UNIT 23

2차원 리스트 사용하기

23 2차원 리스트 사용하기

2차원 리스트 사용하기

- 2차원 리스트는 다음과 같이 가로×세로 형태로 이루어져 있으며 행(row)과 열(column) 모두 0부터 시작함

▼ 그림 2차원 리스트

23.1 2차원 리스트를 만들고 요소에 접근하기

2차원 리스트를 만들고 요소에 접근하기

- 2차원 리스트는 리스트 안에 리스트를 넣어서 만들 수 있으며 안쪽의 각 리스트는 ,(콤마)로 구분함

```
리스트 = [[값, 값], [값, 값], [값, 값]]
```

- 그럼 숫자 2개씩 3묶음으로 리스트를 만들어보자

```
>>> a = [[10, 20], [30, 40], [50, 60]]
>>> a
[[10, 20], [30, 40], [50, 60]]
```

- 리스트를 한 줄로 입력했지만 가로, 세로를 알아보기 쉽게 세 줄로 입력해도 됨

```
a = [[10, 20],
     [30, 40],
     [50, 60] ]
```

23.1 2차원 리스트를 만들고 요소에 접근하기

2차원 리스트의 요소에 접근하기

- 리스트[세로인덱스][가로인덱스]
- 리스트[세로인덱스][가로인덱스] = 값

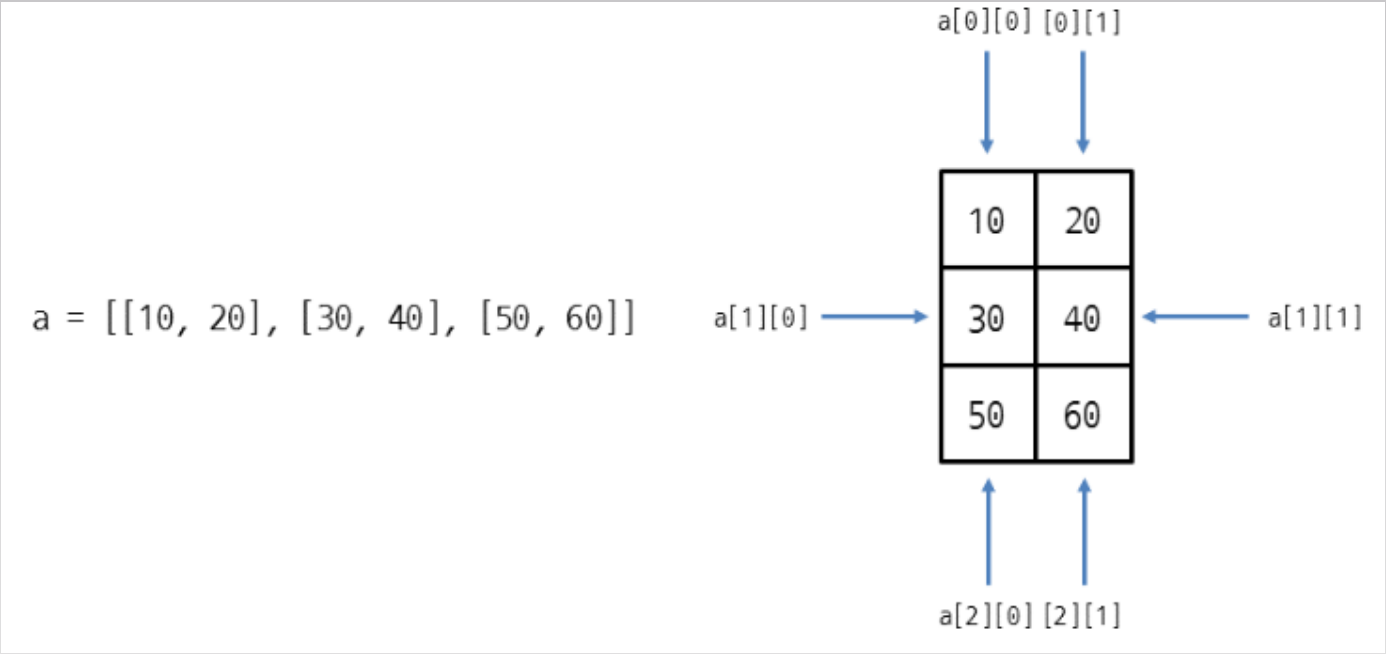
```
>>> a = [[10, 20], [30, 40], [50, 60]]
>>> a[0] [0]            # 세로 인덱스 0, 가로 인덱스 0인 요소 출력
10
>>> a[1][1]           # 세로 인덱스 1, 가로 인덱스 1인 요소 출력
40
>>> a[2][1]           # 세로 인덱스 2, 가로 인덱스 0인 요소 출력
60
>>> a[0][1] = 1000    # 세로 인덱스 0, 가로 인덱스 1인 요소에 값 할당
>>> a[0][1]
1000
```

23.1 2차원 리스트를 만들고 요소에 접근하기

2차원 리스트의 요소에 접근하기

- 2차원 리스트는 다음과 같이 가로×세로 형태로 이루어져 있으며 행(row)과 열(column) 모두 0부터 시작함

▼ 그림 인덱스로 2차원 리스트의 요소에 접근

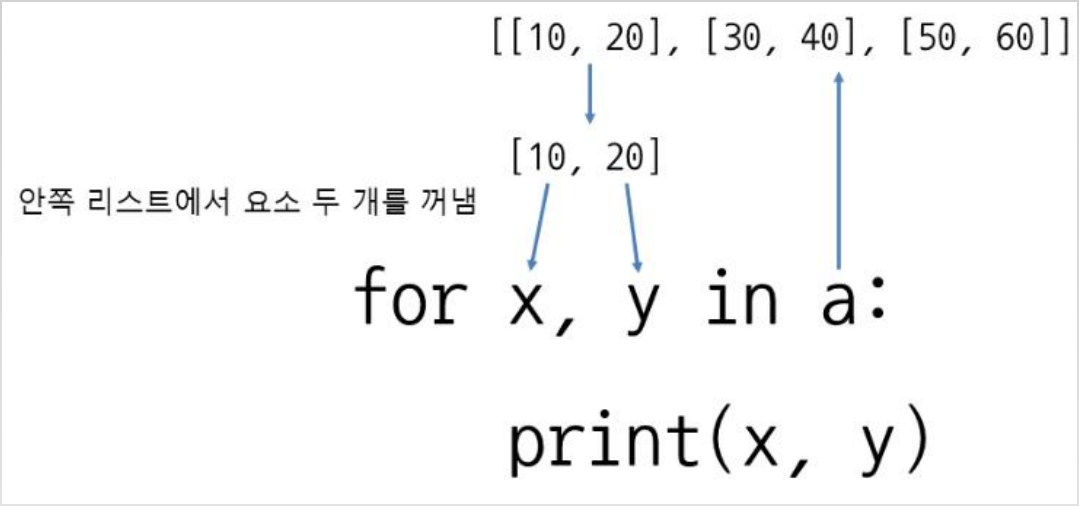


23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

반복문으로 2차원 리스트의 요소를 모두 출력하기

```
>>> a = [[10, 20], [30, 40], [50, 60]]
>>> for x, y in a:    # 리스트의 가로 한 줄(안쪽 리스트)에서 요소 두 개를 꺼냄
...     print(x, y)
...
10 20
30 40
50 60
```

▼ 그림 2차원 리스트에서 for 반복문을 한 번만 사용



23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

for 반복문을 두 번 사용하기

- 다음 내용을 IDLE의 소스 코드 편집 창에 입력한 뒤 실행해보자

two_dimensional_list_for_for.py

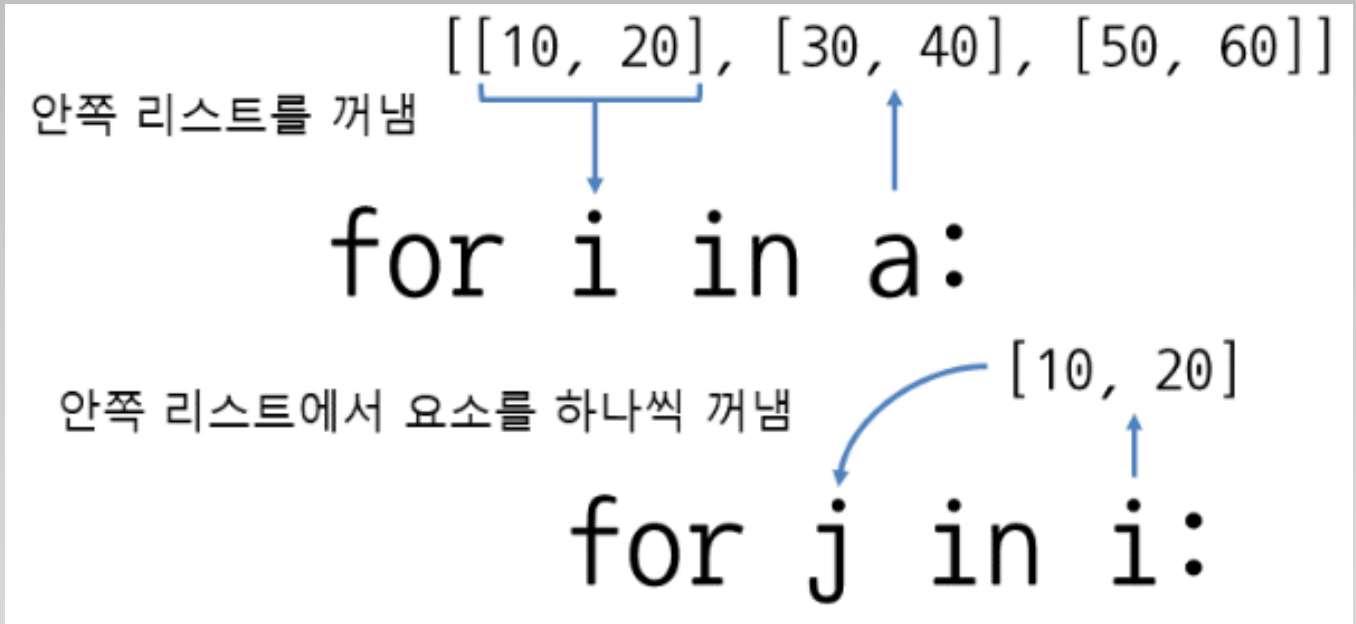
```
a = [[10, 20], [30, 40], [50, 60]]

for i in a:         # a에서 안쪽 리스트를 꺼냄
    for j in i:     # 안쪽 리스트에서 요소를 하나씩 꺼냄
        print(j, end=' ')
    print()
```

실행 결과

```
10 20
30 40
50 60
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

▼ 그림 2차원 리스트에서 for 반복문을 두 번 사용



23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

for와 range 사용하기

- 그럼 이번에는 for range에 세로 크기와 가로 크기를 지정해서 2차원 리스트의 요소를 인덱스로 접근해보자

two_dimensional_list_for_for_range.py

```
a = [[10, 20], [30, 40], [50, 60]]

for i in range(len(a)):            # 세로 크기
    for j in range(len(a[i])):     # 가로 크기
        print(a[i][j], end=' ')
    print()
```

실행 결과

```
10 20
30 40
50 60
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

for와 range 사용하기

- 주의할 점은 len으로 2차원 리스트 a의 크기를 구하면 리스트 안에 들어있는 모든 요소의 개수가 아니라 안쪽 리스트의 개수(세로 크기)가 나온다는 점

```
for i in range(len(a)):            # 세로 크기
    for j in range(len(a[i])):     # 가로 크기
```

- 요소에 접근할 때는 리스트[세로인덱스][가로인덱스] 형식으로 접근함
- 세로 인덱스에 변수 i를, 가로 인덱스에 변수 j를 지정해줌

```
print(a[i][j], end=' ')
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

while 반복문을 한 번 사용하기

two_dimensional_list_while.py

```
a = [[10, 20], [30, 40], [50, 60]]

i = 0
while i < len(a):    # 반복할 때 리스트의 크기 활용(세로 크기)
    x, y = a[i]      # 요소 두 개를 한꺼번에 가져오기
    print(x, y)
    i += 1           # 인덱스를 1 증가시킴
```

실행 결과

```
10 20
30 40
50 60
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

while 반복문을 한 번 사용하기

- while 반복문을 사용할 때도 리스트의 크기를 활용하면 편리함

```
i = 0
while i < len(a):    # 반복할 때 리스트의 크기 활용(세로 크기)
```

- 리스트에 인덱스를 지정하여 값을 꺼내 올 때는 다음과 같이 변수 두 개를 지정해주면 가로 한 줄(안쪽 리스트)에서 요소 두 개를 한꺼번에 가져올 수 있음

```
x, y = a[i]
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

while 반복문을 두 번 사용하기

two_dimensional_list_while_while.py

```
a = [[10, 20], [30, 40], [50, 60]]

i = 0
while i < len(a):           # 세로 크기
    j = 0
    while j < len(a[i]):    # 가로 크기
        print(a[i][j], end=' ')
        j += 1              # 가로 인덱스를 1 증가시킴
    print()
    i += 1                  # 세로 인덱스를 1 증가시킴
```

실행 결과

```
10 20
30 40
50 60
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

while 반복문을 두 번 사용하기

```
i = 0
while i < len(a):           # 세로 크기
    j = 0
    while j < len(a[i]):    # 가로 크기
```

- 요소에 접근할 때는 리스트[세로인덱스][가로인덱스] 형식으로 접근함
- 세로 인덱스에 변수 i를, 가로 인덱스에 변수 j를 지정해줌

```
print(a[i][j], end=' ')
```

23.2 반복문으로 2차원 리스트의 요소를 모두 출력하기

while 반복문을 두 번 사용하기

```
i = 0
while i < len(a):
    j = 0
    while j < len(a[i]):
        print(a[i][j], end=' ')
        j += 1
        i += 1    # 안쪽 while에서 i를 증가시키면 안 됨. 잘못된 방법
    print()
```

23.3 반복문으로 리스트 만들기

while 반복문을 두 번 사용하기

list_create.py

```
a = []    # 빈 리스트 생성

for i in range(10):
    a.append(0)    # append로 요소 추가

print(a)
```

실행 결과

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- for 반복문으로 10번 반복하면서 append로 요소를 추가하면 1차원 리스트를 만들 수 있음

23.3 반복문으로 리스트 만들기

for 반복문으로 2차원 리스트 만들기

two_dimensional_list_create.py

```
a = []    # 빈 리스트 생성

for i in range(3):
    line = []              # 안쪽 리스트로 사용할 빈 리스트 생성
    for j in range(2):
        line.append(0)     # 안쪽 리스트에 0 추가
    a.append(line)         # 전체 리스트에 안쪽 리스트를 추가

print(a)
```

실행 결과

```
[[0, 0], [0, 0], [0, 0]]
```

23.3 반복문으로 리스트 만들기

for 반복문으로 2차원 리스트 만들기

- 먼저 세로 크기만큼 반복하면서 안쪽 리스트로 사용할 빈 리스트 line을 만듬

```
for i in range(3):
    line = []               # 안쪽 리스트로 사용할 빈 리스트 생성
```

```
    for j in range(2):
        line.append(0)      # 안쪽 리스트에 0 추가
    a.append(line)          # 전체 리스트에 안쪽 리스트를 추가
```

- append에 리스트를 넣으면 리스트 안에 리스트가 들어가는 특성을 이용함

23.3 반복문으로 리스트 만들기

리스트 표현식으로 2차원 리스트 만들기

```
>>> a = [[0 for j in range(2)] for i in range(3)]
>>> a
[[0, 0], [0, 0], [0, 0]]
```

- 만약 for 반복문을 한 번만 사용하고 싶다면 다음과 같이 식 부분에서 리스트 자체를 곱해주면 됨

```
>>> a = [[0] * 2 for i in range(3)]
>>> a
[[0, 0], [0, 0], [0, 0]]
```

23.3 반복문으로 리스트 만들기

톱니형 리스트 만들기

- 가로 크기를 알고 있다고 가정하고, 리스트를 만들어보자

jagged_list_create.py

```
a = [3, 1, 3, 2, 5]    # 가로 크기를 저장한 리스트
b = []    # 빈 리스트 생성

for i in a:      # 가로 크기를 저장한 리스트로 반복
    line = []    # 안쪽 리스트로 사용할 빈 리스트 생성
    for j in range(i):    # 리스트 a에 저장된 가로 크기만큼 반복
        line.append(0)
    b.append(line)        # 리스트 b에 안쪽 리스트를 추가

print(b)
```

실행 결과

```
[[0, 0, 0], [0], [0, 0, 0], [0, 0], [0, 0, 0, 0, 0]]
```

23.3 반복문으로 리스트 만들기

톱니형 리스트 만들기

- 사실 이것도 그냥 리스트 표현식을 활용하면 간단하게 만들 수 있음

```
>>> a = [[0] * i for i in [3, 1, 3, 2, 5]]
>>> a
[[0, 0, 0], [0], [0, 0, 0], [0, 0], [0, 0, 0, 0, 0]]
```

23.4 2차원 리스트의 할당과 복사 알아보기

2차원 리스트의 할당과 복사 알아보기

- 2차원 리스트를 만든 뒤 다른 변수에 할당하고, 요소를 변경해보면 두 리스트에 모두 반영됨

```
>>> a = [[10, 20], [30, 40]]
>>> b = a
>>> b[0][0] = 500
>>> a
[[500, 20], [30, 40]]
>>> b
[[500, 20], [30, 40]]
```

- 리스트 a를 copy 메서드로 b에 복사한 뒤 b의 요소를 변경해보면 리스트 a와 b에 모두 반영됨

```
>>> a = [[10, 20], [30, 40]]
>>> b = a.copy()
>>> b[0][0] = 500
>>> a
[[500, 20], [30, 40]]
>>> b
[[500, 20], [30, 40]]
```

23.4 2차원 리스트의 할당과 복사 알아보기

2차원 리스트의 할당과 복사 알아보기

- 2차원 이상의 다차원 리스트는 리스트를 완전히 복사하려면 copy 메서드 대신 copy 모듈의 deepcopy 함수를 사용해야 함

```
>>> a = [[10, 20], [30, 40]]
>>> import copy              # copy 모듈을 가져옴
>>> b = copy.deepcopy(a)     # copy.deepcopy 함수를 사용하여 깊은 복사
>>> b[0][0] = 500
>>> a
[[10, 20], [30, 40]]
>>> b
[[500, 20], [30, 40]]
```