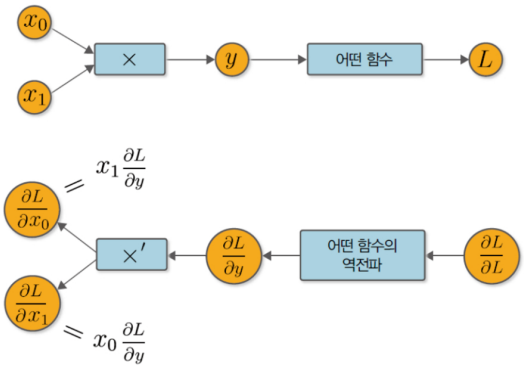
STEP20. 연산자 오버로드 (1)

(DeZero를 더 쉽게 사용하도록 개선하는 작업 필요)
연산자를 지원하도록 Variable 확장
- + 와 * 연산자를 지원하도록 함
- 곱셈을 수행하는 Mul 클래스 구현

Variable 인스턴스를 ndarray 인스턴스 처럼 사용하도록 구성
- y = a * b 처럼 코딩 할 수 있도록
- DeZero 를 평범한 넘파이 코드를 작성하듯 사용할 수 있음

Mul 클래스 구현

곱셈의 순전파와 역전파



- 역전파는 최종 출력인 L 의 미분을, 정확하게는 L 의 각 변수에 대한 미분을 전파함
  - L 은 오차, 다른 말로 손실(loss) 을 뜻함

1. Mul (Multiply)

```
class Mul(Function):
    def forward(self, x0, x1):
        y = x0 * x1
        return y

    def backward(self, gy):
        x0, x1 = self.inputs[0].data, self.inputs[1].data
        # x0 미분값 = 출력값 * x1
        # x1 미분값 = 출력값 * x0
        return gy * x1, gy * x0
```

- Mul 클래스를 파이썬 함수로 사용할 수 있게해주는 코드 추가

```
def mul(x0, x1):
    return Mul()(x0, x1)
```

연산자 오버로드

연산자 오버로드 (operator overload) == 연산자 사용 시 사용자가 설정한 함수 호출

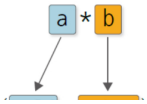
곱셈 연산자 * 오버로드

```
class Variable:
  ```

  def __mul__(self, other):
      return mul(self, other)
```

- 곱셈의 특수 메서드는 `__mul__(self, other)` 임
- `*` 연산자를 사용할 때 `__mul__` 메서드 호출

```
a = Variable(np.array(3.0))
b = Variable(np.array(2.0))
y = a * b
print(y)
```


```

```
__mul__(self, other)
```

- a * b 실행시 호출 순서
  - 먼저 인스턴스 a의 특수 메서드인 `__mul__` 이 호출됨
  - 연산자 * 왼쪽의 a가 인수 self에 전달되고, 오른쪽의 b가 other에 전달됨

  (a : 특수 메서드인 `__mul__` 호출, b : 특수 메서드인 `__mul__` 호출)