

DL_1.ipynb

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

연결 Colab AI

↑ ↓ ↺ ↻ ⌂ ⋮

+

코드

+

텍스트

1 import numpy as np

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

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

[] 1 a + a
array([[2, 4, 6],
 [8, 10, 12]])

[] 1 3 * a
array([[3, 6, 9],
 [12, 15, 18]])

[] 1 type(a)
numpy.ndarray

[] 1 def sumMatrix(A, B):
2 A = np.array(A)
3 B = np.array(B)
4
5 return A + B

[] 1 sumMatrix([1, 2, 3], [4, 5, 6])
array([5, 7, 9])

[] 1 def subMatrix(A, B):
2 A = np.array(A)
3 B = np.array(B)
4
5 return A - B

[] 1 def mulMatrix(A, B):
2 A = np.array(A)
3 B = np.array(B)
4
5 return A * B

[] 1 mulMatrix([1, 2, 3], [4, 5, 6])
array([4, 10, 18])

[] 1 def mulScalarMatrix(A, B):
2 A = np.array(A)
3
4 return A * B

[] 1 mulScalarMatrix([1, 2, 3], 2)
array([2, 4, 6])

[] 1 import numpy as np
2
3 class Matrix:
4
5 def sumMatrix(A, B):
6 A = np.array(A)
7 B = np.array(B)
8
9 return A + B
10
11 def subMatrix(A, B):
12 A = np.array(A)
13 B = np.array(B)
14
15 return A - B
16
17 def mulMatrix(A, B):
18 A = np.array(A)
19 B = np.array(B)
20
21 return A * B
22
23 def mulScalarMatrix(A, B):
24 A = np.array(A)
25
26 return A * B

[] 1 def dotMatrix(A, B):
2 A = np.array(A)
3 B = np.array(B)
4
5 return A @ B

[] 1 dotMatrix([[1, 2], [2, 3]], [[4, 5], [2, 3]])
array([[8, 11],
 [14, 19]])

[] 1 import numpy as np
2
3 class Matrix:
4
5 def sumMatrix(A, B):
6 A = np.array(A)
7 B = np.array(B)
8
9 return A + B
10
11 def subMatrix(A, B):
12 A = np.array(A)
13 B = np.array(B)
14

```

15         return A - B
16
17     def mulMatrix(A, B):
18         A = np.array(A)
19         B = np.array(B)
20
21         return A * B
22
23     def mulScalarMatrix(A, B):
24         A = np.array(A)
25
26         return A * B
27
28     def dotMatrix(A, B):
29         A = np.array(A)
30         B = np.array(B)
31
32         return A @ B

```

```

[ ] 1 def sumMatrix(A, B):
2     result = []
3
4     for i in range(len(A)):
5         for j in range(len(A[i])):
6             if len(A) != len(B) or len(A[i]) != len(B[i]):
7                 return "행렬 크기를 맞춰주세요"
8
9     for i in range(len(A)):
10        tmp = []
11        for j in range(len(A[i])):
12            tmp.append(A[i][j] + B[i][j])
13        result.append(tmp)
14
15    return result

```

```

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

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

```

```

[ ] 1 def subMatrix(A, B):
2     result = []
3
4     for i in range(len(A)):
5         for j in range(len(A[i])):
6             if len(A) != len(B) or len(A[i]) != len(B[i]):
7                 return "행렬 크기를 맞춰주세요"
8
9     for i in range(len(A)):
10        tmp = []
11        for j in range(len(A[i])):
12            tmp.append(A[i][j] - B[i][j])
13        result.append(tmp)
14
15    return result

```

```

[ ] 1 def mulMatrix(A, B):
2     result = []
3
4     for i in range(len(A)):
5         for j in range(len(A[i])):
6             if len(A) != len(B) or len(A[i]) != len(B[i]):
7                 return "행렬 크기를 맞춰주세요"
8
9     for i in range(len(A)):
10        tmp = []
11        for j in range(len(A[i])):
12            tmp.append(A[i][j] * B[i][j])
13        result.append(tmp)
14
15    return result

```

```

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

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

```

```

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

```

```

[ ] 1 def mulMatrix(A, B):
2     result = []
3
4     test_1(A, B)
5
6     for i in range(len(A)):
7         tmp = []
8         for j in range(len(A[i])):
9             tmp.append(A[i][j] * B[i][j])
10        result.append(tmp)
11
12    return result

```

```

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

```

```

-----
IndexError                                Traceback (most recent call last)
<ipython-input-26-34bb58cbbdb> in <cell line: 1>()
----> 1 print(mulMatrix([[1,2, 3], [2,3, 4]], [[3, 4],[5,6]]))

<ipython-input-26-d811e7ae002> 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

```

```

[ ] 1 def scalarMulMatrix(A, B):
2     result = []
3
4     for i in range(len(A)):
5         tmp = []
6         for j in range(len(A[i])):
7             tmp.append(A[i][j] * B)

```

```
7         tmp.append(A[i][j] * B[j])
8     result.append(tmp)
9
10    return result
```

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

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

```
[ ] 1 def mulMatrix(A, B):
2     result = []
3
4     for i in range(len(A)):
5         for j in range(len(A[i])):
6             if len(A[i]) != len(B):
7                 return "행렬 크기를 맞춰주세요"
8
9     for i in range(len(A)):
10        tmp = []
11        for j in range(len(B[i])):
12            tmp.append(A[i][j] * B[j][i])
13        result.append(tmp)
14
15    return result
```

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

행렬 크기를 맞춰주세요

```
[ ] 1 def mulMatrix(A, B):
2     answer = [[0] * len(B[0]) for row in range(len(A))]
3
4     for row in range(len(A)):
5         for col in range(len(B[0])):
6             sum = 0
7             for i in range(len(A[0])):
8                 sum += (A[row][i] * B[i][col])
9             answer[row][col] = sum
10    return answer
```

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

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

```
[ ] 1 class Matrix:
2
3     def sumMatrix(A, B):
4         result = []
5
6         for i in range(len(A)):
7             for j in range(len(A[i])):
8                 if len(A) != len(B) or len(A[i]) != len(B[i]):
9                     return "행렬 크기를 맞춰주세요"
10
11        for i in range(len(A)):
12            tmp = []
13            for j in range(len(A[i])):
14                tmp.append(A[i][j] + B[i][j])
15            result.append(tmp)
16
17        return result
18
19    def subMatrix(A, B):
20        result = []
21
22        for i in range(len(A)):
23            for j in range(len(A[i])):
24                if len(A) != len(B) or len(A[i]) != len(B[i]):
25                    return "행렬 크기를 맞춰주세요"
26
27        for i in range(len(A)):
28            tmp = []
29            for j in range(len(A[i])):
30                tmp.append(A[i][j] - B[i][j])
31            result.append(tmp)
32
33        return result
34
35    def scalarMulMatrix(A, B):
36        result = []
37
38        for i in range(len(A)):
39            tmp = []
40            for j in range(len(A[i])):
41                tmp.append(A[i][j] * B)
42            result.append(tmp)
43
44        return result
45
46    def mulMatrix(A, B):
47
48        if len(A[0]) != len(B):
49            return "행렬 크기를 맞춰주세요"
50
51        answer = [[0] * len(B[0]) for row in range(len(A))]
52
53        for row in range(len(A)):
54            for col in range(len(B[0])):
55                sum = 0
56                for i in range(len(A[0])):
57                    sum += (A[row][i] * B[i][col])
58                answer[row][col] = sum
59        return answer
```

```
[ ] 1 def mulMatrix(A, B):
2
3     if len(A[0]) != len(B):
4         return "행렬 크기를 맞춰주세요"
5
6     answer = [[0] * len(B[0]) for row in range(len(A))]
7
8     for row in range(len(A)):
9         for col in range(len(B[0])):
10            sum = 0
11            for i in range(len(A[0])):
12                sum += (A[row][i] * B[i][col])
13            answer[row][col] = sum
14        return answer
```

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

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

'행렬 크기를 맞춰주세요'

```
[ ] 1 코딩을 시작하거나 AI로 코드를 생성하세요.
```

[Colab 유료 제품](#) - [여기에서 계약 취소](#)

