grammar BKOOL;

@lexer*::header* {

from lexererr import \*

}

options {

    language = Python3;

}

program: (vardecl | funcdecl)+ EOF;

vardecl: typ idlist SEMI;

*//ex: INT A (INT B, C; FLOAT D, E){}*

funcdecl: typ ID LB paramdecl? RB body;

paramdecl: param (SEMI param)\*;

param: typ idlist;

idlist: ID (COMMA ID)\*;

typ: INT | FLOAT;

body: LBR (vardecl | stmt)\* RBR;

stmt: (assignment | call | return\_stmt) SEMI;

assignment: ID EQUAL expr;

call: ID LB (expr (COMMA expr)\*)? RB;

return\_stmt: RETURN expr;

expr: expression1;

expression1: expression2 ADD expression1 | expression2;

expression2: expression3 SUB expression2 | expression3;

expression3: expression3 MULT expression4 | expression3 DIV expression4 | expression4;

expression4: INTLIT | FLOATLIT | ID | call | LB expr RB;

INTLIT: [0-9]+;

FLOATLIT: [0-9]+ '.' [0-9]+;

*// Tokens*

INT: 'int';

FLOAT: 'float';

LB: '(';

RB: ')';

LBR: '{';

RBR: '}';

SEMI: ';';

COMMA: ',';

EQUAL: '=';

RETURN: 'return';

ADD: '+';

SUB: '-';

MULT: '\*';

DIV: '/';

ID: [a-zA-Z]+;

WS: [ \t\r\n] -> skip;

ERROR\_CHAR: . {raise ErrorToken(self.text)};