STEP22. 연산자 오버로드 (3)

표 22-1 이번 단계에서 추가할 연산자들

특수 메서드	예
__neg__(self)	–self
__sub__(self, other)	self – other
__rsub__(self, other)	other – self
__truediv__(self, other)	self / other
__rtruediv__(self, other)	other / self
__pow__(self, other)	self ** other

1. `__neg__(self)` 는 양수를 음수로, 혹은 음수를 양수로 변경하는 부호 변환 연산자 (단항 연산자, 특수 메서드 인수도 하나뿐임) ← 단항 연산자

2. 차례로 뺄셈, 나눗셈, 거듭제곱 ← 이항 연산자

- 뺄셈 | 나눗셈 : 적용 대상이 우항이냐 좌향아냐에 따라 2개의 특수 메서드 중 하나가 선별되어 호출됨
- 거듭제곱 : 좌항이 Variable 인스턴스이고 우항이 상수인 경우 고려

연산자 추가 순서
- Function 클래스를 상속하여 원하는 함수 클래스를 구현
- 파이썬 함수로 사용할 수 있도록 함
- Variable 클래스의 연산자를 오버로드함

음수 (부호 변환)

음수의 미분

```
class Neg(Function):
    def forward(self, x):
        return -x

    def backward(self, gy):
        return -gy

def neg(x):
    return Neg()(x)
```

```
Variable.__neg__ = neg
```

- 역전파 상류(출력 쪽)에서 전해지는 미분에 -1을 곱하여 하류로 흘려보내 줌
- Neg 클래스를 구현한 다음 파이썬 함수로 사용할 수 있도록 neg 함수 구현
- `__neg__` 에 neg를 대입하면 완성

뺄셈

뺄셈의 미분

```
class Sub(Function):
    def forward(self, x0, x1):
        y = x0 - x1
        return y

    def backward(self, gy):
        return gy, -gy

def sub(x0, x1):
    x1 = as_array(x1)
    return Sub()(x0, x1)

def rsub(x0, x1):
    x1 = as_array(x1)
    return Sub()(x1, x0) # x0과 x1의 순서 바꿈
```

```
Variable.__sub__ = sub
Variable.__rsub__ = rsub
```

- 역전파는 상류에서 전해지는 미분값에 1을 곱한 값이 x0의 미분 결과가 되면, -1을 곱한 값이 x1의 미분 결과가 됨
- x0와 x1이 Variable 인스턴스라면 y = x0 – x1 계산을 수행할 수 있음
- x0가 Variable 인스턴스가 아닌 경우 x의 `__rsub__` 메서드가 호출되어 정상 처리 못함
- 해결 방법 – 함수 rsub(x0, x1)을 정의하고 인수의 순서를 바꿔서 Sub()(x1, x0)를 호출

나눗셈

나눗셈의 미분

```
""" STEP22. 연산자 오버로드 ( 3 ) """


class Div(Function):
    def forward(self, x0, x1):
        y = x0 / x1
        return y

    def dackward(self, gy):
        x0, x1 = self.inputs[0].data, self.inputs[1].data
        gx0 = gy / x1
        gx1 = gy * (-x0 / x1 ** 2)
        return gx0, gx1

def div(x0, x1):
    x1 = as_array(x1)
    return Div()(x0, x1)

def rdiv(x0, x1):
    x1 = as_array(x1)
    return Div()(x1, x0) # x0과 x1의 순서 바꿈
```

```
Variable.__div__ = div
Variable.__rdiv__ = rdiv
```

- 나눗셈도 뺄셈과 마찬가지로 좌/우항 중 어느 것에 적용할지에 따라 적용되는 함수가 다름

거듭제곱

거듭제곱의 미분

```
class Pow(Function):
    def __init__(self, c):
        self.c = c

    def forward(self, x):
        y = x ** self.c
        return y

    def backward(self, gy):
        x = self.inputs[0].data
        c = self.c
        gx = c * x ** (c - 1) * gy
        return gx

def pow(x, c):
    return Pow(c)(x)
```

- c는 상수로 취급하여 따로 미분을 계산하지 않음

```
Variable.__pow__ = pow
```

- 순전파 메서드인 forward(x)는 밑에 해당하는 x만 받게 함
- 특수 메서드인 __pow__에 함수 pow를 할당함