Universidad Autónoma de Chiapas

Facultad de Contaduría y Administración, Campus I

Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software

Compiladores

Act. 1.6 Práctica. Unidad 1. Realiza un analizador Léxico en Python 50 tokens, anexar código línea del primer ejercicio REGEX

Elaborado por:

Diego Arturo Anzá Díaz

6°M

Catedrático:

Dr. Luis Gutiérrez Alfaro

Tuxtla Gutiérrez, Chiapas A día domingo 18 de agosto de 2024

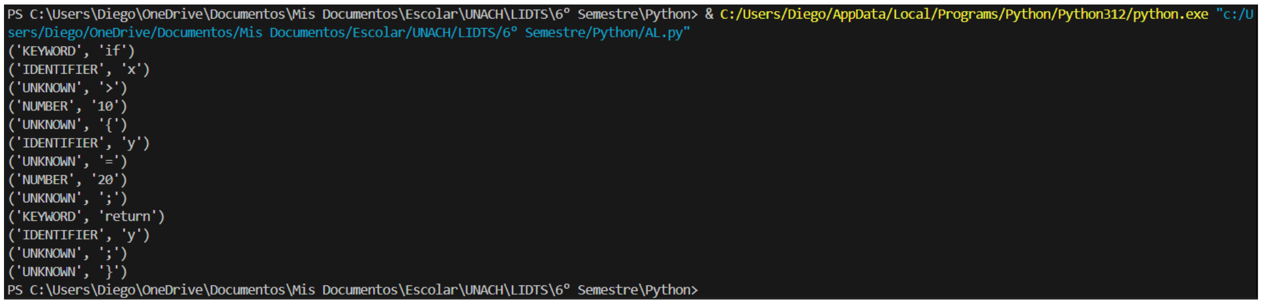
50 ejemplos

Token	***Lexema***	***Patrón (Expresión Regular)***
IF	if	\bif\b
ELSE	else	\belse\b
WHILE	while	\bwhile\b
FOR	for	\bfor\b
RETURN	return	\breturn\b
INT	int	\bint\b
FLOAT	float	\bfloat\b
CHAR	char	\bchar\b
VOID	void	\bvoid\b
CLASS	class	\bclass\b
PUBLIC	public	\bpublic\b
PRIVATE	private	\bprivate\b
PROTECTED	protected	\bprotected\b
STATIC	static	\bstatic\b
NEW	new	\bnew\b
TRUE	true	\btrue\b
FALSE	false	\bfalse\b
NULL	null	\bnull\b
THIS	this	\bthis\b
SUPER	super	\bsuper\b
PLUS	+	\+
MINUS	-	\-
MULTIPLY	*	*
DIVIDE	/	/
MODULO	%	%
EQUAL	==	==
ASSIGN	=	=
NOT_EQUAL	!=	!=
LESS_THAN	<	<
GREATER_THAN	>	>
LESS_EQUAL	<=	<=
GREATER_EQUAL	>=	>=
AND	&&	&&
OR	`	
NOT	!	!
INCREMENT	++	\+\+
DECREMENT	--	--
LEFT_PAREN	(	\(
RIGHT_PAREN	)	\)
LEFT_BRACE	{	\{
RIGHT_BRACE	}	\}

SEMICOLON	;	;
COMMA	,	,
DOT	.	\.
IDENTIFICADOR	variable1	[a-zA-Z_][a-zA-Z0-9_]*
NUMERO_ENTERO	123	\d+
NUMERO_REAL	3.14	\d+\.\d+
CADENA	"hello"	\".*?\"
COMENTARIO	// comentario	//.*
COMENTARIO_BLOQUE	/* comentario */	/*.*?*/

Analizador Léxico

```
import re



# Definir patrones para los tokens

token_patterns = {

    'KEYWORD': r'\b(?:if|else|while|for|return)\b',  # Palabras clave

    'IDENTIFIER': r'\b[a-zA-Z_][a-zA-Z0-9_]*\b',  # Identificadores

    'NUMBER': r'\b\d+\b',  # Números

    'WHITESPACE': r'\s+',  # Espacios en blanco

    'COMMENT': r'//.*',  # Comentarios de una línea

    'UNKNOWN': r'.'  # Cualquier otro carácter

}



def tokenize(code):

    # Crear una lista de expresiones regulares ordenadas por longitud (de mayor a menor)

    sorted_patterns = sorted(token_patterns.items(), key=lambda pair: -len(pair[1]))
    tokens = []
    while code:
        match = None
        for token_name, pattern in sorted_patterns:
            regex = re.compile(pattern)
            match = regex.match(code)
            if match:
                if token_name != 'WHITESPACE' and token_name != 'COMMENT':  # Ignorar
espacios y comentarios
                    tokens.append((token_name, match.group(0)))
                code = code[match.end():]  # Avanzar en el código
                break
        if not match:
            raise ValueError(f"Unexpected character sequence: {code}")

    return tokens


# Ejemplo de código fuente
source_code = '''
if x > 10 {
    y = 20;
    // Esto es un comentario
    return y;
}

'''



# Tokenizar el código fuente

tokens = tokenize(source_code)



# Imprimir los tokens

for token in tokens:

    print(token)
```

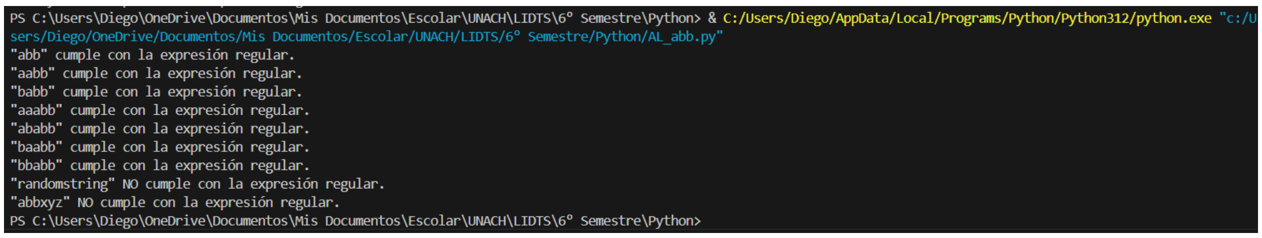


```
PS C:\Users\Diego\OneDrive\Documentos\Mis Documentos\Escolar\UNACH\LIDTS\6° Semestre\Python> & C:/Users/Diego/AppData/Local/Programs/Python/Python312/python.exe "c:/U
sers/Diego/OneDrive/Documentos/Mis Documentos/Escolar/UNACH/LIDTS/6° Semestre/Python/AL.py"
('KEYWORD', 'if')
('IDENTIFIER', 'x')
('UNKNOWN', '>')
('NUMBER', '10')
('UNKNOWN', '{')
('IDENTIFIER', 'y')
('UNKNOWN', '=')
('NUMBER', '20')
('UNKNOWN', ';')
('KEYWORD', 'return')
('IDENTIFIER', 'y')
('UNKNOWN', ';')
('UNKNOWN', '}')
PS C:\Users\Diego\OneDrive\Documentos\Mis Documentos\Escolar\UNACH\LIDTS\6° Semestre\Python>
```

Analizador Léxico del primer ejercicio REGEX

```
import re



# Expresión regular para cadenas que terminen en "abb"

pattern = r'.*abb$'



# Función para verificar si una cadena cumple con la expresión regular

def check_string(s):

    return re.fullmatch(pattern, s) is not None
```

```
# Lista de cadenas para probar

test_strings = [

    "abb",

    "aabb",

    "babb",

    "aaabb",

    "ababb",

    "baabb",

    "bbabb",

    "randomstring",

    "abbxyz",

]


# Probar las cadenas y mostrar los resultados

for string in test_strings:

    if check_string(string):

        print(f'"{string}" cumple con la expresión regular.')

    else:

        print(f'"{string}" NO cumple con la expresión regular.')
```



```
PS C:\Users\Diego\OneDrive\Documentos\Mis Documentos\Escolar\UNACH\LIDTS\6° Semestre\Python> & C:/Users/Diego/AppData/Local/Programs/Python/Python312/python.exe "c:/U
sers/Diego/OneDrive/Documentos/Mis Documentos/Escolar/UNACH/LIDTS/6° Semestre/Python/AL_abb.py"
"abb" cumple con la expresión regular.
"aabb" cumple con la expresión regular.
"babb" cumple con la expresión regular.
"aaabb" cumple con la expresión regular.
"ababb" cumple con la expresión regular.
"baabb" cumple con la expresión regular.
"bbabb" cumple con la expresión regular.
"randomstring" NO cumple con la expresión regular.
"abbxyz" NO cumple con la expresión regular.
PS C:\Users\Diego\OneDrive\Documentos\Mis Documentos\Escolar\UNACH\LIDTS\6° Semestre\Python>
```

GitHub

https://github.com/ArturitoAnDi/Compiladores