Universidad de Guadalajara

Centro Universitario

De

Ciencias Exactas

E

Ingenierías

(CUCEI)

Materia:

SEMINARIO DE SOLUCION DE PROBLEMAS DE

TRADUCTORES DE LENGUAJES II

Nombre: Aceves López Jonathan

Código:

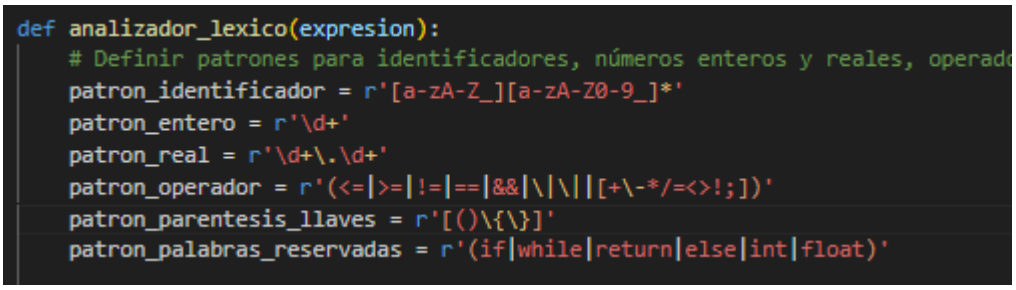
217481363

Este código implementa un simple analizador léxico utilizando la biblioteca re para expresiones regulares y tkinter para la interfaz gráfica. El analizador léxico se encarga de identificar y clasificar tokens en una expresión ingresada por el usuario.

Funcionalidad del Analizador Léxico:

1.- Definición de Patrones:

En esta sección, se definen patrones utilizando expresiones regulares para identificadores, números enteros y reales, operadores, paréntesis, llaves y palabras reservadas.



```
def analizador_lexico(expresion):
    # Definir patrones para identificadores, números enteros y reales, operad
    patron_identificador = r'[a-zA-Z_][a-zA-Z0-9_]*'
    patron_entero = r'\d+'
    patron_real = r'\d+\.\d+'
    patron_operador = r'(<=|>=|!=|==|&&|\|\||[+\-*/=<>!;])'
    patron_parentesis_llaves = r'[()\{\}]'
    patron_palabras_reservadas = r'(if|while|return|else|int|float)'
```

- patron_identificador: Identifica cadenas que siguen las reglas de un identificador válido.
- patron_entero: Identifica secuencias de dígitos que representan números enteros.
- patron_real: Identifica números reales en formato decimal.
- patron_operador: Identifica operadores como <=, >=, !=, ==, &&, ||, y otros operadores aritméticos y lógicos.
- patron_parentesis_llaves: Identifica paréntesis y llaves.
- patron_palabras_reservadas: Identifica palabras reservadas como if, while, return, else, int, float.
- patron_total: Combina todos los patrones en una expresión regular total.
- Estos patrones se utilizarán posteriormente para identificar tokens en la expresión ingresada.

2: Expresión Regular Total y Obtención de Tokens

```
# Encontrar todos los tokens en la expresión
tokens = re.findall(patron_total, expresion)
```

Se utiliza re.findall para encontrar todos los tokens en la expresión según el patrón total definido previamente.

Sección 3: Procesamiento de Tokens y Tipos de Datos

```
# Procesar los tokens y construir el resultado
resultado = ""
for token in tokens:
    for i in range(len(token)):
        if token[i]:
            tipo_dato = obtener_tipo_dato(i, token[i])
            resultado += f'{token[i]} tipo de dato: {tipo_dato}\n'
            break
```

Los tokens se procesan en un bucle, y se utiliza la función obtener_tipo_dato para determinar su tipo de dato.

El resultado incluye el valor del token junto con su tipo de dato.

La función obtener_tipo_operador toma como argumento un operador y devuelve una descripción adicional basada en el tipo de operador.

4.-Operadores Aritméticos:

```
def obtener_tipo_operador(operador):
    if operador == '+':
        return 'suma'
    elif operador == '-':
        return 'resta'
    elif operador == '*':
        return 'multiplicación'
    elif operador == '/':
        return 'división'
    elif operador == '=':
        return 'asignación'
    elif operador == '<':
        return 'menor que'
    elif operador == '>':
        return 'mayor que'
    elif operador == '<=':
        return 'menor o igual que'
    elif operador == '>=':
        return 'mayor o igual que'
    elif operador == '!=':
        return 'diferente de'
    elif operador == '==':
        return 'igual a'
    elif operador == '&&':
        return 'AND lógico'
    elif operador == '||':
        return 'OR lógico'
    elif operador == '!':
        return 'NOT lógico'
    elif operador == ';':
        return 'punto y coma'
    else:
        return 'desconocido'
```

+: 'suma'

-: 'resta'

*: 'multiplicación'

/: 'división'

Operadores de Asignación y Comparación:

=: 'asignación'

<: 'menor que'

>: 'mayor que'

<=: 'menor o igual que'

>=: 'mayor o igual que'

!=: 'diferente de'

==: 'igual a'

Operadores Lógicos:

&&: 'AND lógico'

||: 'OR lógico'

!: 'NOT lógico'

Otros:

;: 'punto y coma'

Si el operador no coincide con ninguno de los anteriores, se devuelve 'desconocido'.

Esta función se utiliza para proporcionar descripciones adicionales para los operadores en la salida del analizador léxico.

Sección 6: Interfaz Gráfica (Tkinter)

```
# Crear la ventana principal
ventana = tk.Tk()
ventana.title("Analizador Léxico")

# Crear elementos de la interfaz
etiqueta = tk.Label(ventana, text="Ingrese la expresión:")
etiqueta.pack(pady=10)

entrada = tk.Entry(ventana, width=30)
entrada.pack(pady=10)

boton_analizar = tk.Button(ventana, text="Analizar", command=analizar_expresion)
boton_analizar.pack(pady=10)

resultado_text = tk.Text(ventana, height=10, width=40, state=tk.DISABLED)
resultado_text.pack(pady=10)
```

Se crea una ventana principal y elementos de interfaz como etiquetas, entrada, botón y área de texto usando la biblioteca Tkinter.

Prueba: