```
from ply import lex
import ply.yacc as yacc

tokens = (
    'PLUS',
    'MINUS',
    'TIMES',
    'DIV',
    'LPAREN',
    'RPAREN',
    'NUMBER',
)
t_ignore = ' \t'
t_PLUS =  r'\+'
t_MINUS = r'-'
t_TIMES = r'\*'
t_DIV = r'/'
t_LPAREN = r'\('
t_RPAREN = r'\)'


def t_NUMBER(t):
    r'[0-9]+'
    t.value = int(t.value)
    return t


def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)


def t_error(t):
    print("Invalid Token:", t.value[0])
    t.lexer.skip(1)


lexer = lex.lex()
precedence = (
    ('left', 'MINUS', 'PLUS'),
    ('left', 'TIMES', 'DIV'),
    ('nonassoc', 'UMINUS')
)


def p_sub(p):
    'expr : expr MINUS expr'
    p[0] = p[1] - p[3]

def p_add(p):
    'expr : expr PLUS expr'
    p[0] = p[1] + p[3]




def p_expr2uminus(p):
    'expr : MINUS expr %prec UMINUS'
    p[0] = - p[2]
```

```
def p_mult_div(p):
    '''expr : expr TIMES expr
            | expr DIV expr'''
    if p[2] == '*':
        p[0] = p[1] * p[3]
    else:
        if p[3] == 0:
            print("Can't divide by 0")
            raise ZeroDivisionError('integer division by 0')
        p[0] = p[1] / p[3]


def p_expr2NUM(p):
    'expr : NUMBER'
    p[0] = p[1]


def p_parens(p):
    'expr : LPAREN expr RPAREN'
    p[0] = p[2]


def p_error(p):
    print("Syntax error in input!")


parser = yacc.yacc()
print(parser.parse("5-8*6-2")) # the input
print(parser.parse("3+5*8/2")) # the input
print(parser.parse("-7-20/5")) # the input
print(parser.parse("70+2*-30")) # the input
```