<<덧셈 - 곱셈이 가능한 계산기 만들기>>

<덧셈 곱셈이 가능한 EBNF 수식 구문법>

expr -> term { + term }

term -> factor { * factor }

factor -> number | (expr)

<파이선 코드 Review)

```
class Parser:
    def __init__(self, expression):
        self.tokens = list(expression.replace(" ", ""))  # 공백을 제거하고 수식을 토큰화
        self.current_token = None  # 현재 토큰
        self.next_token()  # 첫 번째 토큰으로 이동

    def next_token(self):
        """다음 토큰으로 이동"""
        if len(self.tokens) > 0:
            self.current_token = self.tokens.pop(0)  # 리스트의 첫 번째 토큰을 가져옴
        else:
            self.current_token = None  # 더 이상 토큰이 없으면 None으로 설정

    def match(self, token_type):
        """현재 토큰이 기대한 토큰과 일치하는지 확인하고 일치하면 소비"""
        if self.current_token == token_type:
```

```
            self.next_token()  # 토큰을 소비하고 다음 토큰으로 이동
        else:
            raise Exception(f"Unexpected token: {self.current_token}")

def parse(self):
    """수식 분석 시작"""
    return self.expr()

 def expr(self):
        """expr -> term { + term }"""
        result = self.term()  # 첫 번째 term 계산
        while self.current_token == '+':  # 덧셈이 계속 나오면 반복
            self.match('+')  # '+' 기호 소비
            result += self.term()  # 다음 term을 계산하고 더함
        return result

 def term(self):
        """term -> factor { * factor }"""
        result = self.factor()  # 첫 번째 factor 계산
        while self.current_token == '*':  # 곱셈이 계속 나오면 반복
            self.match('*')  # '*' 기호 소비
            result *= self.factor()  # 다음 factor를 계산하고 곱함
        return result
```

```
    def factor(self):
        """factor -> number | ( expr )"""
        if self.current_token == '(':
            self.match('(')  # '(' 괄호를 소비
            result = self.expr()  # 괄호 안의 수식을 처리
            self.match(')')  # ')' 괄호를 소비
            return result
        else:
            return self.number()  # 숫자를 처리

    def number(self):
        """여러 자릿수 숫자 처리"""
        result = 0
        while self.current_token is not None and self.current_token.isdigit():
            result = result * 10 + int(self.current_token)
            self.next_token()  # 숫자를 모두 처리
        return result

# 사용자 입력 수식을 파싱하고 결과 출력
def calculate(expression):
    parser = Parser(expression)
    result = parser.parse()
    return result
```

예시 실행

expression = "2+(3+4) * 5"

result = calculate(expression)

print(f"Result: {result}")

고민해 보자.

2+3+(4+5)*6 를 실행했을 때

1. Result 라는 변수는 몇번 값이 바뀔까 ?
2. Expr, term 은 몇번 실행 될까 ?
3. 단계별로 어떤 처리를 할까 ?