Here's a breakdown considering both the Dragon Book's perspective and the grammar we built:

**Dragon Book's Perspective:**

* **Factor:** The most basic building block of an expression. It's considered indivisible and cannot be further broken down into meaningful subexpressions. Examples include literals (numbers, variables), constants, or function calls without arguments.
* **Term:** A combination of factors using multiplication, division, or sometimes exponentiation. It represents a simple mathematical or logical computation.
* **Expression:** A more complex construct formed by combining terms using addition, subtraction, comparison operators (like greater than, less than), or logical operators (like AND, OR). Expressions can also involve function calls with arguments.

**Relationship with Our Grammar:**

* **Factors:** In our grammar, factors would most closely align with literals (INT, FLOAT, STR, BOOL), or variable names (IDENT). These represent the indivisible building blocks used in expressions.
* **Terms:** While not explicitly defined, terms could be interpreted as expressions involving only multiplication, division, or potentially exponentiation (if supported by the language). However, our grammar uses infix operators like PLUS, MINUS, ASTERISK, and SLASH which could represent both arithmetic and logical operations.
* **Expressions:** In our grammar, expressions encompass a broader range including literals, variables, function calls, prefix expressions (<prefix\_expression>), infix expressions (<infix\_expression>), and grouped expressions (<grouped\_expression>). This aligns more closely with the Dragon Book's definition of expressions.

**Key Differences:**

* The Dragon Book sometimes separates arithmetic and logical operators, whereas our grammar treats them all as infix operators.
* Our grammar doesn't explicitly define terms as distinct from expressions, but the concept of terms involving only specific operators can be applied if the language supports those operations.

**CHOMSKY NORMAL FORM GRAMMAR**

**grammar = {**

**ProgramNode: [**

**[StatementListNode],**

**],**

**StatementListNode: [**

**[StatementNode],**

**[StatementNode, StatementListNode],**

**],**

**StatementNode: [**

**[LetStatementNode],**

**[AssignStatementNode],**

**[ExpressionStatementNode],**

**[ReturnStatementNode],**

**[IfStatementNode],**

**[PrintStatementNode],**

**[ClockStatementNode],**

**[CustomContextNode],**

**],**

**LetStatementNode: [**

**[LET, IdentifierNode, ASSIGN, ExpressionNode, SEMICOLON],**

**[LET, IdentifierNode, SEMICOLON],**

**],**

**AssignStatementNode: [**

**[IdentifierNode, ASSIGN, ExpressionNode, SEMICOLON],**

**],**

**ExpressionStatementNode: [**

**[ExpressionNode, SEMICOLON],**

**],**

**ReturnStatementNode: [**

**[RETURN, ExpressionNode, SEMICOLON],**

**],**

**IfStatementNode: [**

**[IF, LPAREN, ExpressionNode, RPAREN, LBRACE, StatementListNode, RBRACE, ElseClauseNode],**

**],**

**ElseClauseNode: [**

**[ELSE, LBRACE, StatementListNode, RBRACE],**

**[ELSE, IfStatementNode],**

**[], # This represents the absence of an else or else if clause.**

**],**

**PrintStatementNode: [**

**[PRINT, ExpressionNode, SEMICOLON],**

**],**

**ClockStatementNode: [**

**[CLOCK, DOT, ClockFunctionNode, LPAREN, RPAREN, SEMICOLON],**

**],**

**ClockFunctionNode: [**

**[CLOCK],**

**["NOW"],**

**],**

**ExpressionNode: [**

**[IntegerLiteralNode],**

**[FloatLiteralNode],**

**[StringLiteralNode],**

**[BooleanLiteralNode],**

**[IdentifierNode],**

**[FunctionLiteralNode],**

**[CallExpressionNode],**

**[PrefixExpressionNode],**

**[InfixExpressionNode],**

**[GroupedExpressionNode],**

**],**

**FunctionLiteralNode: [**

**[FUNCTION, LPAREN, ParametersNode, RPAREN, LBRACE, StatementListNode, ReturnStatementNode, RBRACE],**

**],**

**CallExpressionNode: [**

**[IdentifierNode, LPAREN, ExpressionNode, RPAREN],**

**],**

**ExpressionList: [**

**[ExpressionNode, ExpressionList],**

**[ExpressionNode],**

**[],**

**],**

**ParametersNode: [**

**[IdentifierNode, ParametersNode],**

**[IdentifierNode]**

**],**

**PrefixExpressionNode: [**

**[PrefixOperatorNode, ExpressionNode],**

**],**

**InfixExpressionNode: [**

**[ExpressionNode, InfixOperatorNode, ExpressionNode],**

**],**

**GroupedExpressionNode: [**

**[LPAREN, ExpressionNode, RPAREN],**

**],**

**PrefixOperatorNode: [**

**[BANG],**

**[MINUS],**

**],**

**InfixOperatorNode: [**

**[PLUS],**

**[MINUS],**

**[ASTERISK]**

**[SLASH],**

**[LT\_EQ],**

**[LT],**

**[GT\_EQ],**

**[GT],**

**[EQ],**

**[NOT\_EQ],**

**],**

**CustomContextNode: [**

**[LBRACE, StatementListNode, RBRACE],**

**[CONTEXT, IDENT, LBRACE, LStatementListNode, RBRACE],**

**],**

**}**