraciones de asignación de variables. A continuación, crea un objeto Lexer y un objeto Analizador, y los utiliza para analizar el código fuente. Finalmente, crea un objeto AnalizadorSemantico y lo utiliza para realizar un análisis semántico del programa.

Si durante el análisis sintáctico o semántico se encuentra algún error, las variables booleanas Test_Analizador_Sintactico y Test_Analizador_Semantico se establecen en False, y se muestra un mensaje de error por pantalla.

Ejemplos.

- Analisis Sintactico Exitoso
- Analisis Semantico Exitoso x = 2 y = x + 3 z = (x + y) * 4
- Se esperaba un token de tipo =
- Analisis Semantico Exitoso x = 2 y { x + 3 z = (x + y) * 4
- Variable no declarada: z
- Analisis Sintactico Exitoso x = 2 y = z + 3 z = (x + y) * 4