

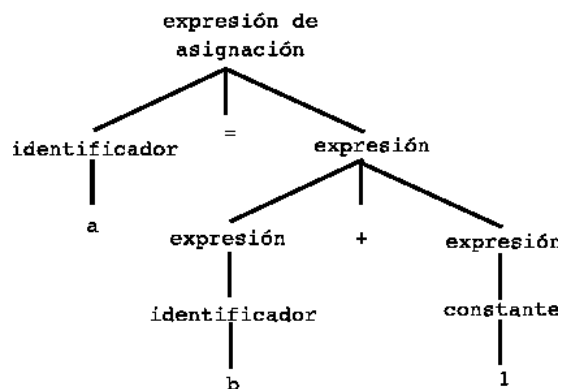
Analizador Sintáctico

Un analizador sintáctico, también conocido como parser, es una herramienta que se utiliza en el procesamiento de lenguaje natural para analizar y comprender la estructura sintáctica de una frase o un texto. El objetivo de un parser es determinar la relación entre las palabras y las frases en un texto y su función gramatical en el contexto del texto completo.

El análisis sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada. Un analizador léxico crea tokens de una secuencia de caracteres de entrada y son estos tokens los que son procesados por el analizador sintáctico para construir la estructura de datos, por ejemplo un árbol de análisis o árboles de sintaxis abstracta.

El lenguaje natural. Es usado para generar diagramas de lenguajes que usan flexión gramatical, como los idiomas romances o el latín. Los lenguajes habitualmente reconocidos por los analizadores sintácticos son los lenguajes libres de contexto. Cabe notar que existe una justificación formal que establece que los lenguajes libres de contexto son aquellos reconocibles por un autómata de pila, de modo que todo analizador sintáctico que reconozca un lenguaje libre de contexto es equivalente en capacidad computacional a un autómata de pila.

Los analizadores sintácticos fueron extensivamente estudiados durante los años 1970, detectándose numerosos patrones de funcionamiento en ellos, cosa que permitió la creación de programas generadores de analizadores sintácticos a partir de una especificación de la sintaxis del lenguaje en forma Backus-Naur por ejemplo, tales como yacc, GNU bison y javaCC.

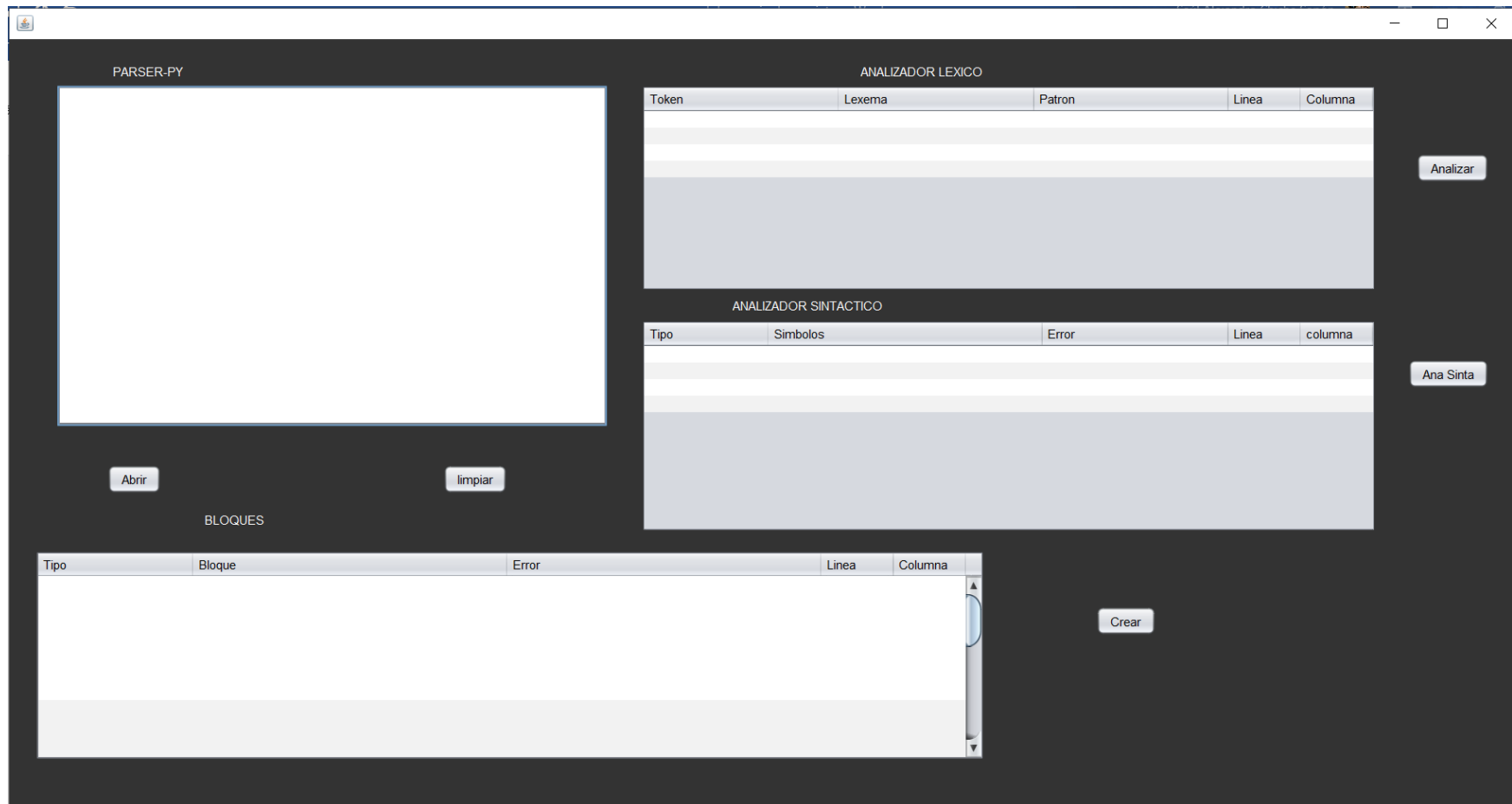


Aquí tienes las instrucciones de cómo utilizar el programa AS conocido por Analizador Sintáctico basado en Python.

Parser_Py fue creado por jdk 17 con netbeans 16.

Inicio del Programa AS

Doble click para iniciar el programa, te daba una imagen como es el programa.



Insertar código

Puedes escribir código Python, dentro el panel blanco escribe la estructura de código y para poder analizar tus token, símbolos y bloques con la ayuda de analizador sintáctico.

The interface is divided into several sections:

- PARSER-PY**: A text area containing the code:

```
Hola mundo
buenos dias tarde noches

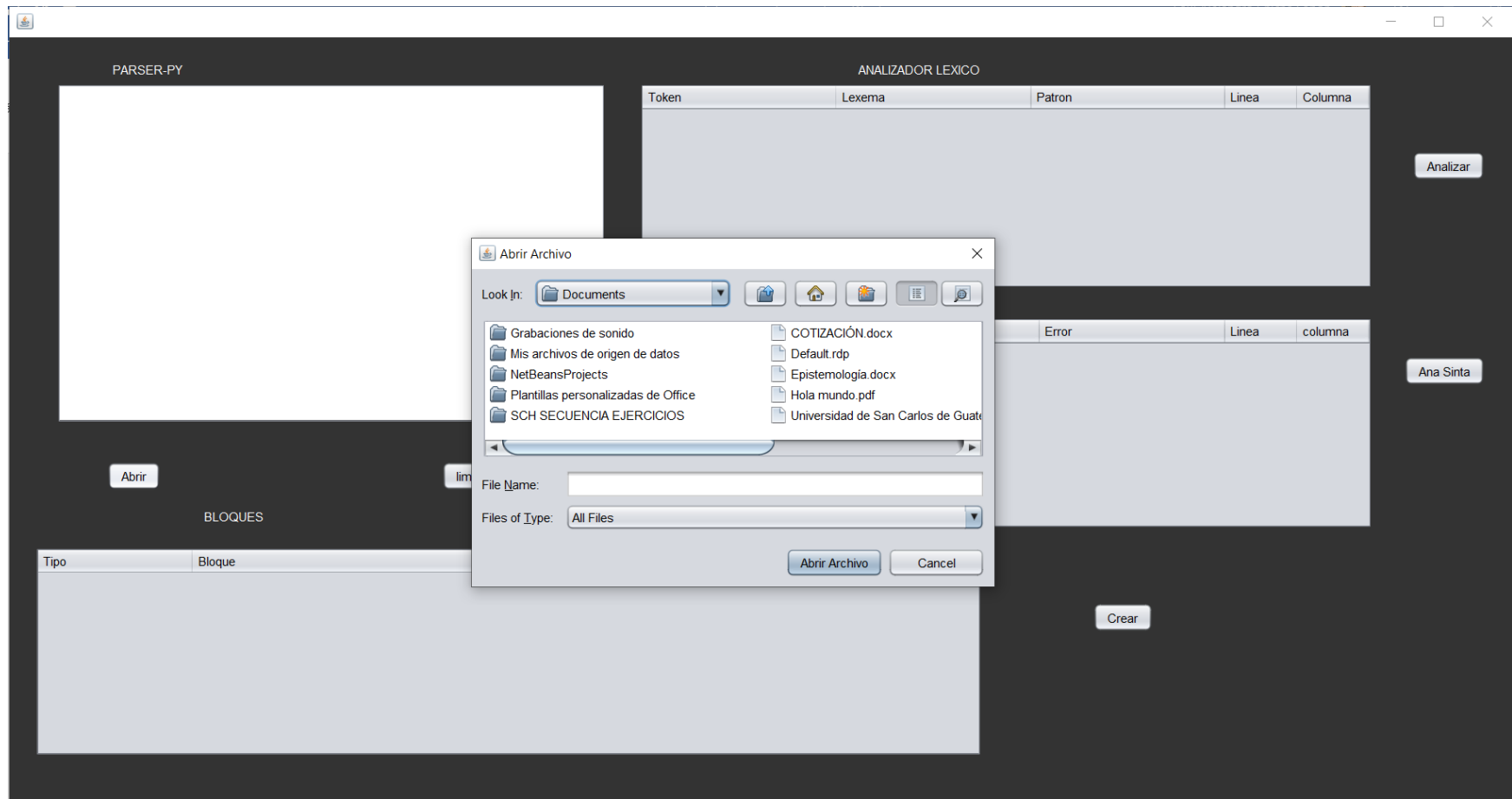
parser_py
```
- ANALIZADOR LEXICO**: A table with columns: Token, Lexema, Patron, Linea, Columna. It is currently empty.
- ANALIZADOR SINTACTICO**: A table with columns: Tipo, Simbolos, Error, Linea, columna. It is currently empty.
- BLOQUES**: A table with columns: Tipo, Bloque, Error, Linea, Columna. It is currently empty.

Buttons for interaction:

- Abrir**: Located below the PARSER-PY text area.
- limpiar**: Located below the PARSER-PY text area.
- Analizar**: Located to the right of the ANALIZADOR LEXICO table.
- Ana Sinta**: Located to the right of the ANALIZADOR SINTACTICO table.
- Crear**: Located at the bottom right of the interface.

Abrir archivo txt

Puedes abrir un archivo txt con el código Python, solo debes localizar el archivo (solo se permite los archivos de fuente txt).



Botón Analizador Léxico

Presiona el botón analizar y abajo observarás una tabla con todos los caracteres ingresados, te muestra los siguientes datos ingresados (el tipo Token, Lexema la palabra ingresada, Patrón basado en expresiones regulares, la Línea donde está localizado el carácter, la Columna donde está localizado el carácter).

Puedes observar que los caracteres cambian de color según su token:

Identificador color negro.

Operadores aritméticos, comparación, lógicos, asignación de color celeste.

Palabras clave color morado.

Constantes color rojo.

Comentarios color gris.

Otros color verde.

Errores léxicos color amarillo.

```
# bucles ciclos
# ejercicio 1
# condiciones
if 10 == 10: # 38 -if-59
    print("if") # 1
elif 10 != 10: # 39 -elif-61
    print("elif") # 1
else: # 40 -else-63
    print("else") # 1
#Funciones
def restar(a, b):
    def restar(a):
    def restar(1, 8):

#return
return a
```

limpiar

Tipo	Bloque	Error	Linea	Columna

Crear

Token	Lexema	Patron	Linea	Columna
Comentario	# bucles ciclos	[# [A-Za-z]+]	1	17
Comentario	# ejercicio 1	[# [A-Za-z]+]	2	14
Comentario	# condiciones	[# [A-Za-z]+]	3	14
Palabra clave	if	[palabras reservadas]	4	2
Constante Entero	10	[0-9]	4	5
Comparacion	==	[== != > < >= <=]	4	8
Constante Entero	10	[0-9]	4	11
Otros	:	[(O O) (, ,)]	4	12
Comentario	# 38 -if-59	[# [A-Za-z]+]	4	25
Identificador	print	[a-z,A-Z][_]	5	9
Otros	([(O O) (, ,)]	5	10

Tipo	Simbolos	Error	Linea	columna

Ana Sinta

Botón Analizador Sintáctico

Para llenar los datos en la tabla del analizador sintáctico debemos crear los tokens hechos por el analizador léxico, teniendo los tokens se puede observa toda la lista de símbolos del analizador sintáctico.

PARSER-PY

```
# bucles ciclos
# ejercicio 1
# condiciones
if 10 == 10: # 38 -if-59
    print("if") # 1
elif 10 != 10: # 39 -elif-61
    print("elif") # 1
else: # 40 -else-63
    print("else") # 1
#Funciones
def restar(a, b):
def restar(a):
def restar(1, 8):

#return
return a
```

Abrir

limpiar

BLOQUES

Tipo	Bloque	Error	Linea	Columna
------	--------	-------	-------	---------

ANALIZADOR LEXICO

Token	Lexema	Patron	Linea	Columna
Comentario	# bucles ciclos	[^# [A-Za-z]+]	1	17
Comentario	# ejercicio 1	[^# [A-Za-z]+]	2	14
Comentario	# condiciones	[^# [A-Za-z]+]	3	14
Palabra clave	if	[palabras reservadas]	4	2
Constante Entero	10	[0-9]	4	5
Comparacion	==	[== != > < >= <=]	4	8
Constante Entero	10	[0-9]	4	11
Otros	:	[(O O O I I :)]	4	12
Comentario	# 38 -if-59	[^# [A-Za-z]+]	4	25
Identificador	print	[a-zA-Z][_]	5	9
Otros	([(O O O I I :)]	5	10

ANALIZADOR SINTACTICO

Tipo	Simbolos	Error	Linea	columna
Comentarios	# bucles ciclos	Ninguno	1	17
Comentarios	# ejercicio 1	Ninguno	2	14
Comentarios	# condiciones	Ninguno	3	14
if	if 10 == 10:	Ninguno	4	2
Comentarios	# 38 -if-59	Ninguno	4	25
Expresiones	print ("if")	Ninguno	5	9
Comentarios	# 1	Ninguno	5	20
elif	elif 10 != 10:	Ninguno	6	4
Comentarios	# 39 -elif-61	Ninguno	6	29
Expresiones	print ("elif")	Ninguno	7	9
Comentarios	# 1	Ninguno	7	22

Ana Sinta

Crear

Botón para crear Bloques

Para crear los bloques (condicionales if, ciclos for, Declaracion de variables) necesitamos los símbolos del analizador sintáctico para verificar el orden de donde están trabajados.

The screenshot displays a Python parser application with the following components:

- PARSER-PY:** A text editor containing Python code with syntax highlighting.
- ANALIZADOR LEXICO:** A table showing the results of lexical analysis.
- ANALIZADOR SINTACTICO:** A table showing the results of syntactic analysis.
- BLOQUES:** A table showing the results of block analysis.
- Buttons:** 'Abrir', 'limpiar', 'Analizar', 'Ana Sinta', and 'Crear'.

PARSER-PY Code:

```
return 3.1416
return u%V
return "hola"
return len(texto)
return 'mundo'

# vamos a crear bloques
# condiciones
if 10 == 10:
    print("if")
elif 10 != 10:
    print("elif")
else:
    print("else")

# condiciones 2
if True:
    print(z)
elif 10 != 10:
```

ANALIZADOR LEXICO Table:

Token	Lexema	Patron	Linea	Columna
Comentario	# bucles ciclos	[^# [A-Za-z]+]	1	17
Comentario	# ejercicio 1	[^# [A-Za-z]+]	2	14
Comentario	# condiciones	[^# [A-Za-z]+]	3	14
Palabra clave	if	[palabras reservadas]	4	2
Constante Entero	10	[0-9]	4	5
Comparacion	==	[== != > < >= <=]	4	8
Constante Entero	10	[0-9]	4	11
Otros	:	[0 8 0 1 :]	4	12
Comentario	# 38 -if-59	[^# [A-Za-z]+]	4	25
Identificador	print	[a-zA-Z][_]	5	9
Otros	([(){} [] :]	5	10

ANALIZADOR SINTACTICO Table:

Tipo	Simbolos	Error	Linea	columna
return	return len (texto)	Ninguno	23	6
return	return 'mundo'	Ninguno	24	6
Comentarios	# vamos a crear bloques	Ninguno	26	24
Comentarios	# condiciones	Ninguno	27	14
if	if 10 == 10:	Ninguno	28	2
Expresiones	print ("if")	Ninguno	29	9
elif	elif 10 != 10:	Ninguno	30	4
Expresiones	print ("elif")	Ninguno	31	9
else	else :	Ninguno	32	4
Expresiones	print ("else")	Ninguno	33	9
Comentarios	# condiciones 2	Ninguno	35	16

BLOQUES Table:

Tipo	Bloque	Error	Linea	Columna
Condicionales	if 10 == 10: print ("if") elif 10 != 10: print ("elif") else : print ("else")	Ninguno	28	2
	if True : print (z) elif 10 != 10:			

Botón limpiar

Darle click al botón limpiar te limpiara el editor de texto y las 3 tablas para que puedes analizar otro código.

The interface is a dark-themed web application for code analysis. It consists of several components:

- PARSER-PY**: A large white text editor area.
- limpiar**: A button located below the text editor, used to clear the editor and the tables.
- ANALIZADOR LEXICO**: A table with columns: Token, Lexema, Patron, Linea, and Columna. It has a **Analizar** button to its right.
- ANALIZADOR SINTACTICO**: A table with columns: Tipo, Simbolos, Error, Linea, and columna. It has an **Ana Sinta** button to its right.
- BLOQUES**: A table with columns: Tipo, Bloque, Error, Linea, and Columna. It has a **Crear** button to its right.