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.

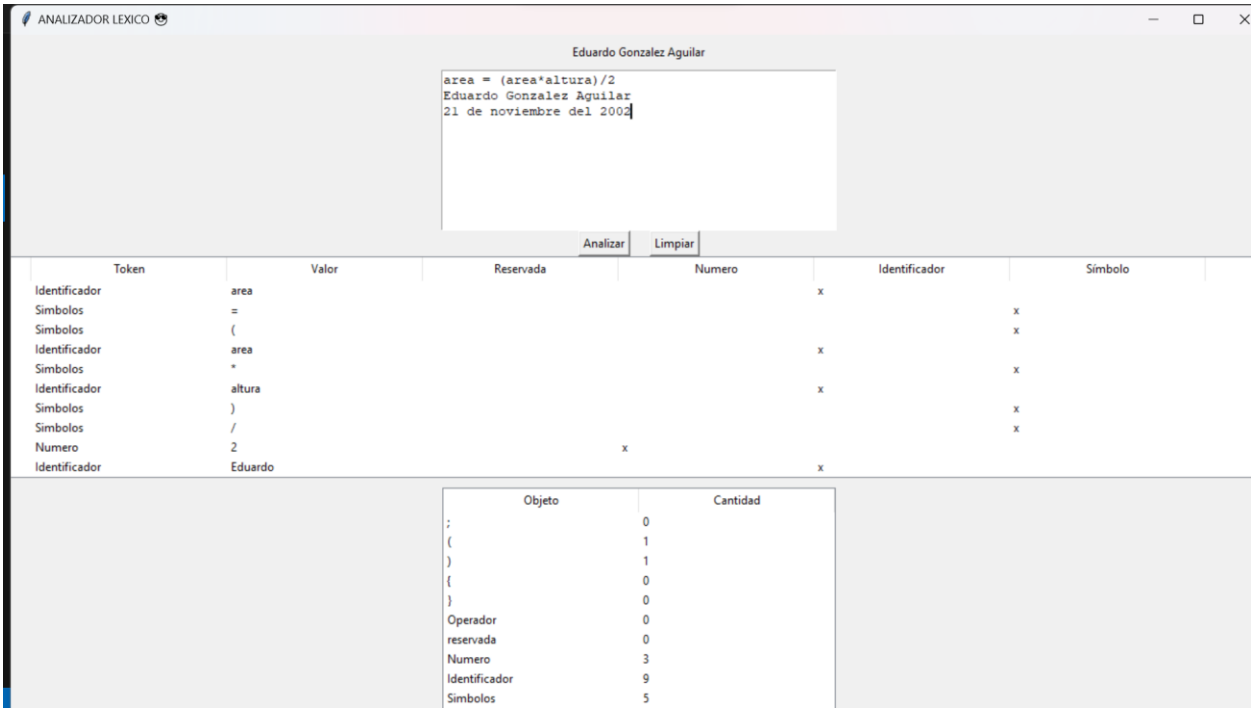
Nombre del alumno:
González Aguilar Eduardo - A211154

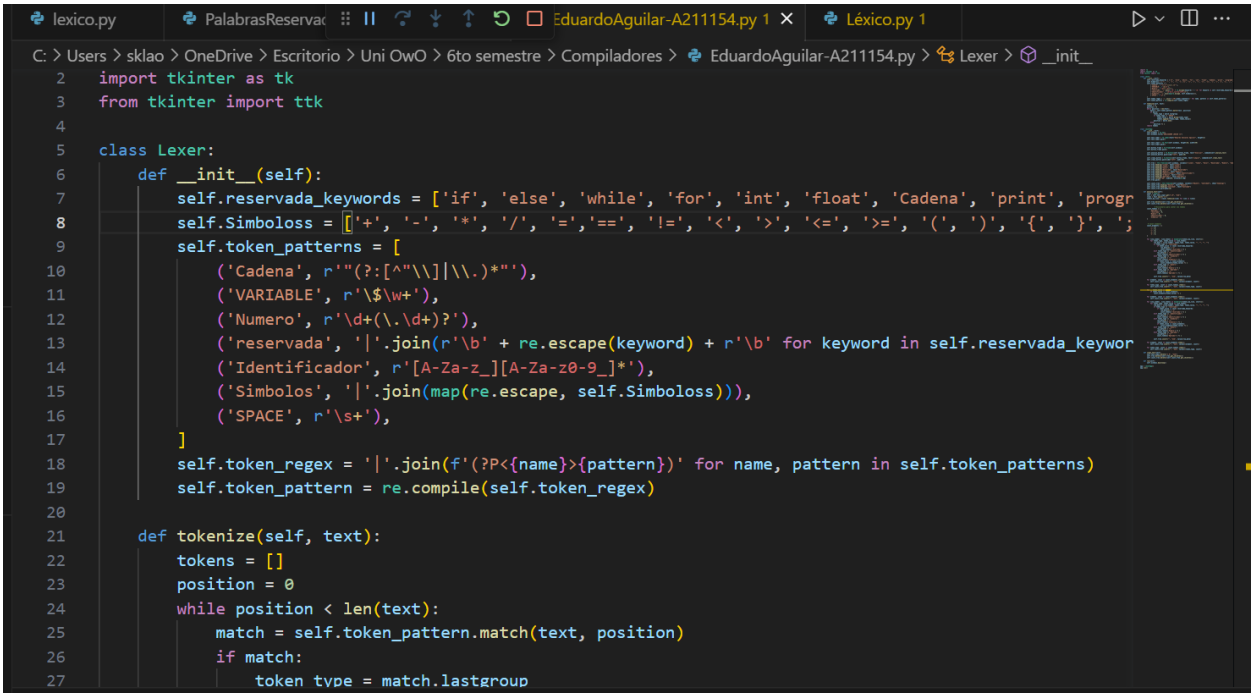
Semestre: 6. **Grupo:** M.

Nombre de la actividad:
Examen 1.

Enlace de actividad:
https://github.com/AguilarEduardo/Compilador

Fecha de entrega:
05/05/2023.


ANALIZADOR LEXICO
Eduardo Gonzalez Aguilar
area = (area*altura)/2
Eduardo Gonzalez Aguilar
21 de noviembre del 2002
Analizar
Limpiar
Token | Valor | Reservada | Numero | Identificador | Simbolo
Identificador | area | | | x |
Simbolos | = | | | | x
Simbolos | (| | | | x
Identificador | area | | | x |
Simbolos | * | | | | x
Identificador | altura | | | x |
Simbolos |) | | | | x
Simbolos | / | | | | x
Numero | 2 | | x | |
Identificador | Eduardo | | | x |
Objeto | Cantidad
; | 0
(| 1
) | 1
{ | 0
} | 0
Operador | 0
reservada | 0
Numero | 3
Identificador | 9
Simbolos | 5


lexico.py
PalabrasReservac
EduardoAguilar-A211154.py 1
Léxico.py 1
C: > Users > sklao > OneDrive > Escritorio > Uni OwO > 6to semestre > Compiladores > EduardoAguilar-A211154.py > Lexer > __init__
import tkinter as tk
from tkinter import ttk
class Lexer:
def __init__(self):
self.reservada_keywords = ['if', 'else', 'while', 'for', 'int', 'float', 'Cadena', 'print', 'progr
self.Simboloss = ['+', '-', '*', '/', '=','==', '!=', '<', '>', '<=', '>=', '(', ')', '{', '}', ';
self.token_patterns = [
('Cadena', r'"(?:[^"\\]|\\.)*"'),
('VARIABLE', r'\$\w+'),
('Numero', r'\d+(\.\d+)?'),
('reservada', '|'.join(r'\b' + re.escape(keyword) + r'\b' for keyword in self.reservada_keywor
('Identificador', r'[A-Za-z_][A-Za-z0-9_]*'),
('Simbolos', '|'.join(map(re.escape, self.Simboloss))),
('SPACE', r'\s+'),
]
self.token_regex = '|'.join(f'(?P<{name}>{pattern})' for name, pattern in self.token_patterns)
self.token_pattern = re.compile(self.token_regex)
def tokenize(self, text):
tokens = []
position = 0
while position < len(text):
match = self.token_pattern.match(text, position)
if match:
token type = match.lastgroup