

Parser-Py

# Manual Tecnico

---

Jerson Emanuel Estrada Estrada ..... 201930199

## Que es un analizador Sintáctico

Un analizador sintáctico, a menudo llamado "parser" en inglés, es una parte esencial de un compilador o intérprete de lenguaje de programación que analiza la estructura gramatical de un programa fuente para comprender su significado. Aquí tienes una breve descripción de los conceptos fundamentales:

**Lenguaje de entrada:** Un analizador sintáctico toma un lenguaje de entrada, que generalmente es un programa fuente escrito en un lenguaje de programación. El analizador se encarga de convertir este texto en una estructura de datos que sea más fácil de manipular y comprender para el compilador o intérprete.

**Gramática:** La gramática es un conjunto de reglas que define la estructura sintáctica válida de un lenguaje. Se utiliza para determinar si las construcciones en el programa fuente son gramaticalmente correctas.


**Árbol de sintaxis abstracta (AST):** El resultado del análisis sintáctico es a menudo representado como un árbol de sintaxis abstracta. Este árbol es una estructura de datos jerárquica que representa la estructura del programa fuente de una manera que es más fácil de procesar. Cada nodo del árbol representa una construcción sintáctica, como una declaración, una expresión, etc.

**Reglas de producción:** En la gramática, las reglas de producción definen cómo se pueden combinar los símbolos no terminales (como "expresión" o "declaración") para formar construcciones válidas en el lenguaje. Estas reglas son utilizadas por el analizador sintáctico para generar el AST.

**Análisis descendente y ascendente:** Estos son dos enfoques comunes para la construcción del AST. El análisis descendente comienza desde la raíz del árbol y busca construir el árbol descendiendo a través de las reglas de producción, mientras que el análisis ascendente comienza desde las hojas y busca combinar símbolos para subir hacia la raíz.

**Tokens:** Antes de que el análisis sintáctico pueda tener lugar, el programa fuente se divide en "tokens" mediante un analizador léxico. Los tokens son unidades léxicas básicas, como palabras clave, identificadores, números y operadores.

**Errores sintácticos:** El analizador sintáctico debe detectar y reportar errores en la estructura del programa fuente. Los errores pueden incluir tokens inesperados, construcciones sintácticas inválidas o construcciones faltantes.



**Recuperación de errores:** En muchos analizadores sintácticos, se incluye una capacidad de recuperación de errores para intentar continuar analizando el programa después de un error en lugar de detenerse por completo.

**Generación de código intermedio:** Después de que el analizador sintáctico haya construido el AST, la siguiente etapa en el proceso de compilación es a menudo la generación de código intermedio. Esto implica transformar el AST en una representación más cercana al código de máquina o a un código intermedio específico del compilador.

## Gramática

La gramática en un analizador sintáctico es un conjunto de reglas que define la estructura sintáctica válida de un lenguaje. Su fin es determinar si las construcciones en el programa fuente son gramaticalmente correctas al aplicar estas reglas, ayudando a comprender y analizar la estructura del código para su procesamiento subsiguiente.

### DECLARACION DE VARIABLES

```
<Declaracion> ::= <DeclaracionVariable>  
                | <DeclaracionArreglo>  
                | <DeclaracionDiccionario>
```

```
<DeclaracionVariable> ::= <Identificador> "=" <Cadena>  
                        | <Identificador> "=" <Decimal>  
                        | <Identificador> "=" <Entero>  
                        | <Identificador> "=" "True"  
                        | <Identificador> "=" "False"
```

```
<DeclaracionArreglo> ::= <Identificador> "=" "[" <ListaValores> "]"
```

```
<ListaValores> ::= <Valor>  
                | <Valor> "," <ListaValores>
```

```
<Valor> ::= <Cadena>  
          | <Entero>  
          | <Decimal>  
          | "True"  
          | "False"
```

```
<DeclaracionDiccionario> ::= <Identificador> "=" "{" <ListaPares> "}"
```

```
<ListaPares> ::= <EntradaDicc>  
              | <EntradaDicc> "," <ListaPares>
```

```
<EntradaDicc> ::= <Identificador> ":" <Valor>
```

```
<Identificador> ::= IDENTIFICADOR
```

**<Cadena> ::= CADENA**

**<Entero> ::= ENTERO**

**<Decimal> ::= DECIMAL**

## ASIGNACIÓN

**<Asignacion> ::= <AsignacionSimple> | <AsignacionMultiple>**

**<AsignacionSimple> ::= <Identificador> <OperadorAsignacion> <Expresion>**

**<AsignacionMultiple> ::= <ListaIdentificadores> <OperadorAsignacion> <ListaExpresiones>**

**<ListaIdentificadores> ::= <Identificador>**

**| <Identificador> "," <ListaIdentificadores>**

**<ListaExpresiones> ::= <Expresion>**

**| <Expresion> "," <ListaExpresiones>**

**<Identificador> ::= IDENTIFICADOR**

**<Expresion> ::= <Cadena>**

**| <Entero>**

**| "True"**

**| "False"**

**<OperadorAsignacion> ::= "="**

## CONDICIONAL

**<Condicional> ::= <If>**

**| <IfElse>**

**| <IfElif>**

**| <IfElifElse>**

**<If>** ::= "if" <Expresion> ":" <BloqueCodigo>

**<IfElse>** ::= "if" <Expresion> ":" <BloqueCodigo> "else" ":" <BloqueCodigo>

**<IfElif>** ::= "if" <Expresion> ":" <BloqueCodigo> <Elif>

**<Elif>** ::= "elif" <Expresion> ":" <BloqueCodigo> <Elif> |  $\epsilon$

# La notación  $\epsilon$  significa que puede haber cero o más bloques 'elif'

**<IfElifElse>** ::= "if" <Expresion> ":" <BloqueCodigo> <Elif> "else" ":" <BloqueCodigo>

**<BloqueCodigo>** ::= <Indentacion> <Sentencias>

# <Indentacion> representa el espacio en blanco o tabulación que precede al bloque de código.

**<Expresion>** ::= <ExpresionLogica>

| <ExpresionAritmetica>

| <ExpresionRelacional>

| <ExpresionBooleana>

**<ExpresionLogica>** ::= <ExpresionLogica> "and" <ExpresionLogica>

| <ExpresionLogica> "or" <ExpresionLogica>

| "not" <ExpresionLogica>

| "(" <ExpresionLogica> ")"

| <ExpresionRelacional>

**<ExpresionAritmetica>** ::= <ExpresionAritmetica> "+" <ExpresionAritmetica>

| <ExpresionAritmetica> "-" <ExpresionAritmetica>

| <ExpresionAritmetica> "\*" <ExpresionAritmetica>

| <ExpresionAritmetica> "/" <ExpresionAritmetica>

| "(" <ExpresionAritmetica> ")"

| <Entero>



| <Decimal>

| <Identificador>

**<ExpresionRelacional>** ::= <ExpresionAritmetica> "==" <ExpresionAritmetica>

| <ExpresionAritmetica> "!=" <ExpresionAritmetica>

| <ExpresionAritmetica> ">" <ExpresionAritmetica>

| <ExpresionAritmetica> "<" <ExpresionAritmetica>

| <ExpresionAritmetica> ">=" <ExpresionAritmetica>

| <ExpresionAritmetica> "<=" <ExpresionAritmetica>

**<ExpresionBooleana>** ::= "True" | "False"

**<Identificador>** ::= IDENTIFICADOR

<Entero> ::= ENTERO

<Decimal> ::= DECIMAL

## **OPERADOR TERNARIO**

**<OperadorTernario>** ::= <Expresion> "?" <Expresion> ":" <Expresion>

| <Expresion> "if" <Expresion> "else" <Expresion>

**<Expresion>** ::= <ExpresionLogica>

| <ExpresionAritmetica>

| <ExpresionRelacional>

| <ExpresionBooleana>

**<ExpresionLogica>** ::= <ExpresionLogica> "and" <ExpresionLogica>

| <ExpresionLogica> "or" <ExpresionLogica>

| "not" <ExpresionLogica>

| "(" <ExpresionLogica> ")"

| <ExpresionRelacional>

## CICLOS FOR

**<CicloFor>** ::= "for" <Identificador> "in" <Iterable> ":" <BloqueCodigo>

**<Iterable>** ::= <Rango>

| <Lista>

| <Diccionario>

| <LlamadaFuncion>

**<Rango>** ::= "range" "(" <Expresion> ")"

**<Lista>** ::= "[" <ListaValores> "]"

**<Diccionario>** ::= "{" <ListaPares> "}"

**<LlamadaFuncion>** ::= <Identificador> "(" <Argumentos> ")"

**<ListaValores>** ::= <Valor> | <Valor> "," <ListaValores>

**<Valor>** ::= <Entero>

| <Decimal>

| <Cadena>

| <Booleano>

| <Arreglo>

| <Diccionario>

**<ListaPares>** ::= <EntradaDicc> | <EntradaDicc> "," <ListaPares>

**<EntradaDicc>** ::= <Identificador> ":" <Valor>

**<Expresion>** ::= <ExpresionLogica>

| <ExpresionAritmetica>

| <ExpresionRelacional>



| <ExpresionBooleana>

| <ExpresionPertenencia>

**<BloqueCodigo> ::= <Indentacion> <Sentencias>**

**<Indentacion> ::= INDENTACION**

**<Identificador> ::= IDENTIFICADOR**

**<Entero> ::= ENTERO**

**<Decimal> ::= DECIMAL**

**<Cadena> ::= CADENA**

**<Booleano> ::= "True" | "False"**

**<Arreglo> ::= "[" <ListaValores> "]"**

## **CICLOS FOR ELSE**

**<CicloFor> ::= "for" <Identificador> "in" <Iterable> ":" <BloqueCodigo>**

**<CicloForConElser> <Identificador> "in" <Iterable> ":" <BloqueCodigo> "else" ":" <BloqueCodigo>**

**<Iterable> ::= <Rango>**

| <Lista>

| <Diccionario>

| <LlamadaFuncion>


**<Rango> ::= "range" "(" <Expresion> ")"**

**<Lista> ::= "[" <ListaValores> "]"**

**<Diccionario> ::= "{" <ListaPares> "}"**

**<LlamadaFuncion> ::= <Identificador> "(" <Argumentos> ")"**

**<ListaValores> ::= <Valor> | <Valor> "," <ListaValores>**



**<Valor> ::= <Entero>**

    | <Decimal>

    | <Cadena>

    | <Booleano>

    | <Arreglo>

    | <Diccionario>

**<ListaPares> ::= <EntradaDicc> | <EntradaDicc> "," <ListaPares>**

**<EntradaDicc> ::= <Identificador> ":" <Valor>**

**<Expresion> ::= <ExpresionLogica>**

    | <ExpresionAritmetica>

    | <ExpresionRelacional>

    | <ExpresionBooleana>

    | <ExpresionPertenencia>

**<BloqueCodigo> ::= <Indentacion> <Sentencias>**

**<Indentacion> ::= INDENTACION**

**<Identificador> ::= IDENTIFICADOR**

**<Entero> ::= ENTERO**

**<Decimal> ::= DECIMAL**

**<Cadena> ::= CADENA**

**<Booleano> ::= "True" | "False"**

**<Arreglo> ::= "[" <ListaValores> "]"**

## WHILE

**<CicloWhile>** ::= "while" <Expresion> ":" <BloqueCodigo>

**<Expresion>** ::= <ExpresionLogica>  
                  | <ExpresionAritmetica>  
                  | <ExpresionRelacional>  
                  | <ExpresionBooleana>  
                  | <ExpresionPertenencia>


**<BloqueCodigo>** ::= <Indentacion> <Sentencias>

**<Indentacion>** ::= INDENTACION

**<ExpresionLogica>** ::= <ExpresionLogica> "and" <ExpresionLogica>  
                      | <ExpresionLogica> "or" <ExpresionLogica>  
                      | "not" <ExpresionLogica>  
                      | "(" <ExpresionLogica> ")"  
                      | <ExpresionRelacional>

**<ExpresionAritmetica>** ::= <ExpresionAritmetica> "+" <ExpresionAritmetica>  
                          | <ExpresionAritmetica> "-" <ExpresionAritmetica>  
                          | <ExpresionAritmetica> "\*" <ExpresionAritmetica>  
                          | <ExpresionAritmetica> "/" <ExpresionAritmetica>  
                          | "(" <ExpresionAritmetica> ")"  
                          | <Entero>  
                          | <Decimal>  
                          | <Identificador>

**<ExpresionRelacional>** ::= <ExpresionAritmetica> "==" <ExpresionAritmetica>  
                          | <ExpresionAritmetica> "!=" <ExpresionAritmetica>



| <ExpresionAritmetica> ">" <ExpresionAritmetica>  
| <ExpresionAritmetica> "<" <ExpresionAritmetica>  
| <ExpresionAritmetica> ">=" <ExpresionAritmetica>  
| <ExpresionAritmetica> "<=" <ExpresionAritmetica>

**<ExpresionBooleana> ::= "True" | "False"**

**<ExpresionPertenencia> ::= <Expresion> "in" <Expresion>**

**<Identificador> ::= IDENTIFICADOR**

**<Entero> ::= ENTERO**

**<Decimal> ::= DECIMAL**

## **FUNCION / METODO**

**<Funcion> ::= "def" <Identificador> "(" <Parametros> ")" ":" <BloqueCodigo>**

**<Parametros> ::= <Parametro> | <Parametro> "," <Parametros>**

**<Parametro> ::= <Identificador>**

**<BloqueCodigo> ::= <Indentacion> <Sentencias>**

**<Indentacion> ::= INDENTACION**

**<Identificador> ::= IDENTIFICADOR**