DL_1.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 3월 5일에 마지막으로 저장됨

댓글 공유 유리

+ 코드 + 텍스트 연결 Colab AI

```
import numpy as np
```

```
a = np.array([[1, 2, 3], [4, 5, 6]])
```

```
a
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
a + a
```

```
array([[ 2,  4,  6],
       [ 8, 10, 12]])
```

```
3 * a
```

```
array([[ 3,  6,  9],
       [12, 15, 18]])
```

```
type(a)
```

```
numpy.ndarray
```

```
def sumMatrix(A, B):
    A = np.array(A)
    B = np.array(B)

    return A + B
```

```
sumMatrix([1, 2, 3], [4, 5, 6])
```

```
array([5, 7, 9])
```

```
def subMatrix(A, B):
    A = np.array(A)
    B = np.array(B)

    return A - B
```

```
def mulMatrix(A, B):
    A = np.array(A)
    B = np.array(B)

    return A * B
```

```
mulMatrix([1, 2, 3], [4, 5, 6])
```

```
array([ 4, 10, 18])
```

```
def mulScalarMatrix(A, B):
    A = np.array(A)

    return A * B
```

```
mulScalarMatrix([1, 2, 3], 2)
```

```
array([2, 4, 6])
```

```
import numpy as np

class Matrix:

    def sumMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A + B

    def subMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A - B

    def mulMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A * B

    def mulScalarMatrix(A, B):
        A = np.array(A)

        return A * B
```

```
def dotMatrix(A, B):
    A = np.array(A)
    B = np.array(B)

    return A @ B
```

```
dotMatrix([[1, 2], [2, 3]], [[4, 5], [2, 3]])
```

```
array([[ 8, 11],
       [14, 19]])
```

```
import numpy as np

class Matrix:

    def sumMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A + B

    def subMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

```

```
        return A - B

    def mulMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A * B

    def mulScalarMatrix(A, B):
        A = np.array(A)

        return A * B

    def dotMatrix(A, B):
        A = np.array(A)
        B = np.array(B)

        return A @ B
```

```
def sumMatrix(A, B):
    result = []

    for i in range(len(A)):
        for j in range(len(A[i])):
            if len(A) != len(B) or len(A[i]) != len(B[i]):
                return "행렬 크기를 맞춰주세요"

    for i in range(len(A)):
        tmp = []
        for j in range(len(A[i])):
            tmp.append(A[i][j] + B[i][j])
        result.append(tmp)

    return result
```

```
print(sumMatrix([[1,2], [2,3]], [[3, 4],[5,6]]))
```

```
[[4, 6], [7, 9]]
```

```
def subMatrix(A, B):
    result = []

    for i in range(len(A)):
        for j in range(len(A[i])):
            if len(A) != len(B) or len(A[i]) != len(B[i]):
                return "행렬 크기를 맞춰주세요"

    for i in range(len(A)):
        tmp = []
        for j in range(len(A[i])):
            tmp.append(A[i][j] - B[i][j])
        result.append(tmp)

    return result
```

```
def mulMatrix(A, B):
    result = []

    for i in range(len(A)):
        for j in range(len(A[i])):
            if len(A) != len(B) or len(A[i]) != len(B[i]):
                return "행렬 크기를 맞춰주세요"

    for i in range(len(A)):
        tmp = []
        for j in range(len(A[i])):
            tmp.append(A[i][j] * B[i][j])
        result.append(tmp)

    return result
```

```
print(mulMatrix([[1,2], [2,3]], [[3, 4],[5,6]]))
```

```
[[3, 8], [10, 18]]
```

```
def test_1(A, B):
    for i in range(len(A)):
        for j in range(len(A[i])):
            if len(A) != len(B) or len(A[i]) != len(B[i]):
                return "행렬 크기를 맞춰주세요"
```

```
def mulMatrix(A, B):
    result = []

    test_1(A, B)

    for i in range(len(A)):
        tmp = []
        for j in range(len(A[i])):
            tmp.append(A[i][j] * B[i][j])
        result.append(tmp)

    return result
```

```
print(mulMatrix([[1,2, 3], [2,3, 4]], [[3, 4],[5,6]]))
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-26-34bb58cbcbdb> in <cell line: 1>()
----> 1 print(mulMatrix([[1,2, 3], [2,3, 4]], [[3, 4],[5,6]]))

<ipython-input-25-dd811e7ae002> in mulMatrix(A, B)
      7         tmp = []
      8         for j in range(len(A[i])):
----> 9             tmp.append(A[i][j] * B[i][j])
     10         result.append(tmp)
     11

IndexError: list index out of range
```

```
def scalarMulMatrix(A, B):
    result = []

    for i in range(len(A)):
        tmp = []
        for j in range(len(A[i])):
            tmp.append(A[i][j] * B)
```

```
            tmp.append(A[i][j] * B)
        result.append(tmp)

    return result
```

```
print(scalarMulMatrix([[1,2, 3], [2,3, 4]], 2))
```

```
[[2, 4, 6], [4, 6, 8]]
```

```
def mulMatrix(A, B):
    result = []

    for i in range(len(A)):
        for j in range(len(A[i])):
            if len(A[i]) != len(B):
                return "행렬 크기를 맞춰주세요"

    for i in range(len(A)):
        tmp = []
        for j in range(len(B[i])):
            tmp.append(A[i][j] * B[j][i])
        result.append(tmp)

    return result
```

```
print(mulMatrix([[1,2], [2,3], [3, 4]], [[3, 4],[5,6], [7, 8]]))
```

```
행렬 크기를 맞춰주세요
```

```
def mulMatrix(A, B):
    answer = [[0] * len(B[0]) for row in range(len(A))]

    for row in range(len(A)):
        for col in range(len(B[0])):
            sum = 0
            for i in range(len(A[0])):
                sum += (A[row][i] * B[i][col])
            answer[row][col] = sum
    return answer
```

```
mulMatrix([[1, 2], [3, 4], [5, 6]], [[2, 4], [1, 2]])
```

```
[[4, 8], [10, 20], [16, 32]]
```

```
class Matrix:

    def sumMatrix(A, B):
        result = []

        for i in range(len(A)):
            for j in range(len(A[i])):
                if len(A) != len(B) or len(A[i]) != len(B[i]):
                    return "행렬 크기를 맞춰주세요"

        for i in range(len(A)):
            tmp = []
            for j in range(len(A[i])):
                tmp.append(A[i][j] + B[i][j])
            result.append(tmp)

        return result

    def subMatrix(A, B):
        result = []

        for i in range(len(A)):
            for j in range(len(A[i])):
                if len(A) != len(B) or len(A[i]) != len(B[i]):
                    return "행렬 크기를 맞춰주세요"

        for i in range(len(A)):
            tmp = []
            for j in range(len(A[i])):
                tmp.append(A[i][j] - B[i][j])
            result.append(tmp)

        return result

    def scalarMulMatrix(A, B):
        result = []

        for i in range(len(A)):
            tmp = []
            for j in range(len(A[i])):
                tmp.append(A[i][j] * B)
            result.append(tmp)

        return result

    def mulMatrix(A, B):

        if len(A[0]) != len(B):
            return "행렬 크기를 맞춰주세요"

        answer = [[0] * len(B[0]) for row in range(len(A))]

        for row in range(len(A)):
            for col in range(len(B[0])):
                sum = 0
                for i in range(len(A[0])):
                    sum += (A[row][i] * B[i][col])
                answer[row][col] = sum
        return answer
```

```
def mulMatrix(A, B):

    if len(A[0]) != len(B):
        return "행렬 크기를 맞춰주세요"

    answer = [[0] * len(B[0]) for row in range(len(A))]

    for row in range(len(A)):
        for col in range(len(B[0])):
            sum = 0
            for i in range(len(A[0])):
                sum += (A[row][i] * B[i][col])
```

```
            answer[row][col] = sum
    return answer
```

```
mulMatrix([[1, 2, 3], [3, 4, 5], [5, 6, 7]], [[2, 4], [1, 2]])
```

'행렬 크기를 맞춰주세요'

```
코딩을 시작하거나 AI로 코드를 생성하세요.
```

Colab 유료 제품 - 여기에서 계약 취소