



Universidad Autónoma de Chiapas
Facultad de contaduría y administración C-I



Carrera:

Lic. En ing en desarrollo y tecnologías de software.

Materia:

Compiladores.

Catedrático:

Mtro. Luis Gutiérrez Alfaro.

Nombre del alumno:

González Aguilar Eduardo - A211154

Semestre: 6. **Grupo:** M.

Nombre de la actividad:

Ejercicios léxico de expresiones regulares.

Fecha de entrega:

25/08/2023.

```

import tkinter as tk

class Lexer:
    def __init__(self):
        self.reservadas = ['for', 'if', 'do', 'while', 'else']
        self.apertura = ['(']
        self.cierre = [')']

    def tokenize(self, text):
        self.tokens = self.reservadas + self.apertura + self.cierre
        arreglo = []
        current_token = ""

        for char in text:
            if char in self.tokens:
                if current_token != "":
                    arreglo.append(current_token)
                    current_token = ""

                arreglo.append(char)
            else:
                if char.isspace():
                    if current_token != "":
                        arreglo.append(current_token)
                        current_token = ""
                    else:
                        current_token += char

        if current_token != "":
            arreglo.append(current_token)
        return arreglo

```

//En esta parte del código declaramos nuestras palabras reservadas y nuestros paréntesis y le decimos al programa que los defina como token para así posteriormente los reconozca con sus respectivos nombres

```
def analyze(self, text):
    arreglo = self.tokenize(text)
    result = ""
    for token in arreglo:
        if token in self.reservadas:
            result += f"{token} | Palabra reservada\n"
        elif token in self.apertura:
            result += f"{token} | Parentecis de apertura\n"
        elif token in self.cierre:
            result += f"{token} | Parentecis de cierre\n"
        else:
            result += f"{token} | Error lexico x_x\n"
    return result
```

//En esta parte del programa le decimos que reconozca cada uno de los nombres que le pusimos con anterioridad a nuestros tokens y si en dado caso colocamos una palabra que no declaramos como nuestros tokens que me mande que es un “Error Léxico”

```
class LexerApp:
    def __init__(self):
        self.window = tk.Tk()
        self.window.title("ANALIZADOR LEXICO 😊")
        self.text_input = tk.Text(self.window, height=10, width=50)
        self.text_input.pack()

        self.analyze_button = tk.Button(self.window, text="Analizar",
command=self.analyze_text)
        self.analyze_button.pack()

        self.clean_button = tk.Button(self.window, text="Limpiar",
command=self.clean_text)
        self.clean_button.pack()
```

//En esta parte iniciamos otra clase para nuestros botones y titulo del programa. En el caso del titulo ajustamos la altura y la anchura para que no nos ocupe tanto espacio, en el caso del botón “Analizar”, declaramos analyze_button/text para que pueda analizar el texto y en caso de “Limpiar” declaramos clean_button/text para que pueda eliminar los que hay dentro de nuestro programa

```

frame = tk.Frame(self.window)
frame.pack()

self.text_label = tk.Label(frame, text="Linea", anchor='w')
self.text_label.pack(side="left")

self.text_label = tk.Label(frame, text="| Lexema |", anchor='center')
self.text_label.pack(side="left")

self.text_label = tk.Label(frame, text="Token", anchor='e')
self.text_label.pack(side="left")

self.result_label = tk.Label(self.window, text=" ", height=25, width=50)
self.result_label.pack()

```

//En este caso decalramos “Frame” para poder colocar los subtítulos debajo de los botones simétricamente. En el caso de “Linea” colocamos el frame hacia la izquierda, en el caso de “Lexema” puse dos separadores para reconocer que los resultados ya tiene un lugar asignado y le puse center para que este valla en medio y se pueda visualizar los operadores. En el caso de “Token” se coloco hacia la izquierda para poner reconocer el valor de los lexema y por ultimo en el label del resultado le puse una altura de “25” para que se pudiera ver bien el programa.

```

def analyze_text(self):
    lexer = Lexer()
    text = self.text_input.get("1.0", "end-1c") # Corregido para evitar el
    último salto de línea
    lines = text.split('\n')
    results = []

    for line_number, line in enumerate(lines, start=1):
        result_line = lexer.analyze(line)
        results.append(f"{line_number} | {result_line}")

    final_result = "\n".join(results)
    self.result_label.config(text=final_result)

```

//En este apartado es para reconocer el que posición se encuentra cada palabra y al momento de dar el resultado final separe las palabras por orden que en cada línea se encuentra

```
//Y como último paso tenemos lo que es el clean para que pueda eliminar
correctamente cada palabra y al final tenemos el código que nos sirve para que
pueda correr el programa.
```

The screenshot shows a Python IDE with a file named 'PalabrasReservadas.py'. The code defines a lexical analyzer for a simple language. It includes a list of reserved words ('for', 'do', 'while', 'else', 'if'), opening and closing characters ('(' and ')'), and a tokenize function that processes the input text into tokens. A pop-up window titled 'ANALIZADOR LEXICO' displays the input text and the resulting tokens: 'for', 'do', 'while', 'else', 'if', '(', ')', 'hola'. The code in the background defines the reserved words, opening and closing characters, and the tokenize function.

```

class Lexer:
    def __init__(self):
        self.reservadas = ['for', 'if', 'do', 'while', 'else']
        self.apertura = ['(']
        self.cierre = [')']

    def tokenize(self, text):
        self.tokens = self.reservadas + self.apertura + self.cierre
        arreglo = []
        current_token = ""

        for char in text:
            if char in self.tokens:
                if current_token != "":
                    arreglo.append(current_token)
                    current_token = ""
                arreglo.append(char)
            else:
                if char.isspace():
                    if current_token != "":
                        arreglo.append(current_token)
                        current_token = ""
                    else:
                        continue
                else:
                    current_token += char

        if current_token != "":
            arreglo.append(current_token)

        return arreglo

lexer = Lexer()
tokens = lexer.tokenize("for do while else if () hola")
print(tokens)

```

Visual Studio Code interface showing a Python file named `PalabrasReservadas.py` and a window titled `ANALIZADOR LEXICO`.

The `ANALIZADOR LEXICO` window displays the following code snippet:

```
for
do
while
else
if
(
)
hola
```

Below the code, there are buttons for `Analizar` and `Limpiar`. A table shows the results of the lexical analysis:

Linea	Lexema	Token
1	for	Palabra reservada
2	do	Palabra reservada
3	while	Palabra reservada
4	else	Palabra reservada
5	if	Palabra reservada
6	(Parentesis de apertura
7)	Parentesis de cierre
8	hola	Error lexico x_x

The `PalabrasReservadas.py` file contains the following Python code:

```
from tkinter import *
from tkinter import messagebox

class Lexer:
    def __init__(self):
        self.reservadas = ['for', 'if', 'do', 'while', 'else']
        self.apertura = ['(']
        self.cierre = [')']

    def tokenize(self, text):
        self.tokens = self.reservadas + self.apertura + self.cierre
        arreglo = []
        current_token = ""

        for char in text:
            if char in self.tokens:
                if current_token != "":
                    arreglo.append(current_token)
                    current_token = ""

                arreglo.append(char)
            else:
                if char.isspace():
                    if current_token != "":
                        arreglo.append(current_token)
                        current_token = ""
```

The bottom status bar shows the file is at `Ln 90, Col 4 (248 selected)` with `Spaces: 4`, `UTF-8` encoding, and `Python 3.11.4 64-bit` interpreter.