REPORT

[과제 : NP 문제지(2)]

과 목 명	파이썬과학프로그래밍기초
교 수 명	김 병 정
학 번	20237107
작 성 자	하 태 영
제 출 일	2025.05.16

한림대학교

문제 NP15-0001

그림과 같이 2차원 배열을 변형하시오.

- 조건
  - np1 : 2차원 배열의 리스트 값 생성 ***(Nested List Comprehension 사용)***
  - np2 : 중심부의 값을 0 로 만드시오. (슬라이싱)

1	2	3	4
5	6	7	8
9	10	11	12

$\cdots \quad np1$

1	2	3	4
5	0	0	8
9	10	11	12

$\cdots \quad np2$

```
import copy  # copy 라이브러리 불러오기

# 3 행 4 열의 2 차원 리스트 생성. 각 원소는 i*4 + j + 1
np1 = [[i*4 + j + 1 for j in range(4)] for i in range(3)]

# np1 을 깊은 복사하여 np2 에 저장 (서로 독립적인 2 차원 리스트가 됨)
np2 = copy.deepcopy(np1)

# np2 의 1 행(두 번째 행)에서 1, 2 번째 열의 값을 0 으로 변경
np2[1][1:3] = [0, 0]

# np1 의 각 행(row)을 출력
for row in np1:
    print(row)
print()

# np2 의 각 행(row)을 출력
for row in np2:
    print(row)
```

```
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]

[1, 2, 3, 4]
[5, 0, 0, 8]
[9, 10, 11, 12]
```

문제 NP15-0002

그림과 같이 2차원 배열을 변형하시오.

- 조건
  - np1 : 2차원 배열의 리스트 값 생성 ***(Nested List Comprehension 사용)***
  - np2 : 상하좌우 테두리에 행과 열을 추가하는 리스트를 만드시오. (numpy 사용)

1	2	3	4
5	6	7	8
9	10	11	12

$\cdots \quad np1$

1	1	2	3	4	4
1	1	2	3	4	4
5	5	6	7	8	8
9	9	10	11	12	12
9	9	10	11	12	12

$\cdots \quad np2$

```python
import numpy as np  # numpy 라이브러리를 np 라는 이름으로 불러오기

# 3 행 4 열의 2 차원 리스트 생성. 각 원소는 i*4 + j + 1
np1 = [[i*4 + j + 1 for j in range(4)] for i in range(3)]

# np1 의 테두리를 1 칸씩 증가하여 np2 에 대입 (가장자리 값을 복제해서 확장)
np2 = np.pad(np1, pad_width=1, mode='edge')

# np2 를 출력
print(np2)
```

```
[[ 1  1  2  3  4  4]
 [ 1  1  2  3  4  4]
 [ 5  5  6  7  8  8]
 [ 9  9 10 11 12 12]
 [ 9  9 10 11 12 12]]
```

문제 NP15-0003

그림과 같이 2차원 배열을 만들고, 1차원 리스트 정보를 출력해보자.

- 조건
  - np1 : 2차원 배열의 리스트 값 생성 ***(Nested List Comprehension 사용)***
  - 왼쪽 상단위치에서 시작해서 시계방향으로의 값을 갖는 리스트를 만드시오.
  - 슬라이싱 이용
  - 리스트의 총 합을 출력하시오.

1	2	3	4
5	6	7	8
9	10	11	12

$\cdots \quad np1$

- 입출력 예

```
[1, 2, 3, 4, 8, 12, 11, 10, 9, 5]
sum 65
```

```python
import numpy as np  # numpy 라이브러리를 np라는 이름으로 불러오기

# 3행 4열의 2차원 리스트 생성. 각 원소는 i*4 + j + 1
np1 = [[i*4+j+1 for j in range(4)] for i in range(3)]

clockwise = (
    np1[0][:] +                 # np1의 0번째 행 전체 (왼쪽에서 오른쪽)
    [np1[1][3]] + [np1[2][3]] +  # np1 1번째 행 3번째 열, np1 2번째 행 3번째 열 (오른쪽 세로)
    np1[2][2::-1] +              # np1 2번째 행의 2~0번째 열 (오른쪽에서 왼쪽으로 역순)
    [np1[1][0]]                 # np1 1번째 행 0번째 열 (왼쪽 세로)
)

# np1의 각 행(row)을 출력
for row in np1:
    print(row)
print()

print(clockwise)         # 시계방향으로 추출한 리스트 출력
print(np.sum(clockwise)) # 시계방향 리스트의 총합 출력
```

```
[1, 2, 3, 4]
[5, 6, 7, 8]
[9, 10, 11, 12]

[1, 2, 3, 4, 8, 12, 11, 10, 9, 5]
65
```

문제 NP15-0004

np1 배열을 만들고, 행의합과 열의합을 순서대로 결합해서 새로운 행렬(np2) 을 출력해보자.

- 조건
  - 방법1 : numpy 배열이용 (반복문x)
  - ~~방법2 : pandas 이용~~

0	1	2	3
4	5	6	7
8	9	10	11

$\cdots \quad np1$

0	1	2	3	6
4	5	6	7	22
8	9	10	11	38
12	15	18	21	66

$\cdots \quad np3$

```
import numpy as np  # numpy 라이브러리를 np 라는 이름으로 불러오기

np1 = np.arange(12).reshape(3, 4)  # 0~11 까지 수를 3 행 4 열 배열로 생성 후 np1 에 저장

# 1. 행(row)별 합계 계산 및 추가
row_sums = np1.sum(axis=1).reshape(-1, 1)  # 각 행의 합을 계산하고 열 벡터로 변환
np3 = np.hstack((np1, row_sums))           # np1 오른쪽에 행 합계 열 추가 (가로로 연결)

# 2. 열(column)별 합계 계산 및 추가
col_sums = np3.sum(axis=0)                 # 각 열의 합 계산 (마지막 행에 추가될 값)
np3 = np.vstack((np3, col_sums))           # np3 아래쪽에 열 합계 행 추가 (세로로 연결)

# 결과 출력
print(np1) # 원본 배열
print()
print(np3) # 행/열 합계 추가된 배열 출력
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

[[ 0  1  2  3  6]
 [ 4  5  6  7 22]
 [ 8  9 10 11 38]
 [12 15 18 21 66]]
```

문제 NP16-0001

다음 그림과 같이 3차원 np1 배열을 만들고, 각 평면에 2와 4를 더한 새로운 3차원 배열을 만드는 프로그램을 작성하시오.

- 조건
  - 브로드케스팅 이용

0	1	2	3
4	5	6	7
8	9	10	11

12	13	14	15
16	17	18	19
20	21	22	23

$\cdots \quad np1$

2	3	4	5
6	7	8	9
10	11	12	13

16	17	18	19
20	21	22	23
24	25	26	27

$\cdots \quad np2$

```
import numpy as np  # numpy 라이브러리를 np 라는 이름으로 불러오기

# 0~23 까지의 숫자를 2 블록(첫 번째 축), 3 행, 4 열로 만들어 np1 에 저장
np1 = np.arange(24).reshape(2, 3, 4)

np2 = np.copy(np1)  # np1 을 복사하여 np2 에 저장 (깊은 복사)

np2[0] = np.add(np2[0], 2)  # 첫 번째 블록(0 번 인덱스)에 2 를 더한다
np2[1] = np.add(np2[1], 4)  # 두 번째 블록(1 번 인덱스)에 4 를 더한다

# 결과 출력
print(np1)  # 원본 배열 출력
print()
print(np2)  # 블록별로 값이 더해진 배열 출력
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]

[[[ 2  3  4  5]
  [ 6  7  8  9]
  [10 11 12 13]]

 [[16 17 18 19]
  [20 21 22 23]
  [24 25 26 27]]]
```