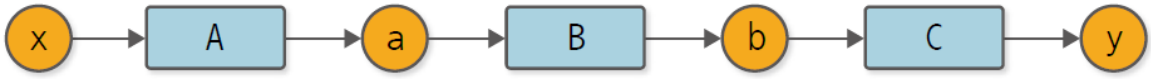
제1 고지 : 미분 자동 계산

STEP 3 : 함수 연결

그림 3-1 여러 함수를 연이어 사용하는 계산 그래프(○은 변수, □은 함수)



- 여러 함수를 순서대로 적용한다면 하나의 함수, 즉 **합성 함수(composite function)**
- $A(x) = x^2, B(x) = e^x, C(x) = x^2$ 이라고 가정한다면 $C(B(A(x))) = (e^{x^2})^2$
- 합성 함수의 이점은 복잡한 함수식 이더라도 순차적인 각 함수의 계산을 통해 계산을 쉽게 할 수 있다

In [ ]:

```
import torch
import numpy as np
import torch.nn as nn


class Variable:
    def __init__(self, data: np.ndarray) -> None:
        self.data = data

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

    def __call__(self, input: Variable) -> Variable:
        x = input.data # 입력 변수
        y = self.forward(x) # 구체적 계산
        return Variable(y) # 출력 변수

    def forward(self, x):
        """
        구체적인 함수 계산 담당
                # NOTE : 0차원의 ndarray 의 경우 np.float64로 변환되는데(넘파이가 의도
        """
        raise NotImplementedError


class Exp(Function):
    """
    y=e ^ x
    """

    def forward(self, x: np.ndarray) -> np.ndarray:
        return np.exp(x)

class Square(Function):
    """
    y= x ^ 2
    """

    def forward(self, x: np.ndarray) -> np.ndarray:
        return x**2
```

```
class Sigmoid(Function):
    """
    y = 1 / (1 + e ^(-x))
    """

    def forward(self, x: np.ndarray) -> np.ndarray:
        return 1 / (1 + np.exp(-x))


class Tanh(Function):
    """
    y= ( e^x - e^{-x} ) / ( e^x + e^{-x} )
    """

    def forward(self, x: np.ndarray) -> np.ndarray:
        return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))
```

In [ ]:

```
# Dezero
A = Square()
B = Exp()
C = Square()

x = Variable(np.array(0.5))
a = A(x)
b = B(a)
c = C(b)

print(c.data)
```

```
1.648721270700128
```

In [ ]:

```
# Dezero ~ Pytorch
## Dezero
x = Variable(np.array(1))
A = Tanh()
B = Sigmoid()
a = A(x)
b = B(a)
print(f"Dezero : {b.data}")

## Pytorch
x = torch.Tensor([1])
A = nn.Tanh()
B = nn.Sigmoid()
a = A(x)
b = B(a)
print(f"PyTorch : {b.data}")
```

```
Dezero : 0.6816997421945262
PyTorch : tensor([0.6817])
```