STEP27. 테일러 급수 미분

sin 함수 미분

sin 함수 구현

1. sin 함수

```python
import numpy as np
from dezero import Function

class Sin(Function):
    def forward(self, x):
        y = np.sin(x)
        return y

    def backward(self, gy):
        x = self.inputs[0].data
        gx = gy * np.cos(x)
        return gx

def sin(x):
    return Sin()(x)
```

- y = sin(x) 일때 그 미분 y' = cos(x)
- Sin 클래스와 sin 함수 구현은 넘파이가 제공하는 np.sin 함수와 np.cos 함수를 사용해 구현
- y값과 x의 미분 모두 0.7071067811865476로 1/np.sqrt(2)와 거의 일치

테일러 급수 이론

테일러 급수 (Taylor Series)

- 테일러 급수 == 어떤 함수를 다항식으로 근사하는 방법

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \frac{1}{3!}f'''(a)(x-a)^3 + \cdots$$

- 점 a에서 f(x)의 테일러 급수, a는 임의의 값이고 f(a)는 점 a에서 f(x)의 값
- 1차미분, 2차미분 ... 식으로 항이 무한히 계속, 어느 시점에서 중단하면 f(x)의 값을 근사함
- 항이 많이 질수록 근사의 정확도가 높음

매클로린 전개(Maclaurin's series)

- a = 0일 때의 테일러 급수

$$f(x) = f(0) + f'(0)x + \frac{1}{2!}f''(0)x^2 + \frac{1}{3!}f'''(0)x^3 + \cdots$$

- f(x) = sin(x) 를 적용하면, f'(x)= cos(x), f''(x)=-sin(x), f'''(x)=-cos(x)... sin(0)=0, cos(0)=1

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = \sum_{i=0}^{\infty}(-1)^i \frac{x^{2i+1}}{(2i+1)!}$$

테일러 급수 구현

2. 테일러 급수 식에 따라 sin 함수를 코드로 구현

```python
import math
import numpy as np
from dezero import Variable, Function
from dezero.utils import plot_dot_graph

# 테일러 급수 구현
def my_sin(x, threshold = 0.0001):    # , threshold = 1e-150)
    y = 0
    for i in range(100000):
        c = (-1) ** i / math.factorial(2 * i + 1)
        t = c * x ** (2 * i + 1)
        y += t
        if abs(t.data) < threshold:
            break
    return y
```

- 팩토리얼 계산은 파이썬의 math 모듈에 있는 math.factorial 함수를 사용

- for 문 안에서 i번째에 추가할 항목을 t로 하여 구현
- threshold로 근사치의 정밀도를 조정
  - 임곗값을 threshold로 지정, threshold 가 작을수록 정밀도가 높아짐
  - t의 절대값이 threshold 보다 낮아지면 for 문을 빠져나오게 함
- 실행 결과
  - 구현한 sin 함수와 거의 같은 결과를 얻음
  - 오차는 무시할 정도로 작고, threshold 값을 줄이면 오차를 더 줄일 수 있음