2024-1
기초 스터디

4. 리스트

목차

리스트

리스트[1] (list)

「명사」

물품이나 사람의 이름 따위를 일정한 순서로 적어 놓은 것.

- **리스트를** 작성하다.
- 승진 **리스트에** 오르다.

- 물품이나 사람의 이름 → 데이터
- 데이터를 **일정한 순서**로 적어 놓은 것

리스트

- 지금까지 언급했던 **데이터 그룹**의 일종
- 데이터들에 **‘순번’**을 매겨서 줄을 세운 형태
- 서로 다른 자료형도 함께 저장할 수 있다.

리스트 생성 (1)

- **[ ]** (대괄호) 안에 데이터들을 ‘ **,** ’ 로 나열한다.

- a = [1, 2, 3]
- b = [“hi-arc”, “하이아크”]
- c = [**2024**, “hi-arc”, “기초스터디”]

리스트 생성 (2)

- **list() 함수**를 사용한다.

- a = **list()** # 빈 리스트 생성 ([ ] 와 동일)
- b = **list(데이터그룹)** # 데이터 그룹을 리스트로 변환
- c = **list(**“하이아크”**)** # 문자열을 리스트로 변환 **(중요)**
  → [“하”, “이”, “아”, “크”]

리스트 생성 (2)

- **list() 함수 + 여러 정수 입력 받기**
  → list(map(int, input().split()))

> **map(자료형, 데이터 그룹)** : 데이터 그룹 안의 모든 데이터를 주어진 자료형으로 바꾸어 새로운 데이터 그룹으로 만든다.

- 여러 정수를 입력 받고,
  입력 받은 정수들로 하나의 리스트를 만든다.

리스트와 산술 연산

- 리스트와 리스트를 더할 수 있다.

```
>>> a = [1, 2, 3]
    b = [4, 5, 6]

>>> a + b
[1, 2, 3, 4, 5, 6]
```

리스트와 산술 연산

- 리스트에 **정수**를 곱할 수 있다.

```
>>> a = [False]
>>> a * 7
[False, False, False, False, False, False, False]
```

```
>>> a * -7
[]
>>> a * 0
[]
```

리스트 활용

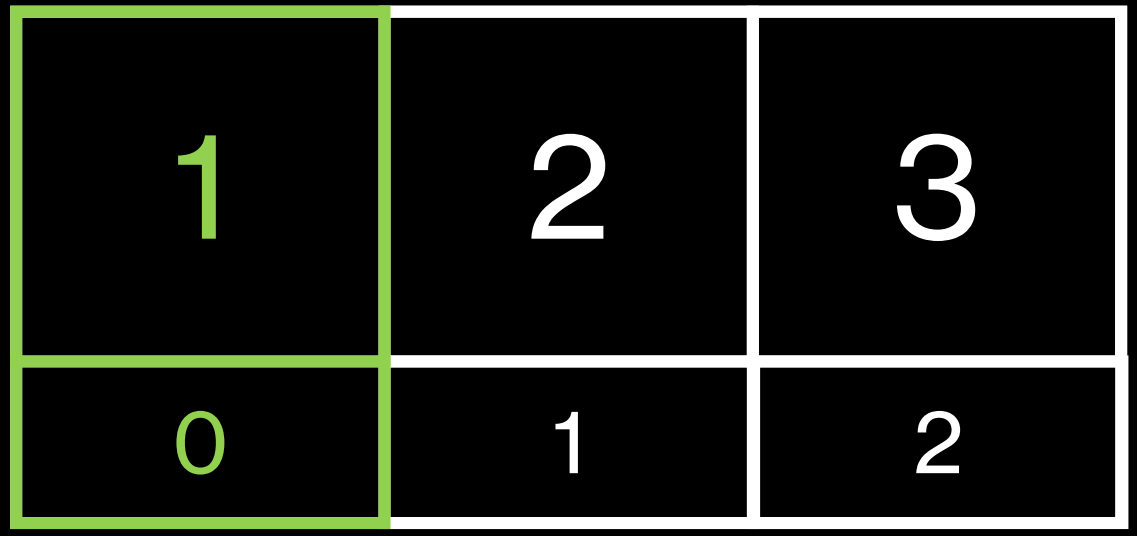
- 리스트는 아래 기능을 수행할 수 있다.

- 리스트 내 데이터 **읽기**
- 리스트 내 데이터 **수정**
- 리스트 내 데이터 **추가**
- 리스트 내 데이터 **삭제**

리스트 내 데이터 읽기

- 문자열과 동일하게 **[ ] 연산자**를 사용
- **리스트[인덱스]** 형태로 사용 (인덱스범위는 문자열과 동일)

```
>>> a = [1, 2, 3]
>>> a[0]
1
```



리스트 내 데이터 읽기

- 문자열과 동일하게 **음수 인덱스** 사용 가능

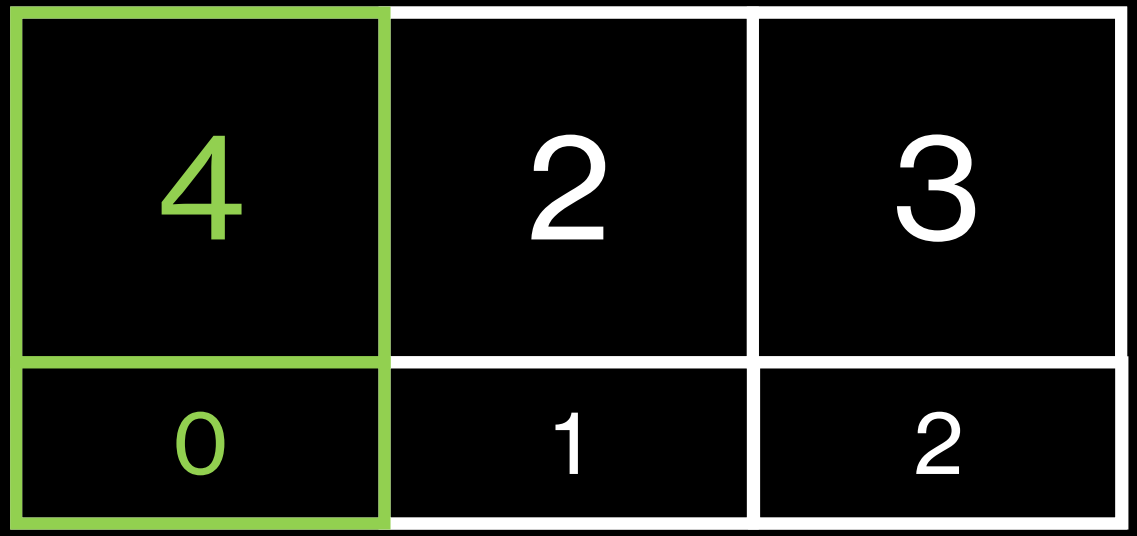
```
>>> a = [1, 2, 3]
>>> a[-2]
2
```

1	2	3
-3	-2	-1

리스트 내 데이터 수정

- 리스트[인덱스] = 수정할 데이터

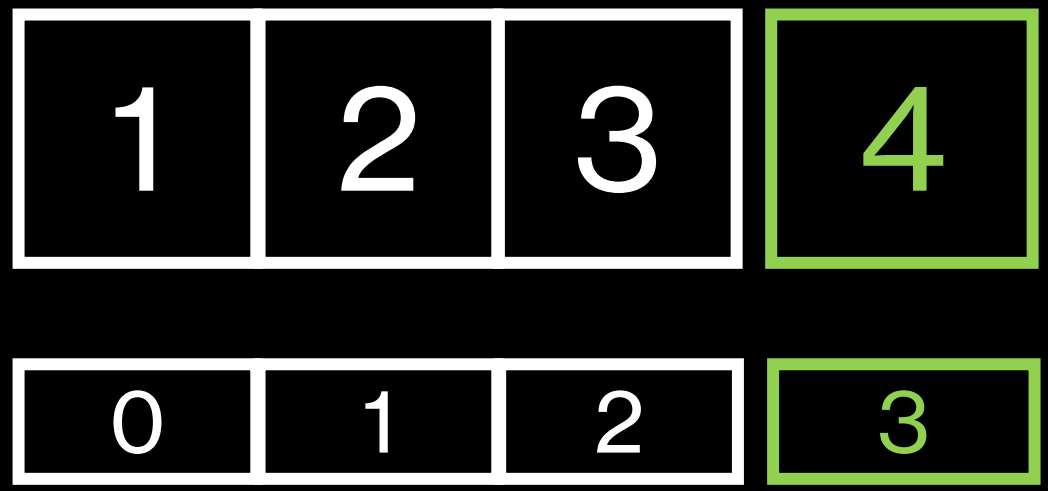
```
>>> a = [1, 2, 3]
>>> a[0] = 4
>>> a
[4, 2, 3]
```



리스트 내 데이터 추가

- **append() 메서드** 사용

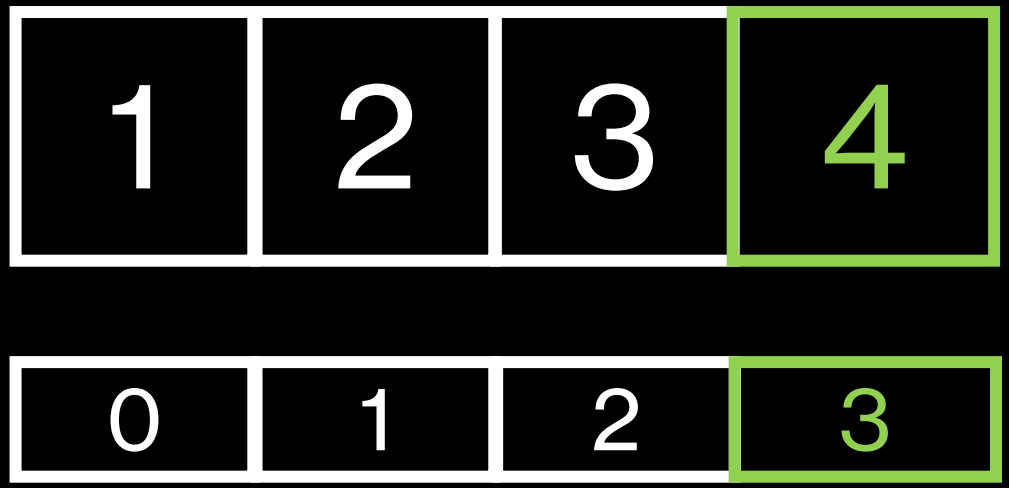
```
>>> a = [1, 2, 3]
>>> a.append(4)
>>> a
[1, 2, 3, 4]
```



리스트 내 데이터 삭제

- **pop()** 메서드 사용
- **리스트 맨 뒤**의 데이터를 빼낸다.

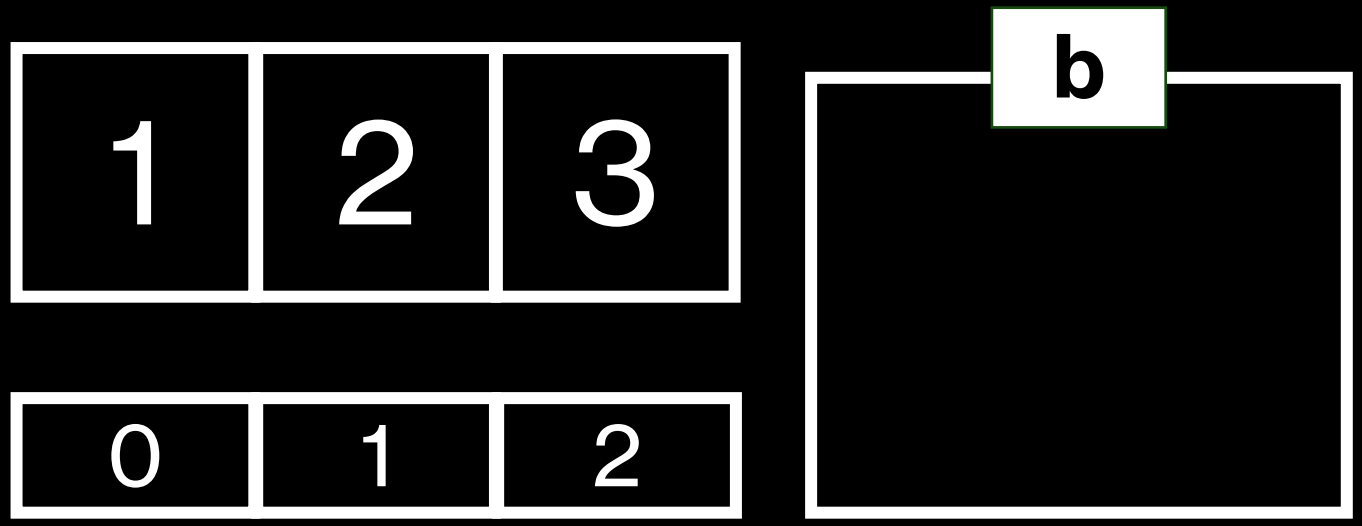
```
>>> a = [1, 2, 3, 4]
>>> a.pop()
4
>>> a
[1, 2, 3]
```



리스트 내 데이터 삭제

- **pop()** 메서드는 리스트에서 데이터를 빼고(pop), 빼낸 데이터를 **돌려준다.**

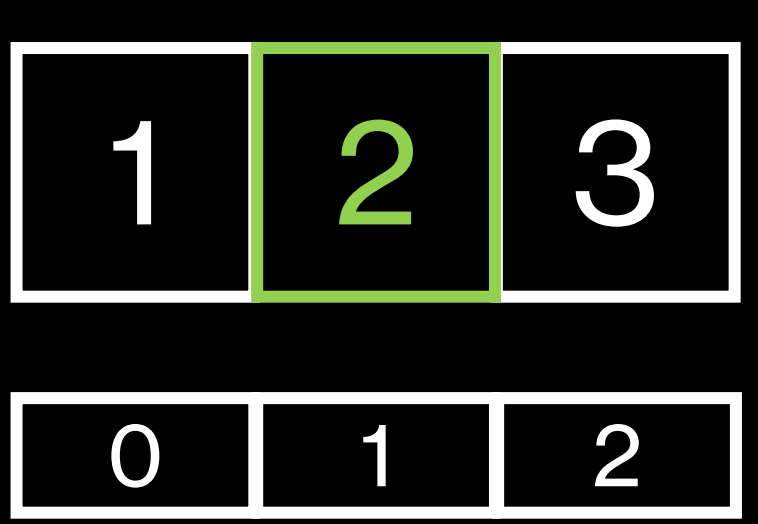
```
>>> a
[1, 2, 3]
>>> b = a.pop()
>>> a
[1, 2]
>>> b
3
```

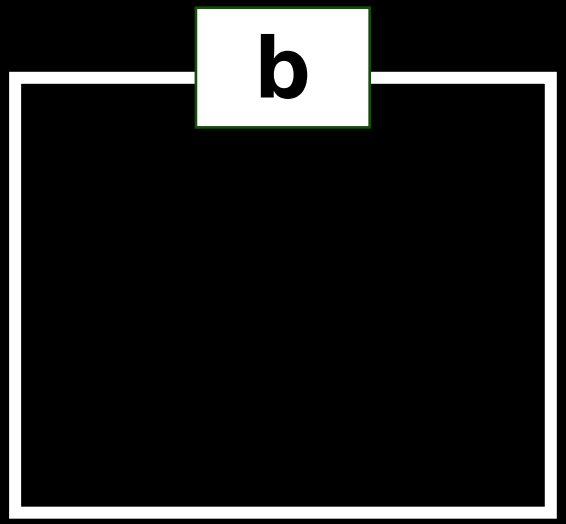


리스트 내 데이터 삭제

- **pop(인덱스)**
- 지정한 인덱스의 데이터를 빼낼 수 있다.

```
>>> a = [1, 2, 3]
>>> b = a.pop(1)
>>> a
[1, 3]
>>> b
2
```





리스트 순회

- 순회: 데이터 그룹의 모든 데이터를 한번씩 읽는 것

- 1. 인덱스로 순회하기
- 2. 데이터로 순회하기

리스트 순회 – 인덱스로 순회

- for문 + range() + 리스트 내 데이터 읽기

```
1 a = [1, 2, 3]
2 for i in range(3):
3     print(a[i])
```

```
>>> %Run ps.py
1
2
3
```

리스트 순회 – 데이터로 순회

- for문 복습

  - 데이터 그룹의 모든 데이터에 한번씩 접근

```
1 for number in [1, 2, 3]:
2     print(number)
```

리스트 순회 – 데이터로 순회

- for문 사용

```
1  a = [1, 2, 3]
2  for data in a:
3      print(data)
```

```
>>> %Run ps.py
 1
 2
 3
```

리스트 순회 – 데이터로 순회

- 데이터로 순회하는 경우, 데이터 '읽기'만 할 수 있다.
- 수정할 데이터의 **위치(인덱스)**를 알 수 없기 때문

→ 데이터를 수정할 때는 **인덱스로 순회**한다.

리스트 속 모든 데이터 출력

1. 리스트 순회 + print(data, end=‘ ’)
2. **spread 연산자**

리스트 속 모든 데이터 출력

- * 는 **spread 연산자**로 쓰이기도 한다.
- 데이터 그룹 앞에 * 을 붙여 출력하면 그룹 속 모든 데이터를 공백으로 구분하여 출력한다.

```
>>> a = [1, 2, 3]
>>> print(a)
 [1, 2, 3]
>>> print(*a)
 1 2 3
```

연습 문제

10871번 X보다 작은 수 성공

문제

정수 N개로 이루어진 수열 A와 정수 X가 주어진다. 이때, A에서 X보다 작은 수를 모두 출력하는 프로그램을 작성하시오.

예제 입력 1 복사

```
10 5
1 10 4 9 2 3 8 5 7 6
```

예제 출력 1 복사

```
1 4 2 3
```

연습 문제

- 수열 A 에서 X 보다 작은 수를 모두 구하기

→ 1. 수열 A의 모든 수를 X와 비교해보면서
2. X보다 작은 수를 찾아 리스트에 저장하고
3. 리스트의 내용을 출력하자

* 리스트에 저장 안하고 바로 띄어쓰기로 출력해도 돼요
하지만 한번 spread 연산자를 활용해 출력해볼게요

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```python
N, X = map(int, input().split())
A = list(map(int, input().split()))

answer_list = []
for number in A:
    if number < X:
        answer_list.append(number)

print(*answer_list)
```

리스트 슬라이싱

- 문자열 슬라이싱과 똑같다.

[시작 : 끝 (: 간격)]

리스트 슬라이싱 – 정정 사항

- range(a, b) 의 더 정확한 의미는
  a부터 **b를 향하는 방향**으로 **b직전**까지 나열한
  데이터 그룹을 만든다는 뜻입니다!

리스트 슬라이싱 – 정정 사항

시작	:	자르기 시작할 인덱스
끝	:	자르기를 끝낼 인덱스 **+1**
간격	:	자를 때 건너 뛸 간격 (생략 가능)

- 지난 주 내용에서도 이 내용은 **시작 < 끝 기준**입니다!
- 더 정확하게는 인덱스를 시작부터 끝까지 차례대로 나열했을 때 끝 인덱스 직전에서 끊어 슬라이싱합니다

리스트 슬라이싱 – 정정 사항

```
1  a = "01234"
2  print(a[4:2:-1])
3
4  "01234를 43210 으로 뒤집는 방향으로 슬라이싱"
5  "이때 2직전에서 멈추므로 43이 나옴"
```

```
>>> %Run -c

  43
```

리스트 슬라이싱

```
>>> a = [1, 2, 3, 4, 5]
>>> a[2:]
[3, 4, 5]
>>> a[:3]
[1, 2, 3]
```

```
>>> a[:]
[1, 2, 3, 4, 5]
>>> a[::2]
[1, 3, 5]
>>> a[::-1]
[5, 4, 3, 2, 1]
```

리스트 관련 함수

- **len(A)** : A 리스트 내 데이터 개수
- **max(A)** : A 리스트 내 데이터 중 최댓값
- **min(A)** : A 리스트 내 데이터 중 최솟값
- **sum(A)** : A 리스트 내 데이터들의 총합

```
>>> a = [1, 2, 4, 5]
```

```
>>> len(a)
4
```

```
>>> max(a)
5
```

```
>>> min(a)
1
```

```
>>> sum(a)
12
```

리스트 관련 함수

- **sorted(A)** : A 리스트를 정렬한 새 리스트 생성
- **reversed(A)** : A 리스트를 뒤집은 새 '그룹' 생성

```
>>> a = [4, 2, 1, 3]
```

```
>>> sorted(a)
[1, 2, 3, 4]
```

```
>>> print(*reversed(a))
3 1 2 4
```

리스트 관련 메소드

- **A.sort()** : A 리스트를 정렬
- **A.reverse()** : A 리스트를 뒤집기

```
>>> a = [4, 2, 1, 3]
>>> a.sort()
>>> a
[1, 2, 3, 4]
```

```
>>> a = [1, 2, 3, 4]
>>> a.reverse()
>>> a
[4, 3, 2, 1]
```

리스트 관련 메소드

- **A.count(data)** : A 리스트 내 'data' 의 개수
- **A.index(data)** : 'data' 의 가장 작은 인덱스

```
>>> a = [1, 1, 2, 3, 3]
```

```
>>> a.count(1)
2
```

```
>>> a.index(1)
0
```

```
>>> a.count(4)
0
```

```
>>> a.index(4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 4 is not in list
```

리스트 관련 메소드

- **구분문자.join(A)**
- 문자열 데이터로만 구성된 A 리스트에 대해 각 데이터 사이에 '구분문자' 를 추가하여 하나의 문자열로 만들어준다.

```
>>> "~".join(["hello","everyone","good"])
'hello~everyone~good'
```

리스트 관련 메소드

- **list(문자열) + 구분문자.join(A)**
- 문자열을 이렇게 수정할 수 있다.

```
1  s = "하이아크 1등!"
2
3  l = list(s)
4  l[5] = "2"
5
6  new_s = "".join(l)
7  print(new_s)
```

```
하이아크 2등!
```

연습 문제

1546번 평균 성공 스페셜 저지

문제

세준이는 기말고사를 망쳤다. 세준이는 점수를 조작해서 집에 가져가기로 했다. 일단 세준이는 자기 점수 중에 최댓값을 골랐다. 이 값을 M이라고 한다. 그리고 나서 모든 점수를 점수/M*100으로 고쳤다.

예를 들어, 세준이의 최고점이 70이고, 수학점수가 50이었으면 수학점수는 50/70*100이 되어 71.43점이 된다.

세준이의 성적을 위의 방법대로 새로 계산했을 때, 새로운 평균을 구하는 프로그램을 작성하시오.

연습 문제

1. 최고점 구하기
2. 최고점에 대한 상대 점수로 모든 점수 환산하기
3. 환산한 점수에 대해 평균 구하기

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```python
n = int(input())
score_list = list(map(int, input().split()))
max_score = max(score_list)
new_score_list = []

for score in score_list:
    new_score = score / max_score * 100
    new_score_list.append(new_score)

average = sum(new_score_list) / n
print(average)
```

2차원 리스트

- 리스트도 하나의 '데이터' 이므로 리스트 속에 들어갈 수 있다.

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

2차원 리스트 생성

- 리스트에 리스트를 추가하여 생성한다.

```
>>> a = []
>>> a.append([1, 2, 3])
>>> a.append([4, 5, 6])
>>> a.append([7, 8, 9])
```

→

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

2차원 리스트 데이터 읽기

- [ ] 연산자를 두 번 사용한다.

```
>>> a = [
            [1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]
        ]
>>> a[0][0]
1
```

a[0] == [1, 2, 3]

a[0][0] == [1, 2, 3][0] == 1

2차원 리스트 데이터 순회

- **for문**을 중첩하여 사용한다.

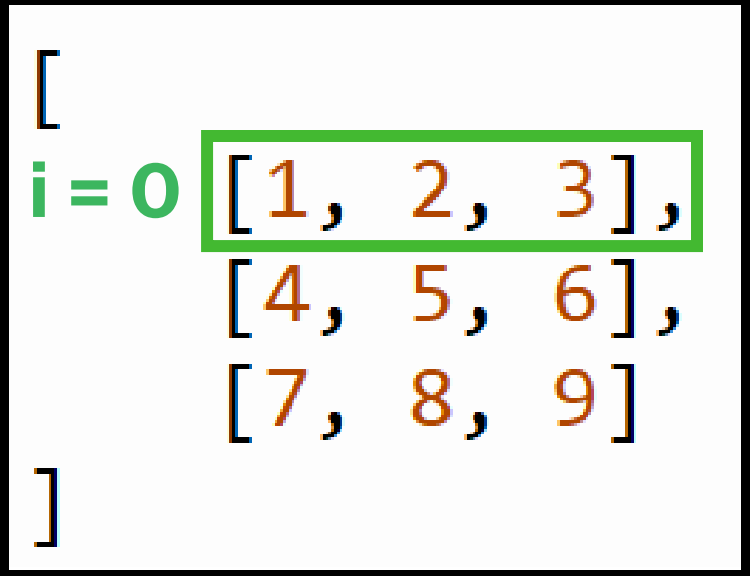
```
>>> a = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]
```

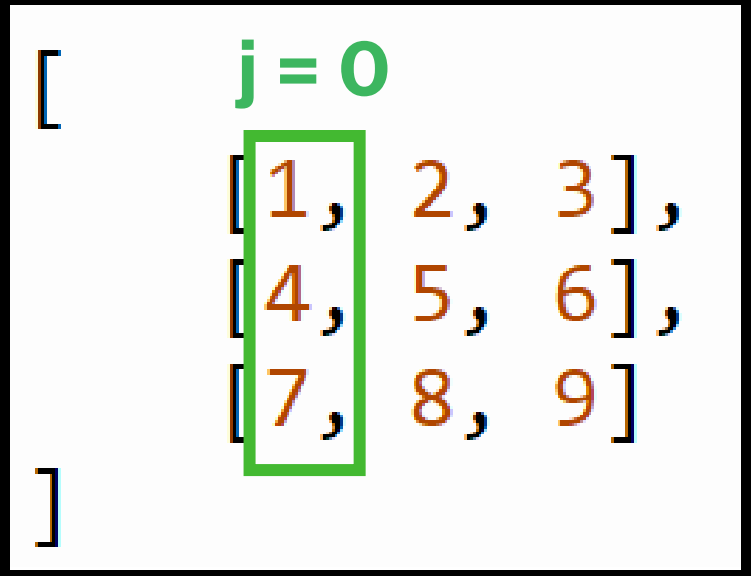
```
for i in range(3):
    for j in range(3):
        print(a[i][j])
```

```
1
2
3
4
5
6
7
8
9
```

2차원 리스트 데이터 순회

- **A[i][j]** 형태로 접근한다.
- 보통 **i** 는 **행(가로)**, **j** 는 **열(세로)**로 사용한다.





2차원 데이터 입력 받기

- 1차원 리스트 형태로 입력 받은 뒤,
  이를 새로운 리스트에 추가하는 과정을 반복

```
1 a = []
2 for _ in range(3):
3     l = list(map(int, input().split()))
4     a.append(l)
```

연습 문제

2738번 행렬 덧셈 성공

문제

N*M크기의 두 행렬 A와 B가 주어졌을 때, 두 행렬을 더하는 프로그램을 작성하시오.

예제 입력 1 복사

```
3 3
1 1 1
2 2 2
0 1 0
3 3 3
4 4 4
5 5 100
```

예제 출력 1 복사

```
4 4 4
6 6 6
5 6 100
```

연습 문제

- 행렬 : 수를 직사각형 모양으로 배열한 것
- 행렬 덧셈 : 같은 위치에 있는 수끼리 더하기

$$\begin{pmatrix} a1 & a2 \\ a3 & a4 \end{pmatrix} + \begin{pmatrix} b1 & b2 \\ b3 & b4 \end{pmatrix} = \begin{pmatrix} a1+b1 & a2+b2 \\ a3+b3 & a4+b4 \end{pmatrix}$$

연습 문제

- 행렬 덧셈 예시 (실제로 이렇게 동작하지는 않습니다!)

```
[[1, 2],     +   [[1, 2],     =   [[1 + 1, 2 + 2],
 [3, 4]]          [3, 4]]          [3 + 3, 4 + 4]]

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

연습 문제

- 2차원 리스트를 이용하여 풀어봅시다.

연습 문제

- 정답 코드

```python
N, M = map(int, input().split())
A, B = [], []
for i in range(N): # 행렬 A에 대해 N 개 가로줄(행)의 입력이 주어진다.
    A.append(list(map(int, input().split())))

for j in range(N): # 행렬 B에 대해 N 개 가로줄(행)의 입력이 주어진다.
    B.append(list(map(int, input().split())))

result = A # A를 결과에 먼저 담아두고, B를 순회하면서 읽은 값을 A에 더해주기
for i in range(N):
    for j in range(M):
        # i 는 행(가로줄), j는 열(세로줄) 이므로, i 는 N번, j 는 M번 반복합니다.
        result[i][j] += B[i][j]

for i in range(N):
    print(*result[i])
```

얕은 복사 vs 깊은 복사

- 리스트 복사
- 과연 b는 a와 완전히 다른 복사본일까?

```
1  a = [1, 2, 3]
2  b = a
3
4  print(a)
5  print(b)
```

```
>>> %Run -c $EDI

  [1, 2, 3]
  [1, 2, 3]
```

얕은 복사 vs 깊은 복사

- b 리스트에서의 수정 작업이 a 리스트에도 적용된다.

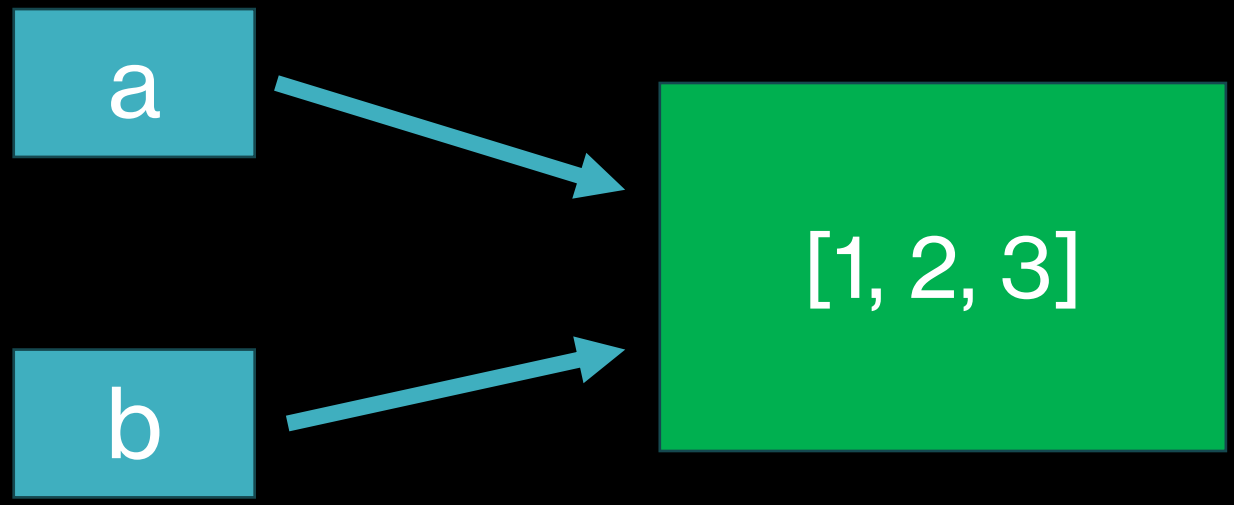
```
1  a = [1, 2, 3]
2  b = a
3
4  b[0] = 99
5  print(a)
6  print(b)
```

```
>>> %Run -c $EDITOR_CONTENT

  [99, 2, 3]
  [99, 2, 3]
```

얕은 복사 vs 깊은 복사

- 사실 a, b는 모두 같은 리스트를 가리키고 있다.



얕은 복사 vs 깊은 복사

- **얕은 복사** : 데이터 그룹을 복사할 때, 참조만 복사
- **깊은 복사** : 데이터 그룹을 복사할 때,
  그룹 안 **모든 데이터를 복사**

cf) 참조 : 데이터를 가리키는 데이터
2081... 이라는 **id**는 [1, 2, 3] 의 참조 값(id)이다.

```
1  a = [1, 2, 3]
2  print(id(a))
```

```
2081452492096
```

얕은 복사 vs 깊은 복사

- 얕은 복사 예시

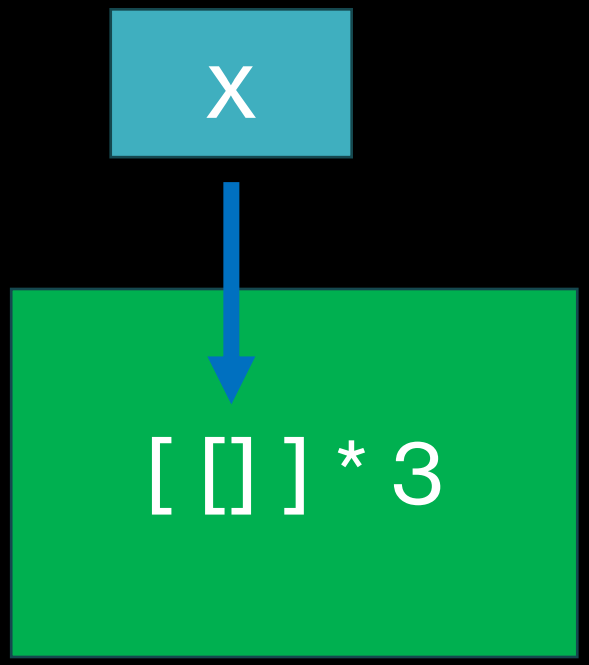
```
1  a = [[]] * 3
2  print(a)
```

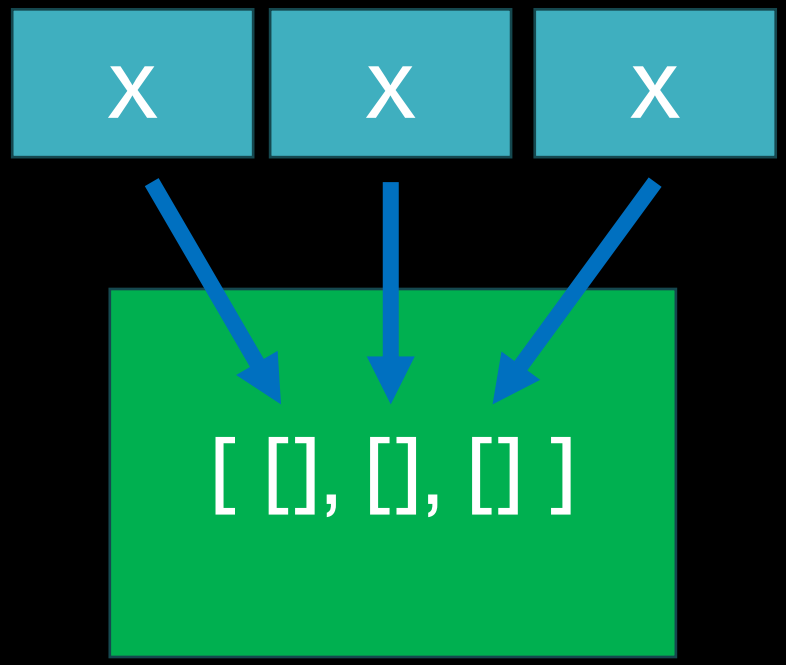
```
[[], [], []]
```

```
4  a[0].append(1)
5  print(a)
```

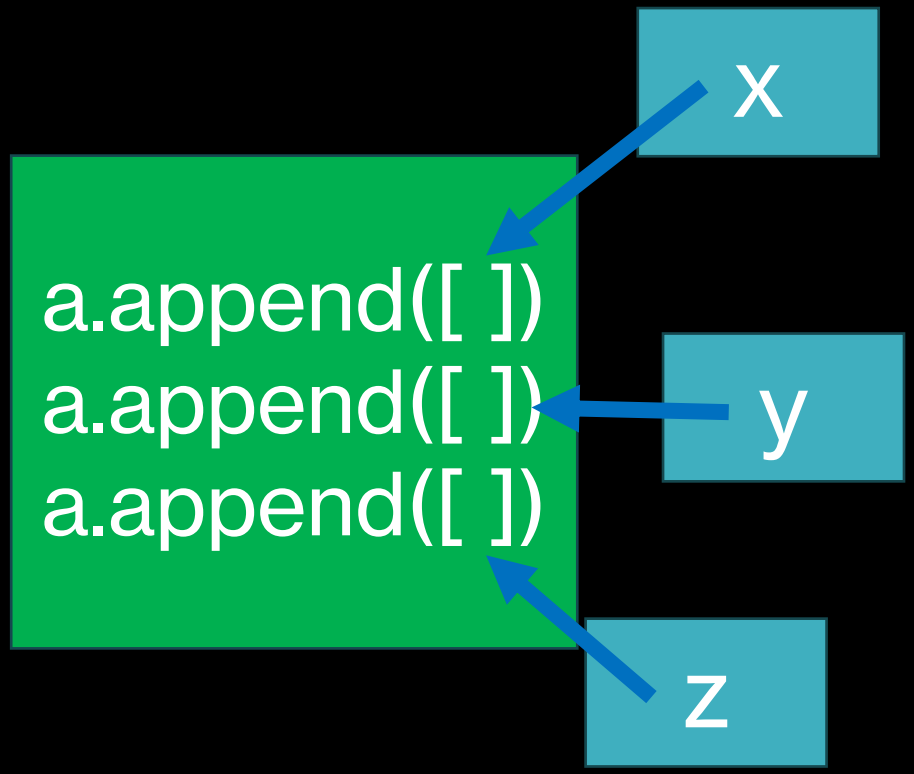
```
[[1], [1], [1]]
```

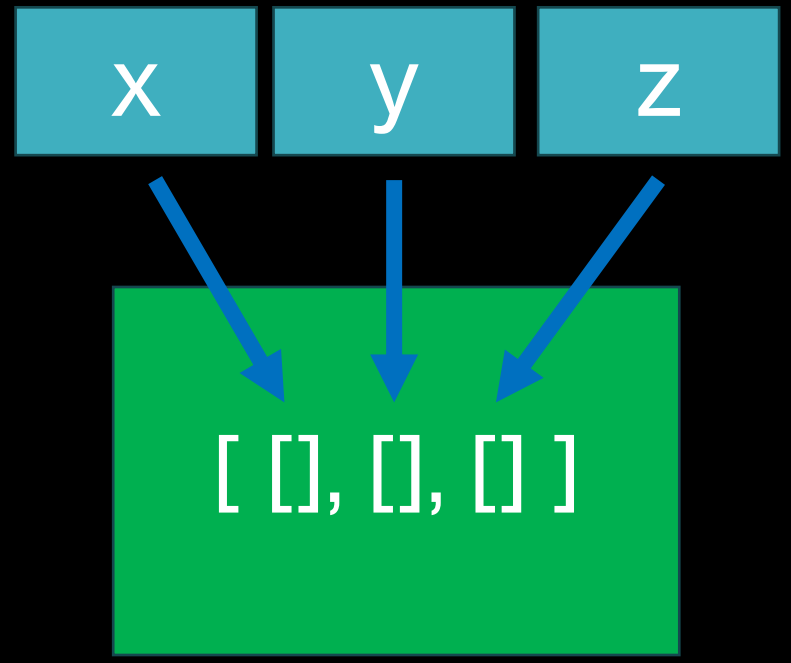
얕은 복사 vs 깊은 복사





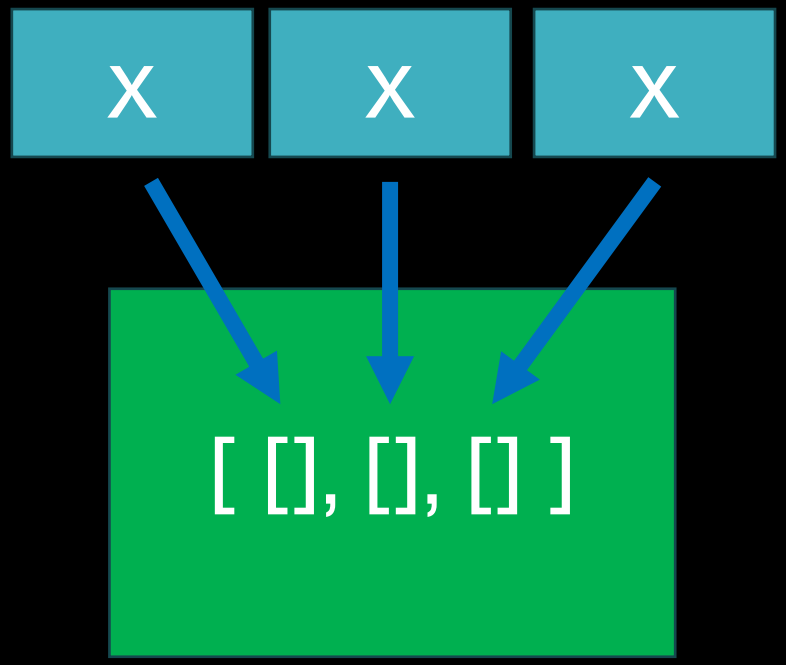
얕은 복사 vs 깊은 복사

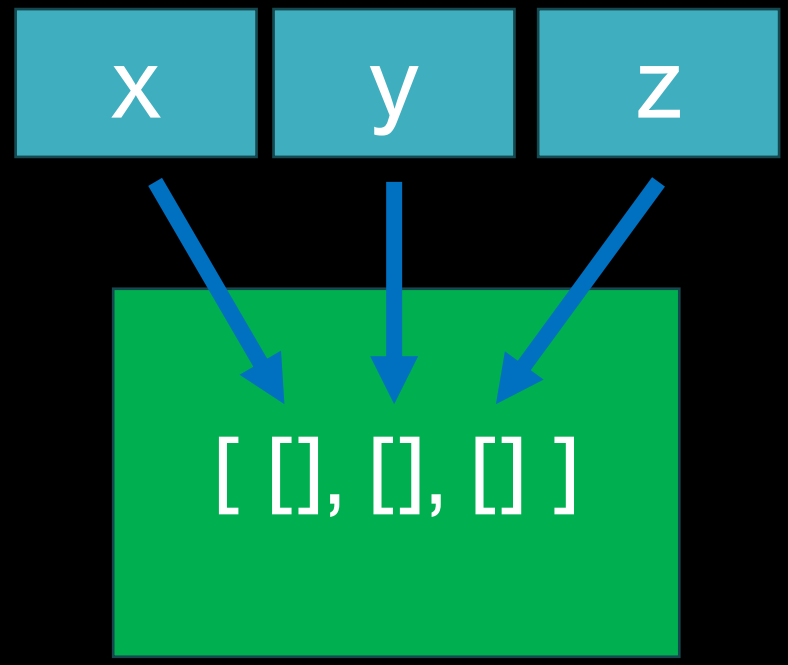




얕은 복사 vs 깊은 복사

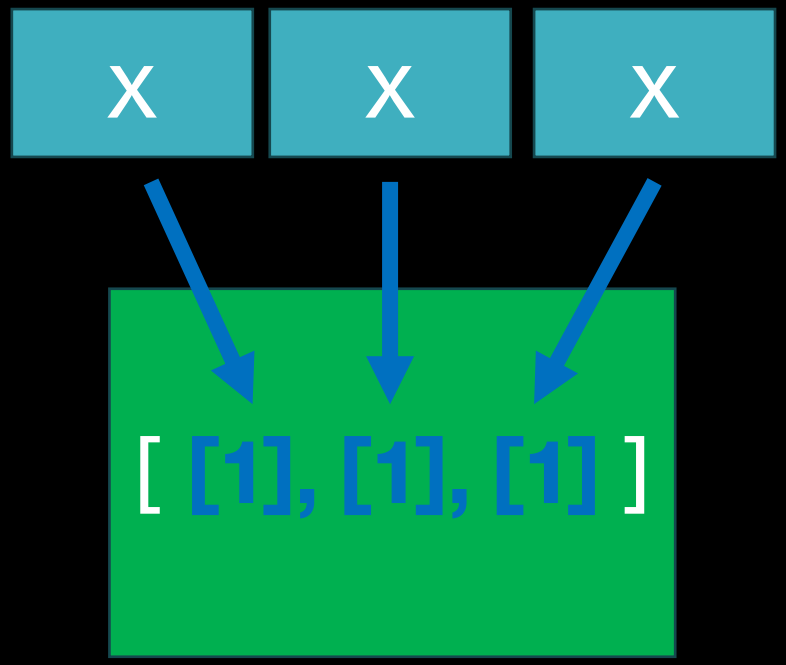
- a[0].append(1) → 첫 번째 리스트(x)에 1을 추가

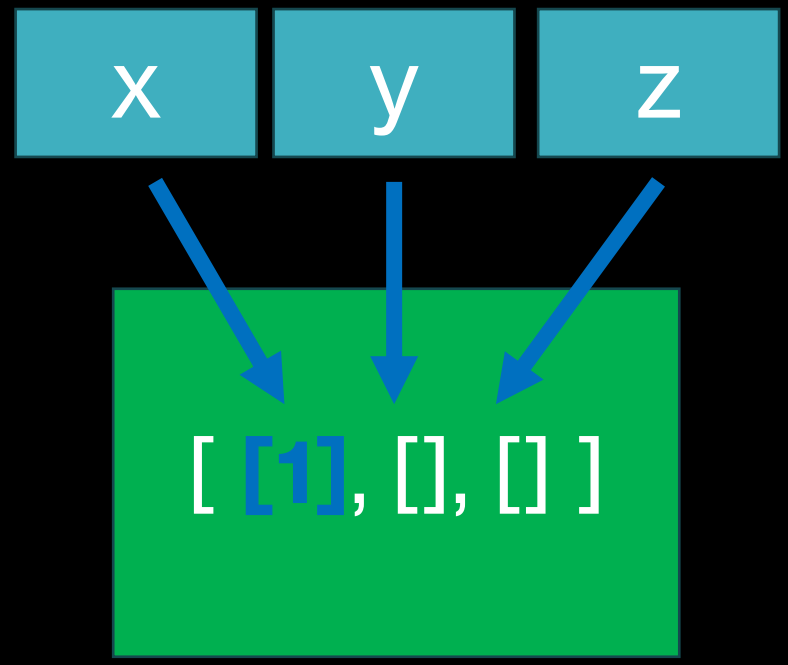




얕은 복사 vs 깊은 복사

- a[0].append(1) → 첫 번째 리스트(x)에 1을 추가





얕은 복사 vs 깊은 복사

- 깊은 복사를 하려면, **'직접 복사'** 하면 된다.

```
1  a = [1, 2, 3]
2  b = []
3  for i in range(3):
4      v = a[i]
5      b.append(v)
6
7  b.append(4)
8  print(a)
9  print(b)
```

```
>>> %Run -c $EDI

  [1, 2, 3]
  [1, 2, 3, 4]
```

얕은 복사 vs 깊은 복사

- 직접 복사하기 어렵다면 **deepcopy 함수**를 사용한다.

```
1  from copy import deepcopy
2
3  a = [1, 2, 3]
4  b = deepcopy(a)
5
6  b.append(4)
7  print(a)
8  print(b)
```

```
>>> %Run -c $ED

  [1, 2, 3]
  [1, 2, 3, 4]
```

얕은 복사 vs 깊은 복사

- **1차원 리스트**라면 **리스트 슬라이싱**을 활용하자.
  (다차원 리스트면 내부에 있는 리스트는 또 얕은 복사로 복사한다!)

```
1  a = [1, 2, 3]
2  b = a[:]
3
4  b[0] = 99
5  print(a)
6  print(b)
```

```
>>> %Run -c $EDI

  [1, 2, 3]
  [99, 2, 3]
```

너무 어려워요..

- 어려운 내용 맞아요. 하지만 매우 중요한 내용이에요.
- 이걸 모르면 문제를 풀 때 의도와 다르게 데이터가 들어가고, 바뀌고, 지워질 수 있습니다.

너무 어려워요..

- 결론만 딱 정리하자면..

- 2차원 이상의 리스트를 만들거나,
  리스트를 복사할 때는
  빈 리스트를 만들어서 **직접 하나하나 넣자!**

더 자세하게 알고 싶어요

- 파이썬 객체(데이터)는 **mutable**(가변), **immutable**(불변)로 분류됩니다.
- mutable 객체는 얕은 복사를 하고,
  immutable 객체는 깊은 복사를 합니다.

- https://blockdmask.tistory.com/576

리스트 축약

- 리스트를 생성하면서 초기 데이터도 넣는 문법
- 리스트 안에 반복문을 작성한다.

리스트 축약

- [“반복마다 넣을 데이터” for 변수 in 데이터그룹]

리스트 축약

- ex) False 가 5개 들어있는 리스트 생성하기
  → **[False for _ in range(5)]**

ex) 1부터 5까지 들어있는 리스트 생성하기
→ **[(i+1) for i in range(5)]**

리스트 축약으로 2차원 리스트 만들기

- 행렬 덧셈 문제를 리스트 축약으로 입력 받아봅시다.

리스트 축약으로 2차원 리스트 만들기

```
A, B = [], []
for i in range(N): # 행렬 A에 대해 N 개 가로줄(행)의 입력이 주어진다.
    A.append(list(map(int, input().split())))
```

반복마다 넣을 데이터

```
a = [ list(map(int, input().split())) for _ in range(N) ]
```

반복마다 넣을 데이터

리스트 축약으로 2차원 리스트 만들기

- 리스트 축약으로 만든 리스트를
  리스트 축약의 데이터로 다시 넣을 수 있습니다.

```
a = [     [False for _ in range(5)]     for _ in range(5)]
```

리스트 축약으로 2차원 리스트 만들기

- 처음에는 이해하기 어려울 수 있습니다.
- 그래도 익숙해지면 나중에 정말 편해져요!
- 리스트 축약은 반복문으로 데이터를 생성하기 때문에 **깊은 복사**하는 효과도 있어요

정리 – 리스트 생성

- **[ ]** 대괄호 이용하기 : [1, 2, 3]
- **list()** 함수 이용하기 : list() 는 빈 리스트
- **list(데이터 그룹)** : 데이터 그룹 → 리스트
- **list(map(int, input().split()))**

  → 정수 여러 개 입력 받아서 리스트로 만들기

정리 – 리스트 연산

- 덧셈 : **[1, 2, 3] + [4, 5]** = [1, 2, 3, 4 5]
- 곱셈 : **[False] * 3** = [False, False, False]

정리 – 리스트 조작

- A = [1, 2, 3, 4]
- 읽기 : **A[인덱스]**
- 수정 : **A[인덱스] = 바꿀 데이터**
- 추가 : **A.append(데이터)** # 맨 뒤에 추가
- 삭제 : **A.pop()** # 맨 뒤에 삭제

정리 – 리스트 순회

- A = [1, 2, 3]
- 1. 데이터 순회
- **for data in A:**
  print(data)

- A = [1, 2, 3]
- 2. 인덱스 순회
- **for i in range(len(A)):**
  print(A[i])

정리 – 리스트 메소드 / 함수

- **len(**리스트**)**
- **max(**리스트**)**
- **min(**리스트**)**
- **sum(**리스트**)**

- 리스트**.sort()**
- 리스트**.count()**
- 리스트**.reverse()**

- **sorted(**리스트**)**
- **reversed(**리스트**)**

정리 – 2차원 리스트

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

- i → 가로 (바깥 반복)
- j → 세로 (안쪽 반복)

- [0][1] = 2
- [2][0] = 7

정리 – 얕은 복사, 깊은 복사

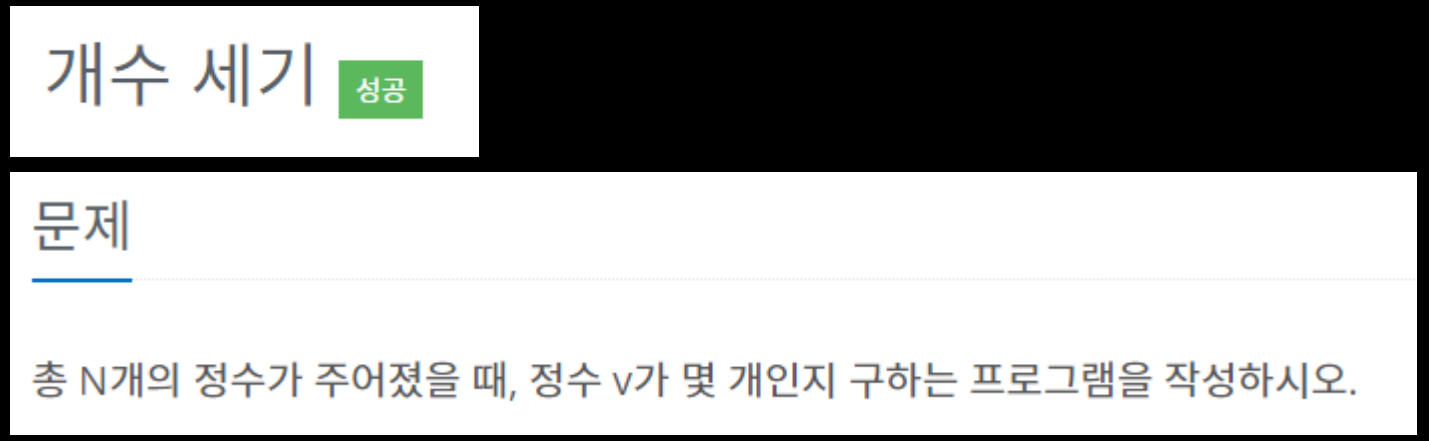
- 얕은 복사 : 데이터 그룹의 참조만 복사
- **깊은 복사 : 데이터 그룹 속 모든 데이터 복사**

정리 – 리스트 축약

- [“반복마다 넣을 데이터” for 변수 in 데이터그룹]

추가 강의 문제

- https://www.acmicpc.net/problem/10807



- 리스트를 순회하면서 v 를 발견할 때마다 count를 증가시킨다. count 함수를 사용해서 풀어도 된다.

추가 강의 문제

- 정답코드

```
1 size = int(input())
2 nums = list(map(int, input().split()))
3 find = int(input())
4 print(nums.count(find))
```

추가 강의 문제

- https://www.acmicpc.net/problem/5597

과제 안 내신 분..?

X대학 M교수님은 프로그래밍 수업을 맡고 있다. 교실엔 학생이 30명이 있는데, 학생 명부엔 각 학생별로 1번부터 30번까지 출석번호가 붙어 있다.

교수님이 내준 특별과제를 28명이 제출했는데, 그 중에서 제출 안 한 학생 2명의 출석번호를 구하는 프로그램을 작성하시오.

추가 강의 문제

- 30명 학생의 과제 제출 여부를 True, False 리스트로 저장해보자.

- **[False] * 31** 로 만들면 문제를 풀 때 더 편하다.
  (30명인데 31로 만든 이유는 인덱스가 0부터 시작하기 때문이다.)
  (False는 깊은 복사되는 데이터이므로 * 연산자로 만들어도 된다.)

추가 강의 문제

- 28명의 학생 번호를 입력 받을 때 리스트에서 해당 학생 번호의 인덱스 값을 True로 바꿔준다.

- 모든 입력을 받은 뒤, 리스트를 인덱스 1부터 차례대로 순회하면서 과제 제출 여부가 False인 번호를 순서대로 출력하자.

추가 강의 문제

- 정답 코드

```python
is_submitted = [False]*31
for _ in range(28):
    student_number = int(input())
    is_submitted[student_number] = True

for student_number in range(1, 31):
    if not is_submitted[student_number]:
        print(student_number)
```

연습문제

- 10818 최소 최대 (여러 정수 입력 + 리스트, min / max 함수)
- 10813 공 바꾸기 (리스트 생성, 리스트 읽기, 리스트 쓰기)
- 2562 최댓값 (리스트 생성, 리스트 순회 or **index()** 메서드)
- 3052 나머지 (리스트 생성, 리스트 순회 or **in 연산자** 복습)
- **10811 바구니 뒤집기** (리스트 생성, 리스트 슬라이싱 or 순회)
- 2566 최댓값 (2차원 리스트 생성, 2차원 리스트 순회)
- 10798 세로읽기 (문자열과 리스트 생성, 2차원 리스트 순회)
- **2563 색종이** (2차원 리스트 생성, 2차원 리스트 순회)

색종이 문제 힌트

- 도화지의 크기는 정해져 있고, 그 안에서만 색종이를 붙이고 있다.
- 색종이의 영역 크기를 수학적으로 계산할 필요 없이 **도화지를 나타내는 2차원 리스트**를 만들고, 색종이를 입력이 주어지는 대로 **직접 붙여보면** 된다.
- 정답은 색종이가 붙은 영역을 직접 카운트하자.