INSTITUTO TECNOLOGICO DE PACHUCA

"El hombre alimenta el ingenio en contacto con la Ciencia"

4.2 Documentación ER y AF para Analizador Léxico

INGENIERIA EN SISTEMAS COMPUTACIONALES

NOMBRE DE LA ASIGNATURA

SISTEMAS OPERATIVOS

INTEGRANTES DEL EQUIPO

Martin Feria Vázquez

21200594

Ramírez Hernández Josué

21200990

Valdez Zuñiga Leonardo Vicente

24200196

Zeron López Germán Eduardo

21200642

PROFESOR DE LA MATERIA

Ing. Rodolfo Baumé Lazcano

PACHUCA, HIDALGO, 14 DE MAYO DEL 2024

Introducción:

El propósito de este trabajo es desarrollar un analizador léxico que identifique y categorice diferentes componentes léxicos (tokens) en un lenguaje de programación en español. Este enfoque puede ser beneficioso en entornos educativos y comunidades de programación hispanohablantes. Python es elegido como base debido a su simplicidad y legibilidad, lo que facilita la implementación y comprensión del analizador.

Tabla de Tokens

Para comenzar, hemos definido una serie de tokens que nuestro analizador léxico debe reconocer. Estos tokens se clasifican en diferentes categorías: palabras clave, identificadores, números, operadores, símbolos especiales y comentarios. A continuación se presenta la tabla de tokens con su tipo y expresión regular correspondiente.

Token	Tipo	Expresión Regular	Explicación
`si`	PALABRA_CLAVE	`\bsi\b`	Coincide con "si", utilizado para condicionales (equivalente a "if").
`sino`	PALABRA_CLAVE	`\bsino\b`	Coincide con "sino", utilizado para condicionales alternativos (equivalente a "else").
`mientras`	PALABRA_CLAVE	`\bmientras\b`	Coincide con "mientras", utilizado para bucles (equivalente a "while").
`para`	PALABRA_CLAVE	`\bpara\b`	Coincide con "para", utilizado para bucles (equivalente a "for").
`función`	PALABRA_CLAVE	`\bfunción\b`	Coincide con "función", utilizado para definir funciones (equivalente a "def").
`retorno`	PALABRA_CLAVE	`\bretorno\b`	Coincide con "retorno", utilizado para devolver valores de funciones (equivalente a "return").
`clase`	PALABRA_CLAVE	`\bclase\b`	Coincide con "clase", utilizado para definir clases (equivalente a "class").
`importar`	PALABRA_CLAVE	`\bimportar\b`	Coincide con "importar", utilizado para importar módulos (equivalente a "import").
`verdadero`	PALABRA_CLAVE	`\bverdadero\b`	Coincide con "verdadero", valor booleano verdadero (equivalente a "true").
`falso`	PALABRA_CLAVE	`\bfalso\b`	Coincide con "falso", valor booleano falso (equivalente a "false").
Identificador	IDENTIFICADOR	`[a-zA-Z_][a-zA-Z0-9_]*`	Coincide con nombres de variables y funciones.
Número	NUMERO	`\d+(\.\d+)?`	Coincide con números enteros y decimales.

Operadores	OPERADOR	`==	!=
Símbolos especiales	SIMBOLO_ESPECIAL	`[{}();,]`	Coincide con símbolos de puntuación.
Comentario	COMENTARIO	`#.*`	Coincide con comentarios.
Espacios en blanco	None	`\s+`	Coincide con espacios en blanco.

1. Palabras Clave

Las palabras clave son términos reservados en el lenguaje que no pueden usarse como identificadores.

Token	Tipo	Expresión Regular
`si`	PALABRA_CLAVE	`\bsi\b`
`sino`	PALABRA_CLAVE	`\bsino\b`
`mientras`	PALABRA_CLAVE	`\bmientras\b`
`para`	PALABRA_CLAVE	`\bpara\b`
`función`	PALABRA_CLAVE	`\bfunción\b`
`retorno`	PALABRA_CLAVE	`\bretorno\b`
`clase`	PALABRA_CLAVE	`\bclase\b`
`importar`	PALABRA_CLAVE	`\bimportar\b`
`verdadero`	PALABRA_CLAVE	`\bverdadero\b`
`falso`	PALABRA_CLAVE	`\bfalso\b`

La expresión regular `\bpalabra\b` asegura que la palabra clave no sea parte de otra palabra, delimitada por límites de palabra (`\b`).

2. Identificadores

Los identificadores son nombres para variables, funciones y otros elementos definidos por el usuario. Deben comenzar con una letra o un guion bajo y pueden contener letras, números y guiones bajos.

Token	Tipo	Expresión Regular
Identificador	IDENTIFICADOR	`[a-zA-Z_][a-zA-Z0-9_]*`

3. Números

Los números pueden ser enteros o decimales. La expresión regular captura ambos formatos.

Token	Tipo	Expresión Regular
Número	NUMERO	`\d+(\.\d+)?`

La expresión regular `\d+(\.\d+)?` permite coincidir con números enteros (`\d+`) y decimales (`\d+\.\d+`).

4. Operadores

Los operadores realizan operaciones sobre los operandos. Incluimos operadores aritméticos y de comparación.

Token	Tipo	Expresión Regular
Operadores	OPERADOR	`==`

5. Símbolos Especiales

Los símbolos especiales incluyen caracteres como paréntesis, llaves, punto y coma, y comas.

Token	Tipo	Expresión Regular
Símbolos especiales	SIMBOLO_ESPECIAL	`[{}();,]`

6. Comentarios

Los comentarios son ignorados por el compilador o intérprete y se utilizan para anotaciones.

Token	Tipo	Expresión Regular
Comentario	COMENTARIO	`#.*`

La expresión regular `#.*` captura cualquier cosa desde el símbolo `#` hasta el final de la línea.

7. Espacios en Blanco

Los espacios en blanco se ignoran durante el análisis léxico, pero son necesarios para separar tokens.

Token	Tipo	Expresión Regular
Espacios en blanco	None	`\s+`

Conclusión:

La implementación del analizador léxico en Python proporciona una sólida base para el reconocimiento de tokens en un lenguaje de programación en español. Utilizando expresiones regulares, se pueden identificar palabras clave, identificadores, números, operadores, símbolos especiales y comentarios, lo que permite analizar un código fuente y convertirlo en una secuencia de tokens para su posterior procesamiento por un analizador sintáctico y semántico.