교수의 파이썬

01_4 다차원 리스트의 참조

창원대학교 정보통신공학과 교수 박동규

널널한 교수의 파이썬

01_4 다차원 리스트의 참조

창원대학교 정보통신공학과 교수 박동규

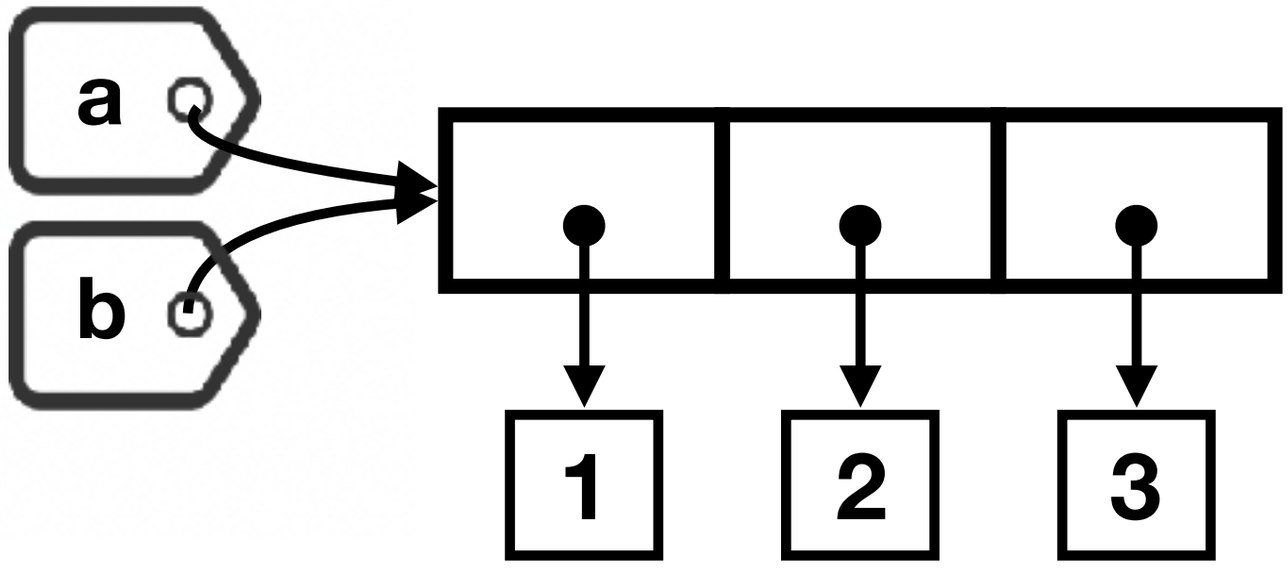
널널한 교수의 고급 파이썬

01_4 다차원 리스트의 참조

창원대학교 정보통신공학과 교수 박동규

파이썬 리스트

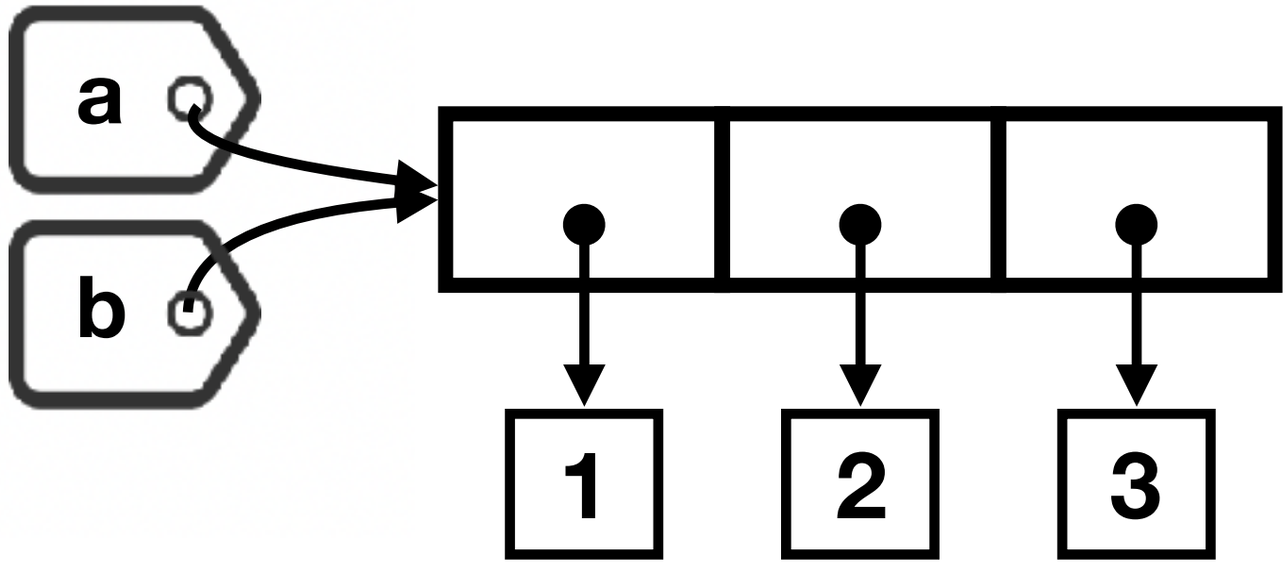
```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



파이썬 리스트

- 리스트는 변경가능(mutable) 자료형이다

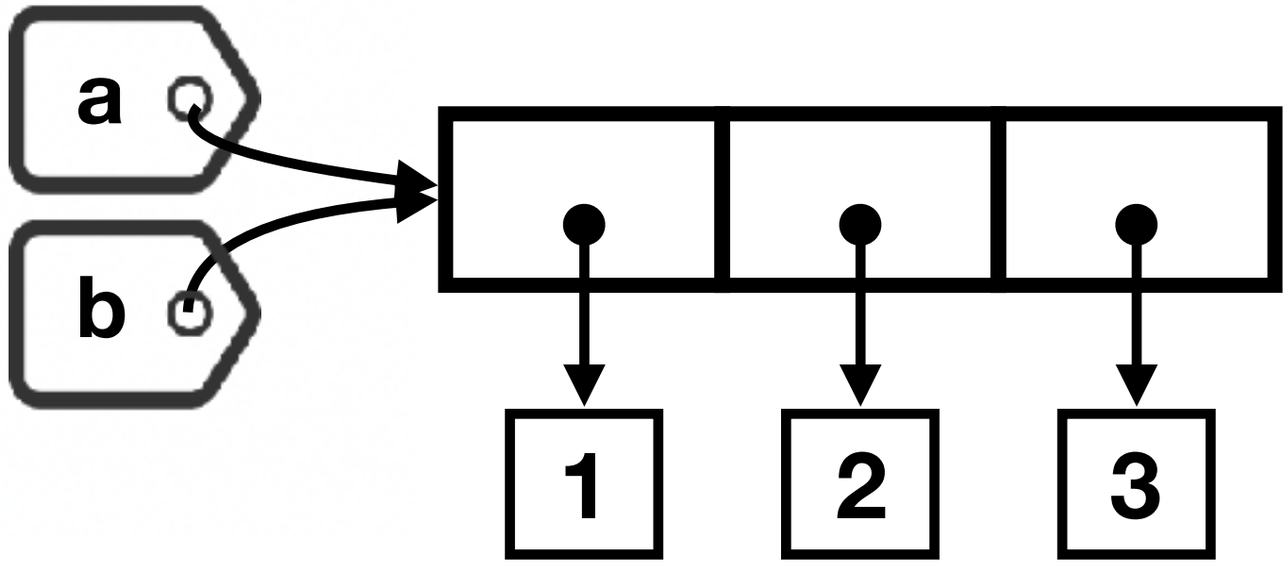
```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



파이썬 리스트

- 리스트는 변경가능(mutable) 자료형이다
- 리스트의 요소는 객체에 대한 참조값이다

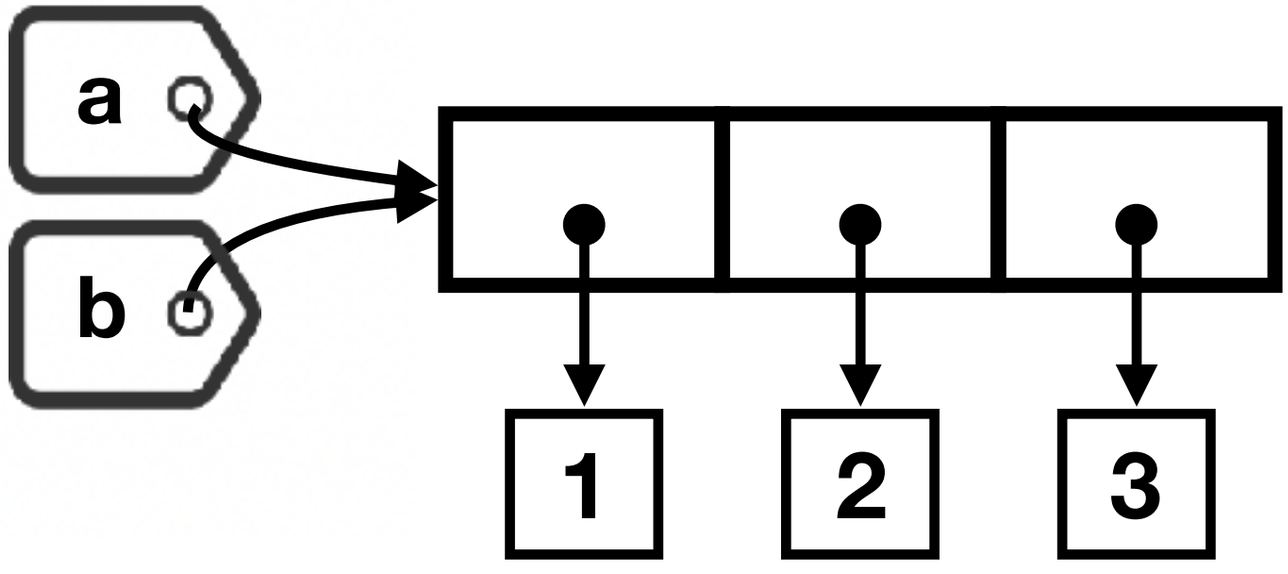
```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



파이썬 리스트

- 리스트는 변경가능(mutable) 자료형이다
- 리스트의 요소는 객체에 대한 참조값이다

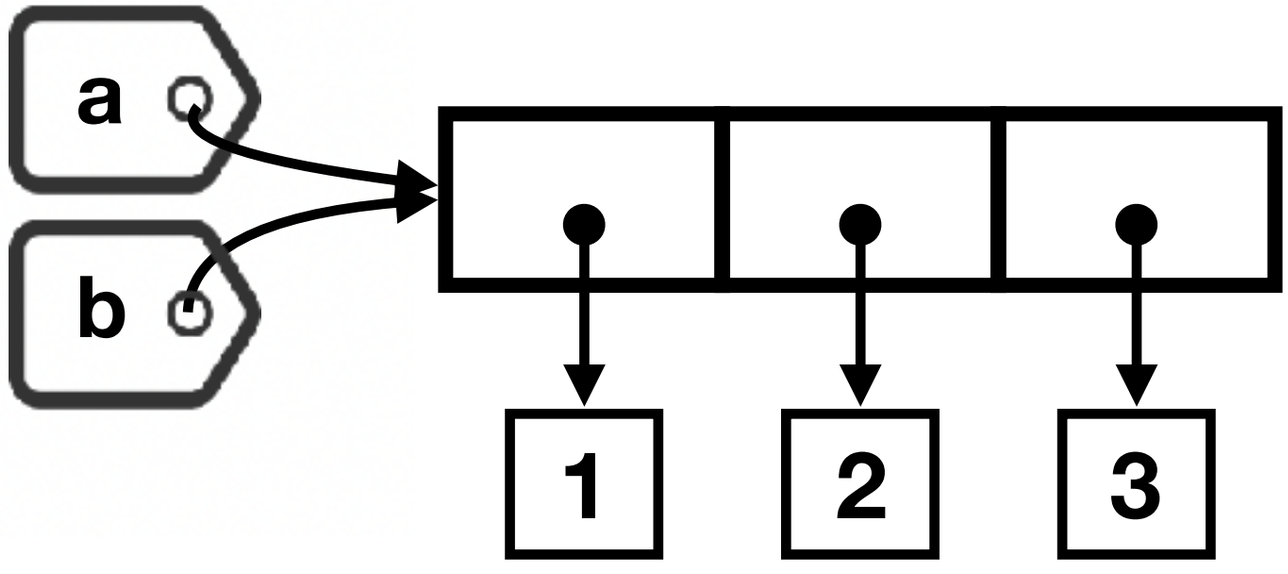
```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



파이썬 리스트

- 리스트는 변경가능(mutable) 자료형이다
- 리스트의 요소는 객체에 대한 참조값이다

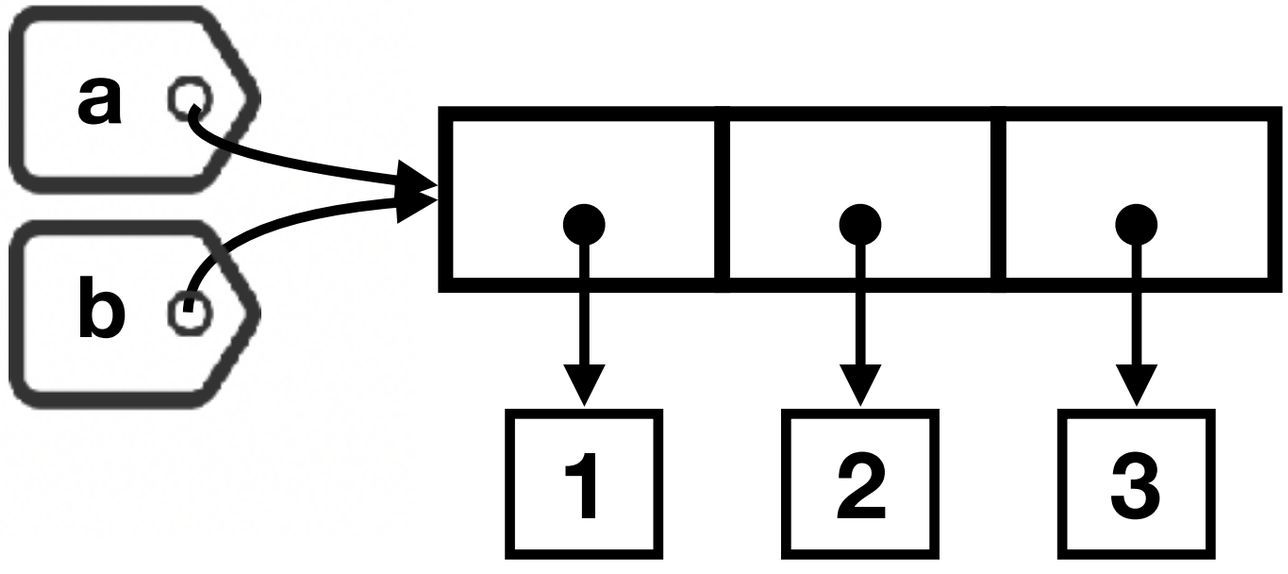
```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



파이썬 리스트

- 리스트는 변경가능(mutable) 자료형이다
- 리스트의 요소는 객체에 대한 참조값이다

```
>>> a = [1, 2, 3]
>>> b = a
>>> print(b)
[1, 2, 3]
>>> id(a)
4512940232
>>> id(b)
4512940232
```



다차원 리스트를 만들어 보자

1 : board = [[0] * cols] * rows

2 : board = [[0] * cols for _ in range(rows)]

두 가지 방법을 비교해보자!!

어떤 차이점이 있을까?

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

어느쪽이 더 빠를까?

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols] * rows
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.000068

```python
import time, sys

rows, cols = 300, 3000
start = time. time()
board = [[0] * cols for _ in range(rows)]
end = time. time()
print("수행시간 : {:10.6f}".format(end - start))
```

수행시간 : 0.007204

1번이 더 빠른 이유는

1 : board = [[0] * cols] * rows

2 : board = [[0] * cols for _ in range(rows)]

rows, cols = 3, 3인 경우에 대해 살펴보자

어떤 차이점?

```
board = [[0] * 3] * 3
```

어떤 차이점?

```
board = [[0] * 3] * 3
```

```
[0] * 3
```

어떤 차이점?

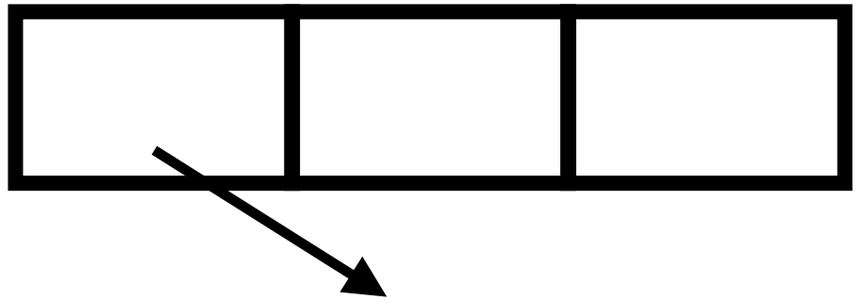
board = [[0] * 3] * 3

[0] * 3

어떤 차이점?

board = [[0] * 3] * 3

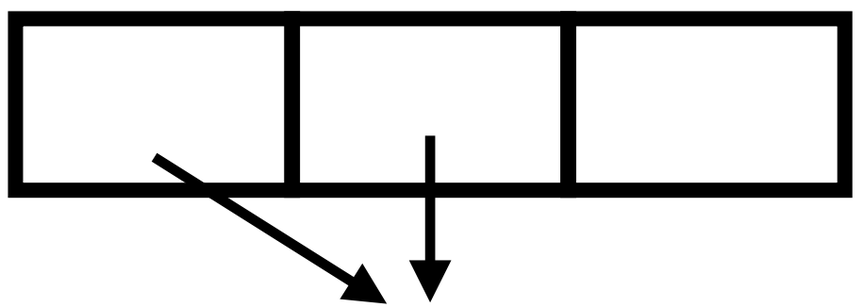
[0] * 3



어떤 차이점?

board = [[0] * 3] * 3

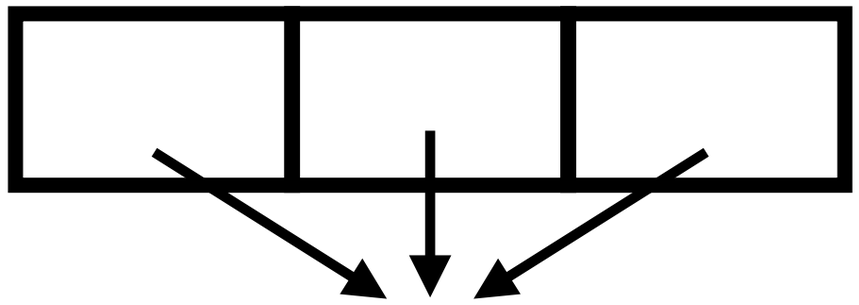
[0] * 3



어떤 차이점?

board = [[0] * 3] * 3

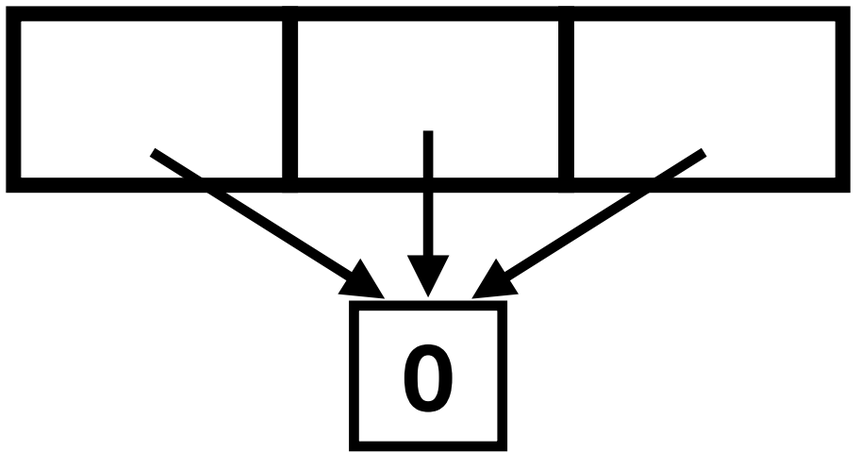
[0] * 3



어떤 차이점?

board = [[0] * 3] * 3

[0] * 3

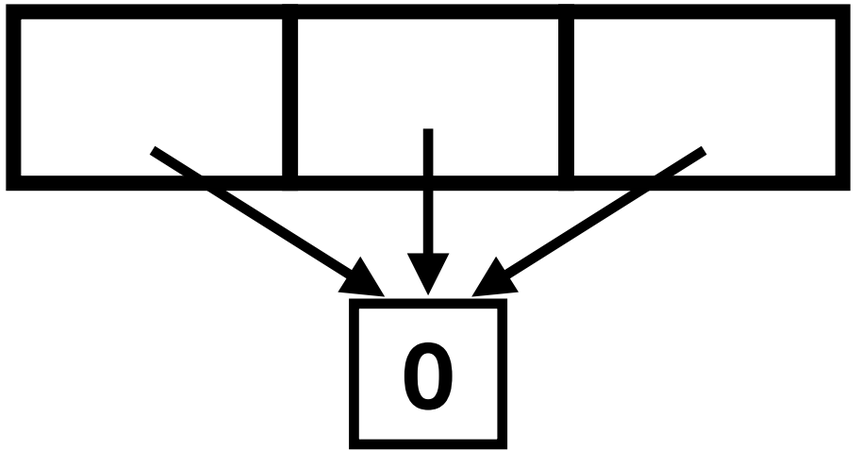


어떤 차이점?

board = [[0] * 3] * 3

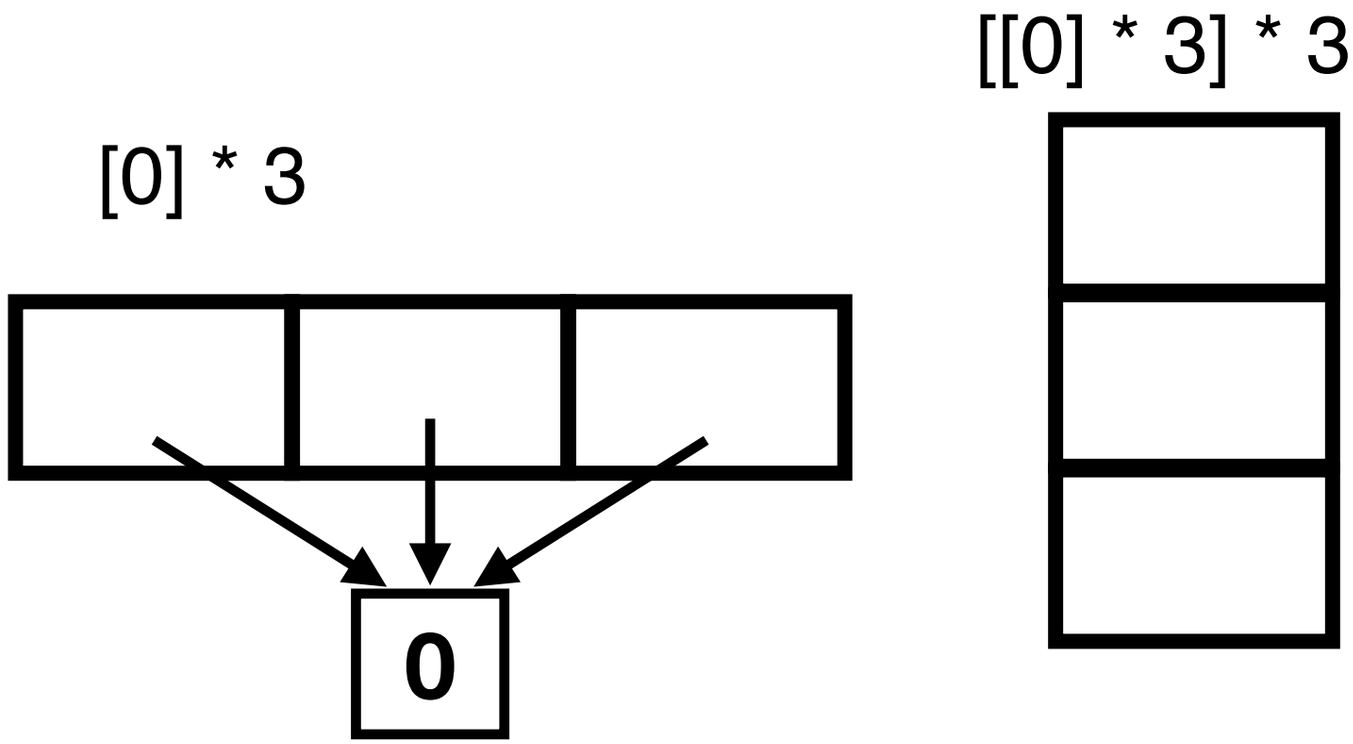
[[0] * 3] * 3

[0] * 3



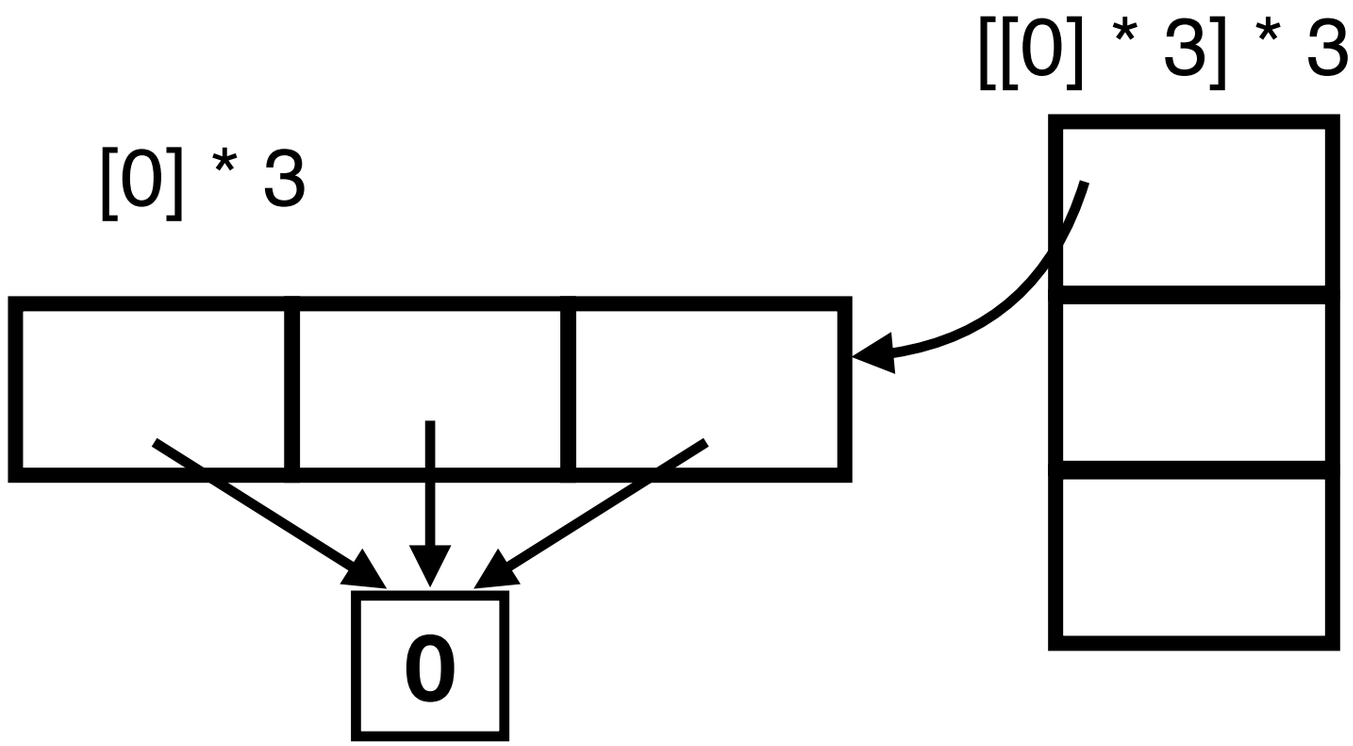
어떤 차이점?

board = [[0] * 3] * 3



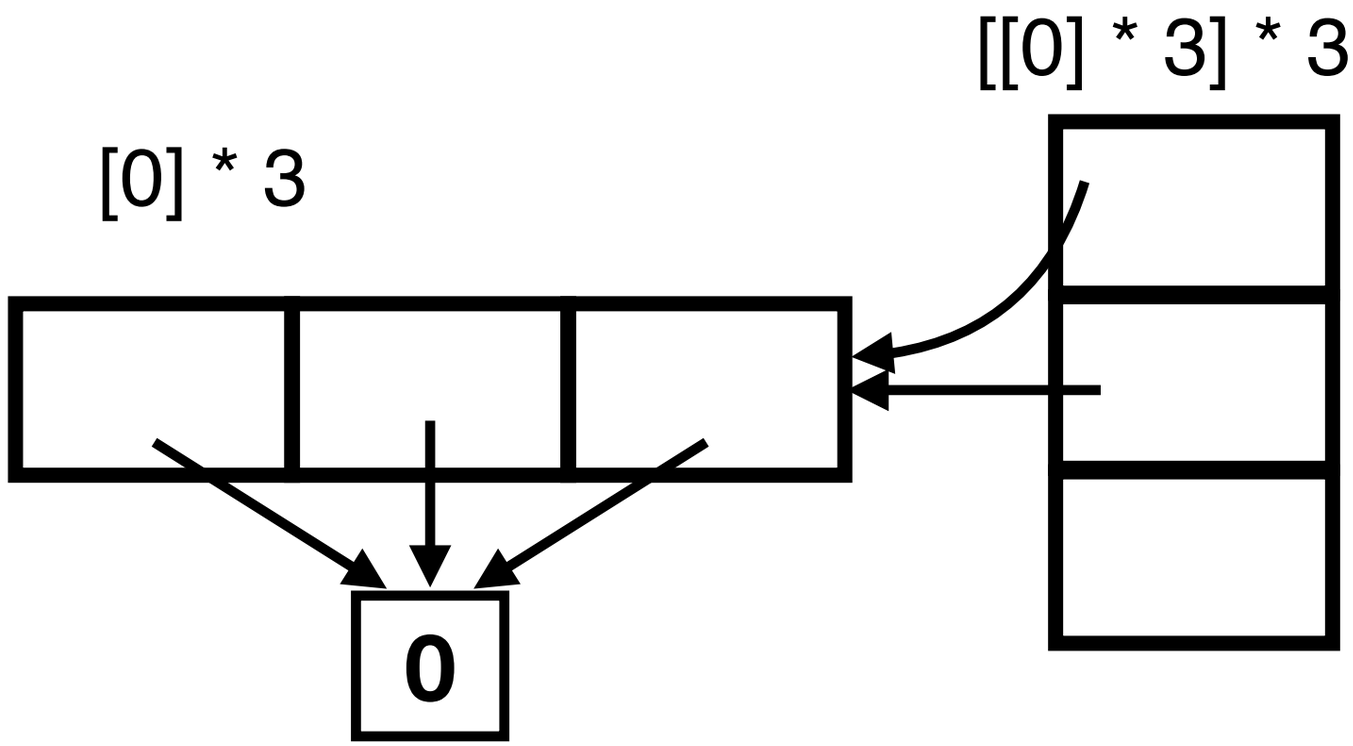
어떤 차이점?

board = [[0] * 3] * 3



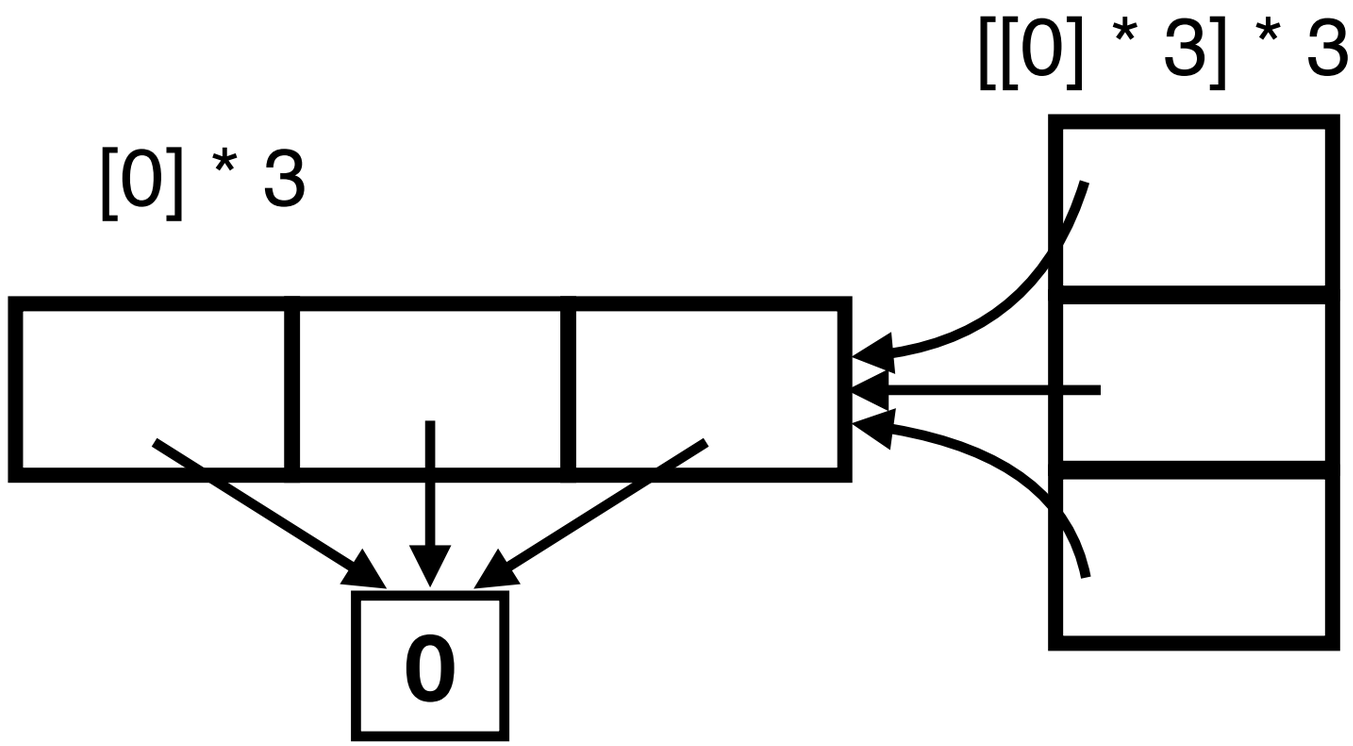
어떤 차이점?

board = [[0] * 3] * 3



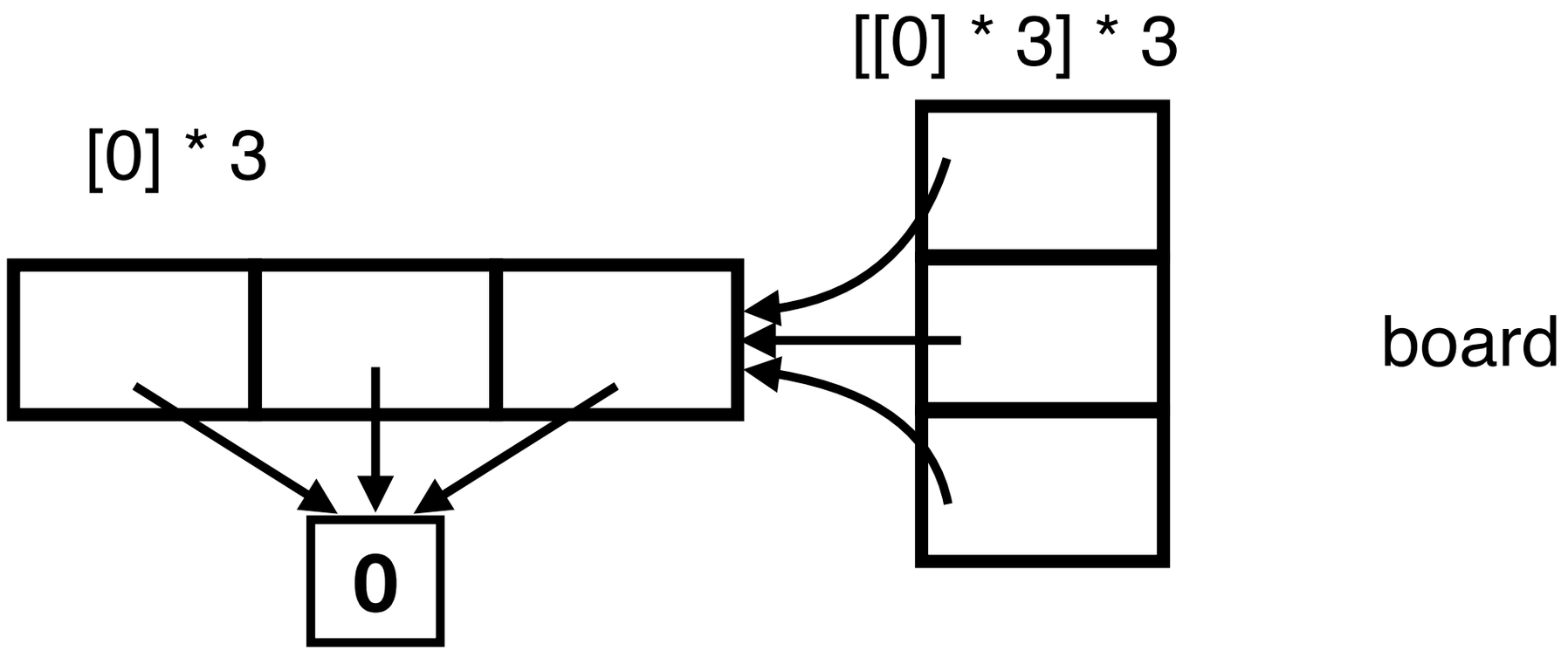
어떤 차이점?

board = [[0] * 3] * 3



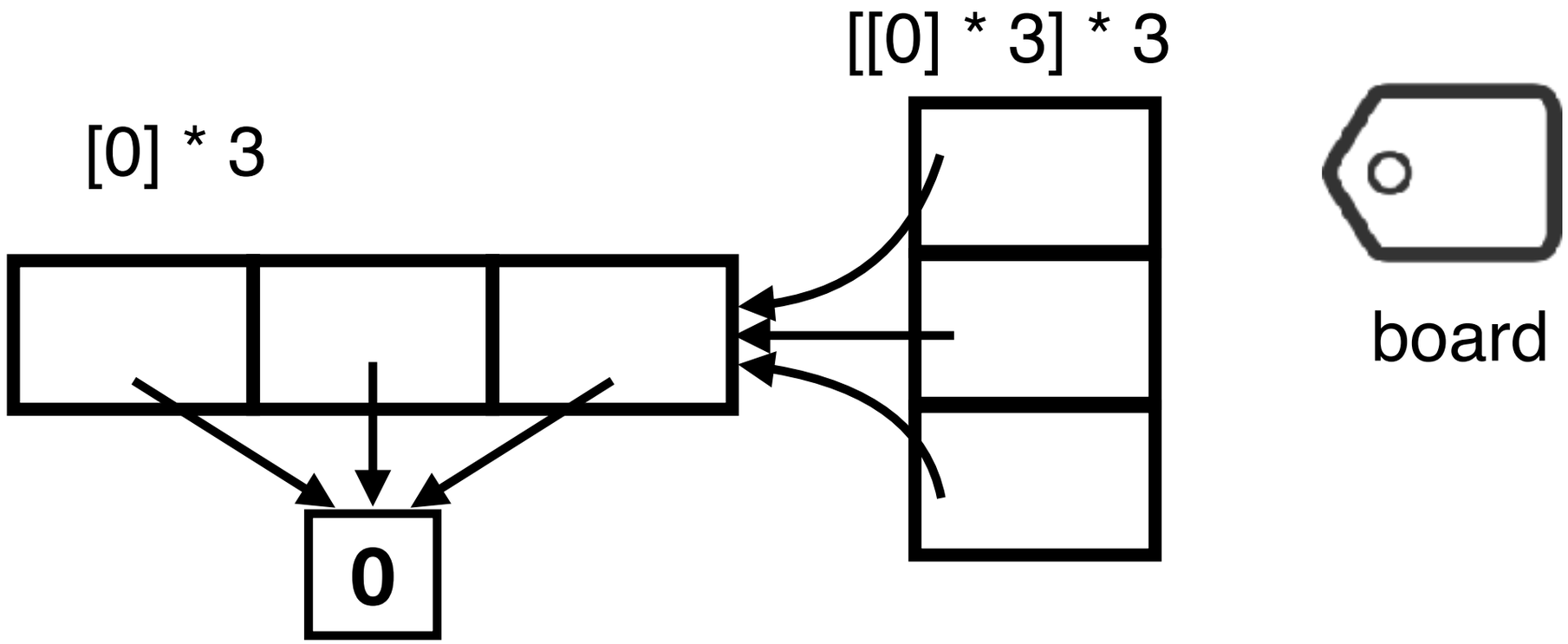
어떤 차이점?

board = [[0] * 3] * 3



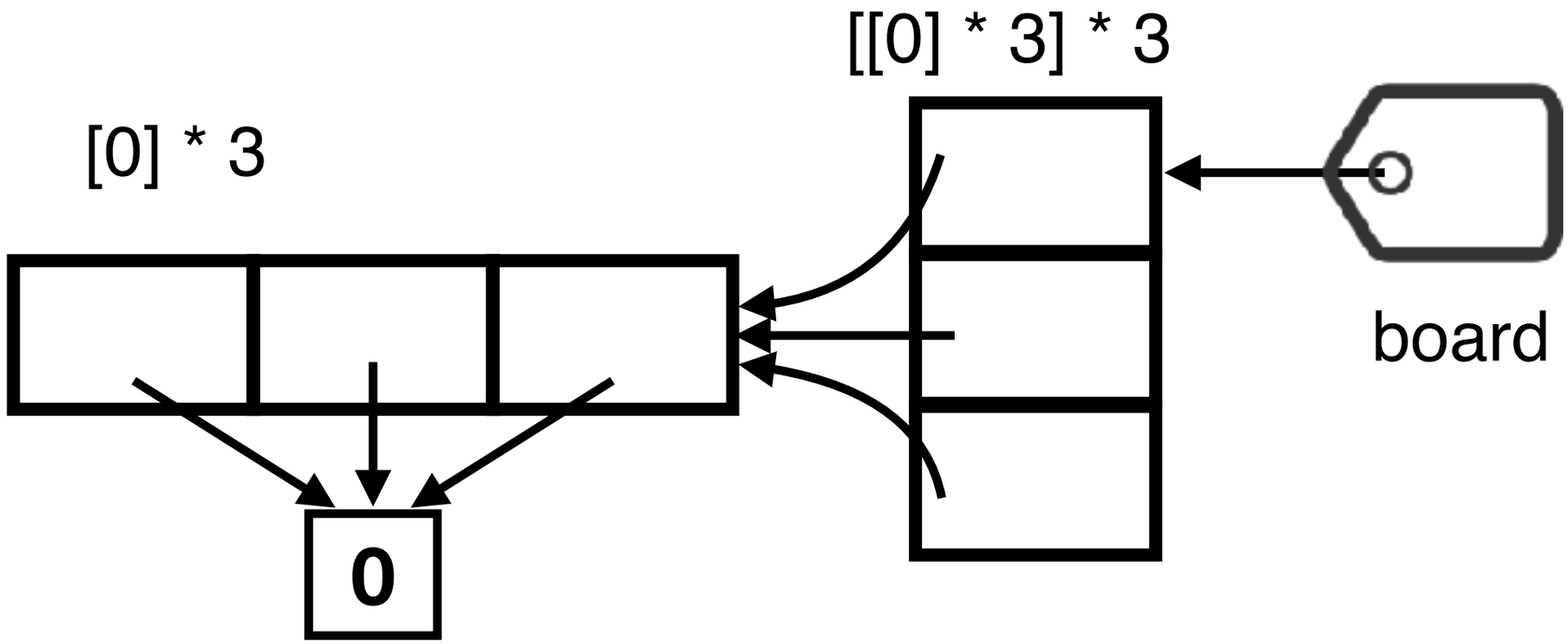
어떤 차이점?

board = [[0] * 3] * 3



어떤 차이점?

board = [[0] * 3] * 3



어떤 차이점?

```
board = [[0] * 3 for _ in range(3)]
```

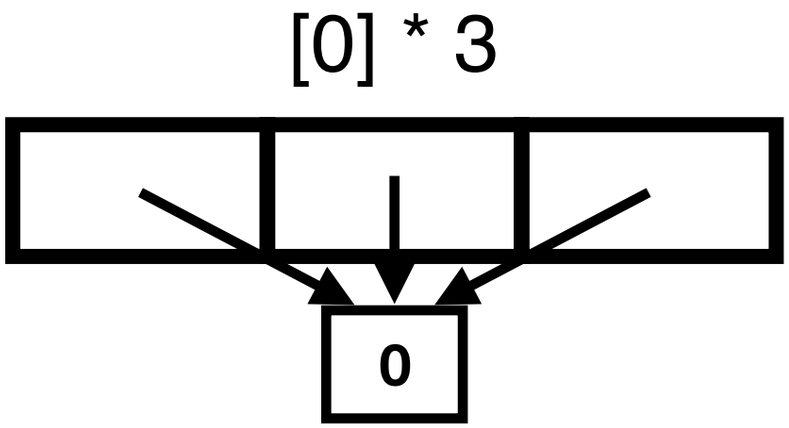
어떤 차이점?

```
board = [[0] * 3 for _ in range(3)]
```

```
[0] * 3
```

어떤 차이점?

board = [[0] * 3 for _ in range(3)]

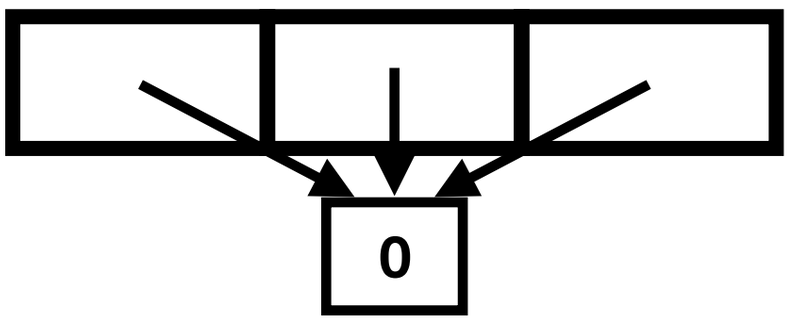


어떤 차이점?

board = [[0] * 3 for _ in range(3)]

[.. for _ in range(3)]

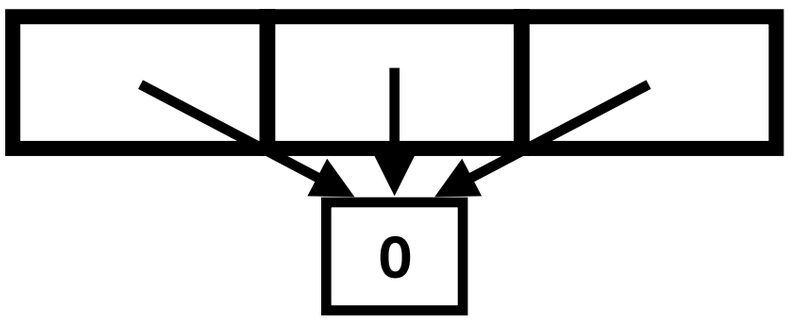
[0] * 3



어떤 차이점?

board = [[0] * 3 for _ in range(3)]

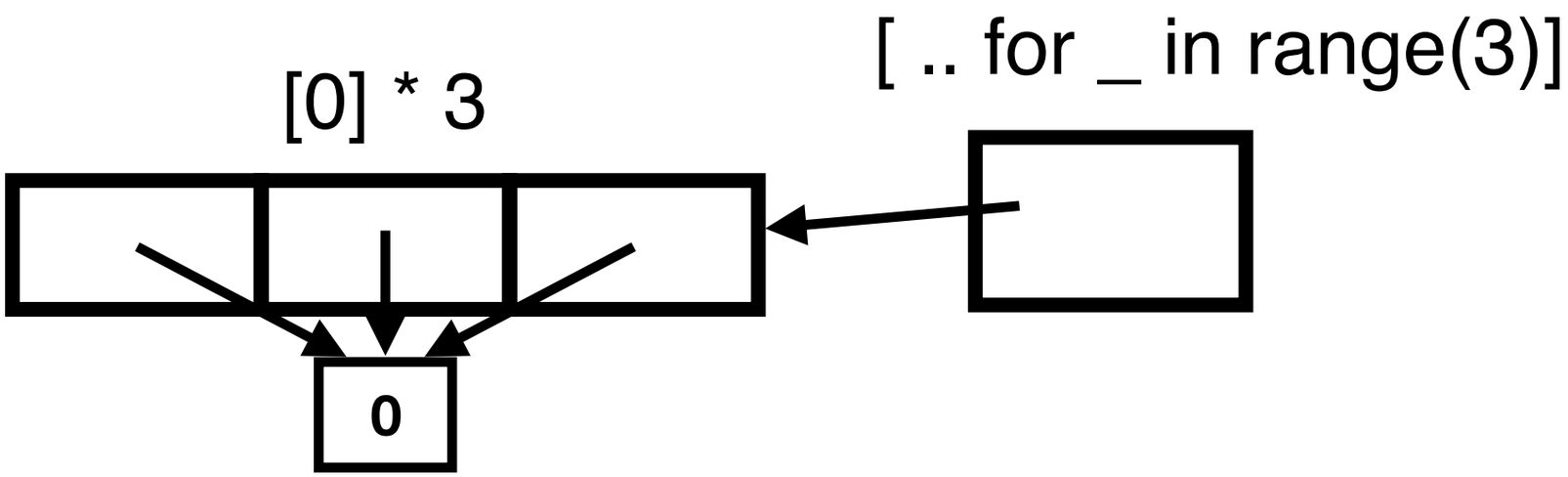
[0] * 3



[.. for _ in range(3)]

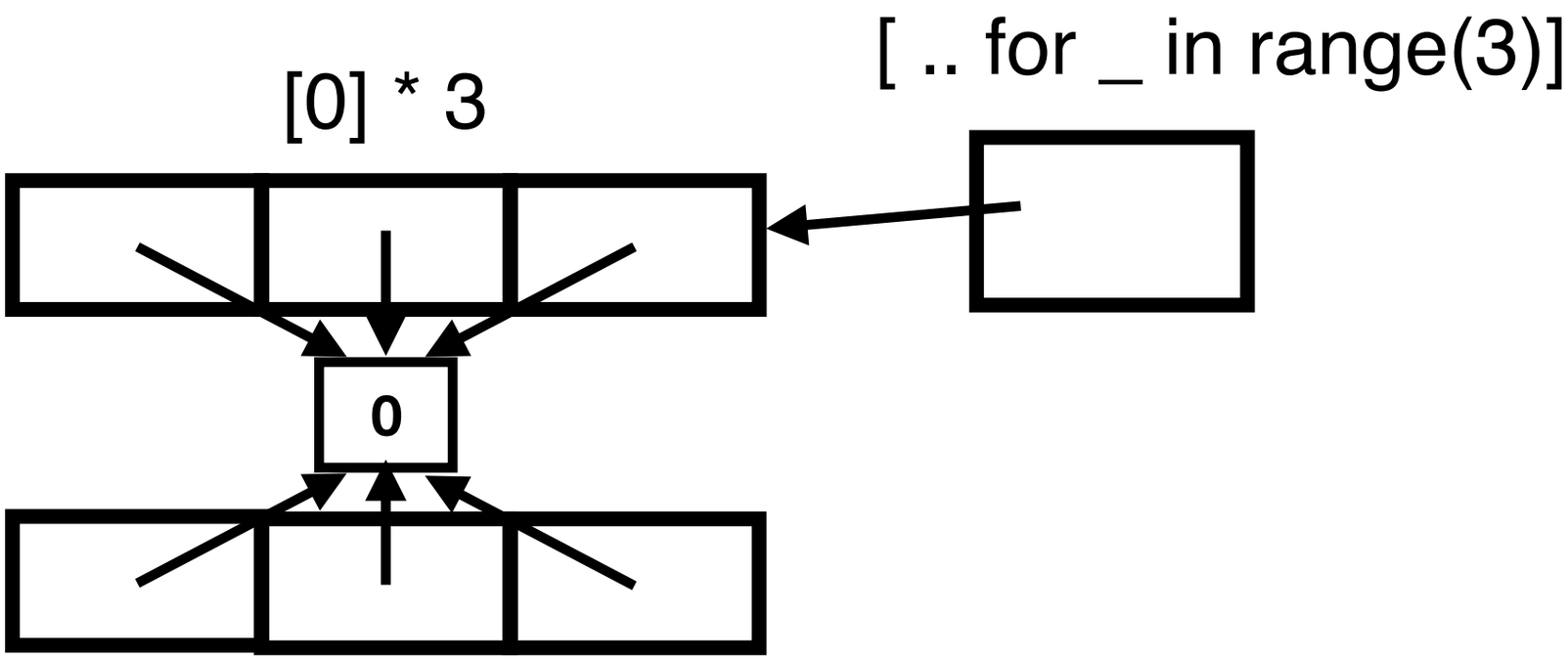
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



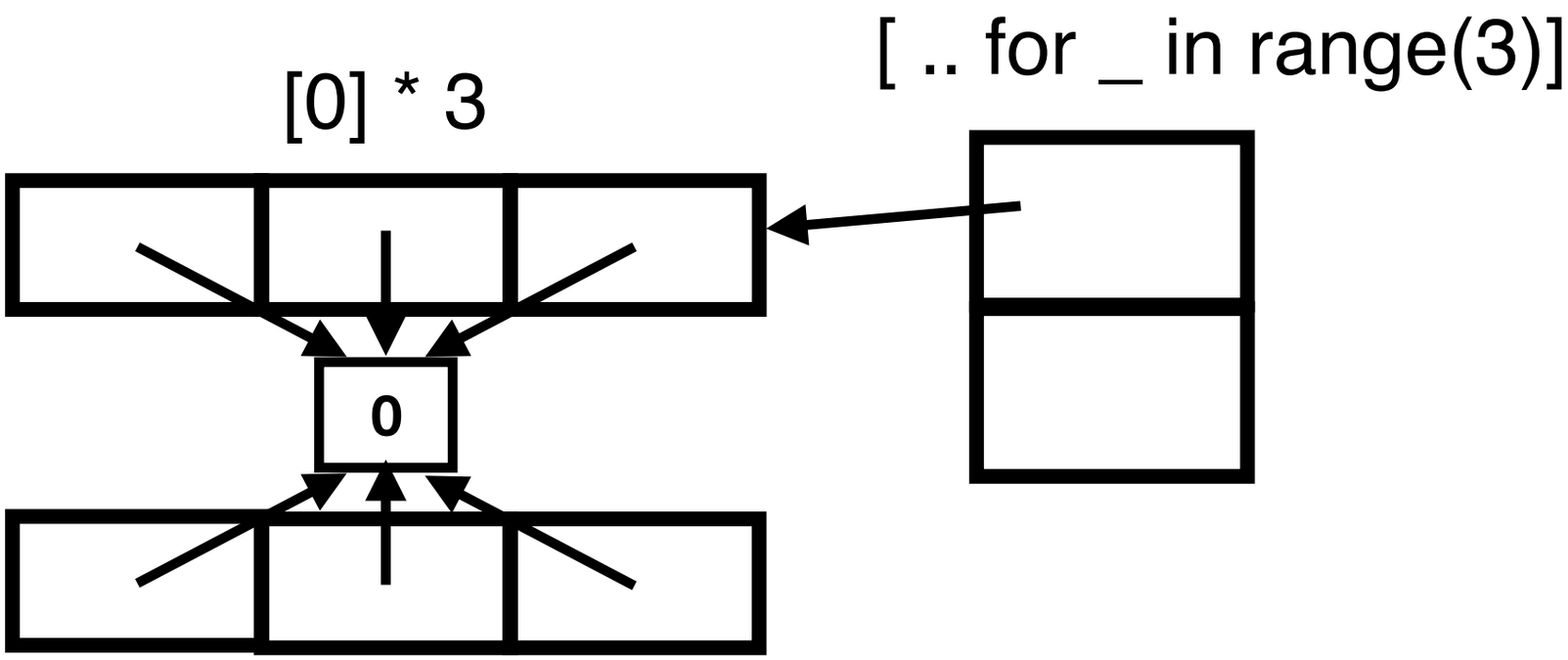
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



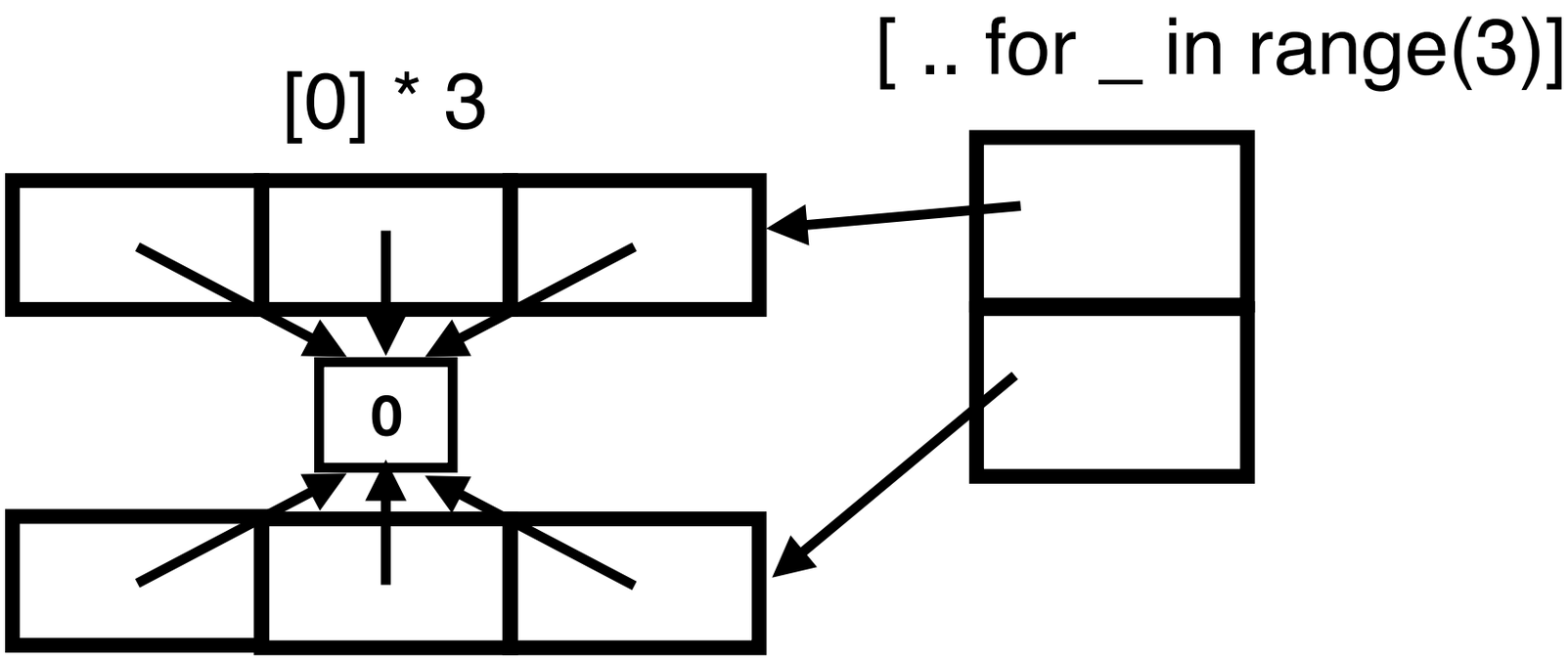
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



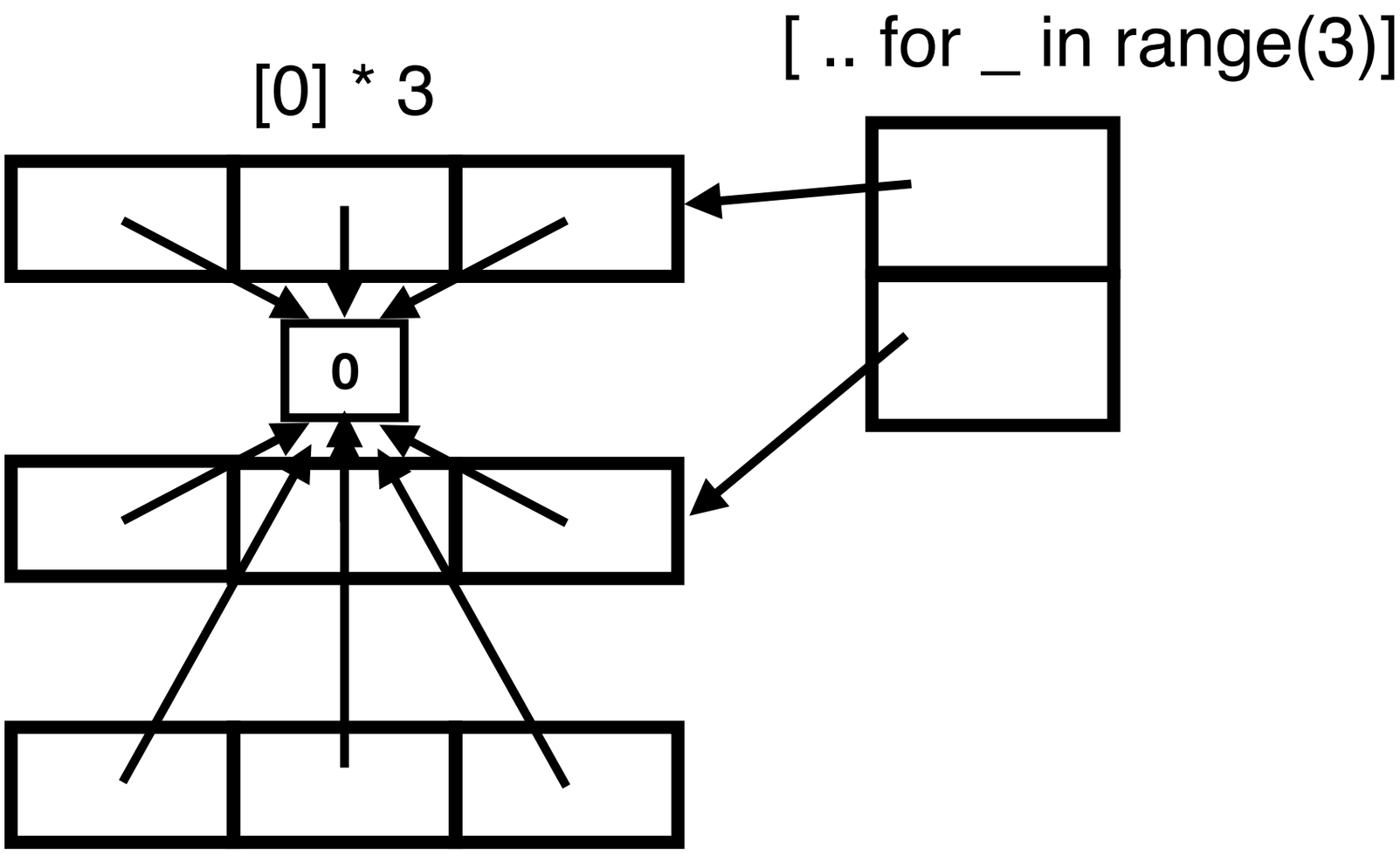
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



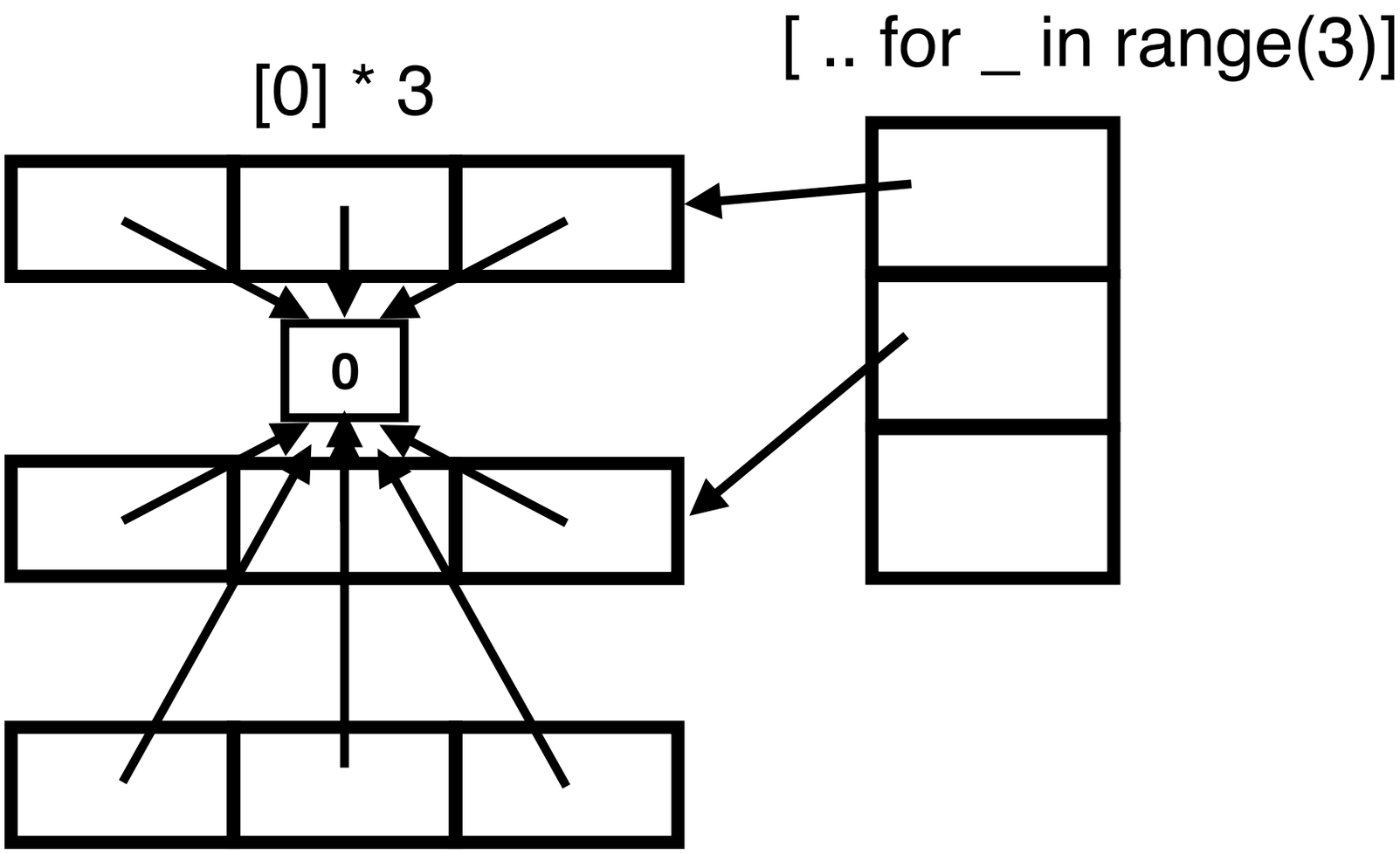
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



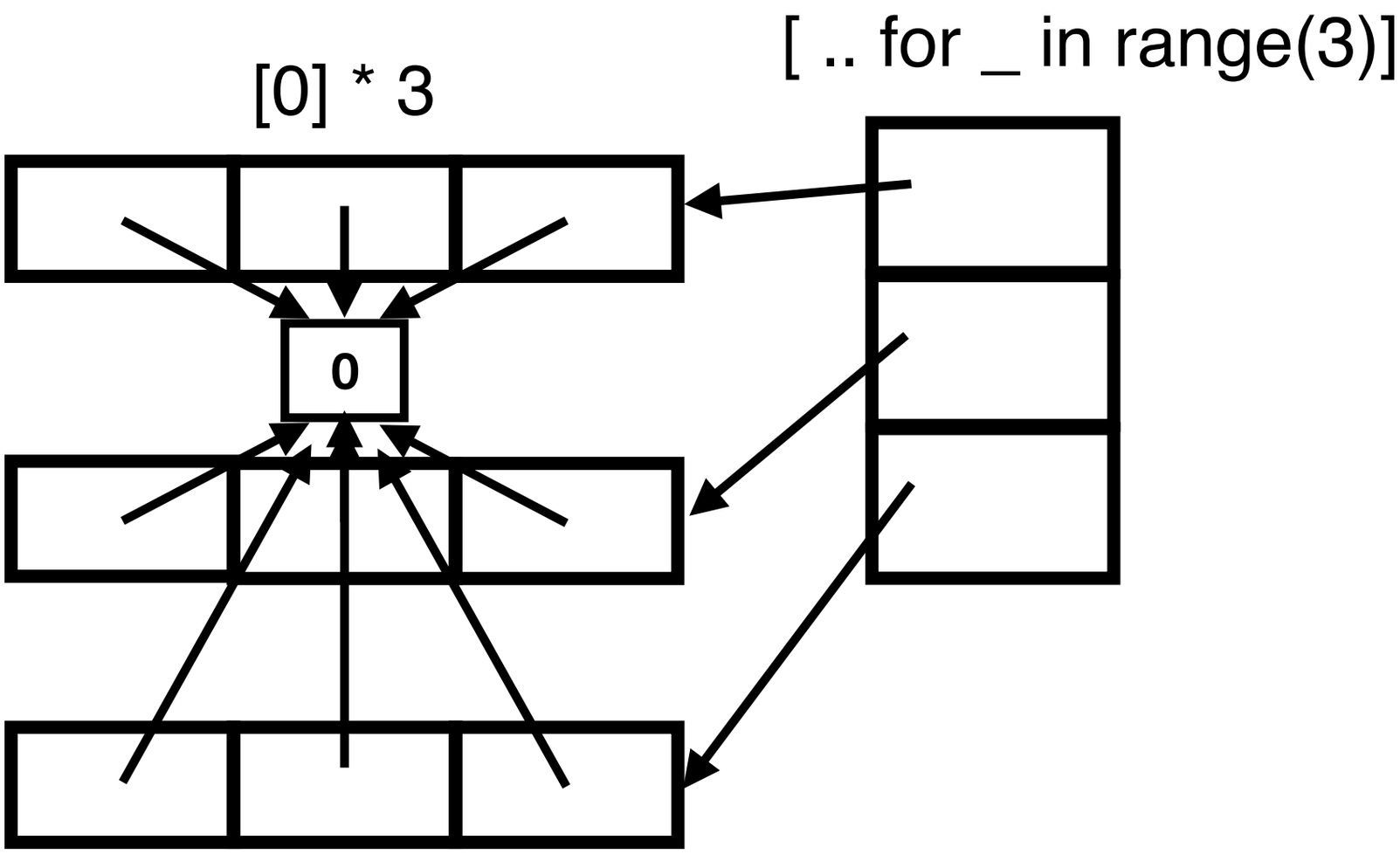
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



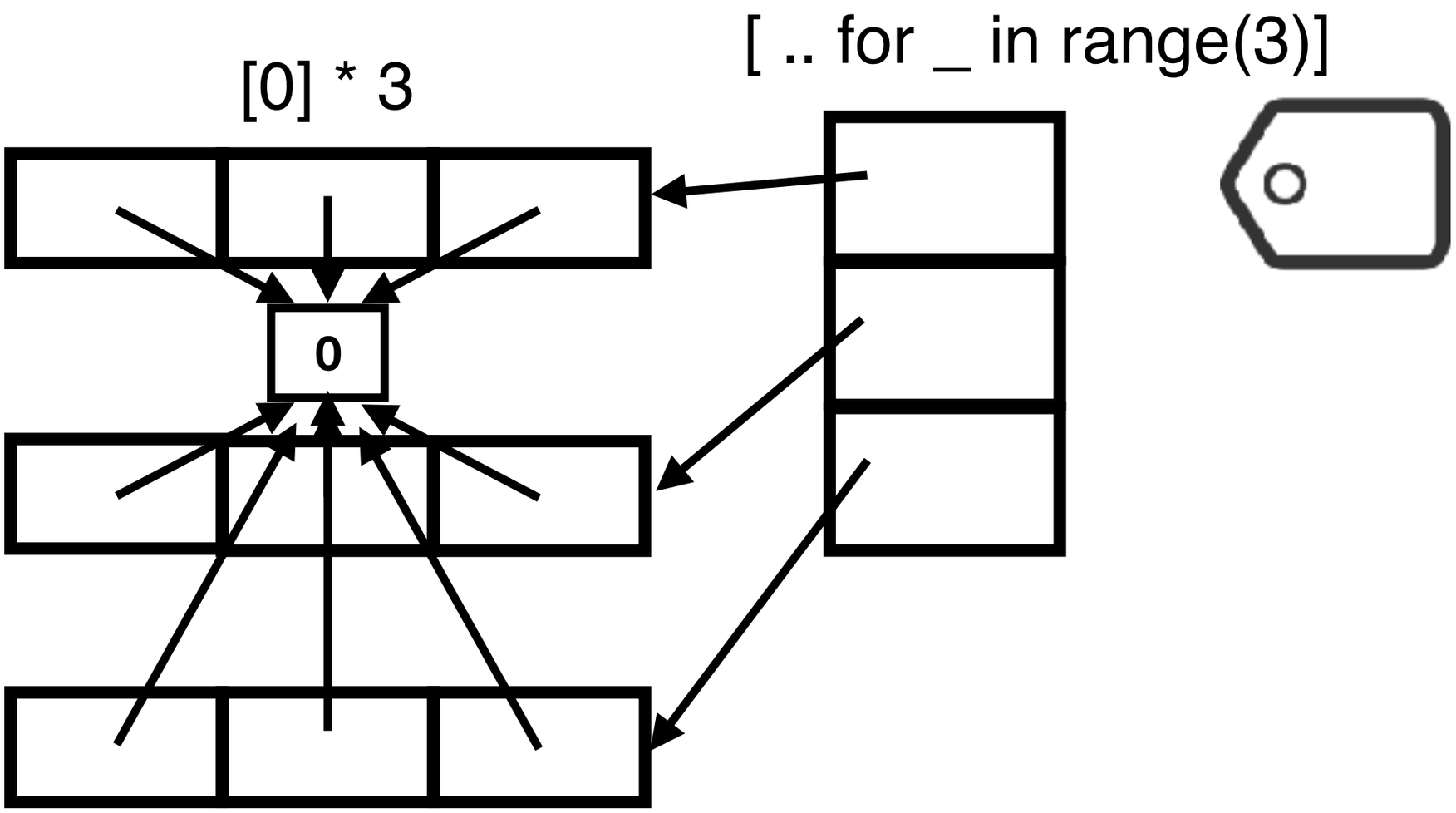
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



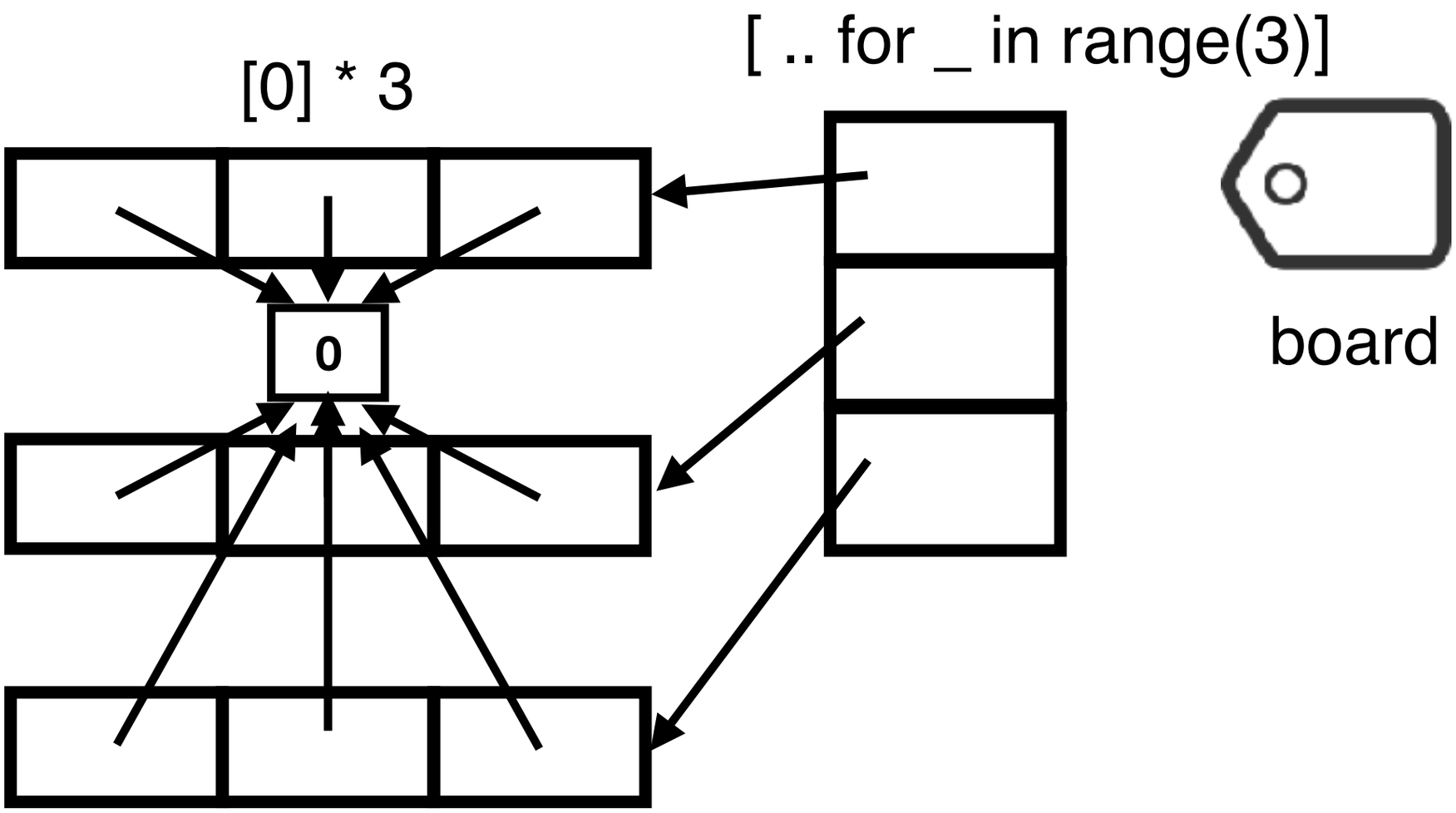
어떤 차이점?

board = [[0] * 3 for _ in range(3)]



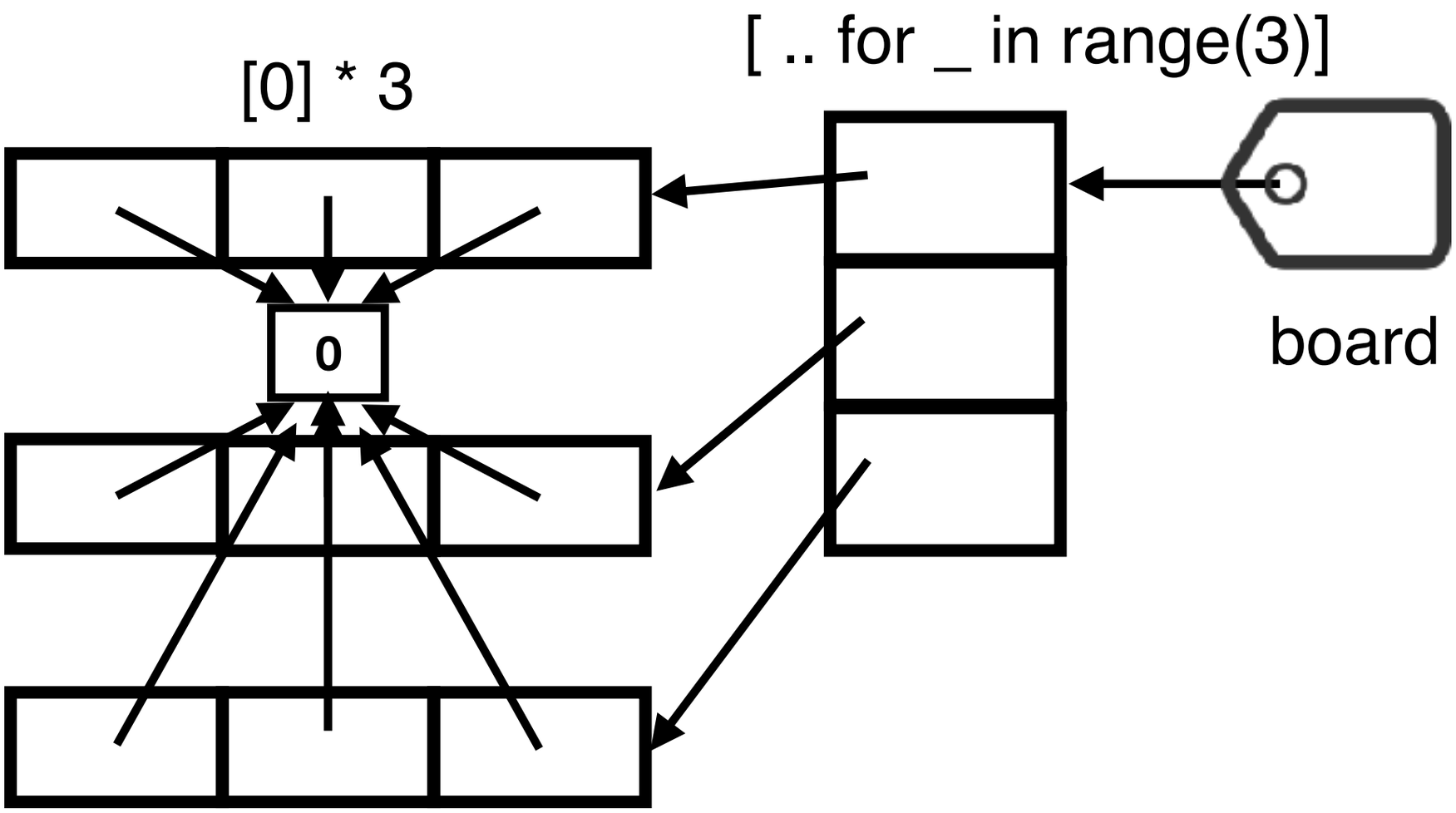
어떤 차이점?

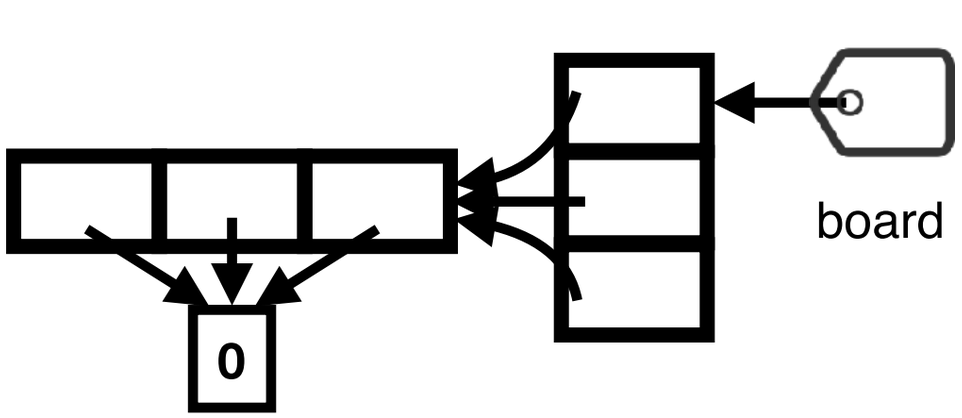
board = [[0] * 3 for _ in range(3)]

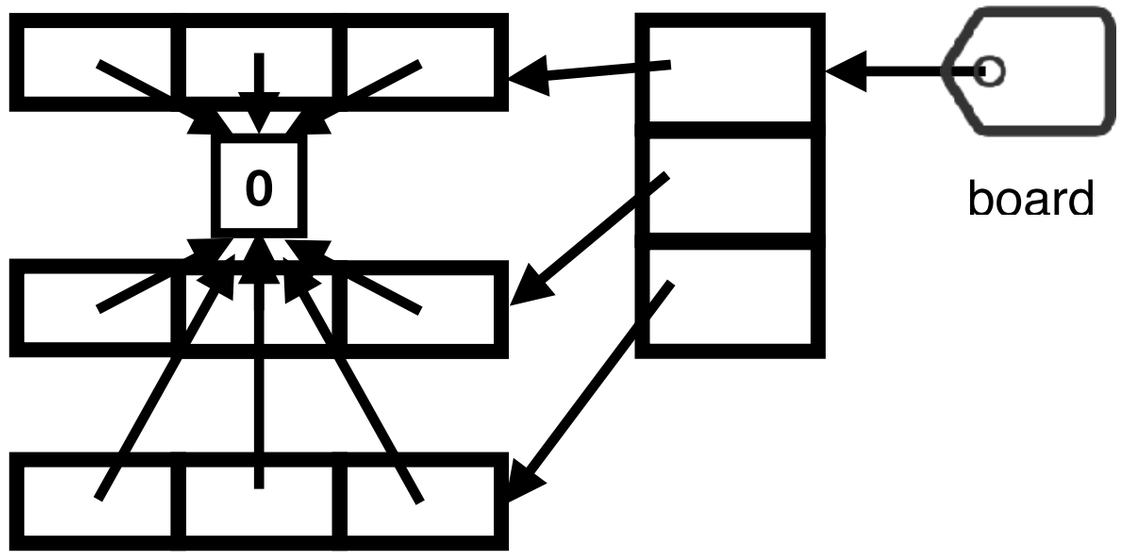


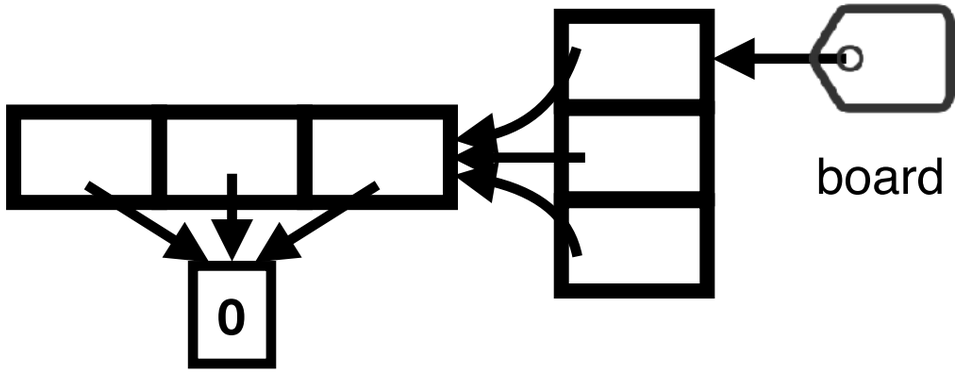
어떤 차이점?

board = [[0] * 3 for _ in range(3)]

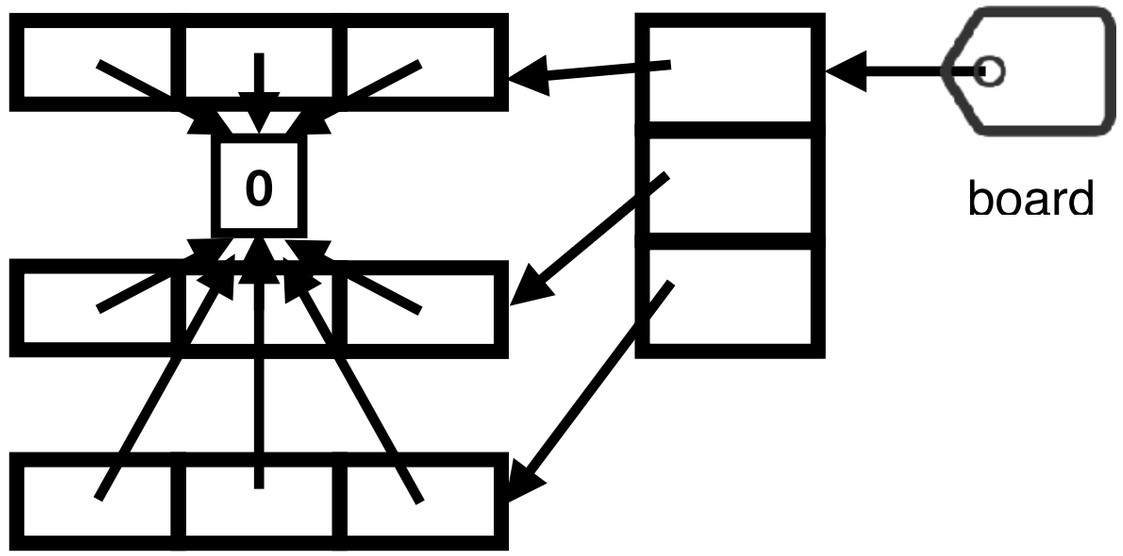


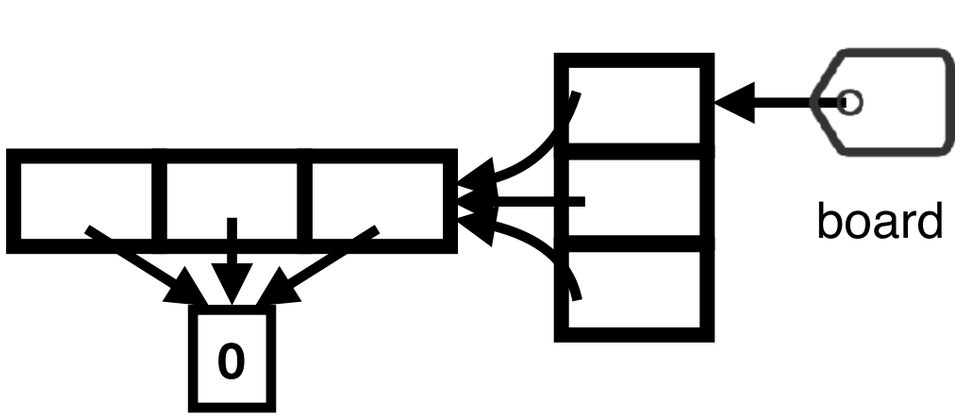

board
0


board
0

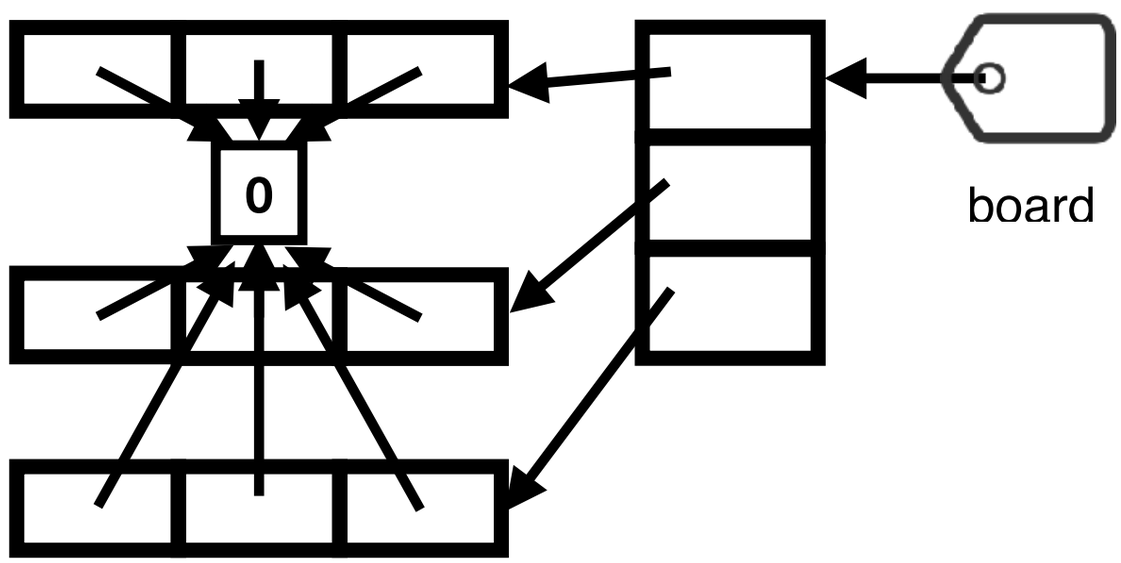


board = [[0] * cols] * rows

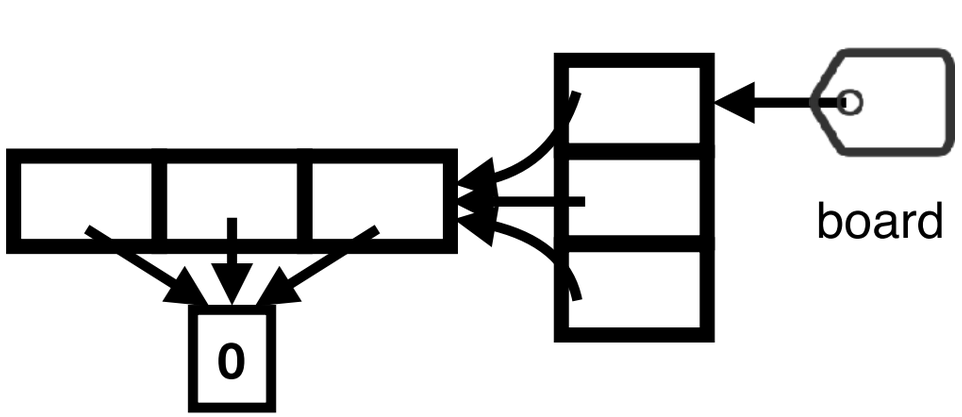




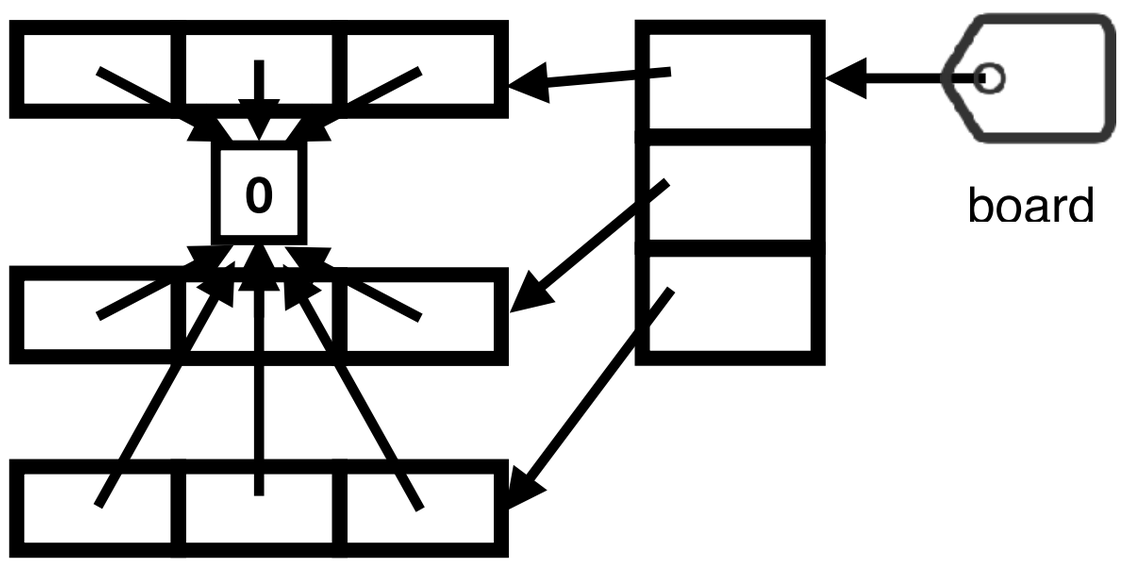
board = [[0] * cols] * rows



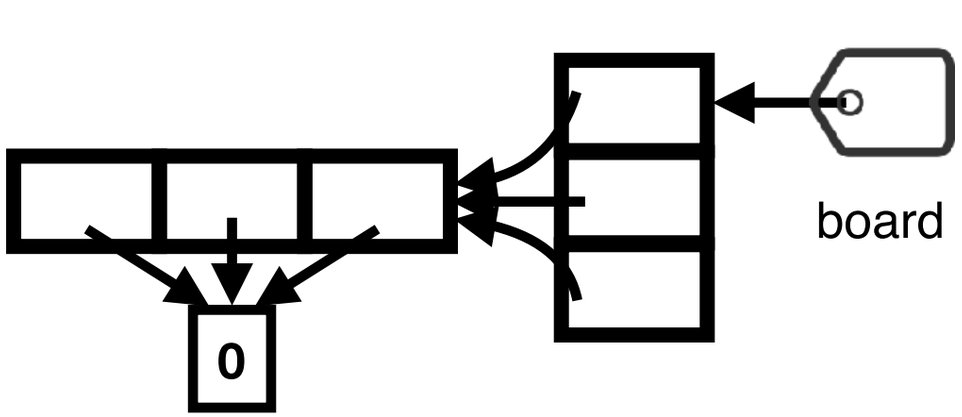
board = [[0] * cols for _ in range(rows)]



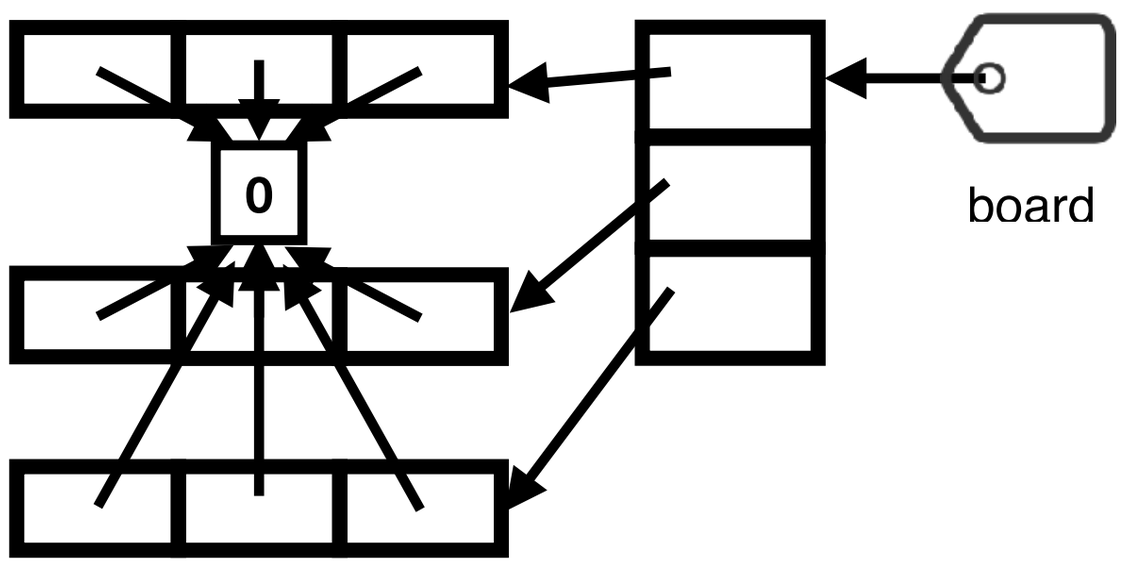
board = [[0] * cols] * rows



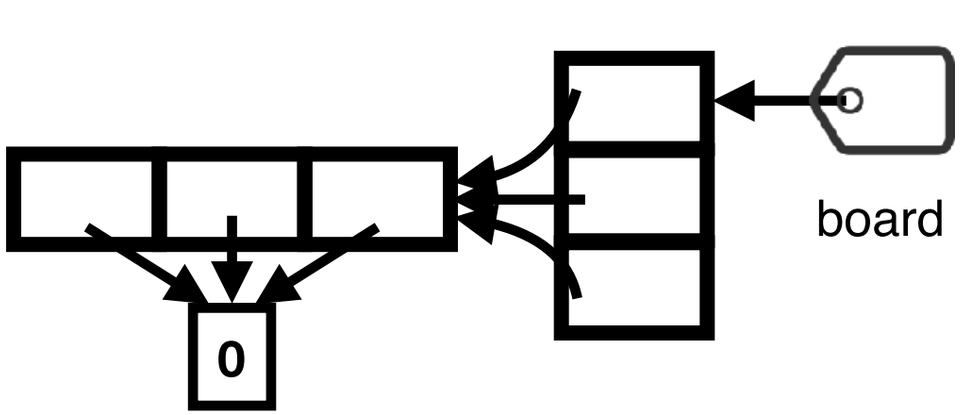
board = [[0] * cols for _ in range(rows)]



board = [[0] * cols] * rows

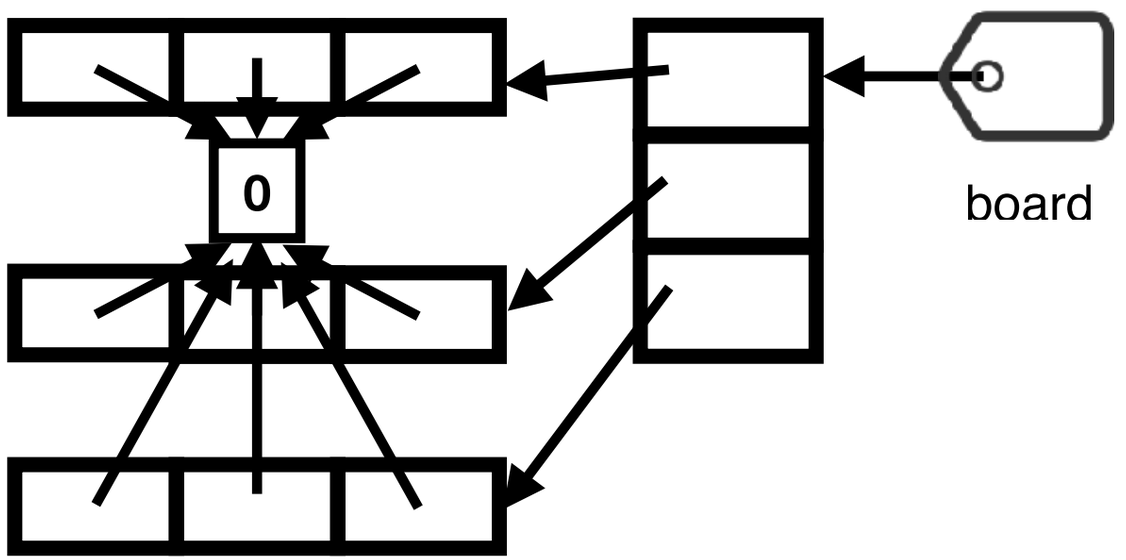


board = [[0] * cols for _ in range(rows)]



board = [[0] * cols] * rows

객체의 수도 적고
참조도 적게한다
= 속도가 빠른 이유



board = [[0] * cols for _ in range(rows)]

Why Bad?

```python
board = [[0] * 3] * 3  # 나쁜 결과
print(board)
print(id(board[0])) # 아래 객체와 동일함
print(id(board[1]))

print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936803144
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [1, 0, 0], [1, 0, 0]]
```

Why Bad?

```python
board = [[0] * 3] * 3  # 나쁜 결과
print(board)
print(id(board[0])) # 아래 객체와 동일함
print(id(board[1]))

print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936803144
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [1, 0, 0], [1, 0, 0]]
```

Why Bad?

```python
board = [[0] * 3] * 3  # 나쁜 결과
print(board)
print(id(board[0])) # 아래 객체와 동일함
print(id(board[1]))

print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936803144
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [1, 0, 0], [1, 0, 0]]
```

Why Good?

```python
board = [[0] * 3 for _ in range(3)]
print(board)
print(id(board[0]))      # 아래 객체와 다름
print(id(board[1]))
print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936802568
139869936802888
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Why Good?

```python
board = [[0] * 3 for _ in range(3)]
print(board)
print(id(board[0]))      # 아래 객체와 다름
print(id(board[1]))
print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936802568
139869936802888
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Why Good?

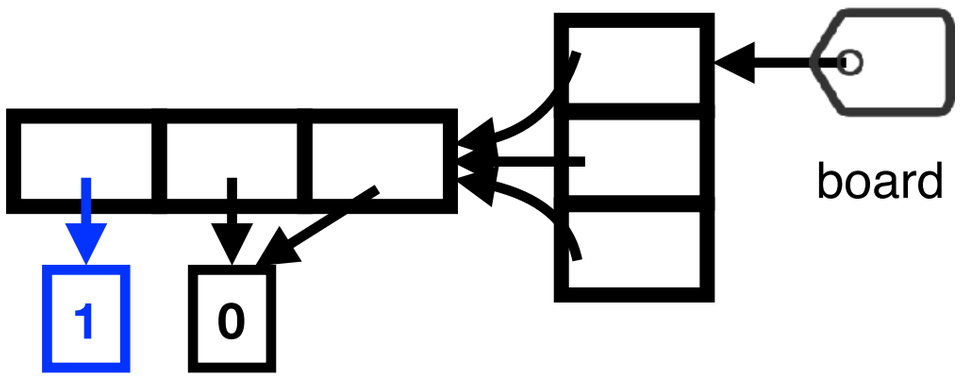
```python
board = [[0] * 3 for _ in range(3)]
print(board)
print(id(board[0]))      # 아래 객체와 다름
print(id(board[1]))
print(id(board[0][0]))
print(id(board[0][1]))
print(id(board[0][2]))
print(id(board[1][0]))
print(id(board[1][1]))
print(id(board[1][2]))

# 첫 항목 값만 고침
board[0][0] = 1
print(board)
```

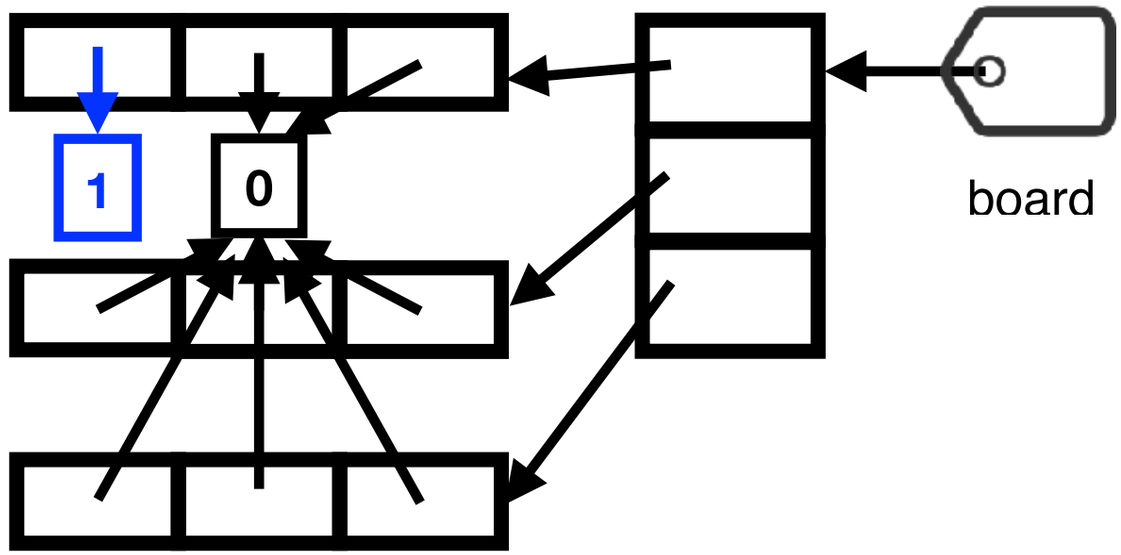
```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
139869936802568
139869936802888
10910368
10910368
10910368
10910368
10910368
10910368
10910368
[[1, 0, 0], [0, 0, 0], [0, 0, 0]]
```

board[0][0] = 1 결과

board = [[0] * cols] * rows

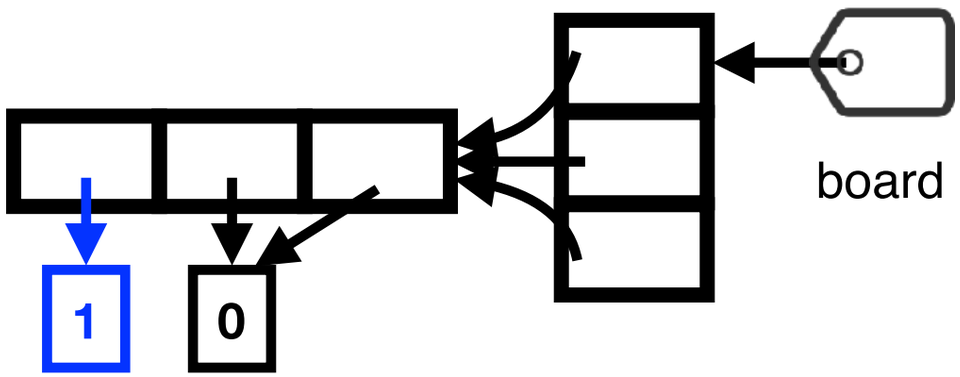


board = [[0] * cols for _ in range(rows)]

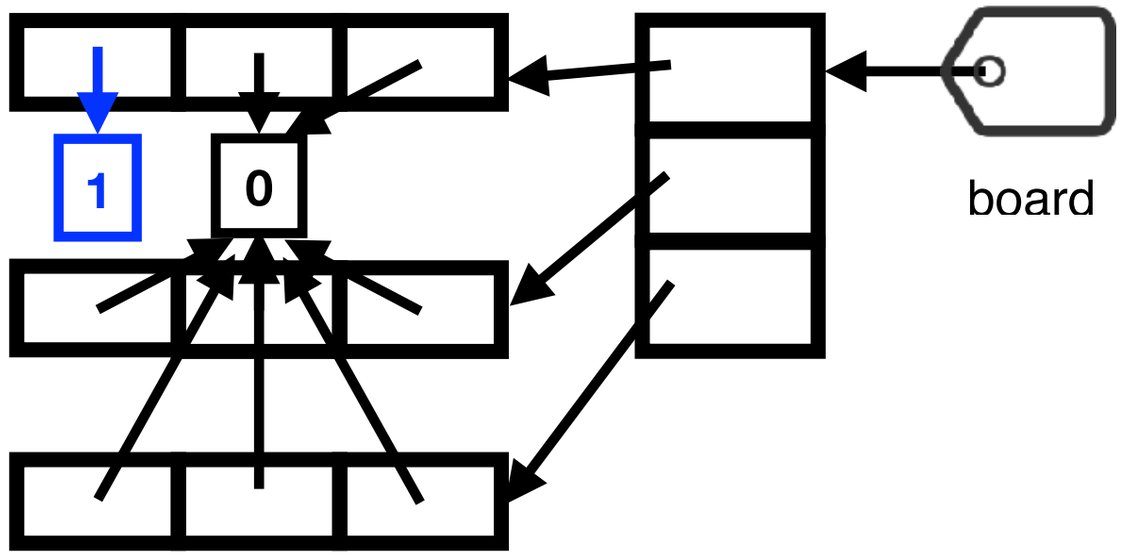


board[0][0] = 1 결과

board = [[0] * cols] * rows

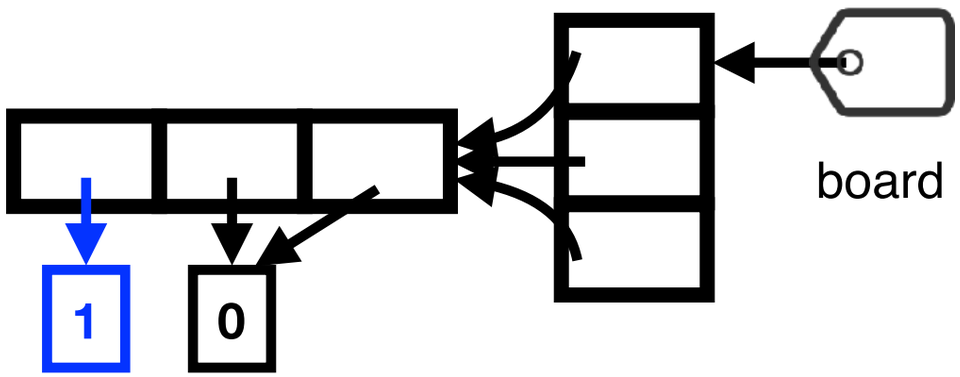


board = [[0] * cols for _ in range(rows)]

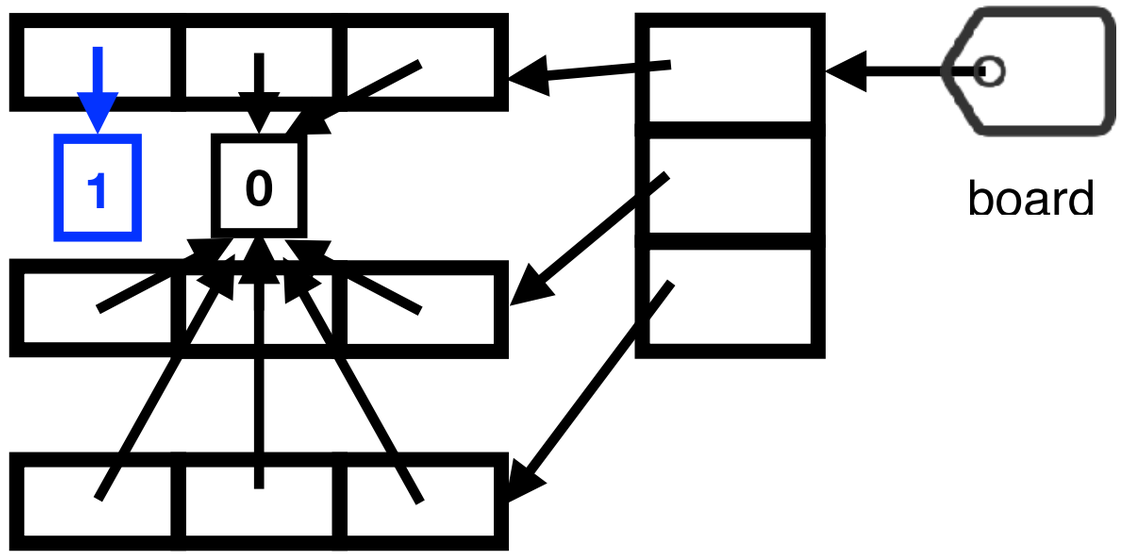


board[0][0] = 1 결과

board = [[0] * cols] * rows

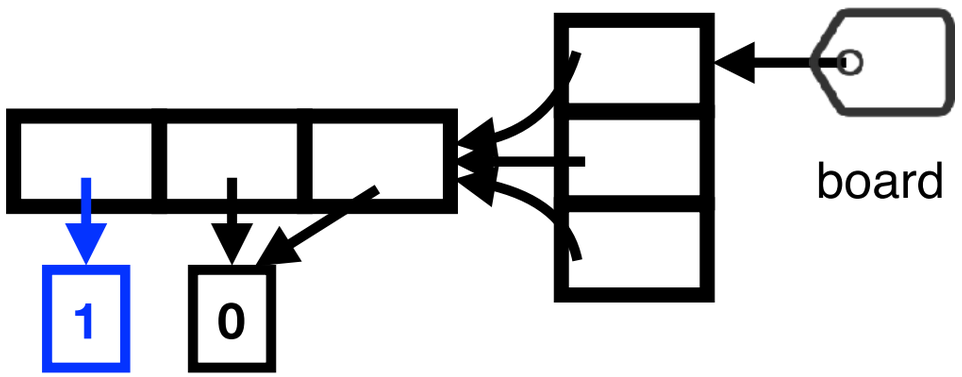


board = [[0] * cols for _ in range(rows)]

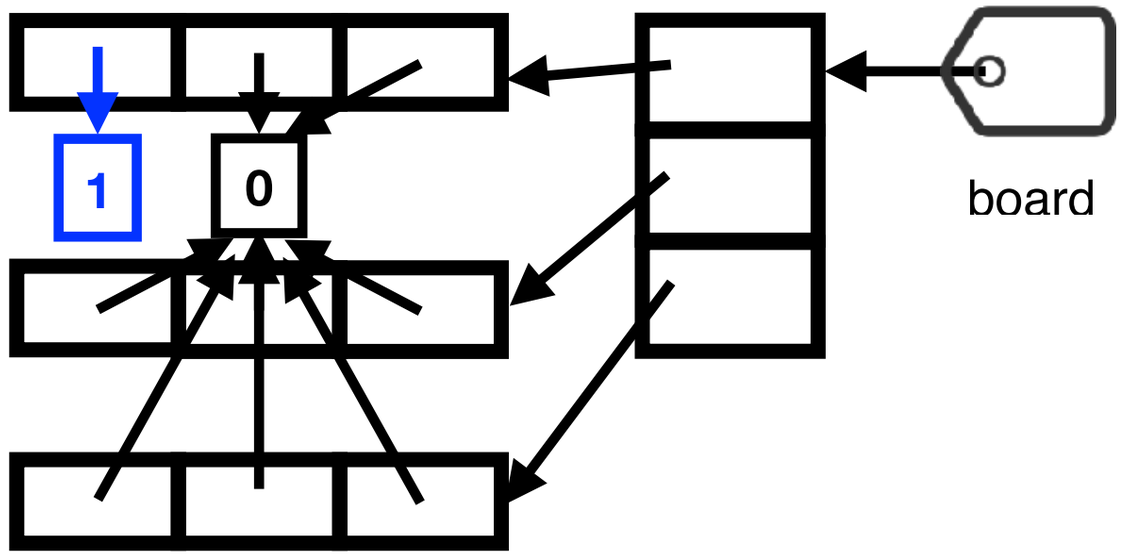


board[0][0] = 1 결과

board = [[0] * cols] * rows



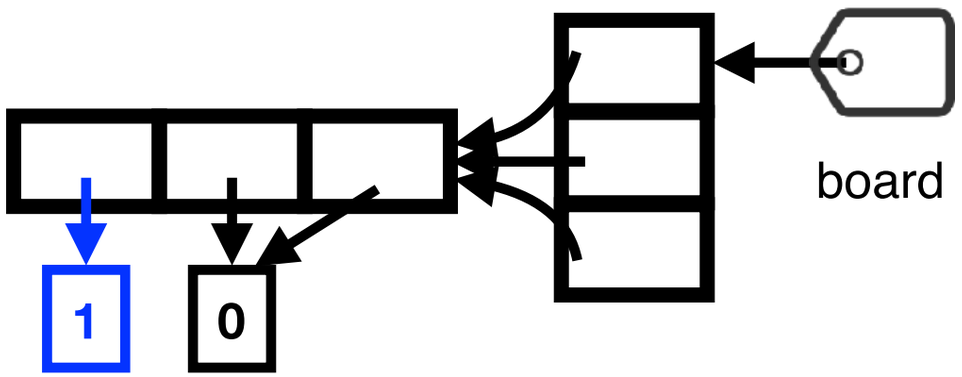
board = [[0] * cols for _ in range(rows)]



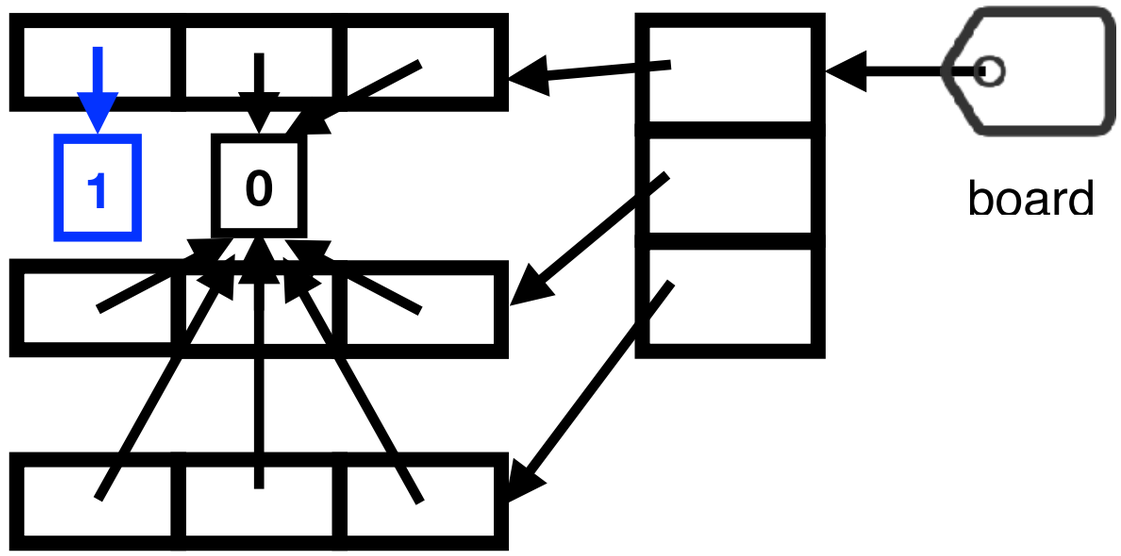
board[0] = [1, 0, 0]
board[1] = [1, 0, 0]
board[2] = [1, 0, 0]

board[0][0] = 1 결과

board = [[0] * cols] * rows



board = [[0] * cols for _ in range(rows)]

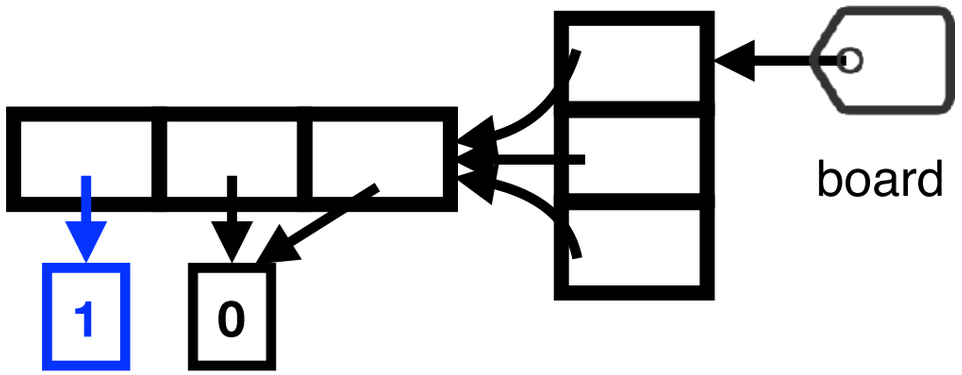


board[0] = [1, 0, 0]
board[1] = [1, 0, 0]
board[2] = [1, 0, 0]

Bad!!

board[0][0] = 1 결과

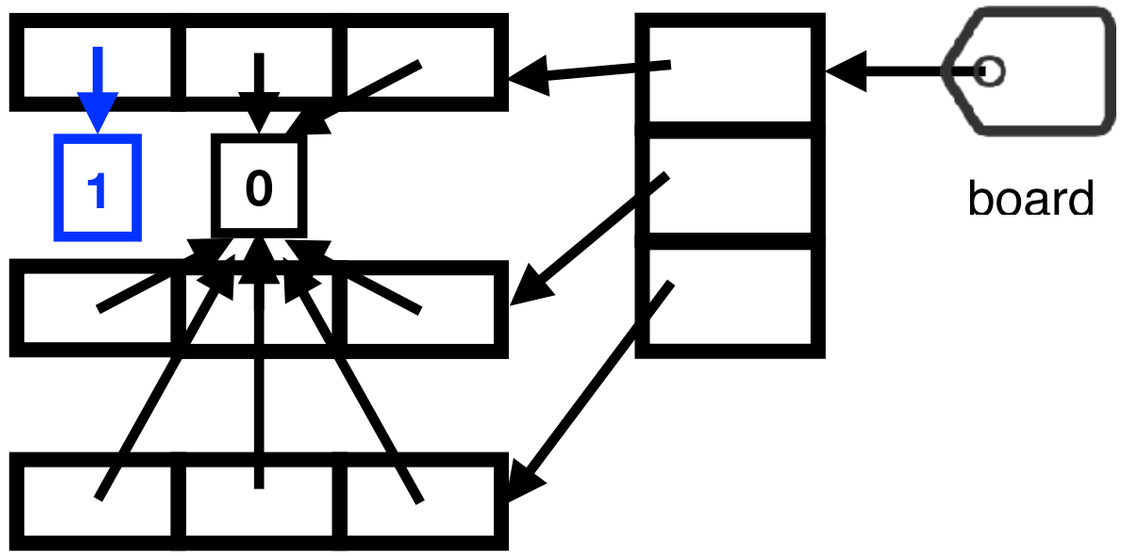
board = [[0] * cols] * rows



board[0] = [1, 0, 0]
board[1] = [1, 0, 0]
board[2] = [1, 0, 0]

Bad!!

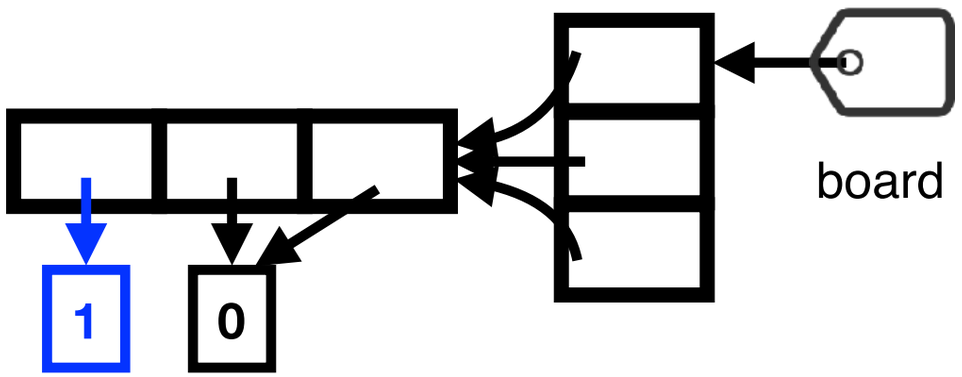
board = [[0] * cols for _ in range(rows)]



board[0] = [1, 0, 0]
board[1] = [0, 0, 0]
board[2] = [0, 0, 0]

board[0][0] = 1 결과

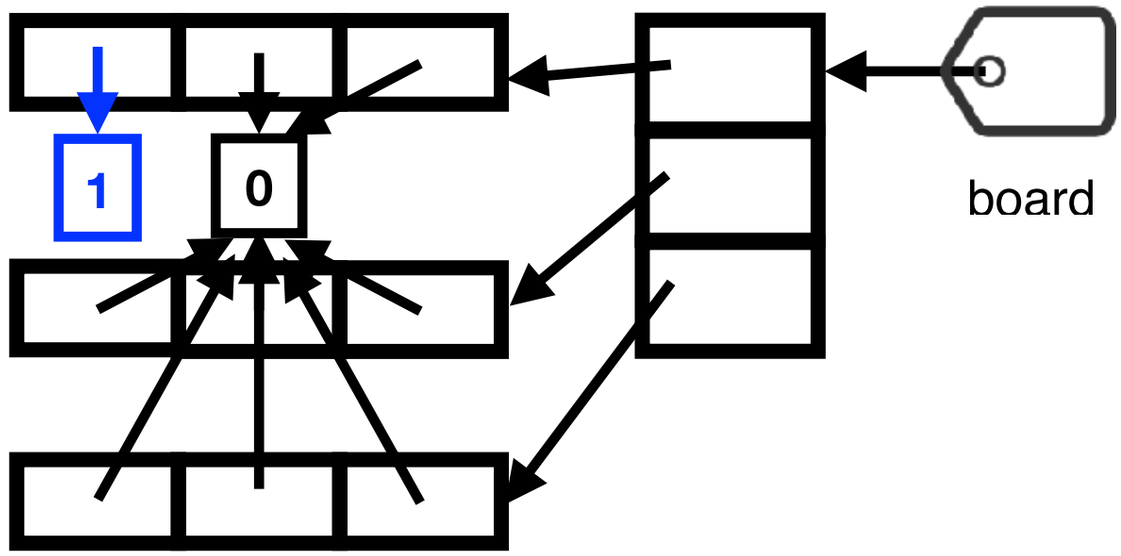
board = [[0] * cols] * rows



board[0] = [1, 0, 0]
board[1] = [1, 0, 0]
board[2] = [1, 0, 0]

Bad!!

board = [[0] * cols for _ in range(rows)]



board[0] = [1, 0, 0]
board[1] = [0, 0, 0]
board[2] = [0, 0, 0]

Good!!

Lab

감사합니다.