UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 “M”

COMPILADORES

SUBCOMPETENCIA I.- ANÁLISIS LÉXICO

ACT 1.3 PRÁCTICA I. UNIDAD 1. EJERCICIOS LÉXICOS. - REALIZA EL SIGUIENTE EJERCICIO LÉXICO PYTHON

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
MARTES, 22 DE AGOSTO DE 2023

Analizador Léxico

A continuación, se muestra el código en python del Analizador Léxico

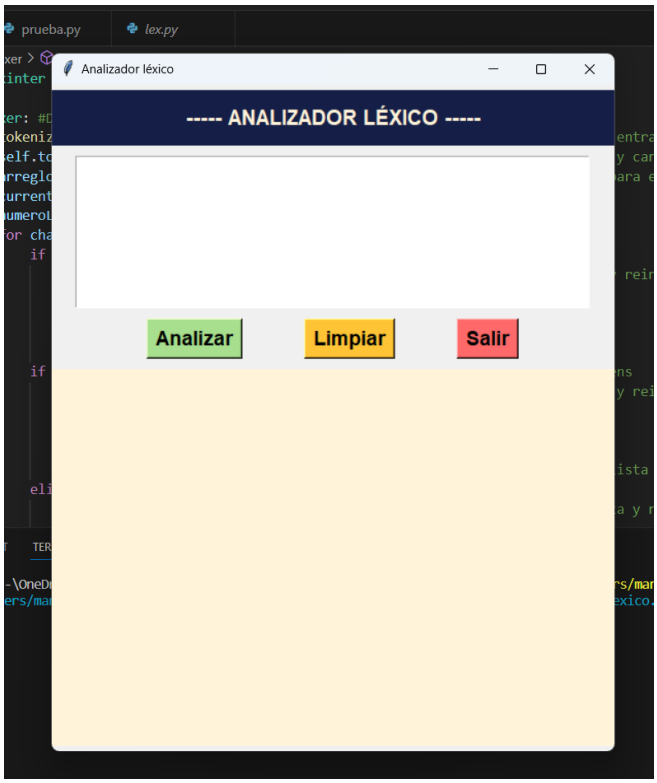
Class Lexer

```python
import tkinter as tk #importa la biblioteca tkinter

class Lexer: #Define una clase llamda Lexer para realizar el analisis lexico
    def tokenize(self, text): #Define un metodo llamado tokenize que toma el texto de entrada y divide en tokens
        self.tokens = ['for', 'do', 'while', 'if', 'else', '(', ')'] #Lista de tokens y caracteres especiales
        arreglo = [] #Crea una lista vacia para almacenar los tokens y otra variable para el token actual
        current_token = ""
        numeroLinea = 1   #Inicializa el nimero de linea en 1
        for char in text: #Itera a traves de cada caracter en el texto de entrada
            if char == '\n': #Verifica si el carácter es un salto de línea
                if current_token != "": #Si hay un token actual, agrégalo a la lista y reinicia el token actual
                    arreglo.append((current_token, numeroLinea))
                    current_token = ""
                numeroLinea += 1  #Incrementa el número de línea
                continue   #Salta al siguiente caracter en la iteracion
            if char in self.tokens:  #Verifica si el carácter está en la lista de tokens
                if current_token != "":  #Si hay un token actual, agrégalo a la lista y reinicia el token actual
                    arreglo.append((current_token, numeroLinea))
                    current_token = ""

                arreglo.append((char, numeroLinea))  #Agrega el carácter actual a la lista como un token
            elif char.isspace(): #Verifica si el carácter es un espacio en blanco
                if current_token != "":  ## Si hay un token actual, agrégalo a la lista y reinicia el token actual
                    arreglo.append((current_token,numeroLinea))
                    current_token = ""
            else:
                current_token += char   #Agrega el carácter actual al token actual
        if current_token != "":  #Si hay un token actual al final, agrégalo a la lista
            arreglo.append((current_token,numeroLinea))
        return arreglo #Retorna la lista de tokens

    def analyze(self, text):  #Define un método llamado "analyze" que toma el texto de entrada y realiza el análisis léxico
        arreglo = self.tokenize(text)  #Obtiene la lista de tokens utilizando el método "tokenize"
        result = ""  #Inicializa una cadena vacía para almacenar los resultados
        for token, numeroLinea in arreglo:  #Itera a través de cada token y número de línea en la lista de tokens
            if token in self.tokens:
                if token == "(":  #Verifica si el token es un paréntesis de apertura "("
                    result += f"Línea {numeroLinea} [ {token} ] Paréntesis De Apertura \n" #Agrega un mensaje de resultado para un paréntesis de apertura
                elif token == ")": #Verifica si el token es un paréntesis de cierre ")"
                    result += f"Línea {numeroLinea} [ {token} ] Paréntesis De Cierre \n" #Agrega un mensaje de resultado para un paréntesis de cierre
                else:
                    result += f"Línea {numeroLinea} ({token}) Palabra Reservada \n" # Agrega un mensaje de resultado para una palabra reservada
            else:
                result += f"Línea {numeroLinea} ({token}) Error Léxico \n" #Agrega un mensaje de resultado para un error léxico

        return result #Retorna la cadena de resultados
```

Class LexerApp

```python
class LexerApp:
    def __init__(self):  #Define una clase llamda LexerApp para la interfaz grafica de usuario
        self.windows = tk.Tk() #Crea una ventana de la clase
        self.windows.title("Analizador léxico") #Establece el titulo de la ventana

        #Crea una etiqueta para el titulo de la aplicacion
        self.text_label = tk.Label(text="----- ANALIZADOR LÉXICO -----", height=2, width=45, font=
("Arial", 15, 'bold'), fg="#FFF3DA", bg="#141E46")
        self.text_label.pack(pady=5)

        #Crea un cuadro de texto para la entrada de texto
        self.text_input = tk.Text(self.windows, height=8, width=55, font=("Arial", 12))
        self.text_input.pack(pady=5)

        #Crea un marco para los botones
        self.button_frame = tk.Frame(self.windows)
        self.button_frame.pack()

        #crea un boton para realizar el analisis lexico del texto de entrada
        self.analyze_button = tk.Button(self.button_frame, text="Analizar", command=self.analyze_text,
bg="#A8DF8E", font=("Arial", 14, 'bold'))
        self.analyze_button.grid(row=0, column=0, padx=30, pady=5)

        #crea un boton para limpiar el cuadro de texto
        self.clean_button = tk.Button(self.button_frame, text="Limpiar", command=self.clean_text,
bg="#FFC436", font=("Arial", 14, 'bold'))
        self.clean_button.grid(row=0, column=1, padx=30, pady=5)

        #Crea un boton para salir del programa
        self.exit_button = tk.Button(self.button_frame, text="Salir", command=self.exit_app,
bg="#FF6969", font=("Arial", 14, 'bold'))
        self.exit_button.grid(row=0, column=2, padx=30, pady=5)

        #Crea una etiqueta para mostrar los resultados
        self.result_label = tk.Label(self.windows, text="", height=20, width=60, bg="#FFF3DA", font=
("Arial", 12))
        self.result_label.pack(pady=5)

    def analyze_text(self):  #Define un metodo para realizar el analisis lexico del texto de entrada
        lexer = Lexer()     #crea una instancia de la clase Lexer
        text = self.text_input.get("1.0", "end") #Obtiene el texto de entrada del cuadro de texto
        result = lexer.analyze(text)  #Realiza el analisis lexico utilizando el metodo "analyze" de
Lexer
        self.result_label.config(text=result) #Configura la etiqueta de resultados con el resultado del
analisis

    def clean_text(self): #Define un metodo para limpiar el cuadro de texto y la etiqueta de resultados
        self.text_input.delete("1.0", "end") #Borra el contenido del cuadro del texto
        self.result_label.config(text="") #Limpia el contenido de la etiqueta de resultados

    def exit_app(self): #Define un metodo para salir del programa
        self.windows.destroy()

    def run(self): #Define un metodo para ejecutar la aplicacion de la interfaz grafica
        self.windows.mainloop() #inicia el bucle de la interfaz grafica

app = LexerApp() #Crea una instancia de la clase Lexer app
app.run()   #Ejecuta la aplicacion de la interfaz grafica llamando al metodo "Run"
```

Pruebas


prueba.py
lex.py
Analizador léxico
----- ANALIZADOR LÉXICO -----
Analizar
Limpiar
Salir


prueba.py
lex.py
Analizador léxico
----- ANALIZADOR LÉXICO -----
( ) marco do if
do
while else
if
for
do if else (
Analizar
Limpiar
Salir
Línea 1 [( ] Paréntesis De Apertura
Línea 1 [ )] Paréntesis De Cierre
Línea 1 (marco) Error Léxico
Línea 1 (do) Palabra Reservada
Línea 1 (if) Palabra Reservada
Línea 2 (do) Palabra Reservada
Línea 3 (while) Palabra Reservada
Línea 3 (else) Palabra Reservada
Línea 4 (if) Palabra Reservada
Línea 5 (for) Palabra Reservada
Línea 6 (do) Palabra Reservada
Línea 6 (if) Palabra Reservada
Línea 6 (else) Palabra Reservada
Línea 6 [( ] Paréntesis De Apertura