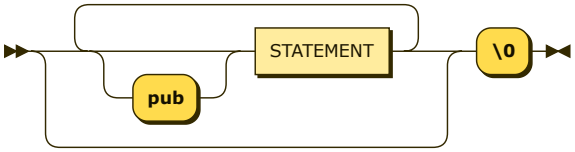
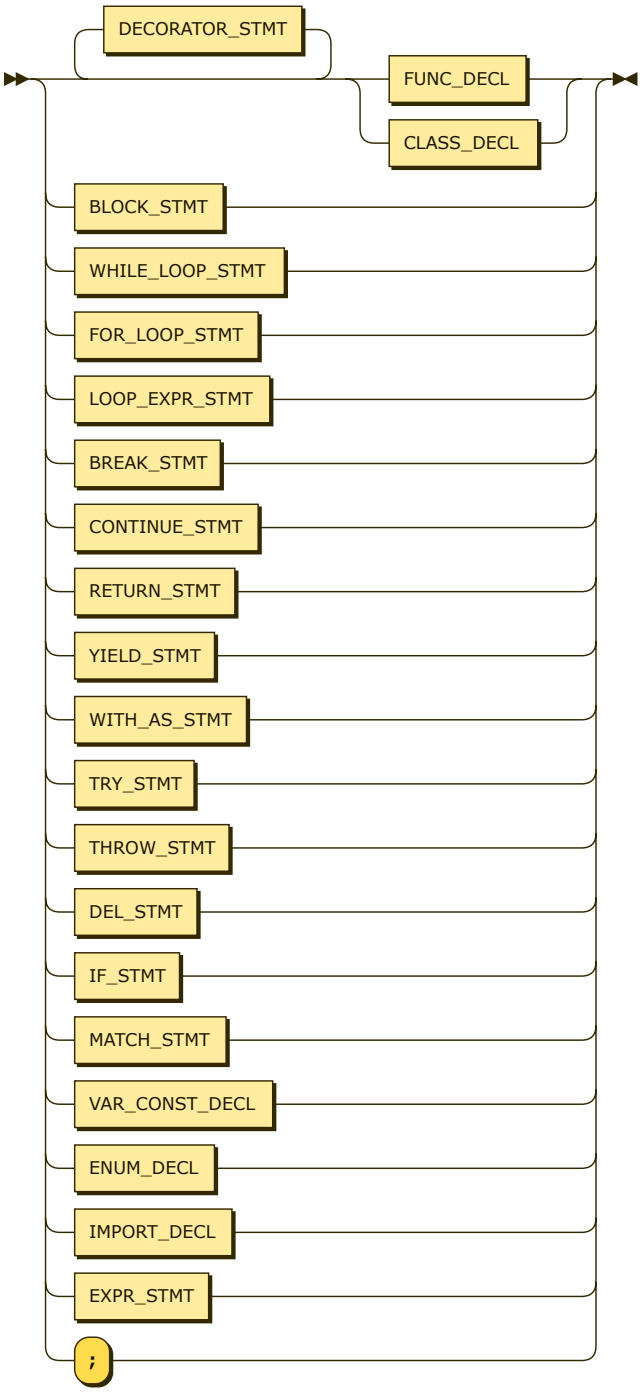
MODULE:




```
MODULE   ::= ( 'pub'? STATEMENT )* '\0'
```

no references

STATEMENT:



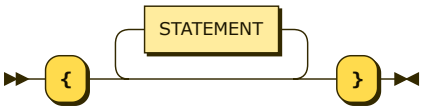

```
STATEMENT
         ::= BLOCK_STMT
           | WHILE_LOOP_STMT
           | FOR_LOOP_STMT
           | LOOP_EXPR_STMT
```

```
        | BREAK_STMT
        | CONTINUE_STMT
        | RETURN_STMT
        | YIELD_STMT
        | WITH_AS_STMT
        | TRY_STMT
        | THROW_STMT
        | DEL_STMT
        | IF_STMT
        | MATCH_STMT
        | VAR_CONST_DECL
        | ENUM_DECL
        | IMPORT_DECL
        | DECORATOR_STMT* ( FUNC_DECL | CLASS_DECL )
        | EXPR_STMT
        | ';'
```

referenced by:

- BLOCK_STMT
- MODULE

BLOCK_STMT:

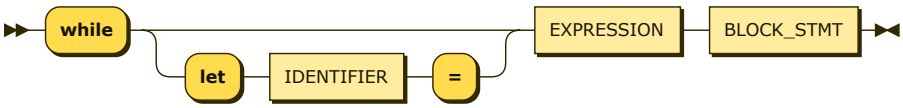



```
BLOCK_STMT
        ::= '{' STATEMENT* '}'
```

referenced by:

- CATCH_PART
- FINALLY_PART
- FOR_LOOP_STMT
- FUNC_DECL
- IF_STMT
- LAMBDA_EXPR
- LOOP_EXPR_STMT
- MATCH_ARM
- OPERATOR_OVERLOAD
- STATEMENT
- TRY_STMT
- WHILE_LOOP_STMT
- WITH_AS_STMT

WHILE_LOOP_STMT:




```
WHILE_LOOP_STMT
        ::= 'while' ( 'let' IDENTIFIER '=' )? EXPRESSION BLOCK_STMT
```

referenced by:

- STATEMENT

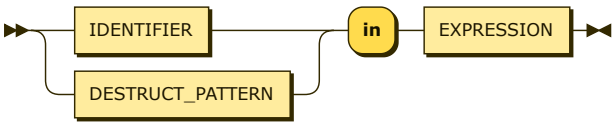
FOR_LOOP_STMT:


```
FOR_LOOP_STMT
        ::= 'for' FOR_LOOP_HEAD BLOCK_STMT
```

referenced by:

- STATEMENT

FOR_LOOP_HEAD:

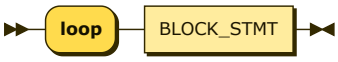



```
FOR_LOOP_HEAD
        ::= ( IDENTIFIER | DESTRUCT_PATTERN ) 'in' EXPRESSION
```

referenced by:

- COMPACT_FOR_LOOP
- FOR_LOOP_STMT

LOOP_EXPR_STMT:

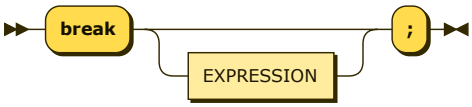



```
LOOP_EXPR_STMT
        ::= 'loop' BLOCK_STMT
```

referenced by:

- LARGE_EXPR
- STATEMENT

BREAK_STMT:




```
BREAK_STMT
        ::= 'break' EXPRESSION? ';'
```

referenced by:

- STATEMENT

CONTINUE_STMT:


```
CONTINUE_STMT
        ::= 'continue' ';'
```

referenced by:

- STATEMENT

RETURN_STMT:


```
RETURN_STMT
        ::= 'return' EXPRESSION ';'
```

referenced by:

- STATEMENT

YIELD_STMT:


```
YIELD_STMT
        ::= 'yield' EXPRESSION ';'
```

referenced by:

- STATEMENT

THROW_STMT:


```
THROW_STMT
        ::= 'throw' EXPRESSION ';'
```

referenced by:

- STATEMENT

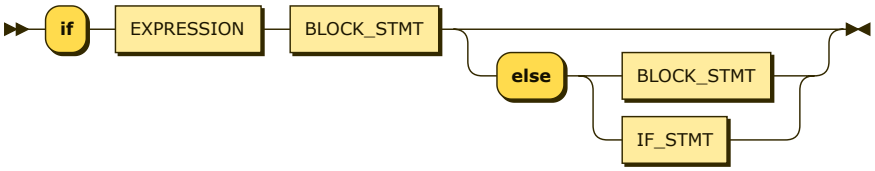
DEL_STMT:


```
DEL_STMT ::= 'del' EXPRESSION ';'
```

referenced by:

- STATEMENT

IF_STMT:

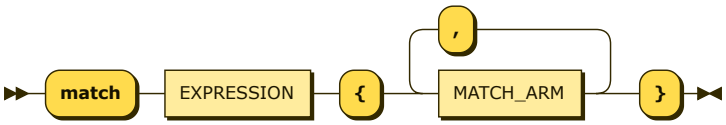



```
IF_STMT  ::= 'if' EXPRESSION BLOCK_STMT ( 'else' ( BLOCK_STMT | IF_STMT ) )?
```

referenced by:

- IF_STMT
- STATEMENT

MATCH_STMT:

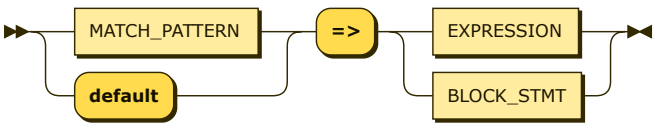



```
MATCH_STMT
        ::= 'match' EXPRESSION '{' MATCH_ARM ( ',' MATCH_ARM )* '}'
```

referenced by:

- STATEMENT

MATCH_ARM:

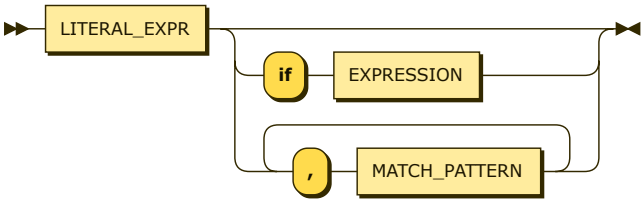



```
MATCH_ARM
        ::= ( MATCH_PATTERN | 'default' ) '=>' ( EXPRESSION | BLOCK_STMT )
```

referenced by:

- MATCH_STMT

MATCH_PATTERN:

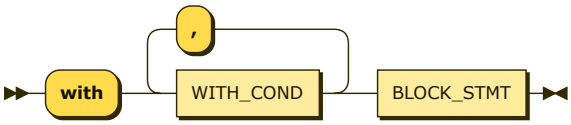



```
MATCH_PATTERN
        ::= LITERAL_EXPR ( 'if' EXPRESSION | ( ',' MATCH_PATTERN )* )
```

referenced by:

- MATCH_ARM
- MATCH_EXPR_ARM
- MATCH_PATTERN

WITH_AS_STMT:

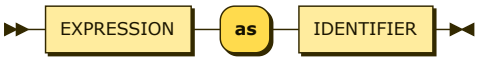



```
WITH_AS_STMT
        ::= 'with' WITH_COND ( ',' WITH_COND )* BLOCK_STMT
```

referenced by:

- STATEMENT

WITH_COND:

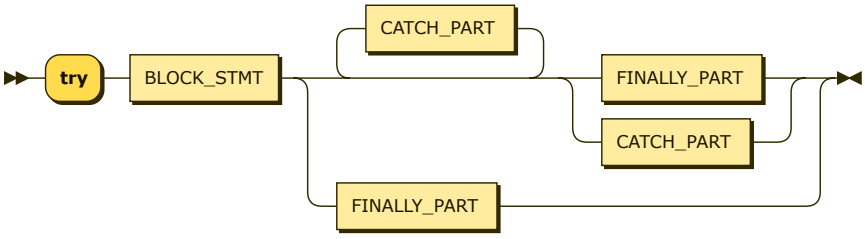



```
WITH_COND
        ::= EXPRESSION 'as' IDENTIFIER
```

referenced by:

- WITH_AS_STMT

TRY_STMT:

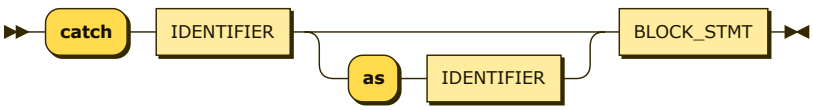



```
TRY_STMT ::= 'try' BLOCK_STMT ( CATCH_PART* ( FINALLY_PART | CATCH_PART ) | FINALLY_PART )
```

referenced by:

- STATEMENT

CATCH_PART:

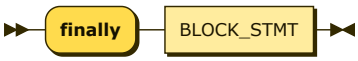


```
CATCH_PART
        ::= 'catch' IDENTIFIER ( 'as' IDENTIFIER )? BLOCK_STMT
```

referenced by:

- TRY_STMT

FINALLY_PART:




```
FINALLY_PART
        ::= 'finally' BLOCK_STMT
```

referenced by:

- TRY_STMT

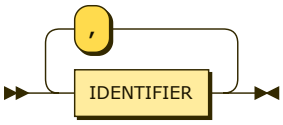
EXPR_STMT:


```
EXPR_STMT
        ::= EXPRESSION ';'
```

referenced by:

- STATEMENT

IDENTIFIER_LIST:

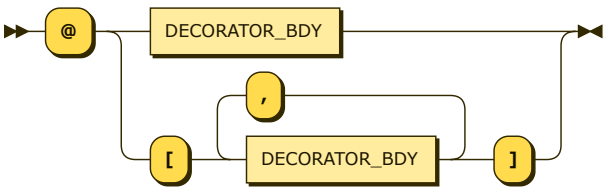



```
IDENTIFIER_LIST
        ::= IDENTIFIER ( ',' IDENTIFIER )*
```

referenced by:

- CLASS_EXTEND
- CLASS_IMPL
- DESTRUCT_PATTERN
- ENUM_DECL
- GRANULAR_IMPORT
- PARAMETERS

DECORATOR_STMT:

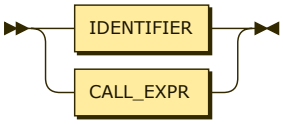



```
DECORATOR_STMT
        ::= '@' ( DECORATOR_BDY | '[' DECORATOR_BDY ( ',' DECORATOR_BDY )* ']' )
```

referenced by:

- CLASS_MEMBER
- STATEMENT

DECORATOR_BDY:

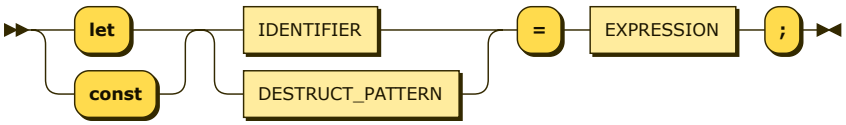



```
DECORATOR_BDY
        ::= IDENTIFIER
          | CALL_EXPR
```

referenced by:

- DECORATOR_STMT

VAR_CONST_DECL:

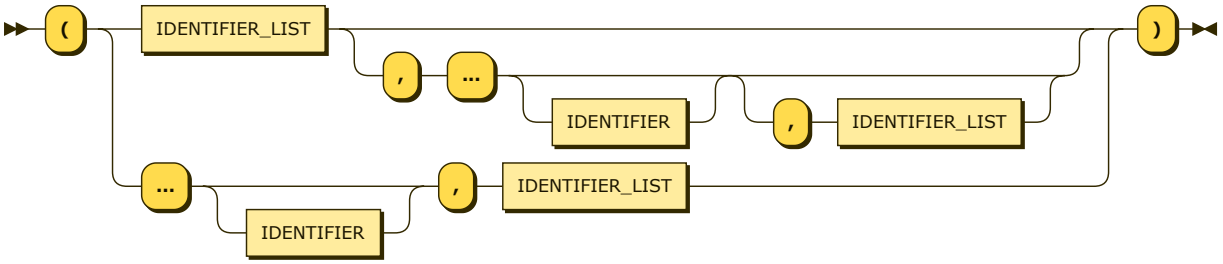



```
VAR_CONST_DECL
        ::= ( 'let' | 'const' ) ( IDENTIFIER | DESTRUCT_PATTERN ) '=' EXPRESSION ';'
```

referenced by:

- CLASS_MEMBER
- STATEMENT

DESTRUCT_PATTERN:

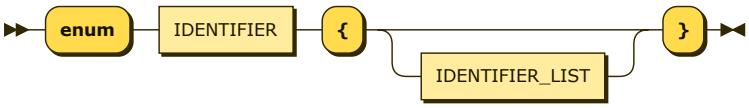



```
DESTRUCT_PATTERN
        ::= '(' ( IDENTIFIER_LIST ( ',' '...' IDENTIFIER? ( ',' IDENTIFIER_LIST )? )? | '...' IDENTIFIER? ',' IDENTIFIER_LIST ) ')'
```

referenced by:

- FOR_LOOP_HEAD
- VAR_CONST_DECL

ENUM_DECL:

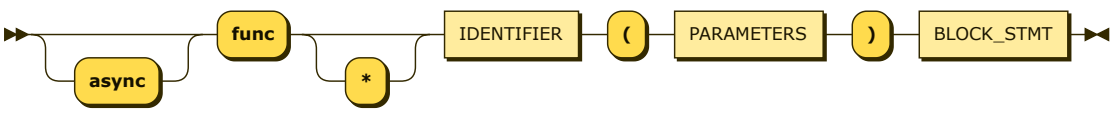



```
ENUM_DECL
        ::= 'enum' IDENTIFIER '{' IDENTIFIER_LIST? '}'
```

referenced by:

- STATEMENT

FUNC_DECL:

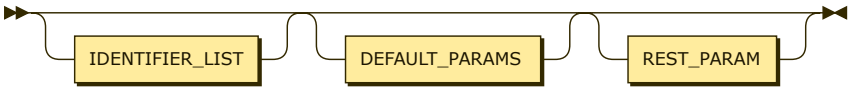



```
FUNC_DECL
        ::= 'async'? 'func' '*'? IDENTIFIER '(' PARAMETERS ')' BLOCK_STMT
```

referenced by:

- CLASS_MEMBER
- STATEMENT

PARAMETERS:

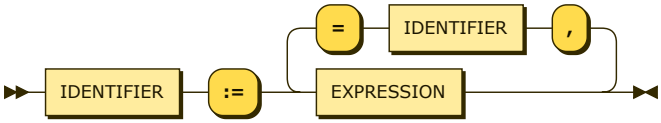



```
PARAMETERS
        ::= IDENTIFIER_LIST? DEFAULT_PARAMS? REST_PARAM?
```

referenced by:

- FUNC_DECL
- LAMBDA_EXPR
- OPERATOR_OVERLOAD

DEFAULT_PARAMS:




```
DEFAULT_PARAMS
        ::= IDENTIFIER ':=' EXPRESSION ( ',' IDENTIFIER '=' EXPRESSION )*
```

referenced by:

- PARAMETERS

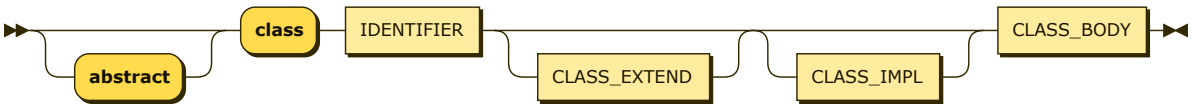
REST_PARAM:


```
REST_PARAM
        ::= '...' IDENTIFIER
```

referenced by:

- PARAMETERS

CLASS_DECL:




```
CLASS_DECL
        ::= 'abstract'? 'class' IDENTIFIER CLASS_EXTEND? CLASS_IMPL? CLASS_BODY
```

referenced by:

- STATEMENT

CLASS_EXTEND:


```
CLASS_EXTEND
        ::= '->' IDENTIFIER_LIST
```

referenced by:

- CLASS_DECL

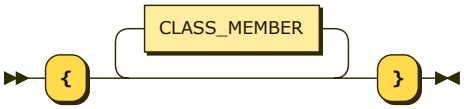
CLASS_IMPL:


```
CLASS_IMPL
        ::= 'impl' IDENTIFIER_LIST
```

referenced by:

- CLASS_DECL

CLASS_BODY:

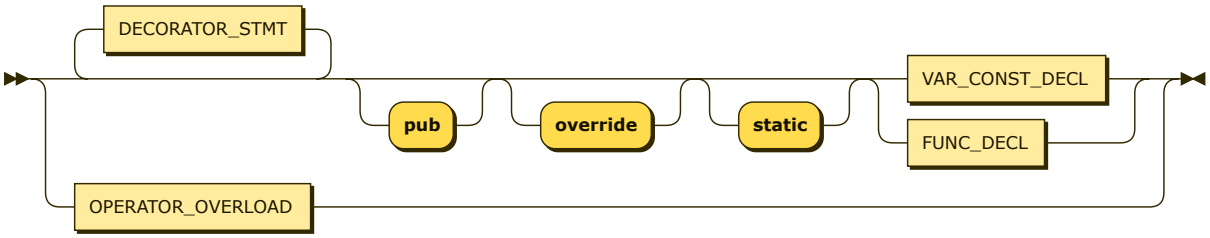



```
CLASS_BODY
        ::= '{' CLASS_MEMBER* '}'
```

referenced by:

- CLASS_DECL

CLASS_MEMBER:

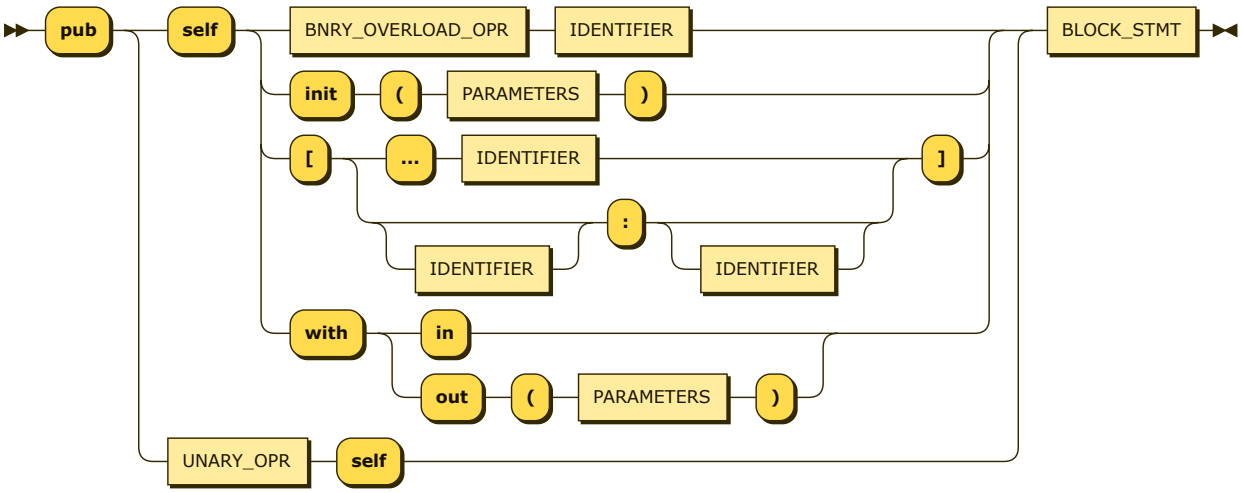



```
CLASS_MEMBER
        ::= DECORATOR_STMT* 'pub'? 'override'? 'static'? ( VAR_CONST_DECL | FUNC_DECL )
          | OPERATOR_OVERLOAD
```

referenced by:

- CLASS_BODY

OPERATOR_OVERLOAD:

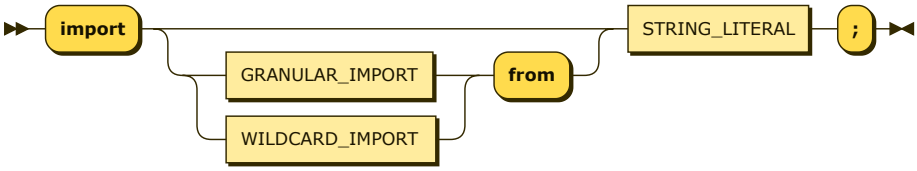



```
OPERATOR_OVERLOAD
        ::= 'pub' ( 'self' ( BNRY_OVERLOAD_OPR IDENTIFIER | 'init' '(' PARAMETERS ')' | '[' ( '...' IDENTIFIER | IDENTIFIER? ':'
            IDENTIFIER? ) ']' | 'with' ( 'in' | 'out' '(' PARAMETERS ')' ) ) | UNARY_OPR 'self' ) BLOCK_STMT
```

referenced by:

- CLASS_MEMBER

IMPORT_DECL:




```
IMPORT_DECL
        ::= 'import' ( ( GRANULAR_IMPORT | WILDCARD_IMPORT ) 'from' )? STRING_LITERAL ';'
```

referenced by:

- STATEMENT

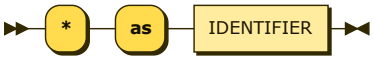
GRANULAR_IMPORT:


```
GRANULAR_IMPORT
        ::= '{' IDENTIFIER_LIST '}'
```

referenced by:

- IMPORT_DECL

WILDCARD_IMPORT:




```
WILDCARD_IMPORT
        ::= '*' 'as' IDENTIFIER
```

referenced by:

- IMPORT_DECL

EXPRESSION:

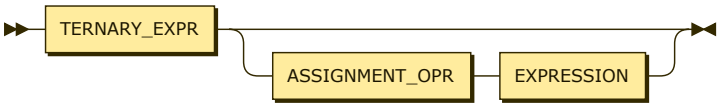

```
EXPRESSION
        ::= REASSIGNMENT_EXPR
```

referenced by:

- ARR_TPL_LIST
- ARR_TPL_REPEAT
- BREAK_STMT
- COMPACT_ARR_TPL
- COMPACT_FOR_LOOP
- DEFAULT_PARAMS
- DEL_STMT
- EXPR_ARGUMENTS
- EXPR_STMT
- FOR_LOOP_HEAD
- IF_STMT
- INDEXER
- KEY_VAL_PAR
- LAMBDA_EXPR
- LITERAL_EXPR
- MATCH_ARM
- MATCH_EXPR
- MATCH_EXPR_ARM
- MATCH_PATTERN
- MATCH_STMT
- NAMED_ARGUMENTS
- REASSIGNMENT_EXPR
- RETURN_STMT
- SINGLE_SPREAD_EXPR
- SLICE
- STRING_SEQUENCE

- TERNARY_EXPR
- THROW_STMT
- VAR_CONST_DECL
- WHILE_LOOP_STMT
- WITH_COND
- YIELD_STMT

REASSIGNMENT_EXPR:

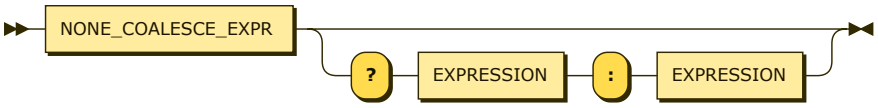



```
REASSIGNMENT_EXPR
        ::= TERNARY_EXPR ( ASSIGNMENT_OPR EXPRESSION )?
```

referenced by:

- EXPRESSION

TERNARY_EXPR:

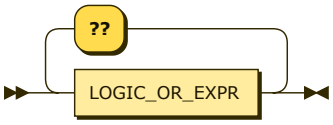



```
TERNARY_EXPR
        ::= NONE_COALESCE_EXPR ( '?' EXPRESSION ':' EXPRESSION )?
```

referenced by:

- REASSIGNMENT_EXPR

NONE_COALESCE_EXPR:

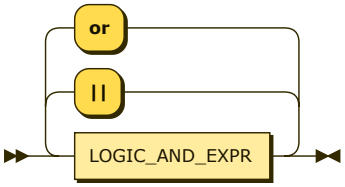



```
NONE_COALESCE_EXPR
        ::= LOGIC_OR_EXPR ( '??' LOGIC_OR_EXPR )*
```

referenced by:

- TERNARY_EXPR

LOGIC_OR_EXPR:

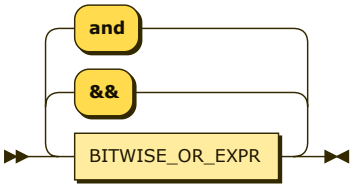



```
LOGIC_OR_EXPR
        ::= LOGIC_AND_EXPR ( ( '||' | 'or' ) LOGIC_AND_EXPR )*
```

referenced by:

- NONE_COALESCE_EXPR

LOGIC_AND_EXPR:

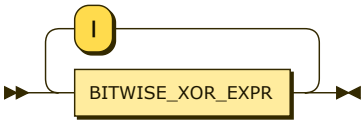



```
LOGIC_AND_EXPR
        ::= BITWISE_OR_EXPR ( ( '&&' | 'and' ) BITWISE_OR_EXPR )*
```

referenced by:

- LOGIC_OR_EXPR

BITWISE_OR_EXPR:

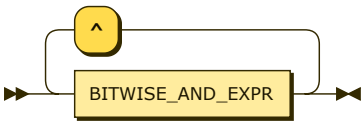



```
BITWISE_OR_EXPR
        ::= BITWISE_XOR_EXPR ( '|' BITWISE_XOR_EXPR )*
```

referenced by:

- LOGIC_AND_EXPR

BITWISE_XOR_EXPR:




```
BITWISE_XOR_EXPR
        ::= BITWISE_AND_EXPR ( '^' BITWISE_AND_EXPR )*
```

referenced by:

- BITWISE_OR_EXPR

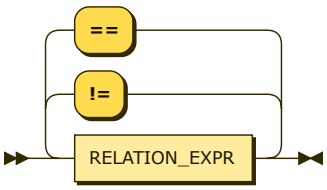
BITWISE_AND_EXPR:


```
BITWISE_AND_EXPR
        ::= EQUALITY_EXPR ( '&' EQUALITY_EXPR )*
```

referenced by:

- BITWISE_XOR_EXPR

EQUALITY_EXPR:

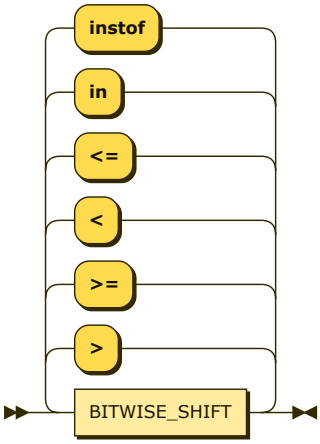



```
EQUALITY_EXPR
        ::= RELATION_EXPR ( ( '!=' | '==' ) RELATION_EXPR )*
```

referenced by:

- BITWISE_AND_EXPR

RELATION_EXPR:

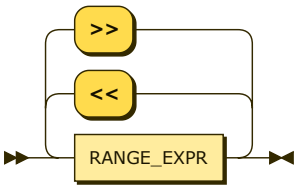



```
RELATION_EXPR
        ::= BITWISE_SHIFT ( ( '>' | '>=' | '<' | '<=' | 'in' | 'instof' ) BITWISE_SHIFT )*
```

referenced by:

- EQUALITY_EXPR

BITWISE_SHIFT:

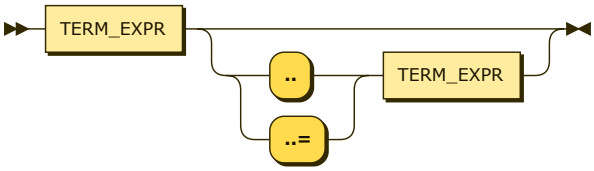



```
BITWISE_SHIFT
        ::= RANGE_EXPR ( ( '<<' | '>>' ) RANGE_EXPR )*
```

referenced by:

- RELATION_EXPR

RANGE_EXPR:

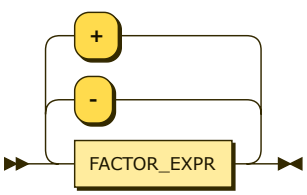



```
RANGE_EXPR
        ::= TERM_EXPR ( ( '..' | '..=' ) TERM_EXPR )?
```

referenced by:

- BITWISE_SHIFT

TERM_EXPR:

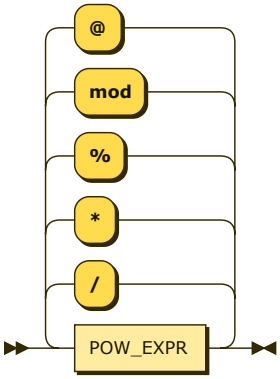


```
TERM_EXPR
        ::= FACTOR_EXPR ( ( '−' | '+' ) FACTOR_EXPR )*
```

referenced by:

- RANGE_EXPR

FACTOR_EXPR:

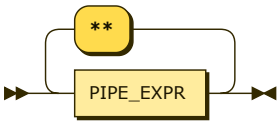



```
FACTOR_EXPR
        ::= POW_EXPR ( ( '/' | '*' | '%' | 'mod' | '@' ) POW_EXPR )*
```

referenced by:

- TERM_EXPR

POW_EXPR:

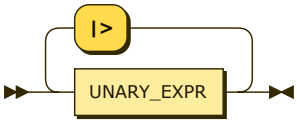



```
POW_EXPR ::= PIPE_EXPR ( '**' PIPE_EXPR )*
```

referenced by:

- FACTOR_EXPR

PIPE_EXPR:

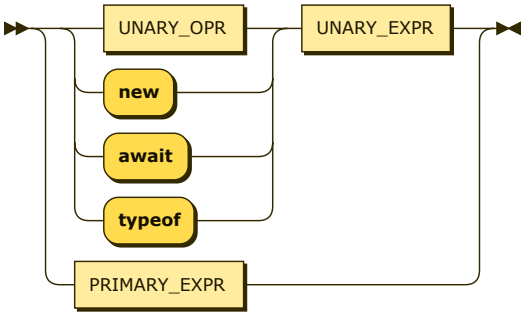



```
PIPE_EXPR
        ::= UNARY_EXPR ( '|>' UNARY_EXPR )*
```

referenced by:

- POW_EXPR

UNARY_EXPR:

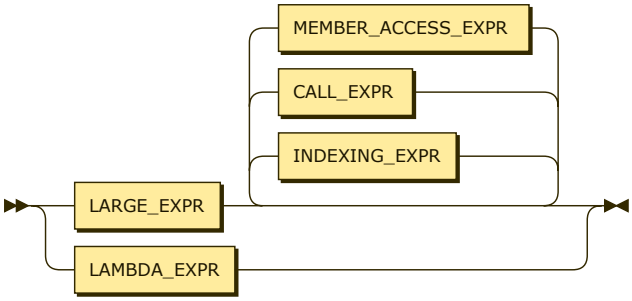



```
UNARY_EXPR
          ::= ( UNARY_OPR | 'new' | 'await' | 'typeof' ) UNARY_EXPR
            | PRIMARY_EXPR
```

referenced by:

- PIPE_EXPR
- UNARY_EXPR

PRIMARY_EXPR:

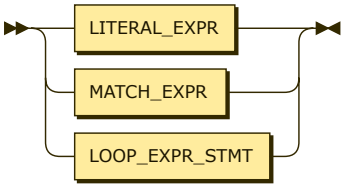



```
PRIMARY_EXPR
          ::= LAMBDA_EXPR
            | LARGE_EXPR ( INDEXING_EXPR | CALL_EXPR | MEMBER_ACCESS_EXPR )*
```

referenced by:

- UNARY_EXPR

LARGE_EXPR:

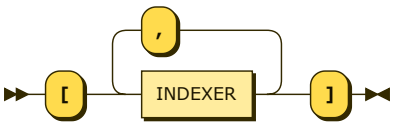



```
LARGE_EXPR
          ::= LITERAL_EXPR
            | MATCH_EXPR
            | LOOP_EXPR_STMT
```

referenced by:

- PRIMARY_EXPR

INDEXING_EXPR:

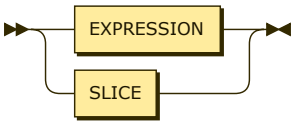



```
INDEXING_EXPR
          ::= '[' INDEXER ( ',' INDEXER )* ']'
```

referenced by:

- PRIMARY_EXPR

INDEXER:

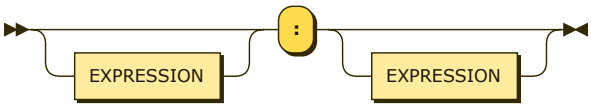



```
INDEXER  ::= EXPRESSION
           | SLICE
```

referenced by:

- INDEXING_EXPR

SLICE:

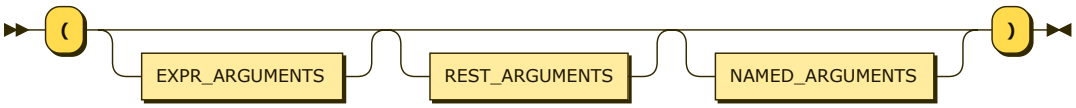



```
SLICE    ::= EXPRESSION? ':' EXPRESSION?
```

referenced by:

- INDEXER

CALL_EXPR:

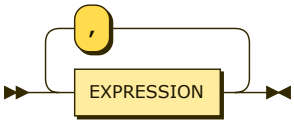



```
CALL_EXPR
         ::= '(' EXPR_ARGUMENTS? REST_ARGUMENTS? NAMED_ARGUMENTS? ')'
```

referenced by:

- DECORATOR_BDY
- PRIMARY_EXPR

EXPR_ARGUMENTS:

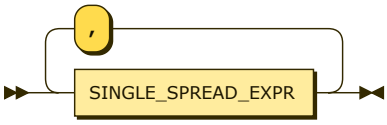



```
EXPR_ARGUMENTS
         ::= EXPRESSION ( ',' EXPRESSION )*
```

referenced by:

- CALL_EXPR

REST_ARGUMENTS:

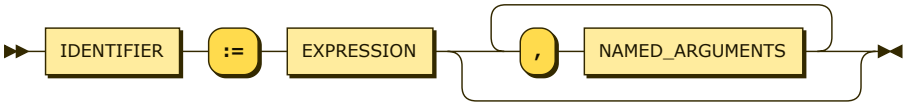



```
REST_ARGUMENTS
         ::= SINGLE_SPREAD_EXPR ( ',' SINGLE_SPREAD_EXPR )*
```

referenced by:

- CALL_EXPR

NAMED_ARGUMENTS:

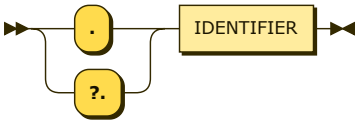



```
NAMED_ARGUMENTS
        ::= IDENTIFIER ':=' EXPRESSION ( ',' NAMED_ARGUMENTS )*
```

referenced by:

- CALL_EXPR
- NAMED_ARGUMENTS

MEMBER_ACCESS_EXPR:

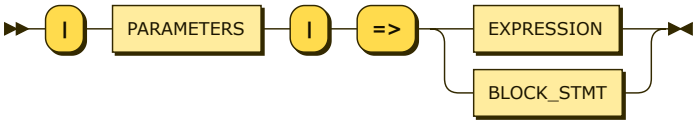



```
MEMBER_ACCESS_EXPR
        ::= ( '.' | '?.' ) IDENTIFIER
```

referenced by:

- PRIMARY_EXPR

LAMBDA_EXPR:

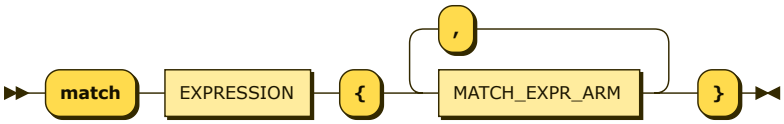



```
LAMBDA_EXPR
        ::= '|' PARAMETERS '|' '=>' ( EXPRESSION | BLOCK_STMT )
```

referenced by:

- PRIMARY_EXPR

MATCH_EXPR:

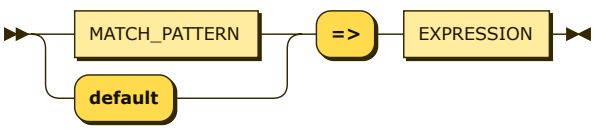



```
MATCH_EXPR
        ::= 'match' EXPRESSION '{' MATCH_EXPR_ARM ( ',' MATCH_EXPR_ARM )* '}'
```

referenced by:

- LARGE_EXPR

MATCH_EXPR_ARM:

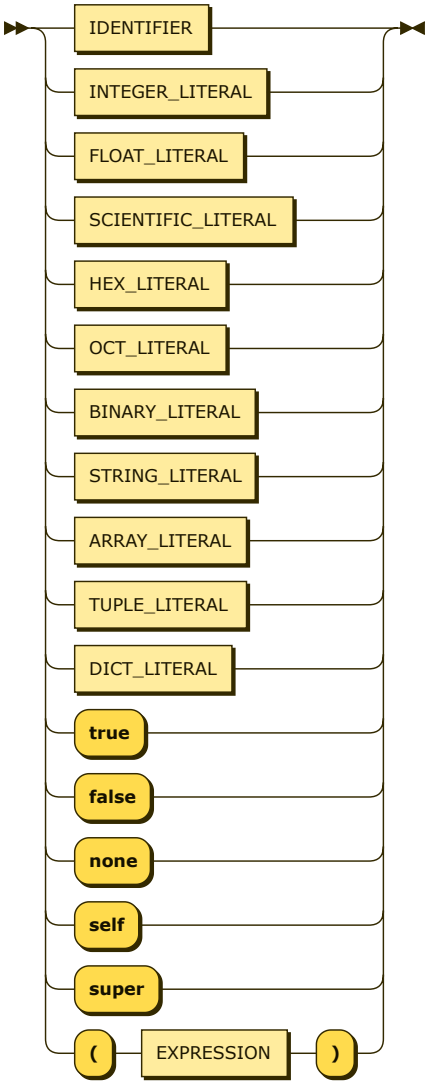



```
MATCH_EXPR_ARM
        ::= ( MATCH_PATTERN | 'default' ) '=>' EXPRESSION
```

referenced by:

- MATCH_EXPR

LITERAL_EXPR:

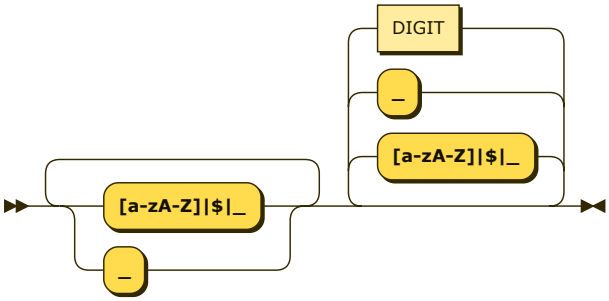



```
LITERAL_EXPR
          ::= IDENTIFIER
            | INTEGER_LITERAL
            | FLOAT_LITERAL
            | SCIENTIFIC_LITERAL
            | HEX_LITERAL
            | OCT_LITERAL
            | BINARY_LITERAL
            | STRING_LITERAL
            | ARRAY_LITERAL
            | TUPLE_LITERAL
            | DICT_LITERAL
            | 'true'
            | 'false'
            | 'none'
            | 'self'
            | 'super'
            | '(' EXPRESSION ')'
```

referenced by:

- LARGE_EXPR
- MATCH_PATTERN

IDENTIFIER:

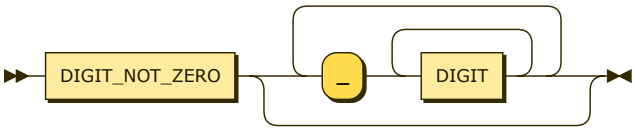



```
IDENTIFIER
        ::= ( '[a-zA-Z]|$|_' | '_' )+ ( '[a-zA-Z]|$|_' | '_' | DIGIT )*
```

referenced by:

- CATCH_PART
- CLASS_DECL
- DECORATOR_BDY
- DEFAULT_PARAMS
- DESTRUCT_PATTERN
- ENUM_DECL
- FOR_LOOP_HEAD
- FUNC_DECL
- IDENTIFIER_LIST
- KEY_VAL_PAR
- LITERAL_EXPR
- MEMBER_ACCESS_EXPR
- NAMED_ARGUMENTS
- OPERATOR_OVERLOAD
- REST_PARAM
- VAR_CONST_DECL
- WHILE_LOOP_STMT
- WILDCARD_IMPORT
- WITH_COND

INTEGER_LITERAL:

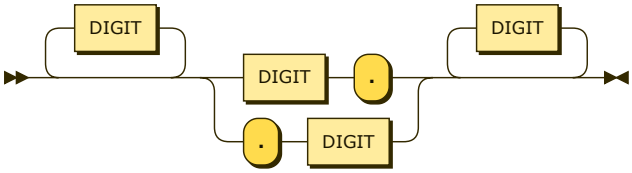



```
INTEGER_LITERAL
        ::= DIGIT_NOT_ZERO ( '_' DIGIT+ )*
```

referenced by:

- KEY_VAL_PAR
- LITERAL_EXPR
- SCIENTIFIC_LITERAL

FLOAT_LITERAL:

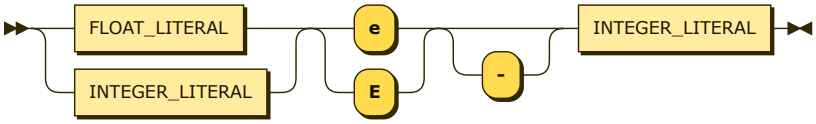



```
FLOAT_LITERAL
        ::= DIGIT* ( DIGIT '.' | '.' DIGIT ) DIGIT*
```

referenced by:

- LITERAL_EXPR
- SCIENTIFIC_LITERAL

SCIENTIFIC_LITERAL:

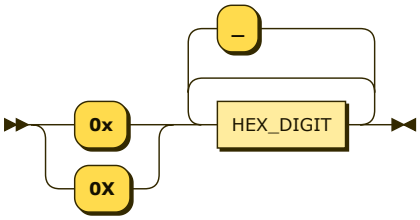



```
SCIENTIFIC_LITERAL
        ::= ( FLOAT_LITERAL | INTEGER_LITERAL ) ( 'e' | 'E' ) '-'? INTEGER_LITERAL
```

referenced by:

- LITERAL_EXPR

HEX_LITERAL:

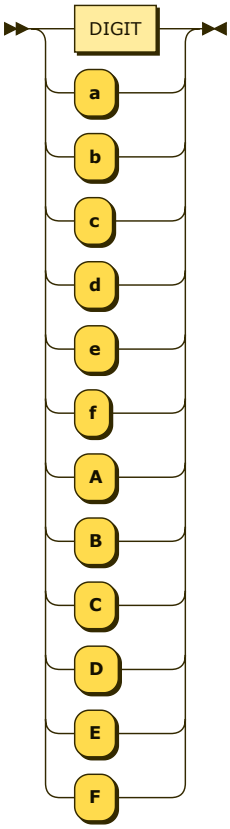



```
HEX_LITERAL
        ::= ( '0x' | '0X' ) HEX_DIGIT ( '_'? HEX_DIGIT )*
```

referenced by:

- KEY_VAL_PAR
- LITERAL_EXPR

HEX_DIGIT:



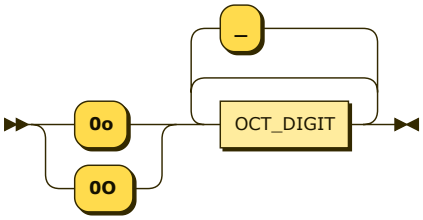

```
HEX_DIGIT
        ::= DIGIT
          | 'a'
          | 'b'
          | 'c'
          | 'd'
          | 'e'
          | 'f'
          | 'A'
```

```
          | 'B'
          | 'C'
          | 'D'
          | 'E'
          | 'F'
```

referenced by:

- HEX_LITERAL

OCT_LITERAL:

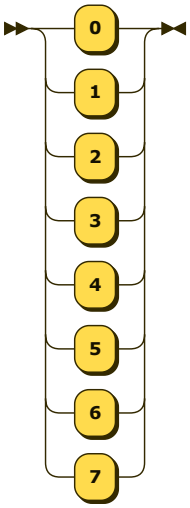



```
OCT_LITERAL
         ::= ( '0o' | '0O' ) OCT_DIGIT ( '_'? OCT_DIGIT )*
```

referenced by:

- KEY_VAL_PAR
- LITERAL_EXPR

OCT_DIGIT:

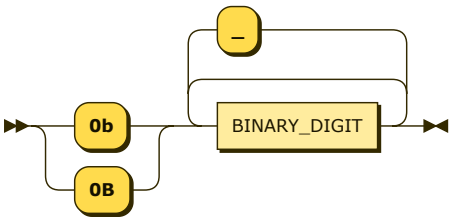



```
OCT_DIGIT
         ::= '0'
           | '1'
           | '2'
           | '3'
           | '4'
           | '5'
           | '6'
           | '7'
```

referenced by:

- OCT_LITERAL

BINARY_LITERAL:

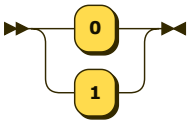


```
BINARY_LITERAL
        ::= ( '0b' | '0B' ) BINARY_DIGIT ( '_'? BINARY_DIGIT )*
```

referenced by:

- KEY_VAL_PAR
- LITERAL_EXPR

BINARY_DIGIT:

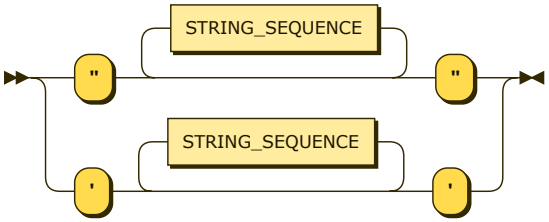



```
BINARY_DIGIT
        ::= '0'
          | '1'
```

referenced by:

- BINARY_LITERAL

STRING_LITERAL:

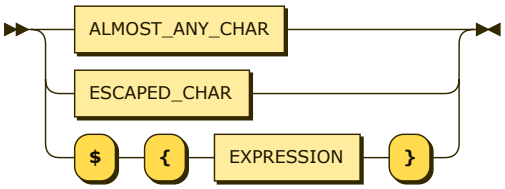



```
STRING_LITERAL
        ::= '"' STRING_SEQUENCE* '"'
          | "'" STRING_SEQUENCE* "'"
```

referenced by:

- IMPORT_DECL
- KEY_VAL_PAR
- LITERAL_EXPR

STRING_SEQUENCE:

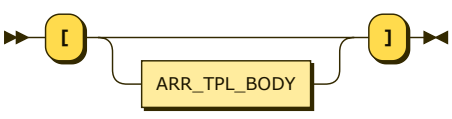



```
STRING_SEQUENCE
        ::= ALMOST_ANY_CHAR
          | ESCAPED_CHAR
          | '$' '{' EXPRESSION '}'
```

referenced by:

- STRING_LITERAL

ARRAY_LITERAL:

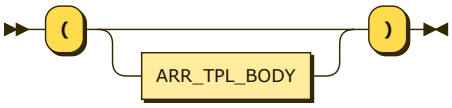



```
ARRAY_LITERAL
        ::= '[' ARR_TPL_BODY? ']'
```

referenced by:

- LITERAL_EXPR

TUPLE_LITERAL:

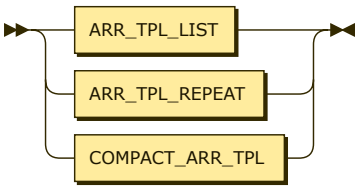



```
TUPLE_LITERAL
        ::= '(' ARR_TPL_BODY? ')'
```

referenced by:

- KEY_VAL_PAR
- LITERAL_EXPR

ARR_TPL_BODY:

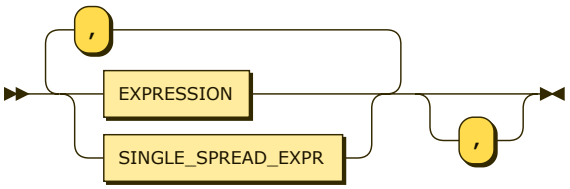



```
ARR_TPL_BODY
        ::= ARR_TPL_LIST
          | ARR_TPL_REPEAT
          | COMPACT_ARR_TPL
```

referenced by:

- ARRAY_LITERAL
- TUPLE_LITERAL

ARR_TPL_LIST:

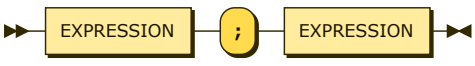



```
ARR_TPL_LIST
        ::= ( EXPRESSION | SINGLE_SPREAD_EXPR ) ( ',' ( EXPRESSION | SINGLE_SPREAD_EXPR ) )* ','?
```

referenced by:

- ARR_TPL_BODY

ARR_TPL_REPEAT:

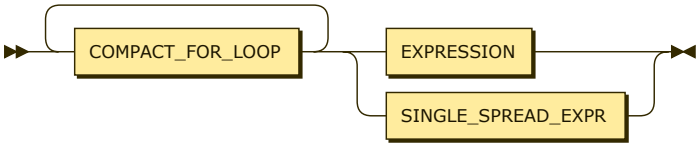



```
ARR_TPL_REPEAT
        ::= EXPRESSION ';' EXPRESSION
```

referenced by:

- ARR_TPL_BODY

COMPACT_ARR_TPL:

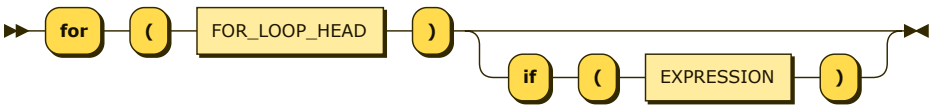



```
COMPACT_ARR_TPL
        ::= COMPACT_FOR_LOOP+ ( EXPRESSION | SINGLE_SPREAD_EXPR )
```

referenced by:

- ARR_TPL_BODY

COMPACT_FOR_LOOP:

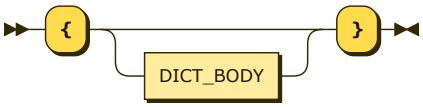



```
COMPACT_FOR_LOOP
        ::= 'for' '(' FOR_LOOP_HEAD ')' ( 'if' '(' EXPRESSION ')' )?
```

referenced by:

- COMPACT_ARR_TPL
- COMPACT_DICT

DICT_LITERAL:

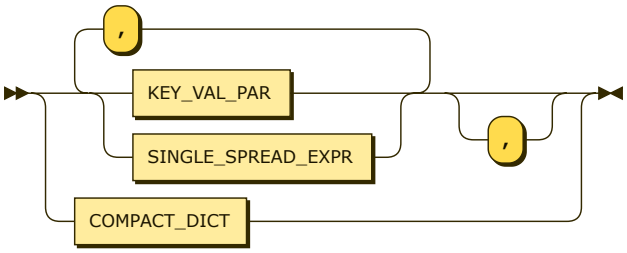



```
DICT_LITERAL
        ::= '{' DICT_BODY? '}'
```

referenced by:

- LITERAL_EXPR

DICT_BODY:

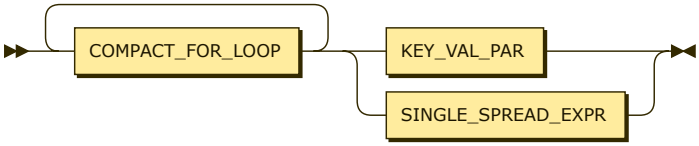



```
DICT_BODY
        ::= ( KEY_VAL_PAR | SINGLE_SPREAD_EXPR ) ( ',' ( KEY_VAL_PAR | SINGLE_SPREAD_EXPR ) )* ','?
          | COMPACT_DICT
```

referenced by:

- DICT_LITERAL

COMPACT_DICT:

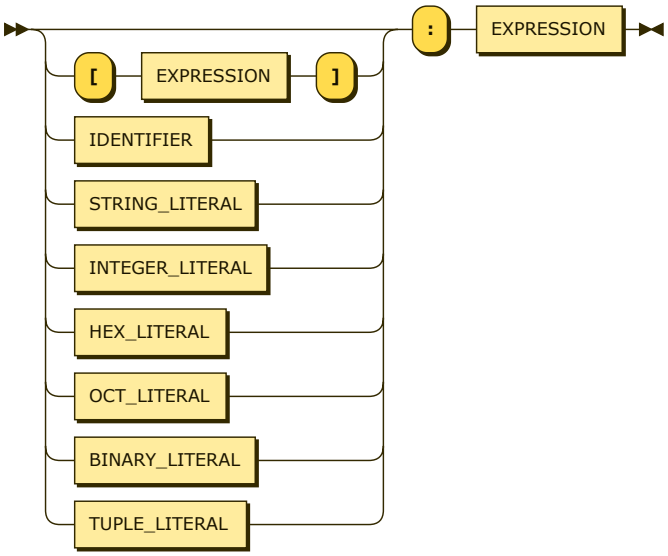



```
COMPACT_DICT
        ::= COMPACT_FOR_LOOP+ ( KEY_VAL_PAR | SINGLE_SPREAD_EXPR )
```

referenced by:

- DICT_BODY

KEY_VAL_PAR:




```
KEY_VAL_PAR
         ::= ( '[' EXPRESSION ']' | IDENTIFIER | STRING_LITERAL | INTEGER_LITERAL | HEX_LITERAL | OCT_LITERAL | BINARY_LITERAL |
             TUPLE_LITERAL )? ':' EXPRESSION
```

referenced by:

- COMPACT_DICT
- DICT_BODY

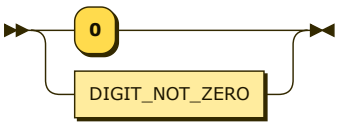
SINGLE_SPREAD_EXPR:


```
SINGLE_SPREAD_EXPR
         ::= '...' EXPRESSION
```

referenced by:

- ARR_TPL_LIST
- COMPACT_ARR_TPL
- COMPACT_DICT
- DICT_BODY
- REST_ARGUMENTS

DIGIT:

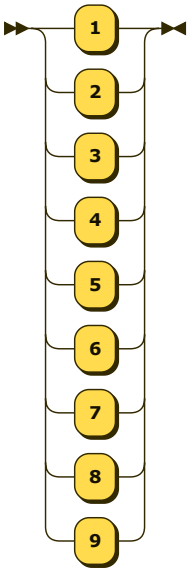



```
DIGIT    ::= '0'
           | DIGIT_NOT_ZERO
```

referenced by:

- FLOAT_LITERAL
- HEX_DIGIT
- IDENTIFIER
- INTEGER_LITERAL

DIGIT_NOT_ZERO:

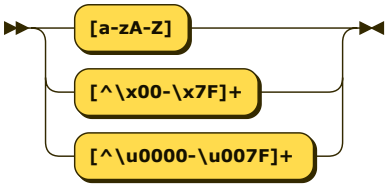



```
DIGIT_NOT_ZERO
          ::= '1'
            | '2'
            | '3'
            | '4'
            | '5'
            | '6'
            | '7'
            | '8'
            | '9'
```

referenced by:

- DIGIT
- INTEGER_LITERAL

ALMOST_ANY_CHAR:

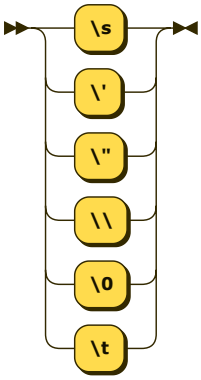



```
ALMOST_ANY_CHAR
          ::= '[a-zA-Z]'
            | '[^\x00-\x7F]+'
            | '[^\u0000-\u007F]+'
```

referenced by:

- STRING_SEQUENCE

ESCAPED_CHAR:



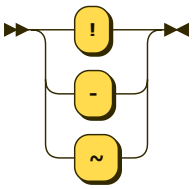

```
ESCAPED_CHAR
```

```
          ::= '\s'
            | "\'"
            | '\"'
            | '\\'
            | '\0'
            | '\t'
```

referenced by:

- STRING_SEQUENCE

UNARY_OPR:

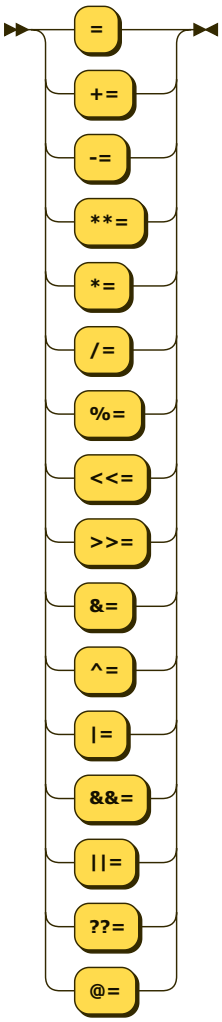



```
UNARY_OPR
          ::= '!'
            | '-'
            | '~'
```

referenced by:

- OPERATOR_OVERLOAD
- UNARY_EXPR

ASSIGNMENT_OPR:



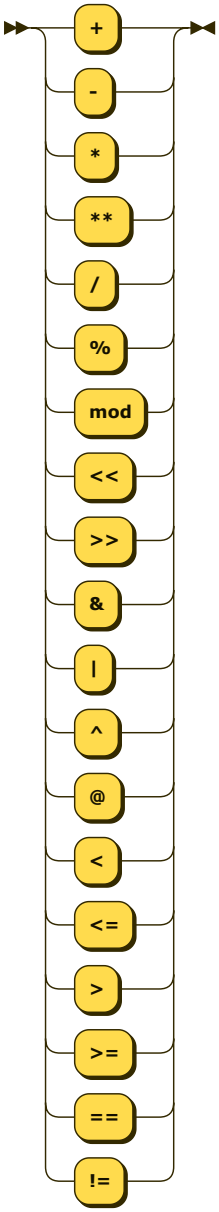

```
ASSIGNMENT_OPR
          ::= '='
            | '+='
```

```
              | '-='
              | '**='
              | '*='
              | '/='
              | '%='
              | '<<='
              | '>>='
              | '&='
              | '^='
              | '|='
              | '&&='
              | '||='
              | '??='
              | '@='
```

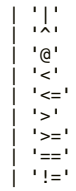
referenced by:

- REASSIGNMENT_EXPR

BNRY_OVERLOAD_OPR:




```
BNRY_OVERLOAD_OPR
          ::= '+'
            | '-'
            | '*'
            | '**'
            | '/'
            | '%'
            | 'mod'
            | '<<'
            | '>>'
            | '&'
```



referenced by:

- OPERATOR_OVERLOAD

... generated by RR - Railroad Diagram Generator