UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

ING. EN DESARROLLO Y TECNOLOGÍAS DE SOFTWARE

6 “M”

COMPILADORES

SUBCOMPETENCIA II.- ANÁLISIS SINTÁCTICO

ACTIVIDAD. 2.3. INDIVIDUAL REALIZAR LA SIGUIENTE PRÁCTICA EN PYTHON

ALUMNO: MARCO ANTONIO ZÚÑIGA MORALES – A211121

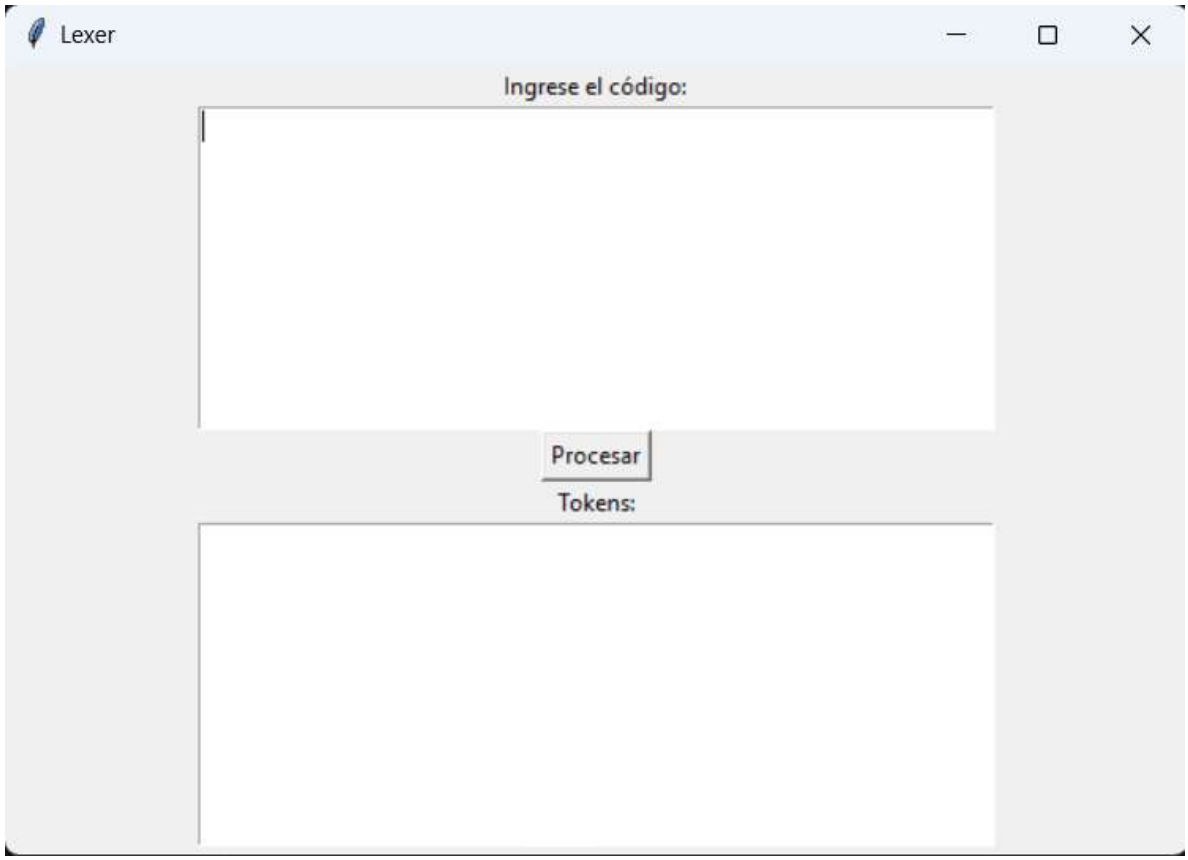
DOCENTE: DR. LUIS GUTIÉRREZ ALFARO

TUXTLA GUTIÉRREZ, CHIAPAS
VIERNES, 29 DE SEPTIEMBRE DE 2023

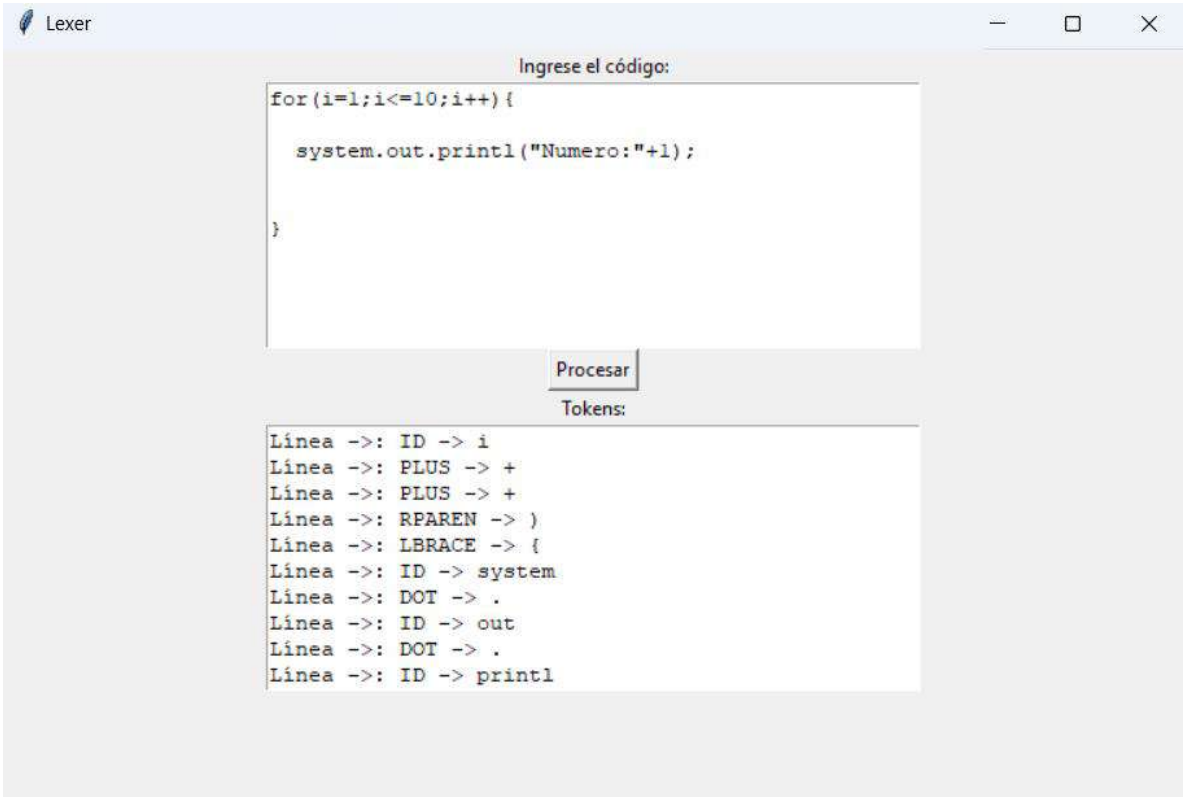
```
#Importando Librerias
import re
import tkinter as tk
from tkinter import messagebox
from ply import lex, yacc

#Definir los tokens
tokens = (
    'FOR',
    'INT',
    'ID',
    'NUM',
    'STRING',
    'PLUS',
    'SEMICOLON',
    'LPAREN',
    'RPAREN',
    'LBRACE',
    'RBRACE',
    'DOT',
    'EQUALS',
    'LEQ'
)
t_PLUS = r'\+'
t_SEMICOLON = r';'
t_LPAREN = r'\('
t_RPAREN = r'\)'
t_LBRACE = r'{'
t_RBRACE = r'}'
t_DOT = r'\.'
t_EQUALS = r'='
t_LEQ = r'<='

def t_STRING(t):
    r'\".*?\"'
    t.value = t.value[1:-1]
    return t

def t_ID(t):
    r'[a-zA-Z][a-zA-Z0-9]*'
    if t.value == 'for':
        t.type = 'FOR'
    elif t.value == 'int':
        t.type = 'INT'
    return t
```

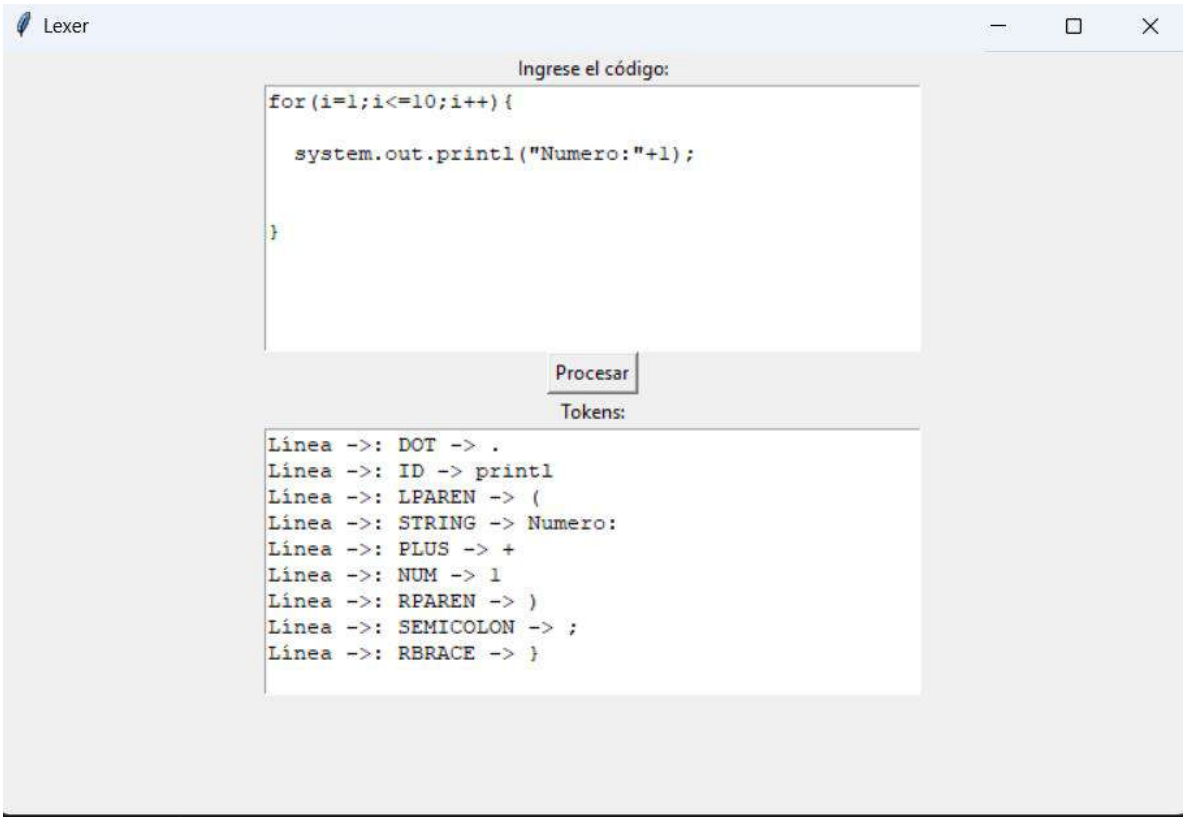
```
# Regla para identificar números
def t_NUM(t):
    r'\d+'
    t.value = int(t.value)
    return t

# Ignorar espacios en blanco y saltos de línea
t_ignore = ' \t\n'

def t_error(t):
    error_message(f"Token desconocido '{t.value[0]}'", t.lineno)
    t.lexer.skip(1)

# Construcción del lexer
lexer = lex.lex()

# Definición de la gramática para el análisis sintáctico
def p_for_loop(p):
    '''for_loop : FOR LPAREN INT ID EQUALS NUM SEMICOLON ID LEQ NUM
SEMICOLON ID PLUS PLUS RPAREN LBRACE ID DOT ID DOT ID LPAREN STRING PLUS NUM
RPAREN SEMICOLON RBRACE'''
    pass

# Manejo de errores de sintaxis
def p_error(p):
    if p:
        error_message(f"Error de sintaxis en '{p.value}'", p.lineno)
    else:
        error_message("Error de sintaxis: final inesperado del código",
len(code_text.get("1.0", "end-1c").split('\n')))

# Construcción del parser
parser = yacc.yacc()

# Función para el análisis léxico
def lex_analyzer(code):
    lexer.input(code)
    tokens = []
    while True:
        token = lexer.token()
        if not token:
            break
        tokens.append((token.lineno, token.type, token.value))
    return tokens
```

```
# Función para el análisis sintáctico
def parse_code(code):
    parser.parse(code, lexer=lexer)

def error_message(message, line_number):
    messagebox.showerror("Error de sintaxis", f"{message}\nEn la línea
{line_number}")

# Función para procesar el código ingresado
def process_code():
    code = code_text.get("1.0", "end-1c")
    tokens = lex_analyzer(code)
    result_text.delete("1.0", "end")
    for token in tokens:
        line_number, token_type, token_value = token
        result_text.insert("end", f"Línea ->: {token_type} ->
{token_value}\n")
    parse_code(code)


# Creación de la ventana de la interfaz gráfica
window = tk.Tk()
window.title("Lexer")
window.geometry("600x400")

# Etiqueta y campo de texto para ingresar el código
code_label = tk.Label(window, text="Ingrese el código:")
code_label.pack()

code_text = tk.Text(window, height=10, width=50)
code_text.pack()

# Botón para procesar el código
process_button = tk.Button(window, text="Procesar", command=process_code)
process_button.pack()

# Etiqueta y campo de texto para mostrar los tokens
result_label = tk.Label(window, text="Tokens:")
result_label.pack()

result_text = tk.Text(window, height=10, width=50)
result_text.pack()

# Ejecución de la interfaz gráfica
window.mainloop()
```


Lexer
Ingrese el código:
Procesar
Tokens:


Lexer
Ingrese el código:
for(i=1;i<=10;i++){
system.out.println("Numero:"+1);
}
Procesar
Tokens:
Línea ->: FOR -> for
Línea ->: LPAREN -> (
Línea ->: ID -> i
Línea ->: EQUALS -> =
Línea ->: NUM -> 1
Línea ->: SEMICOLON -> ;
Línea ->: ID -> i
Línea ->: LEQ -> <=
Línea ->: NUM -> 10
Línea ->: SEMICOLON -> ;


Lexer
Ingrese el código:
for(i=1;i<=10;i++){
system.out.println("Numero:"+1);
}
Procesar
Tokens:
Línea ->: ID -> i
Línea ->: PLUS -> +
Línea ->: PLUS -> +
Línea ->: RPAREN ->)
Línea ->: LBRACE -> {
Línea ->: ID -> system
Línea ->: DOT -> .
Línea ->: ID -> out
Línea ->: DOT -> .
Línea ->: ID -> println


Lexer
Ingrese el código:
for(i=1;i<=10;i++){
system.out.println("Numero:"+1);
}
Procesar
Tokens:
Línea ->: DOT -> .
Línea ->: ID -> println
Línea ->: LPAREN -> (
Línea ->: STRING -> Numero:
Línea ->: PLUS -> +
Línea ->: NUM -> 1
Línea ->: RPAREN ->)
Línea ->: SEMICOLON -> ;
Línea ->: RBRACE -> }