# simpL

## Rules

- program
- command
- body
- declaration
- assignment
- statement
- simple\_expr ■ compound\_expr
- if stmt
- func\_def
- param list
- arith\_expr
- bool\_expr
- bool\_operator
- arith\_operator
- basic\_type
- identifier
- primitive
- value\_keyword
- word\_keyword
- assign\_num
- assign text
- value
- text
- NUMBER
- TEXT
- EOS
- ASSIGN ■ CONV
- LPAREN
- RPAREN
- LBRACKET ■ RBRACKET
- IF ■ ELSE
- ELSE\_IF
- DEF
- RETURN ■ EQUIV
- NOT
- GT
- LT ■ LTE
- GTE
- ADD
- SUB ■ MUL
- DIV
- POW
- ID
- WS NUMERIC

program Тор

### Text notation:

```
program : command+ ;
```

#### Visual notation:

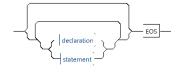


## command Top

### Text notation:

```
command : (declaration | statement)* EOS ;
```

#### Visual notation:

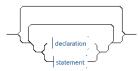


## body Top

### Text notation:

```
body : (declaration | statement)*;
```

## Visual notation:



## declaration Top

## Text notation:

```
declaration : primitive identifier | assignment ;
```

### Visual notation:



## assignment Top

#### Text notation:

```
assignment : assign_num | assign_text ;
```

### Visual notation:



# statement Top

Text notation:

```
statement : simple_expr | compound_expr ;
Visual notation:
       compound expr
simple_expr Top
Text notation:
   simple_expr : bool_expr | arith_expr ;
Visual notation:
      bool_expr
compound_expr Top
Text notation:
   compound_expr : if_stmt | func_def ;
Visual notation:
if_stmt
Text notation:
   if_stmt : If LPAREN simple_expr RPAREN LBRACKET body RBRACKET (ELSE_IF LPAREN simple_expr RPAREN LBRACKET body RBRACKET) * (ELSE_LBRACKET body RBRACKET)? ;
Visual notation:
          LPAREN
                                RPAREN
                                          LBRACKET
                                                             RBRACKET
                                                     body
                                                                                                                                                                      ELSE
                                                                                                                                                                             LBRACKET body
                                                                                                                                                                                                RBRACKET
                                                                                         LPAREN
                                                                                                                RPAREN
                                                                                                                          LBRACKET
                                                                                                                                            RBRACKET
func_def Top
Text notation:
   func_def : DEF ID LPAREN param_list RPAREN LBRACKET body RBRACKET ;
Visual notation:
                 LPAREN
                                      RPAREN
                                                LBRACKET
                                                                   RBRACKET
param_list Top
Text notation:
   param_list : (primitive ID (',' primitive ID)*)? ;
Visual notation:
arith_expr Top
```

```
Text notation:
   arith_expr : (value | ID) (arith_operator (value | ID))+ ;
Visual notation:
bool_expr Top
Text notation:
   bool_expr : value bool_operator value | identifier bool_operator identifier ;
Visual notation:
               bool_operator --- value -
   bool_operator Top
Text notation:
   bool_operator : EQUIV | NOT | GT | LT | LTE | GTE ;
Visual notation:
     EQUIV
     NOT
arith_operator Top
Text notation:
   arith_operator : ADD | SUB | MUL | DIV | POW ;
Visual notation:
     MUL
basic_type Top
Text notation:
   basic_type : text | value ;
Visual notation:
```

identifier Top

```
Text notation:
   identifier : ID ;
Visual notation:
     ID
primitive Top
Text notation:
   primitive : value_keyword | word_keyword ;
Visual notation:
     value_keyword -
value_keyword Top
Text notation:
   value_keyword : NUMBER ;
Visual notation:
     NUMBER
word_keyword Top
Text notation:
   word_keyword : TEXT ;
Visual notation:
     TEXT
assign_num Top
Text notation:
   assign_num : NUMBER? ID ASSIGN (value | arith_expr | ID) EOS ;
Visual notation:
                          ASSIGN
                                                    EOS
        NUMBER
assign_text Top
Text notation:
   assign_text : TEXT? ID ASSIGN (text | ID) EOS ;
Visual notation:
value Top
Text notation:
   value : NUMERIC ;
Visual notation:
```

NUMERIC
text Top
Text notation:
text: '\'' ID '\'';
Visual notation:
'text'
NUMBER Top
Text notation:
NUMBER: 'number';
Visual notation:
'number'
TEXT Top
Text notation:
TEXT: 'text';
Visual notation:
'text'
EOS Top
Text notation:
BOS: ';';
Visual notation:
ASSIGN Top
Text notation:
ASSIGN: '=';
Visual notation:
CONV Top
Text notation:
CONV: '\' ;
Visual notation:
LPAREN Top
Text notation:
LPAREN: '(';
Visual notation:

_	
R	RPAREN Top
Te	ext notation:
	RPAREN: ')';
Vis	isual notation:
	T Control of the cont
L	LBRACKET Top
Te	ext notation:
	LBRACKET : '{' ;
Vis	isual notation:
R	RERACKET Top
Te	ext notation:
	RBRACKET : '}' ;
Vi	isual notation:
I	ГР Тор
Te	ext notation:
	IF: 'if';
Vis	isual notation:
	TIP
E	ELSE Top
Te	ext notation:
	ELSE: 'else';
Vis	isual notation:
	'else'
	ELSE_IF Top
Te	ext notation:
	ELSE_IF: 'else if';
Vis	isual notation:
	'else if'
D	DEF Top
Te	ext notation:
	DEF: 'def';
Vis	isual notation:
	'def'
	RETURN Top

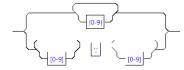
Text notation:
RETURN: 'return';
Visual notation:
'return'
EQUIV Top
Text notation:
EQUIV: 'is';
Visual notation:
NOT Top
Text notation:
NOT: 'not';
Visual notation:
'not'
GT Top
Text notation:
GT: '>' ;
Visual notation:
LT Top
Text notation:
LT : '<' ;
Visual notation:
LTE Top
Text notation:
LTE: '<=';
Visual notation:
GTE Top
Text notation:
GTE : '>=' ;
Visual notation:
'>='
ADD Top
Text notation:
ADD: '+';

Visual notation:
SUB Top
Text notation:
SUB: '-';
Visual notation:
MUL Top
Text notation:
MUL: *** ;
Visual notation:
DIV Top
Text notation:
DIV: '/';
Visual notation:
POW Top
Text notation:
POW : '^' ;
Visual notation:
ID Top
Text notation:
ID : [a-z\text{\fig2}]+[0-9a-z\text{\fig2}]*;
Visual notation:
[a-zA-Z] [0-9a-zA-Z]
WS Top
Text notation:  WS : [ \t\r\n]+ -> skip ;
Visual notation:
NUMERIC Top

# Text notation:

NUMERIC : ([0-9]+ | [0-9]+.[0-9]+) ;

### Visual notation:



Generated by: ANTLR 4 IDE. Copyright (c) 2013 Edgar Espina