1-3. 행렬연산

1. 행렬 덧셈 (Matrix Addition)

   행렬 덧셈은 두 행렬의 같은 위치에 있는 원소끼리 더하는 연산입니다. 두 행렬 A와 B가 동일한 크기를 가질 때에만 덧셈이 가능합니다.

   예를 들어, A와 B가 둘다 m x n 행렬이라면, 덧셈 결과인 행렬 C는 다음과 같습니다.

   수학적표현

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$
$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

```
A = np.array([[a11, a12], [a21, a22]])
B = np.array([[b11, b12], [b21, b22]])

# 행렬 덧셈
C = A + B
print(C)
```

2. 행렬 곱셈 (Matrix Multiplication)

   행렬 곱셈은 두 행렬 A와 B의 곱을 통해 새로운 행렬 C를 생성하는 연산

   행렬 A는 m x n 크기, B는 n x p 크기 일 때 곱셈이 가능합니다. 결과 행렬 C는 m x p 크기를 갖습니다.

   - 행렬 A : 크기가 m x n(즉, m개의 행과 n개의 열이 있는 행렬)
   - 행렬 B : 크기가 n x p(즉, n개의 행과, p개의 열이 있는 행렬)

   행렬 곱셈이 가능하려면, 첫 번째 행렬 A의 열 개수n와 두 번째 행렬 B의 행 개수n가 동일해야 합니다.

   결과 행렬 C의 크기는 m x p가 됩니다.

   수학적 표현

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$
$$C_{11} = (1 \times 5) + (2 \times 7) = 19$$
$$C_{12} = (1 \times 6) + (2 \times 8) = 22$$
$$C_{21} = (3 \times 5) + (4 \times 7) = 43$$
$$C_{22} = (3 \times 6) + (4 \times 8) = 50$$
$$C = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

```
import numpy as np

# 행렬 A와 B 정의
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

# 행렬 곱셈
C = np.dot(A, B)
print(C)

'''
result = [[19 22]
                    [43 50]]
'''
```

3. 전치 행렬 (Transpose of a Matrix)

   전치 행렬은 행렬의 행과 열을 바꾼 새로운 행렬

   예를들어, 행렬 A의 전치 행렬 A^T는 A의 행이 열로, 열이 행으로 변환

   예를 들어, A가 다음과 같은 행렬이라고 가정해 보겠습니다.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

   행렬의 전치 A^T는 다음과 같음

$$A^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

계산식 표현

$$A_{ij}^T = A_{ji}$$

```
import numpy as np

# 행렬 A 정의
A = np.array([[1, 2, 3], [4, 5, 6]])

# 전치 행렬
A_T = A.T
print(A_T)

'''
result = [[1 4]
                    [2 5]
                    [3 6]]
'''
```

4. 역행렬 (Inverse of a Matrix)

   역행렬은 주어진 행렬을 곱했을 때, 단위행렬이 되는 행렬입니다. 역행렬은 정방 행렬에서만 존재합니다.

   즉, 행과 열의 크기가 같은 행렬에서만 역행렬을 구할 수 있습니다.

   행렬 A와 그 역행렬 A^{-1}가 있을 때, 이들의 곱은 다음과 같습니다.

$$A \cdot A^{-1} = I$$

   여기서 I는 단위 행렬로, 대각선의 값이 모두 1이고 나머지가 0인 행렬입니다.

   수학적 표현

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad A^{-1} = \frac{1}{\det(A)} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$
$$det(A) = ad - bc$$

   역행렬은 행렬식 det(A)가 0이 아닌 경우에만 존재합니다. 위 예제에서, A의 역행렬은 다음과 같이 계산됩니다.

```
import numpy as np

# 행렬 A 정의
A = np.array([[1, 2], [3, 4]])

# 역행렬 계산
A_inv = np.linalg.inv(A)
print(A_inv)

'''
result = [[-2.   1. ]
                   [ 1.5 -0.5]]
'''
```

위 예제에서 $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$의 역행렬 $A^{-1} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$

5. 행렬식 (Determinant of a Matrix)

   행렬식은 정방 행렬의 고유한 값으로, 그 행렬이 역행렬을 가질 수 있는지를 결정하는 중요한 지표입니다.

   행렬식이 0이 아닌 경우에만 역행렬이 존재합니다. 2x2 행렬의 행렬식은 다음과 같이 계산 됩니다.

$$det(A) = a \cdot d - b \cdot c$$

   여기서 $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$입니다.

   예를 들어, A가 다음과 같은 2 x 2 행렬이라고 하면

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

   행렬식은 다음과 같이 계산됩니다.

$$\det(A) = 1 \cdot 4 - 2 \cdot 3 = -2$$

$$det(A) = a_{11}a_{22} - a_{12}a_{21}$$

```
import numpy as np

# 행렬 A 정의
A = np.array([[1, 2], [3, 4]])

# 행렬식 계산
det_A = np.linalg.det(A)
print(det_A)

# result = -2.0
```

행렬 A의 행렬식은 -2로, 0이 아니므로 역행렬이 존재합니다.

요약

- **전치 행렬**: 행과 열을 서로 바꾸는 행렬 변환입니다. 수식: AT
  A^T
- **역행렬**: 행렬과 곱했을 때 단위 행렬을 만들어주는 행렬입니다. 수식: A−1
  A^{-1}
- **행렬식**: 행렬이 역행렬을 가질 수 있는지 결정하는 값입니다. 수식: det(A)
  $det(A)$