UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software

Act 1.3 Práctica I. Unidad 1. Ejercicios Léxico.- Realiza el siguiente ejercicio Léxico PYTHON

6 “M”

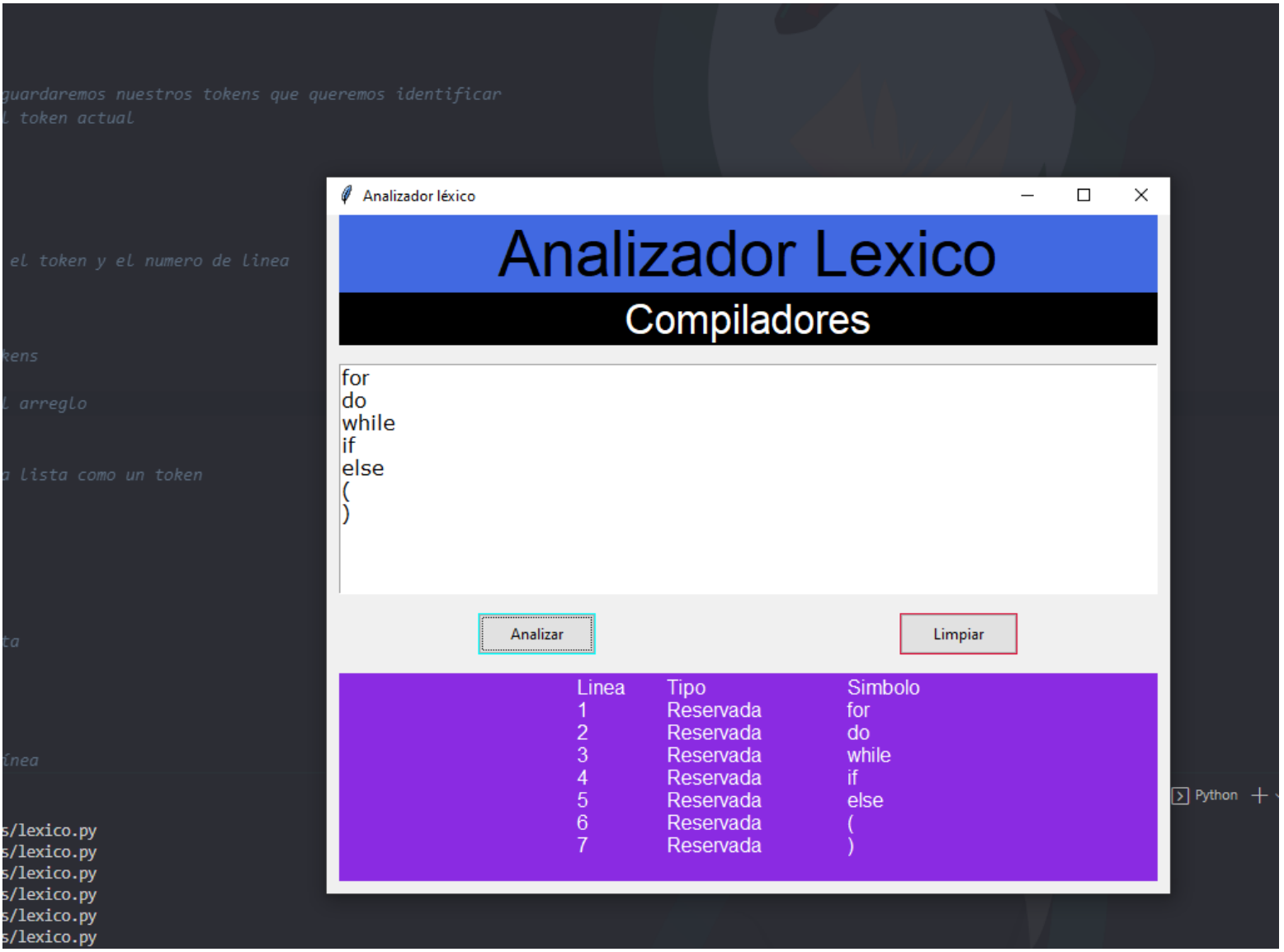
Materia: Compiladores

Docente: Luis Gutiérrez Alfaro

ALUMNO:
A 211387
Steven de Dios Montoya Hernández
GIT:

TUXTLA GUTIÉRREZ, CHIAPAS

Viernes, 25 de Agosto de 2023, 12:00


guardaremos nuestros tokens que queremos identificar
l token actual
el token y el numero de linea
kens
l arreglo
a lista como un token
ta
inea
s/lexico.py
s/lexico.py
s/lexico.py
s/lexico.py
s/lexico.py
s/lexico.py
Python
Analizador léxico
Analizador Lexico
Compiladores
for
do
while
if
else
(
)
Analizar
Limpiar
Linea Tipo Simbolo
1 Reservada for
2 Reservada do
3 Reservada while
4 Reservada if
5 Reservada else
6 Reservada (
7 Reservada)

```python
import tkinter as tk  # Importamos la librería tkinter
from tkinter import ttk

# Clase Lexer para analizar el texto y tokens
class Lexer:
    def tokenize(self, text):#Iniciamos el metodo tokenize
        self.tokens = ['for', 'do', 'while', 'if', 'else', '(', ')',"[","]"]#Aqui guardaremos nuestros tokens que queremos identificar
        arreglo = []#creamos un arreglo para agregar los tokens y variables para el token actual
        current_token = ""#El token actual
        numeroLinea = 1#Para ver el numero de linea
        for char in text:#Iteramos cada caracter del texto que hay entrada
            if char == '\n':#Verifica el salto de linea
                if current_token != "":#Si es diferente de vacio
                    arreglo.append((current_token, numeroLinea))#Agrega al arreglo el token y el numero de linea
                    current_token = ""#Reiniciamos el token
                numeroLinea += 1#Sumamos el numero de linea
                continue#continua
            if char in self.tokens:#Verifica si el carácter está en la lista de tokens
                if current_token != "":#El token actual si es diferente
                    arreglo.append((current_token, numeroLinea))#agrega el token al arreglo
                    current_token = ""#Token actual lo pone en vacio

                arreglo.append((char, numeroLinea)) #Agrega el carácter actual a la lista como un token
            elif char.isspace():#Si no el caracter es espacio en blanco
                if current_token != "":#Si hay un token, lo agregara al arreglo
                    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):
        arreglo = self.tokenize(text)  # Obtener la lista de tokens y números de línea
        result = "Linea\tTipo\t\tSimbolo\n"  # Encabezado del resultado
        for token, numeroLinea in arreglo:
            result += f"{numeroLinea}\t"
            if token in self.tokens:
                result += f"Reservada\t{token}\n"  # Token es una palabra reservada
            else:
                result += f"Error Léxico\t{token}\n"  # Token es un error léxico

        return result#Retorna la cadena de resultados

# Clase LexerApp para la interfaz de usuario
```

```
class LexerApp:
    def __init__(self):
        self.windows = tk.Tk()  # Crear la ventana principal
        self.windows.title("Analizador léxico")  # Establecer título de la ventana

        # Etiqueta de título
        self.text_label = ttk.Label(self.windows, text="Analizador Lexico", font=
("Ubuntu Medium", 38), anchor="center", background="#4169E1")
        self.text_label.grid(row=0, column=0, columnspan=2, padx=10, pady=0,
sticky="ew")

        # Etiqueta de subtítulo
        self.subtitle_label = ttk.Label(self.windows, text="Compiladores", font=
("Ubuntu Medium", 25), anchor="center", foreground="#ffffff", background="#000000")
        self.subtitle_label.grid(row=1, column=0, columnspan=2, padx=10, pady=(0,15),
sticky="ew")

        # Área de entrada de texto
        self.text_input = tk.Text(self.windows, height=10, width=65, font=("Verdana",
12))
        self.text_input.grid(row=2, column=0, columnspan=2, padx=10)

        # Botón "Analizar"
        self.analyze_button = ttk.Button(self.windows, text="Analizar",
command=self.analyze_text, style="Analyze.TButton")
        self.analyze_button.grid(row=3, column=0, padx=4, pady=(15, 5))

        # Botón "Limpiar"
        self.clean_button = ttk.Button(self.windows, text="Limpiar",
command=self.clean_text, style="Clean.TButton")
        self.clean_button.grid(row=3, column=1, padx=4, pady=(15, 5))

        # Etiqueta de resultado
        self.result_label = ttk.Label(self.windows, text="", font=("Ubuntu Medium",
12), background="#8A2BE2", foreground="#FFFFFF", anchor="center")
        self.result_label.grid(row=4, column=0, columnspan=3, padx=10, pady=10,
sticky="ew")

        # Configuración de estilos
        self.style = ttk.Style()
        self.style.configure("Analyze.TButton", background="#00FFFF", padding=(10, 5))
        self.style.configure("Clean.TButton", background="#DC143C", padding=(10, 5))
```

```python
# Método para analizar el texto ingresado
    def analyze_text(self):
        lexer = Lexer()
        text = self.text_input.get("1.0", "end")
        result = lexer.analyze(text)
        self.result_label.config(text=result)
```

```python
    # Método para limpiar el área de texto y resultado
    def clean_text(self):
        self.text_input.delete("1.0", "end")
        self.result_label.config(text="")
```

```
    # Método para ejecutar la interfaz
    def run(self):
        self.windows.mainloop()  # Iniciar el bucle principal de la interfaz

# Crear una instancia de la clase LexerApp y ejecutar la interfaz
app = LexerApp()
app.run()
```