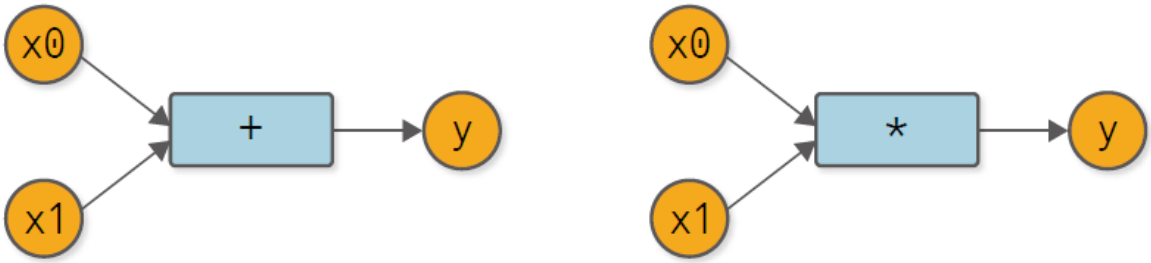
제2 고지 : 자연스러운 코드로

STEP 11 : 가변길이 인수 (순전파 편)

11.1 Function 클래스 수정

그림 11-1 '덧셈' 계산 그래프와 '곱셈' 계산 그래프(곱셈 연산은 *로 표시)



- 지금까지는 함수에 입출력 변수가 하나씩인 경우만 생각해왔지만, 함수에 따라 여러개의 입력/출력이 존재할 수 있다.
- **입력은 리스트로 바꿔서 여러개의 입력**을 받을 수 있도록, **출력은 튜플로 바꿔서 여러 개의 출력**에 대응할 수 있도록 한다.

In [ ]:

```
class Function:
    """
    Function Base Class

    """

    def __call__(self, inputs):
        ######################################
        # 여러개의 입력
        xs = [x.data for x in inputs]
        ######################################
        ys = self.forward(xs)

        ######################################
        # 여러개의 출력 및 creator 설정
        outputs = [Variable(as_array(y)) for y in ys]
        for output in outputs:
            output.set_creator(self)
        ######################################

        self.inputs = inputs
        self.outputs = outputs

        return outputs

    def forward(self, xs):
        """
        구체적인 함수 계산 담당
        """
        raise NotImplementedError()

    def backward(self, gys):
        """
        역전파
```

```
        """
        raise NotImplementedError()
```

11.2 Add 클래스 구현

In [ ]:

```
class Add(Function):
    def forward(self, xs):
        x0, x1 = xs
        y = x0 + x1
        return (y,)
```

코드

In [ ]:

```
import numpy as np


def as_array(x):

    """
    0차원 ndarray / ndarray가 아닌 경우
    """
    if np.isscalar(x):
        return np.array(x)
    return x


class Variable:
    def __init__(self, data: np.ndarray) -> None:
        if data is not None:
            if not isinstance(data, np.ndarray):
                raise TypeError(f"{type(data)}은(는) 지원하지 않습니다.")
        self.data = data
        self.grad = None  # gradient
        self.creator = None  # creator

    def set_creator(self, func) -> None:
        self.creator = func

    def backward(self):
        """
        자동 역전파 (반복)
        """
        if self.grad is None:
            self.grad = np.ones_like(self.data)
        funcs = [self.creator]
        while funcs:
            f = funcs.pop()  # 1. 함수를 가져온다
            x, y = f.input, f.output  # 2. 함수의 입력 / 출력을 가져온다
            x.grad = f.backward(y.grad)  # 3. 역전파를 계산한다

            if x.creator is not None:
                funcs.append(x.creator)  # 하나 앞의 함수를 리스트에 추가한다.


class Function:
    """
    Function Base Class

    """

    def __call__(self, inputs):
```

```
        xs = [x.data for x in inputs]
        ys = self.forward(xs)
        outputs = [Variable(as_array(y)) for y in ys]

        for output in outputs:
            output.set_creator(self)
        self.inputs = inputs
        self.outputs = outputs

        return outputs

    def forward(self, xs):
        """
        구체적인 함수 계산 담당
        """
        raise NotImplementedError()

    def backward(self, gys):
        """
        역전파
        """
        raise NotImplementedError()


class Add(Function):
    def forward(self, xs):
        x0, x1 = xs
        y = x0 + x1
        return (y,)


xs = [Variable(np.array(2)), Variable(np.array(3))]
f = Add()
ys = f(xs)
y = ys[0]
print(y.data)
```

```
5
```