제 3고지 27단계. 테일러 급수 미분

sin 함수의 미분은 해석적으로 풀림

정공법으로 sin 함수를 구현하여 미분을 테일러 급수를 이용해 계산

- *y = sin(x)* 일때 그 미분은 *y'=cos(x)* 이다.
- np.sin과 np.cos를 이용하여 구현

```
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)
```

테일러 급수 이론

- **테일러 급수**
    - **어떤 함수 fx를 다항식으로 근사하는 방법**
    - 점 a를 기준으로 무한히 계속되는 항을 이용해 어느 시점에서 fx 값 근사가 가능
    - 당연한 말이지만 항이 많아질수록 근사의 정확도가 높음
    - a = 0 일 때의 테일러 급수를 매클로린 전개(Malclaurin's series)라고 할 수 있음

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

    - f'(x) = cos(x), f''(x) = -sin(x), f'''(x) = -cos(x), f''''(x) = sin(x), ... 의 형태가 반복되며 sin(0) = 0, cos(0) = 1이므로 다음과 같은 식을 도출할 수 있다.

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

    - i 가 커질수록 근사 정밀도가 좋아지며, 시그마 내 식의 절대값은 작아짐
    - 0에 수렴하도록 지그재그 형태가 표현되며, 일정 수준을 지나면 추가해도 함수에 영향을 미치지 않음
    - 적절한 i 값 결정이 필요함
- 구현

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