Parser - parsing Expressions

```
# Expression := Term (("+"|"-") Term)*
def parse_expression(tokens, pos):
    node, pos = parse_term(tokens, pos)
    while True:
        tok = peek(tokens, pos)
        if tok == lexer.OPPLUS:
            pos += 1
            right, pos = parse_term(tokens, pos)
            node = ast_types.AddExp(node, right)
        elif tok == lexer.OPMINUS:
            pos += 1
            right, pos = parse_term(tokens, pos)
            node = ast_types.SubExp(node, right)
        else:
            break
    return node, pos
```

Parser - parsing Terms

```
# Term := Factor (("*"|"/") Factor)*
def parse_term(tokens, pos):
    node, pos = parse_factor(tokens, pos)
    while True:
        tok = peek(tokens, pos)
        if tok == lexer.OPMUL:
            pos += 1
            right, pos = parse_factor(tokens, pos)
            node = ast_types.MulExp(node, right)
        elif tok == lexer.OPDIV:
            pos += 1
            right, pos = parse_factor(tokens, pos)
            node = ast_types.DivExp(node, right)
        else:
            break
    return node, pos
```