교수의 파이썬

01_2 리스트의 요소는 참조형이다

창원대학교 정보통신공학과 교수 박동규

널널한 교수의 파이썬

01_2 리스트의 요소는 참조형이다

창원대학교 정보통신공학과 교수 박동규

널널한 교수의 고급 파이썬

01_2 리스트의 요소는 참조형이다

창원대학교 정보통신공학과 교수 박동규

리스트

```
>>> list1 = ["one", "two", 3, 4]
>>> list2 = [1, 2, 3, 4]
>>> list3 = ["one", "two", "three", "four"]
>>> list3[1]
'two'
```

리스트

- 파이썬의 리스트는 타 언어의 배열과 비슷해 보인다(??).

```
>>> list1 = ["one", "two", 3, 4]
>>> list2 = [1, 2, 3, 4]
>>> list3 = ["one", "two", "three", "four"]
>>> list3[1]
'two'
```

리스트

- 파이썬의 리스트는 타 언어의 배열과 비슷해 보인다(??).
- 하지만 하나의 리스트에 서로 다른 자료형의 항목을 포함할 수 있다.(리스트, 딕셔너리 등을 포함할 수 있다)

```
>>> list1 = ["one", "two", 3, 4]
>>> list2 = [1, 2, 3, 4]
>>> list3 = ["one", "two", "three", "four"]
>>> list3[1]
'two'
```

리스트

- 파이썬의 리스트는 타 언어의 배열과 비슷해 보인다(??).
- 하지만 하나의 리스트에 서로 다른 자료형의 항목을 포함할 수 있다.(리스트, 딕셔너리 등을 포함할 수 있다)
- 매우 강력한 기능이 있는 자료형이다.

```
>>> list1 = ["one", "two", 3, 4]
>>> list2 = [1, 2, 3, 4]
>>> list3 = ["one", "two", "three", "four"]
>>> list3[1]
'two'
```

C

배열

Python

리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

Python

리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a

Python

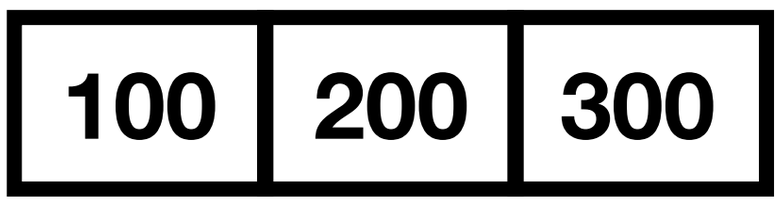
리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a



Python

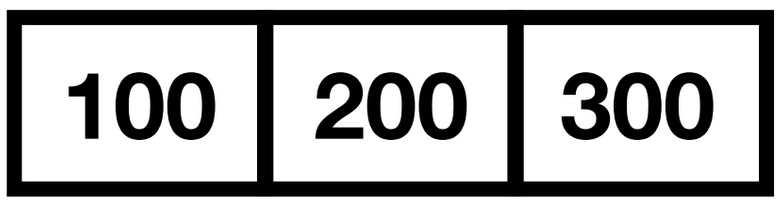
리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a



배열 원소의 크기는 int 형으로 고정됨

Python

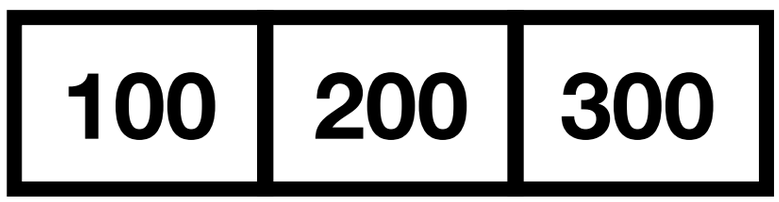
리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a



배열 원소의 크기는 int 형으로 고정됨

Python

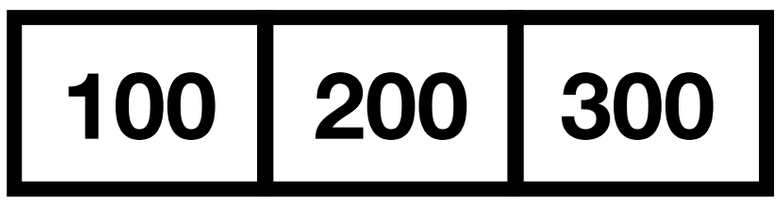
리스트

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a



배열 원소의 크기는 int 형으로 고정됨

Python

리스트

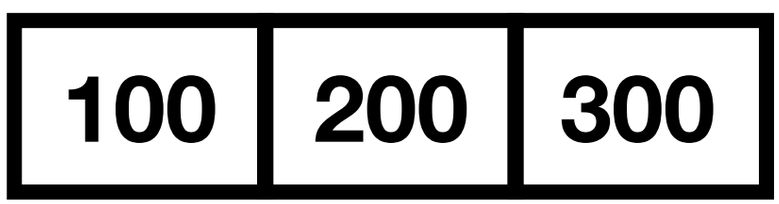
```
# 리스트 객체 생성
a = [100, 200, 300]
```

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a



배열 원소의 크기는 int 형으로 고정됨

Python

리스트

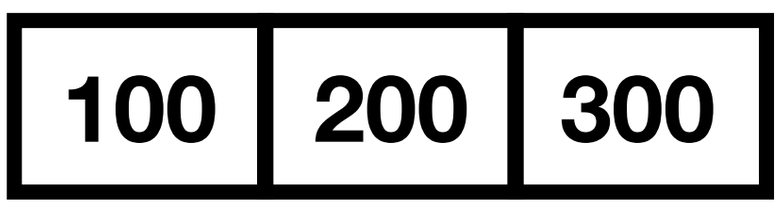
```
# 리스트 객체 생성
a = [100, 200, 300]
```

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a

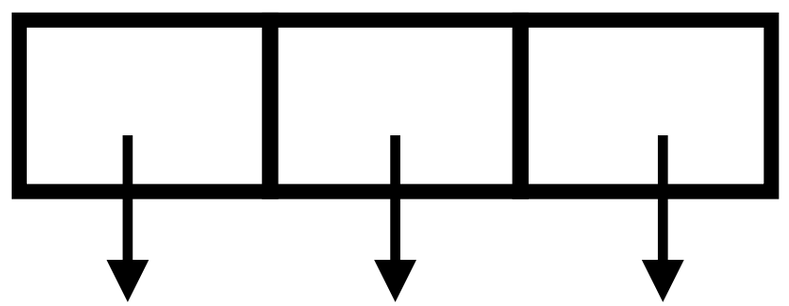


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

```
# 리스트 객체 생성
a = [100, 200, 300]
```

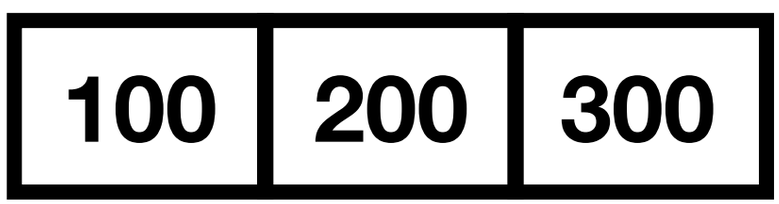


C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

a

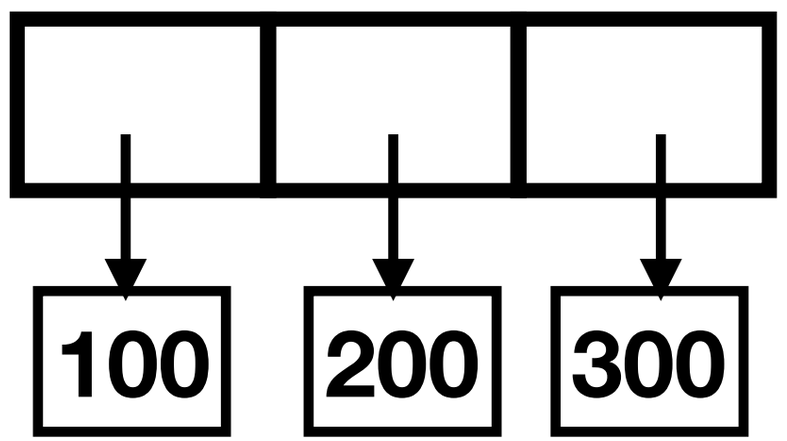


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

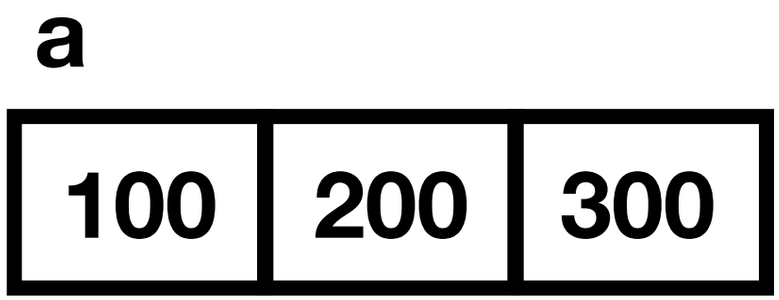
```
# 리스트 객체 생성
a = [100, 200, 300]
```



C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

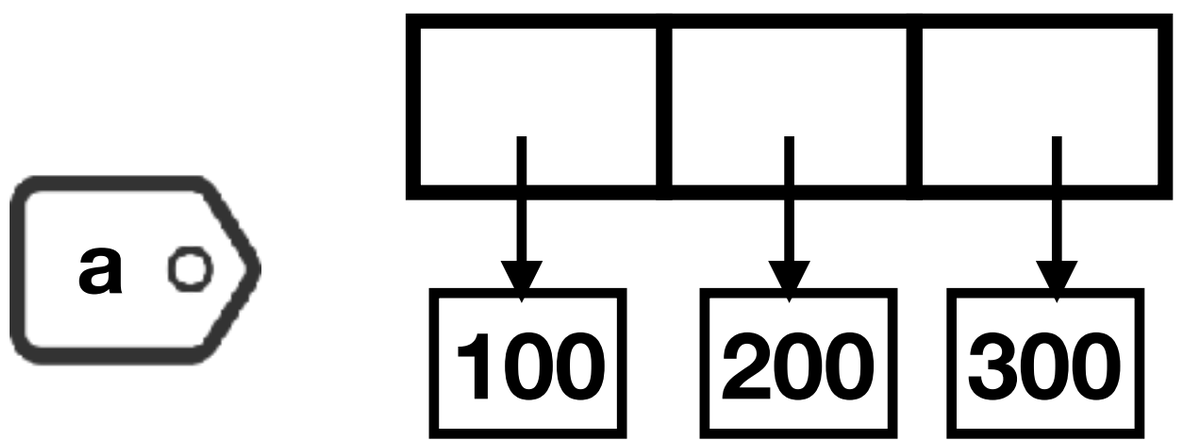


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

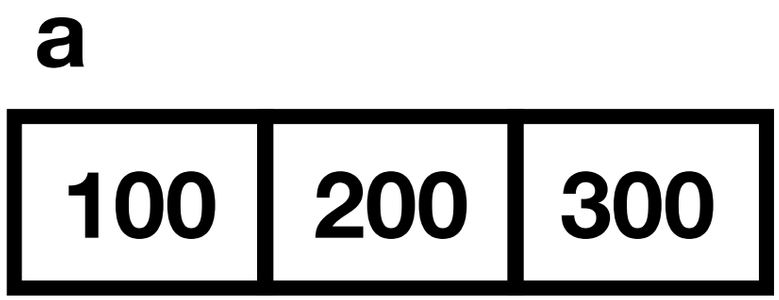
```
# 리스트 객체 생성
a = [100, 200, 300]
```



C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

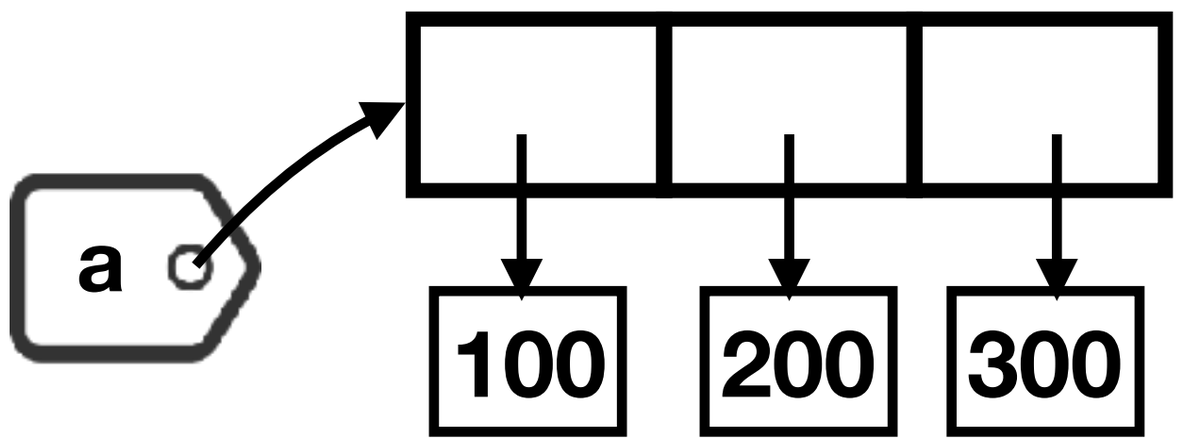


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

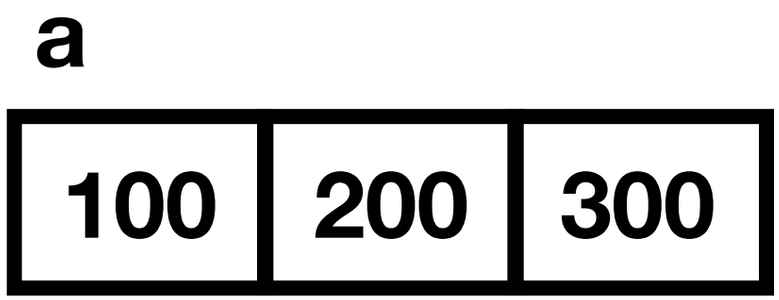
```
# 리스트 객체 생성
a = [100, 200, 300]
```



C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

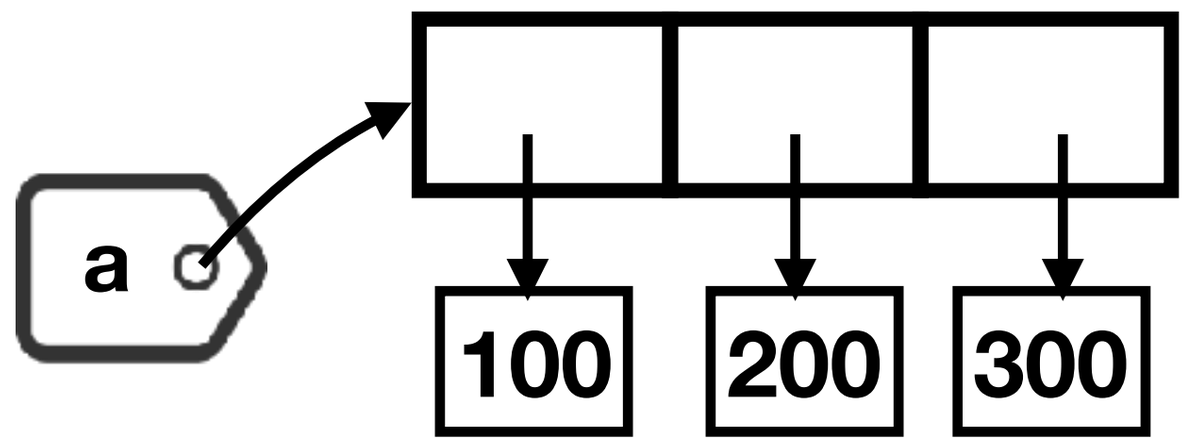


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

```
# 리스트 객체 생성
a = [100, 200, 300]
```

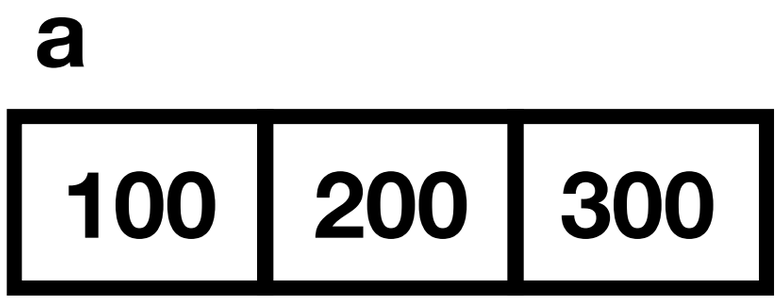


리스트의 항목은 서로 다른 자료형도 가능
리스트 요소 a[0], a[1], a[2]는 참조형임

C

배열

```
// 배열의 선언과 초기화
int a[3] = {100, 200, 300};
```

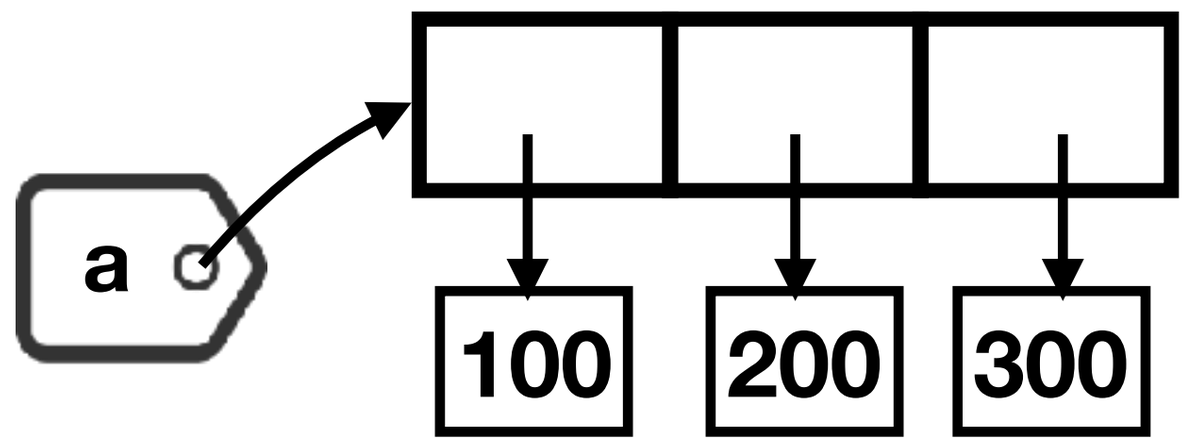


배열 원소의 크기는 int 형으로 고정됨

Python

리스트

```
# 리스트 객체 생성
a = [100, 200, 300]
```

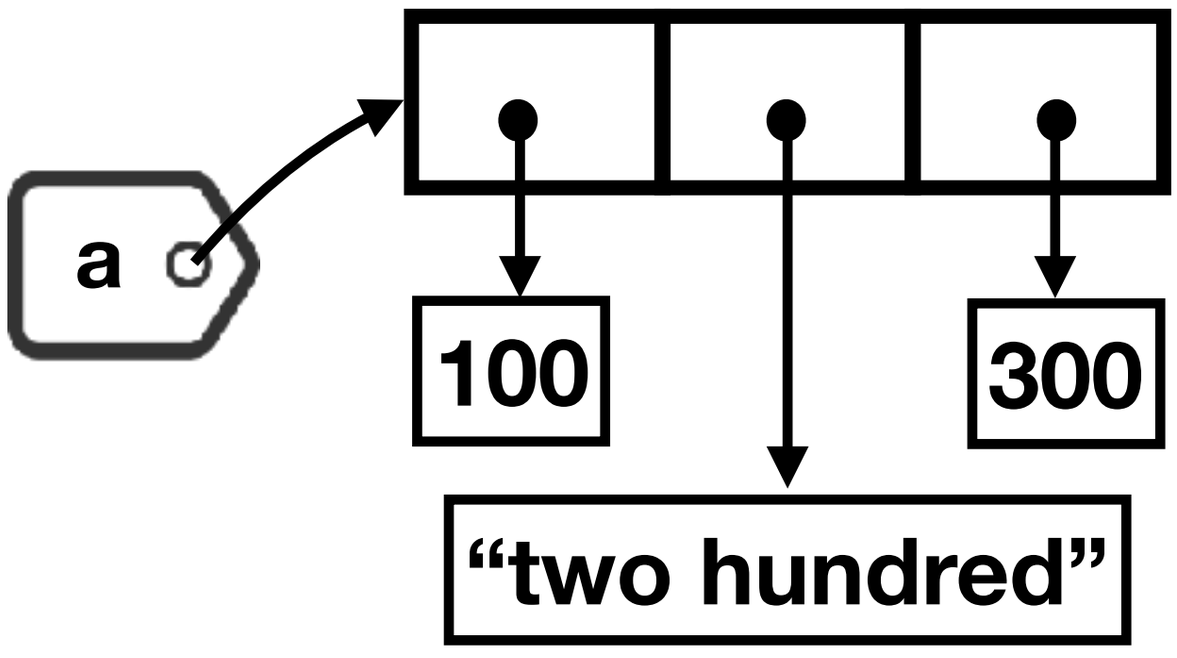


리스트의 항목은 서로 다른 자료형도 가능
리스트 요소 a[0], a[1], a[2]는 참조형임

elements are references
리스트 요소는 참조형이다!!

 # 리스트 객체 생성

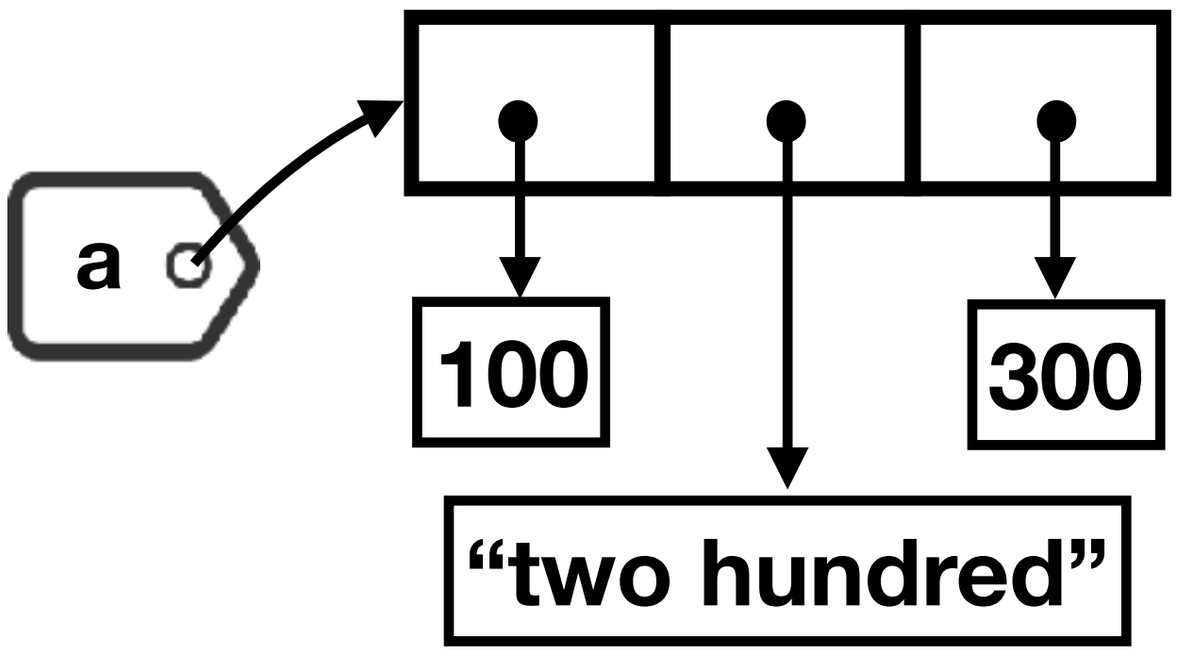
a = [100, “two hundred”, 300]




```
>>> print(a[0])
100
>>> print(a[1])
two hundred
>>> id(a)
4512940232
>>> id(a[0])
4509016272
>>> id(a[1])
4512941744
```

elements are references
리스트 요소는 참조형이다!!

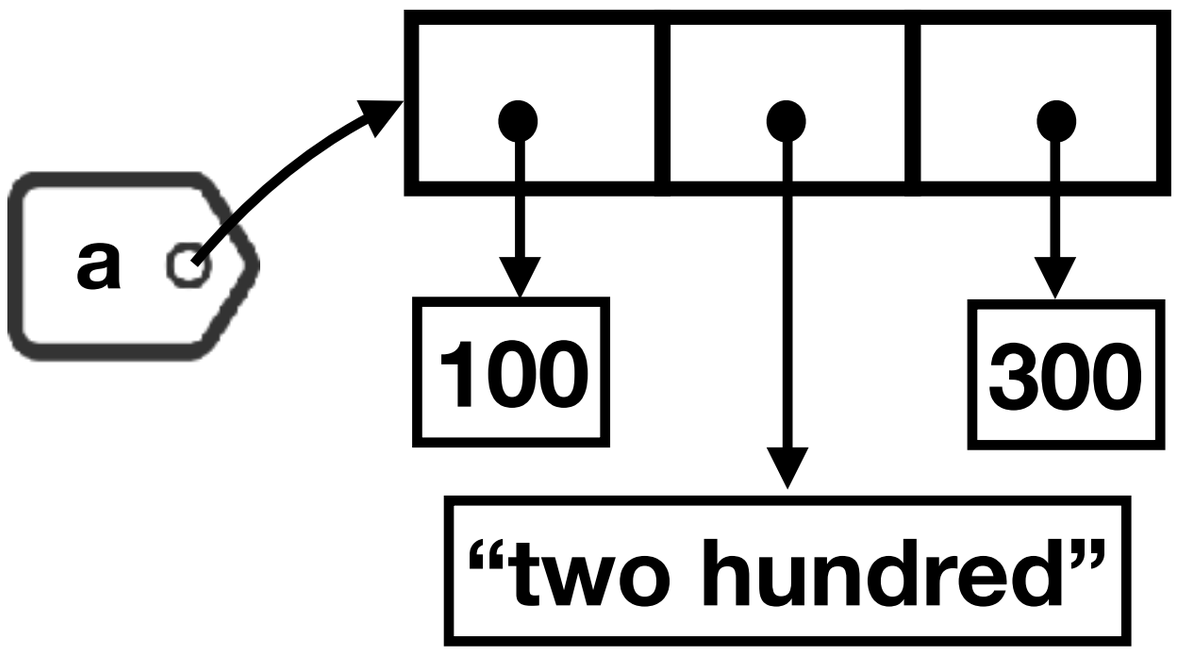
```
# 리스트 객체 생성
a = [100, “two hundred”, 300]
print(a[0])
```




```
>>> print(a[0])
100
>>> print(a[1])
two hundred
>>> id(a)
4512940232
>>> id(a[0])
4509016272
>>> id(a[1])
4512941744
```

elements are references
리스트 요소는 참조형이다!!

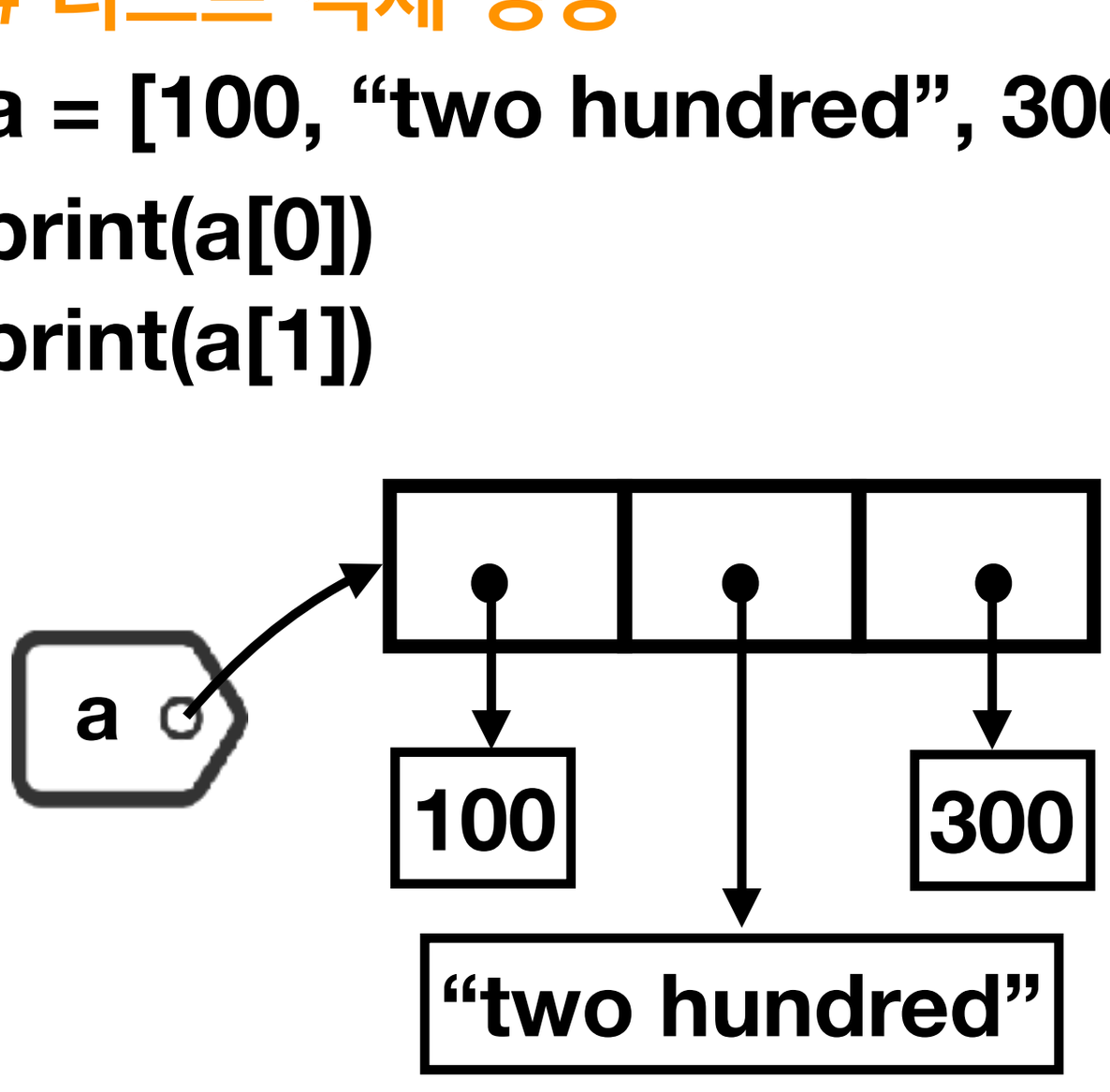
```
# 리스트 객체 생성
a = [100, “two hundred”, 300]
print(a[0])
```




```
>>> print(a[0])
100
>>> print(a[1])
two hundred
>>> id(a)
4512940232
>>> id(a[0])
4509016272
>>> id(a[1])
4512941744
```

elements are references
리스트 요소는 참조형이다!!

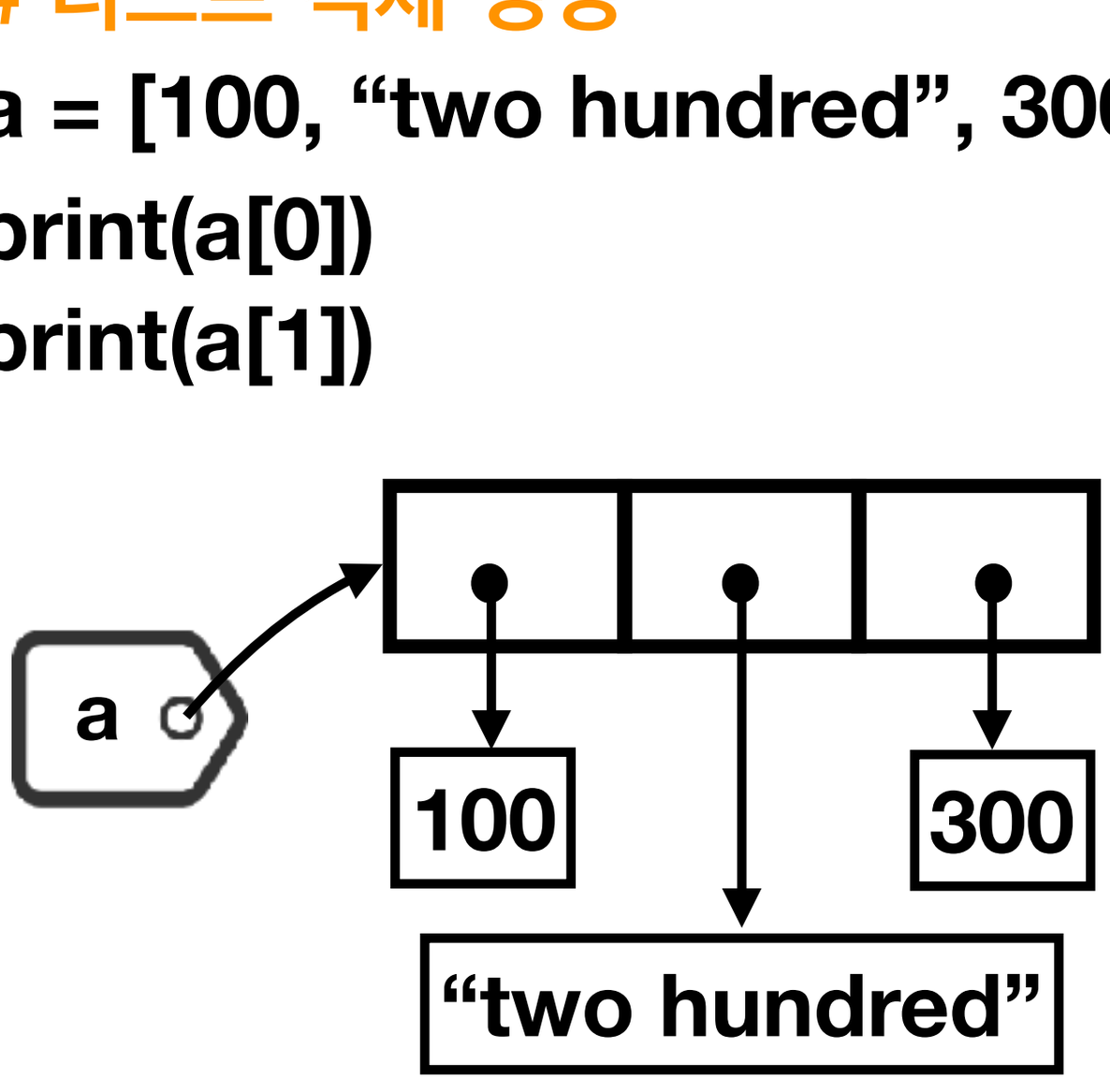
```
# 리스트 객체 생성
a = [100, “two hundred”, 300]
print(a[0])
print(a[1])
```




```
>>> print(a[0])
100
>>> print(a[1])
two hundred
>>> id(a)
4512940232
>>> id(a[0])
4509016272
>>> id(a[1])
4512941744
```

elements are references
리스트 요소는 참조형이다!!

```
# 리스트 객체 생성
a = [100, “two hundred”, 300]
print(a[0])
print(a[1])
```

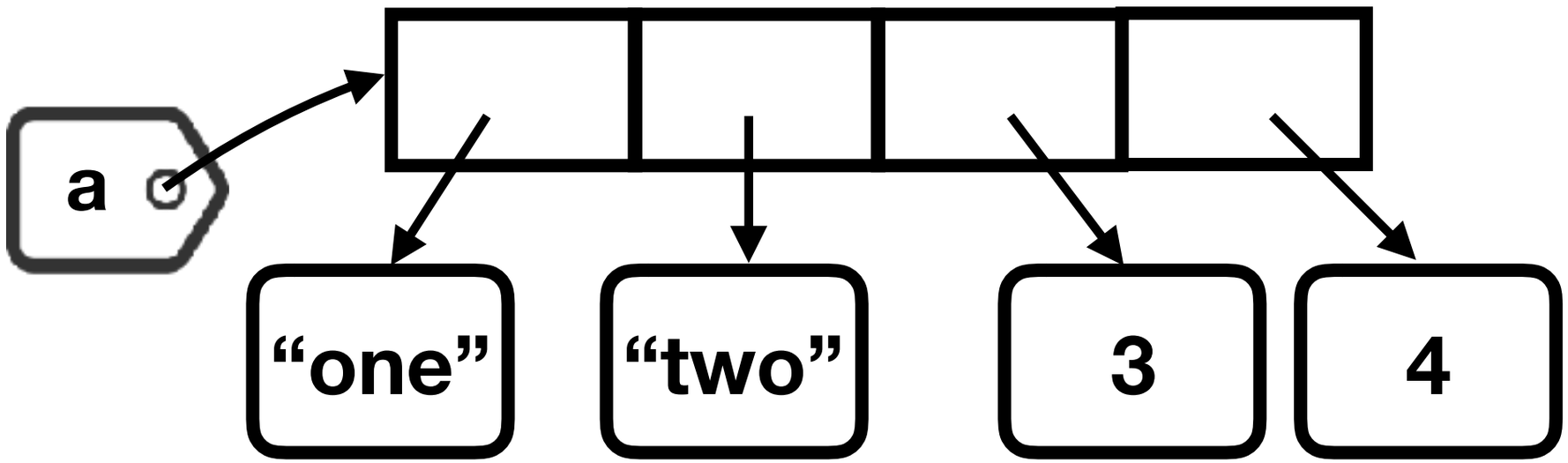



```
>>> print(a[0])
100
>>> print(a[1])
two hundred
>>> id(a)
4512940232
>>> id(a[0])
4509016272
>>> id(a[1])
4512941744
```

리스트 요소의 재할당

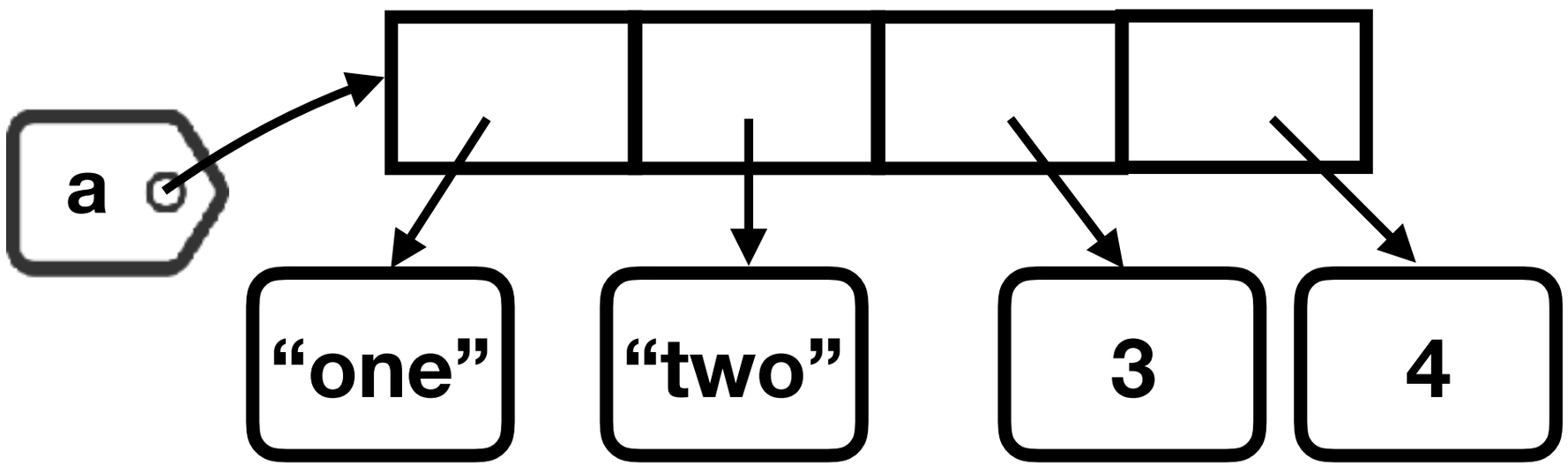
 # 리스트 객체 생성

a = [“one”, “two”, 3, 4]



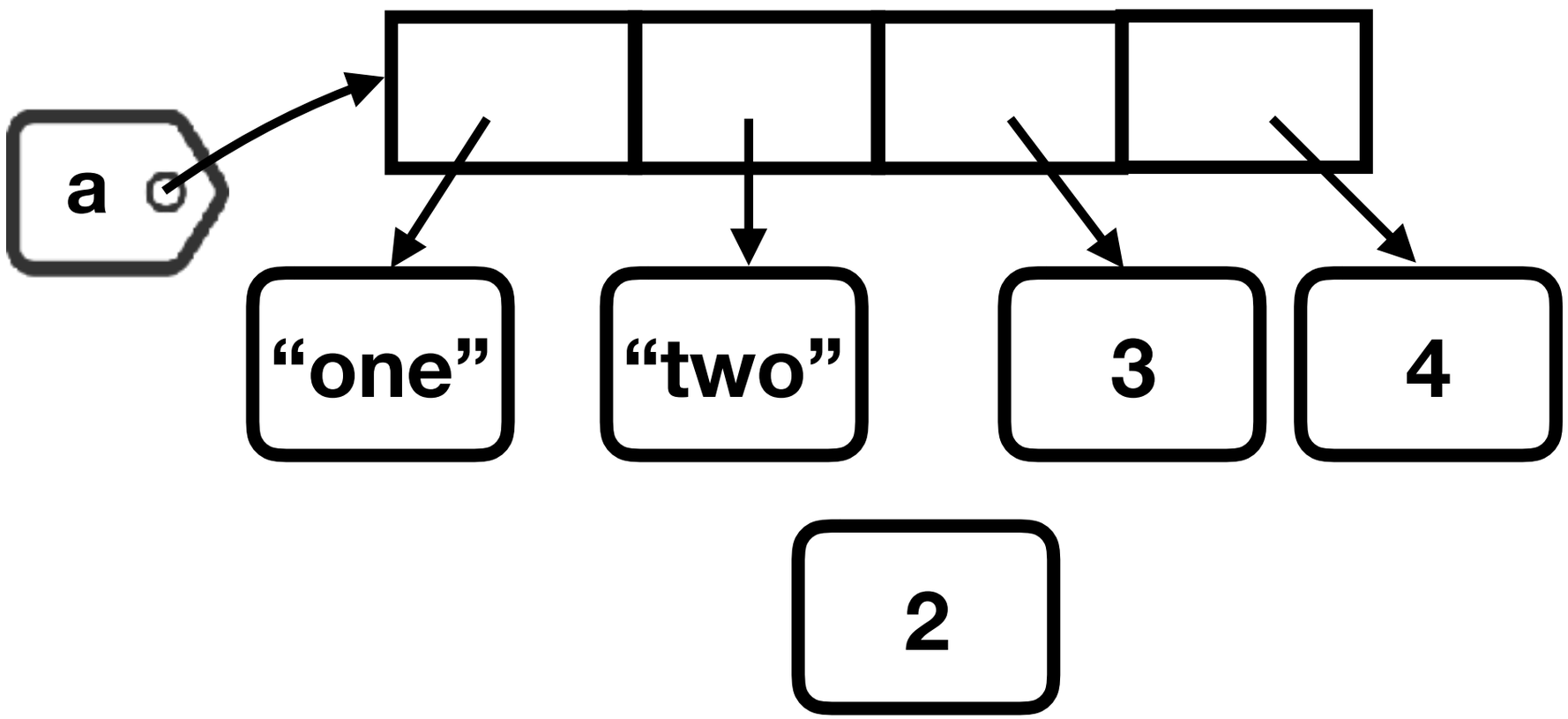
리스트 요소의 재할당

```
# 리스트 객체 생성
a = [“one”, “two”, 3, 4]
# 리스트 요소의 재할당
a[1] = 2
```



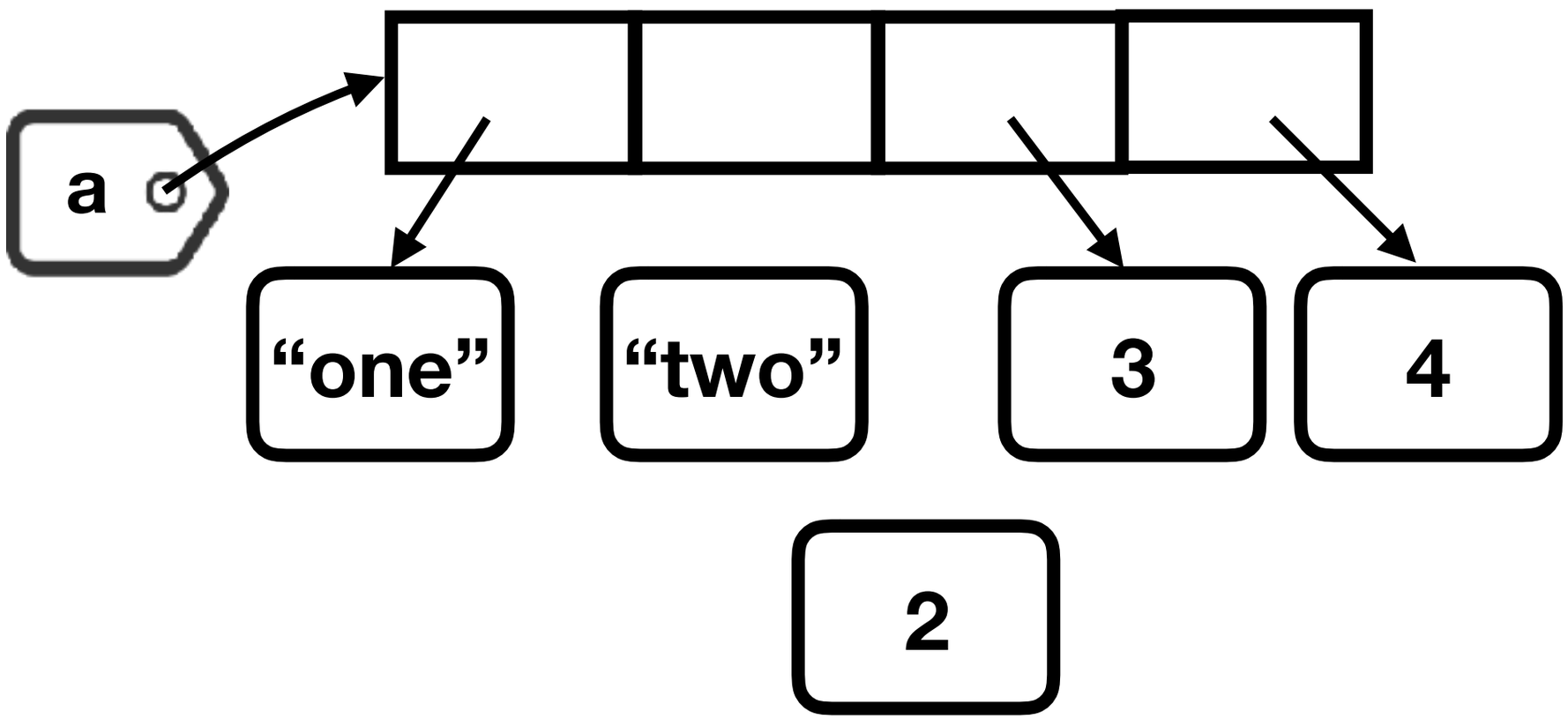
리스트 요소의 재할당

```
# 리스트 객체 생성
a = [“one”, “two”, 3, 4]
# 리스트 요소의 재할당
a[1] = 2
```



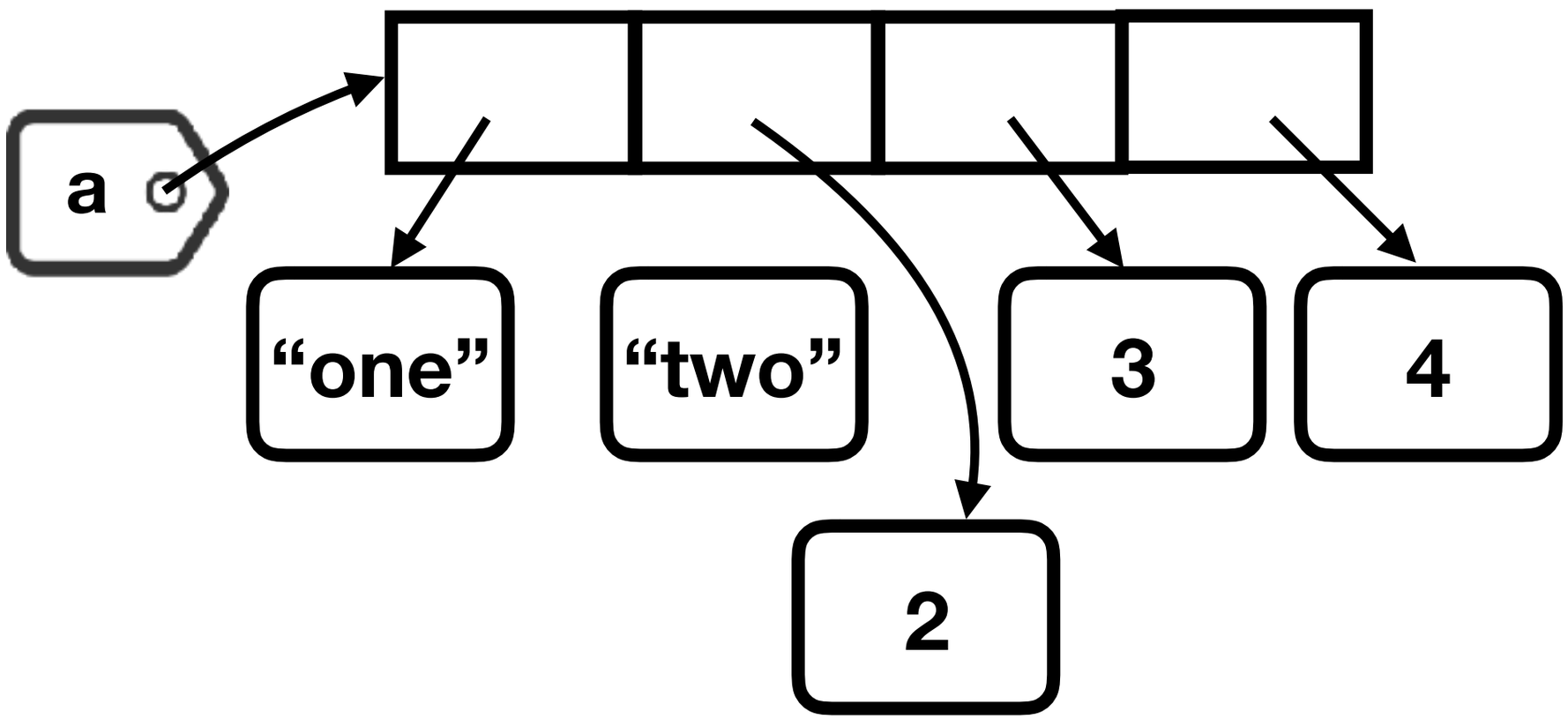
리스트 요소의 재할당

```
# 리스트 객체 생성
a = [“one”, “two”, 3, 4]
# 리스트 요소의 재할당
a[1] = 2
```



리스트 요소의 재할당

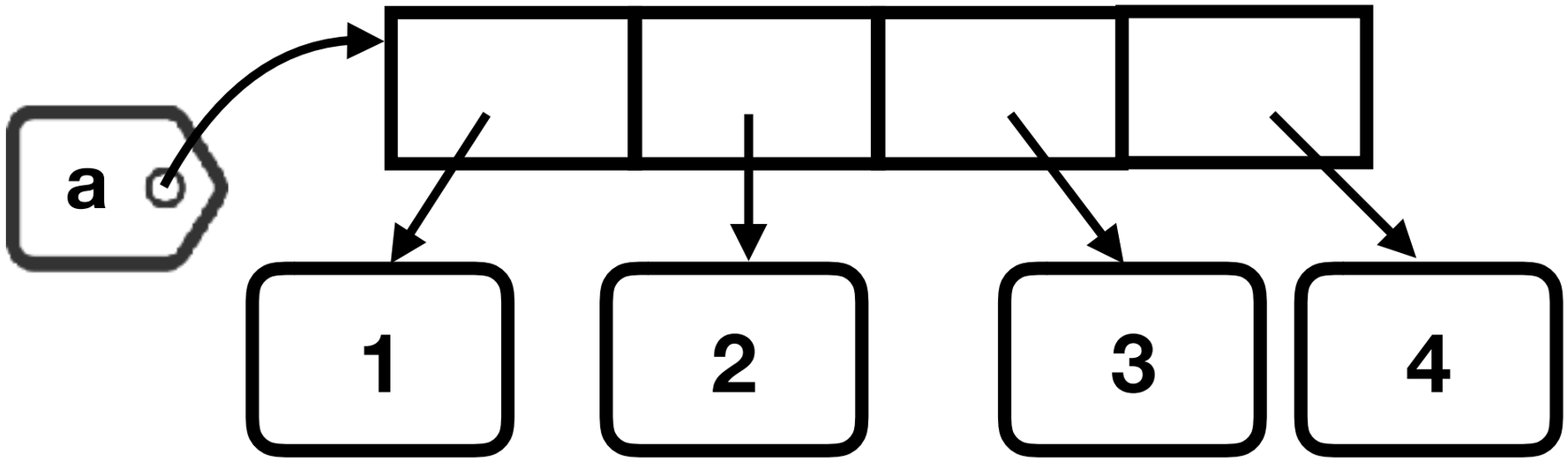
```
# 리스트 객체 생성
a = [“one”, “two”, 3, 4]
# 리스트 요소의 재할당
a[1] = 2
```



리스트 요소의 추가

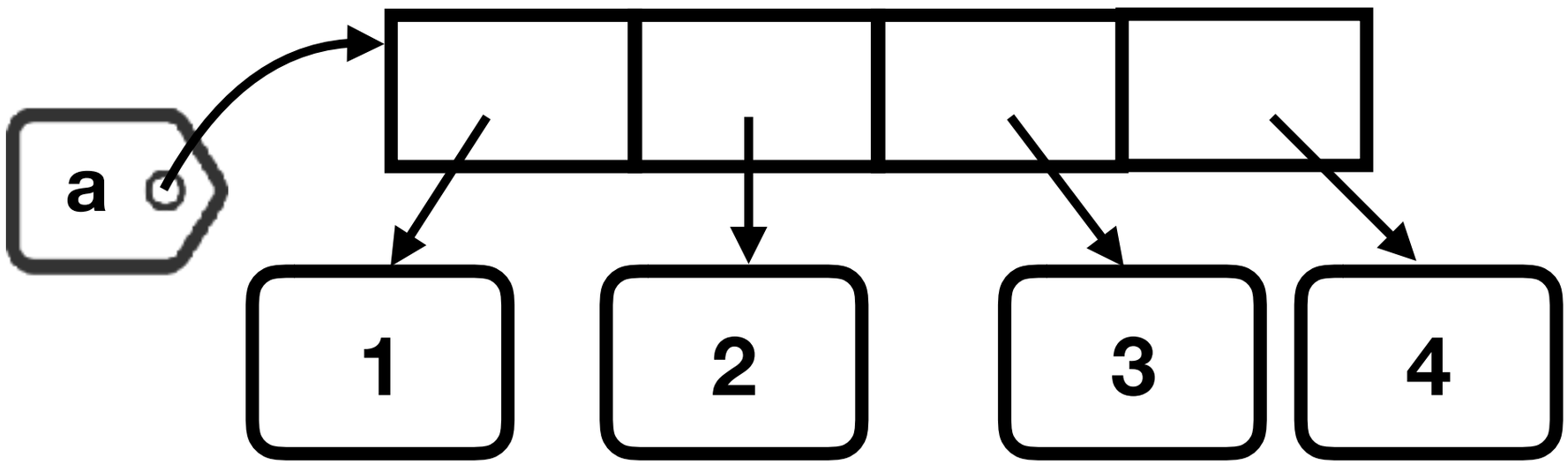
 # 리스트 객체 생성

a = [1, 2, 3, 4]



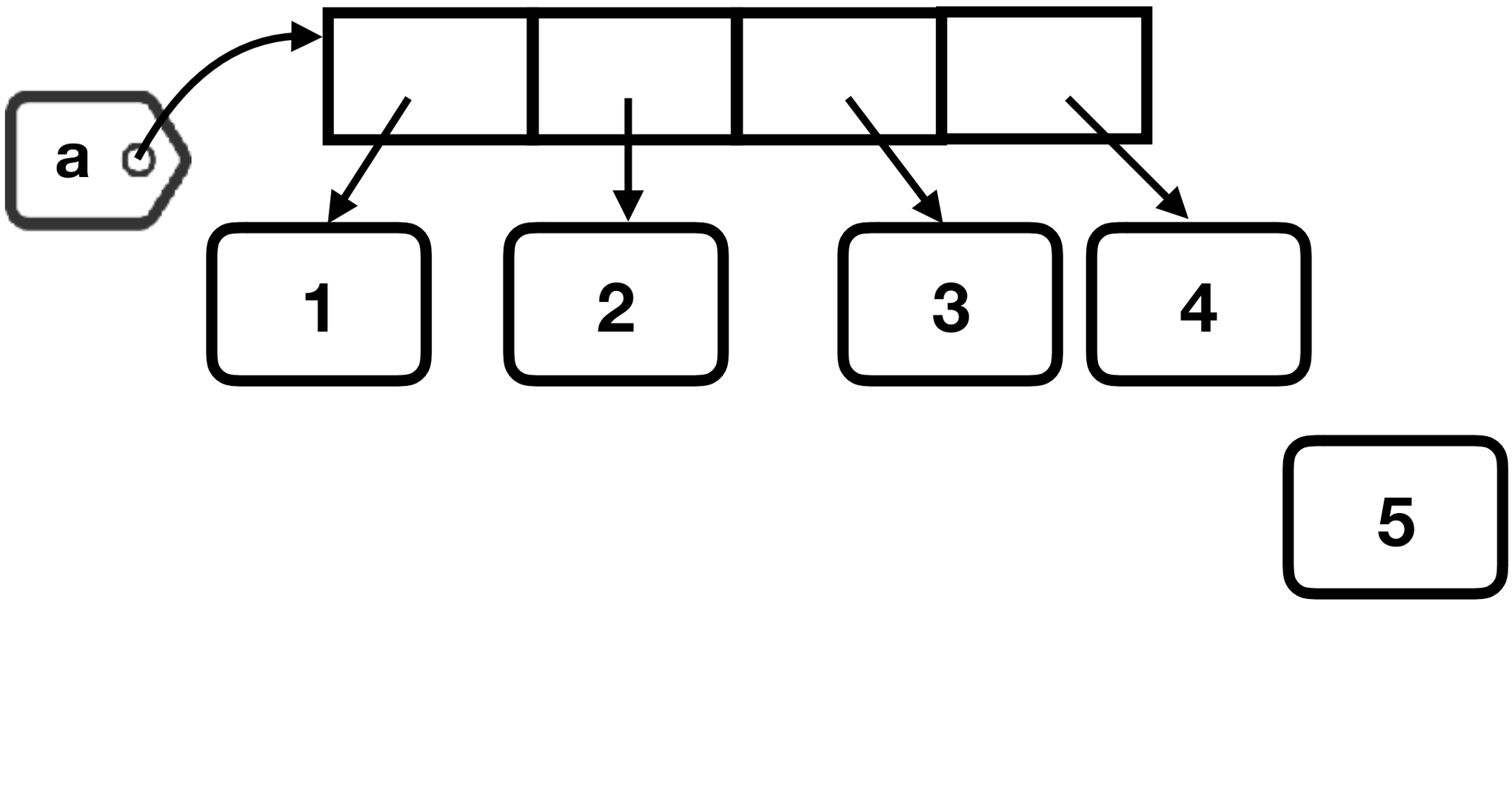
리스트 요소의 추가

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a.append(5)    # 리스트 객체의 변경(mutating)
```



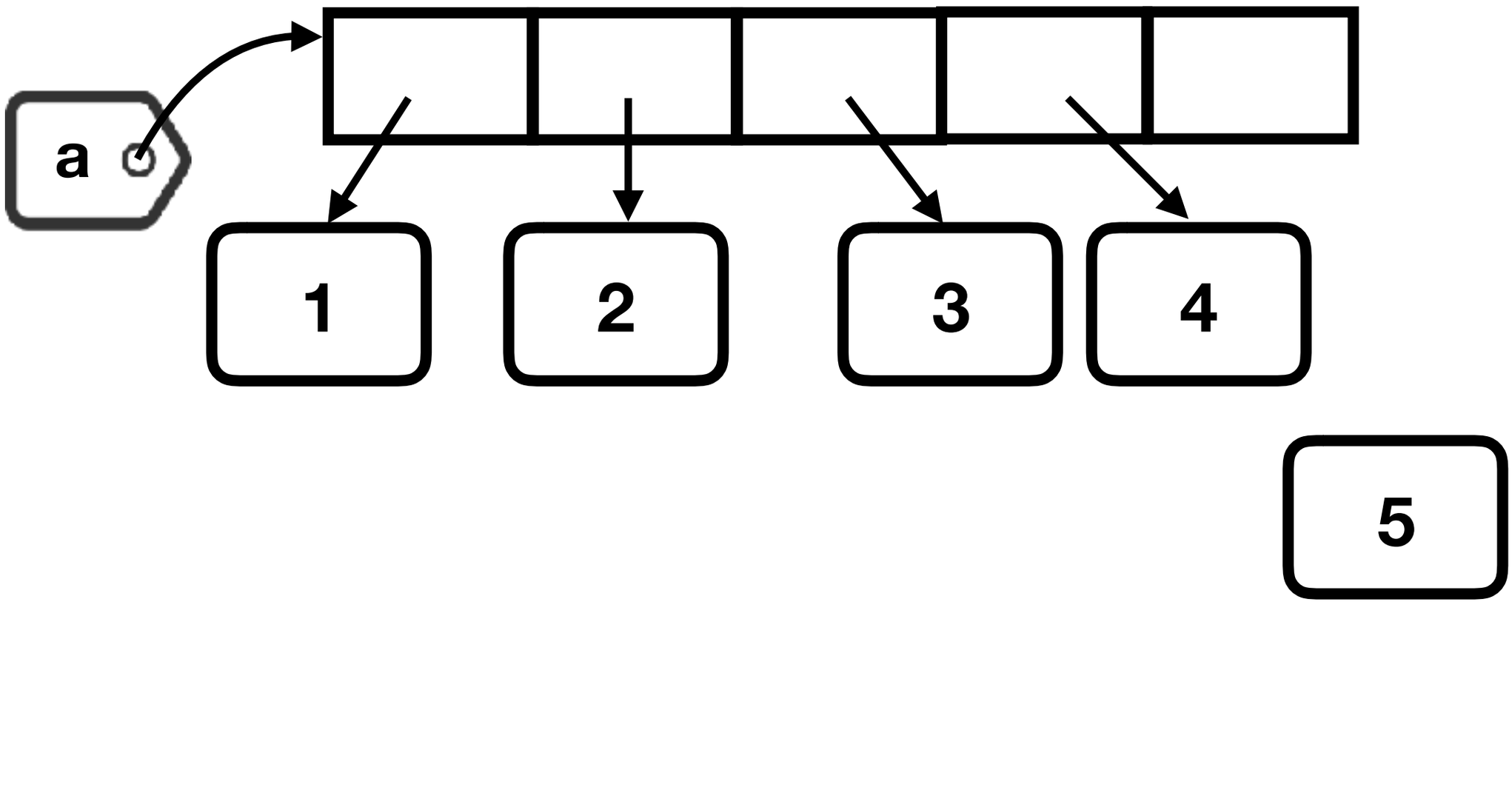
리스트 요소의 추가

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a.append(5)    # 리스트 객체의 변경(mutating)
```



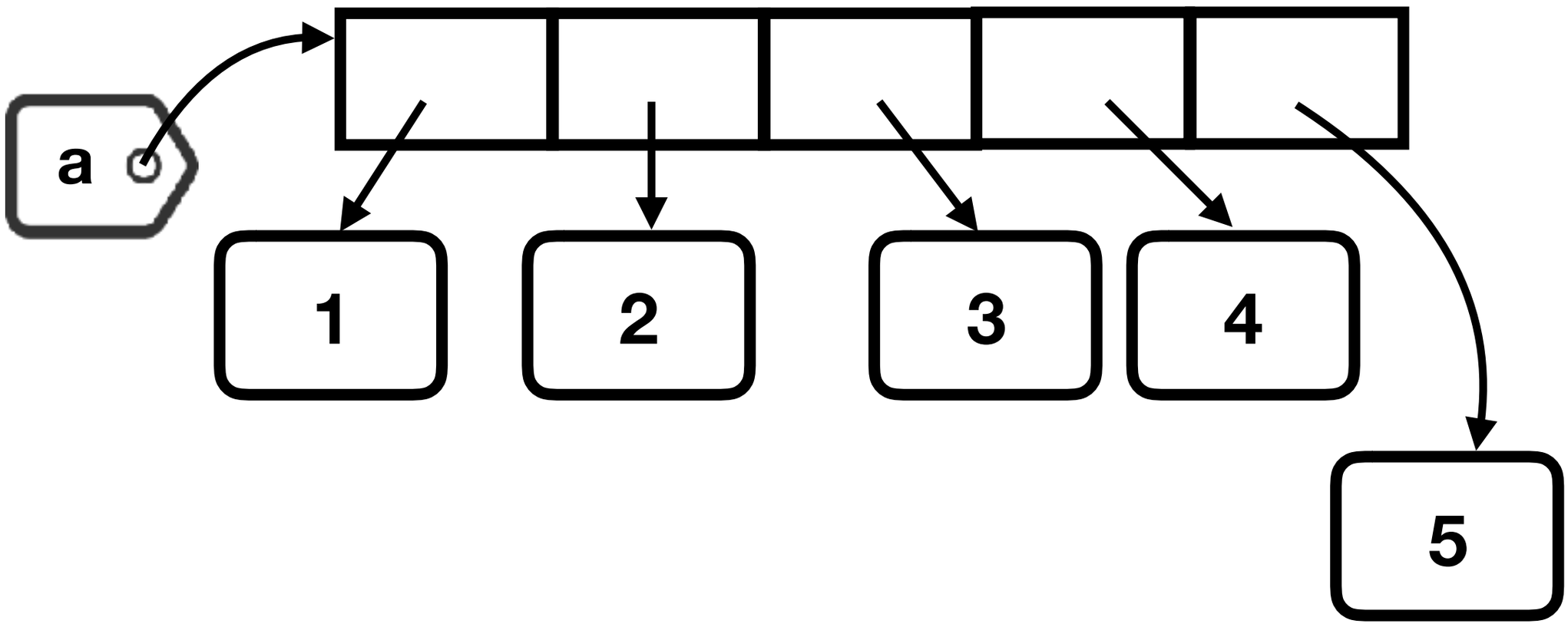
리스트 요소의 추가

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a.append(5)    # 리스트 객체의 변경(mutating)
```



리스트 요소의 추가

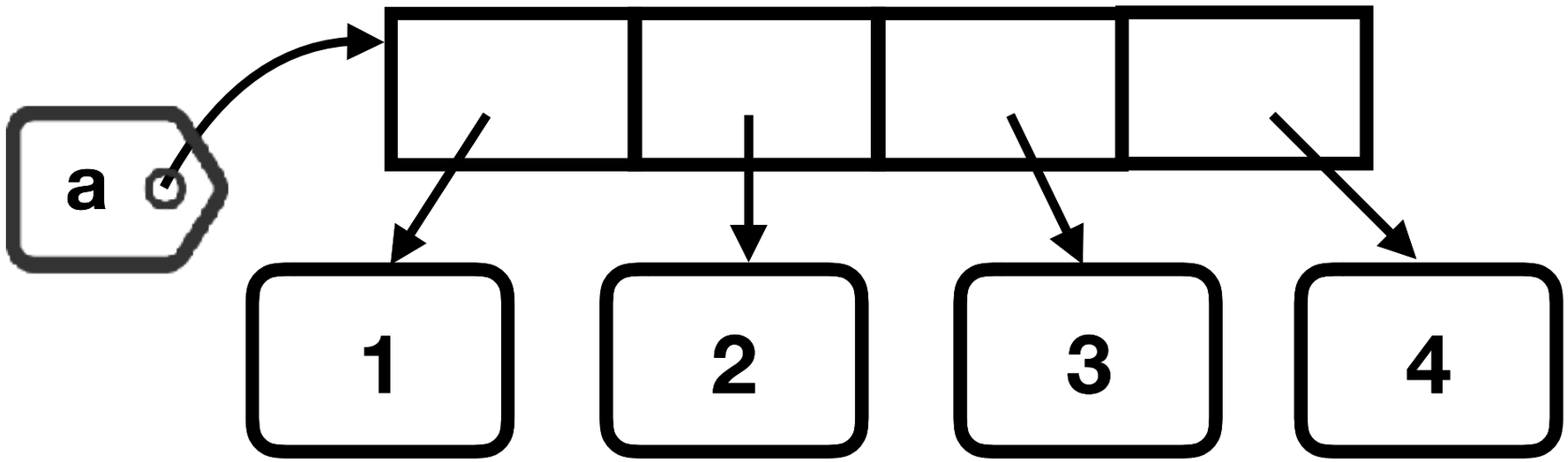
```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a.append(5)    # 리스트 객체의 변경(mutating)
```



리스트의 덧셈과 재할당

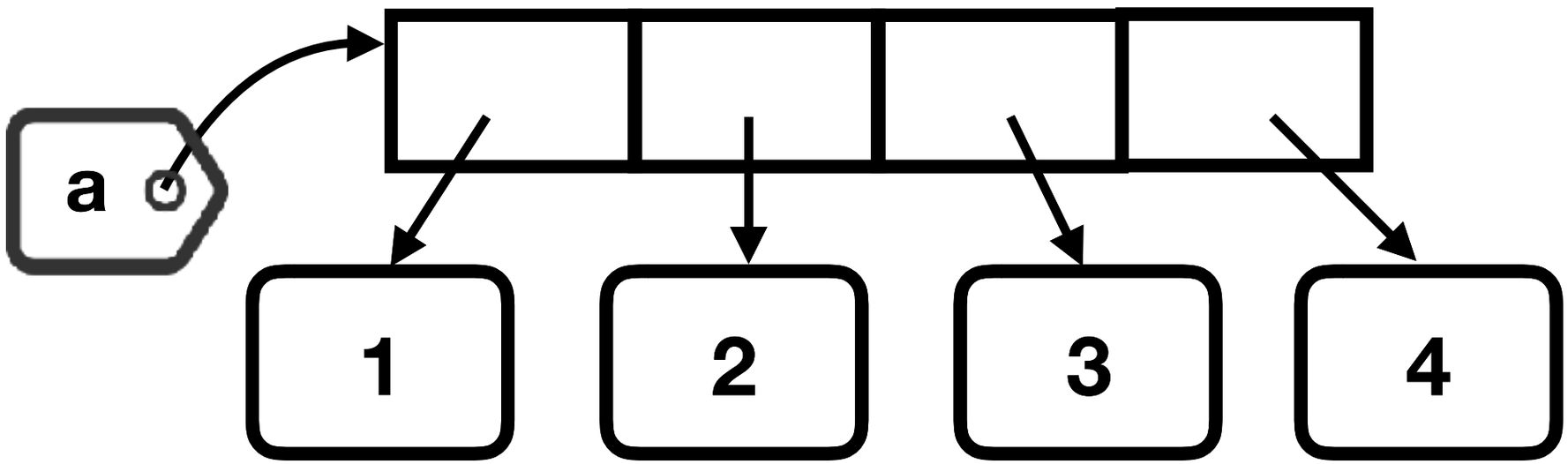
 # 리스트 객체 생성

a = [1, 2, 3, 4]



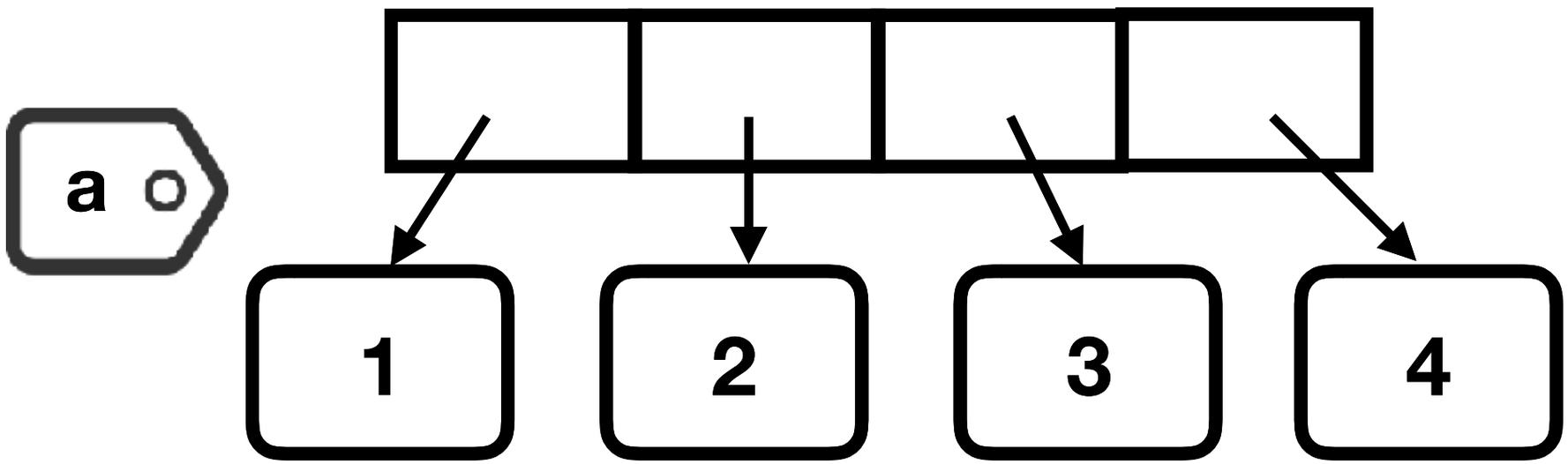
리스트의 덧셈과 재할당

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a = a + [5]   # 리스트 객체의 재바인딩(rebinding)
```



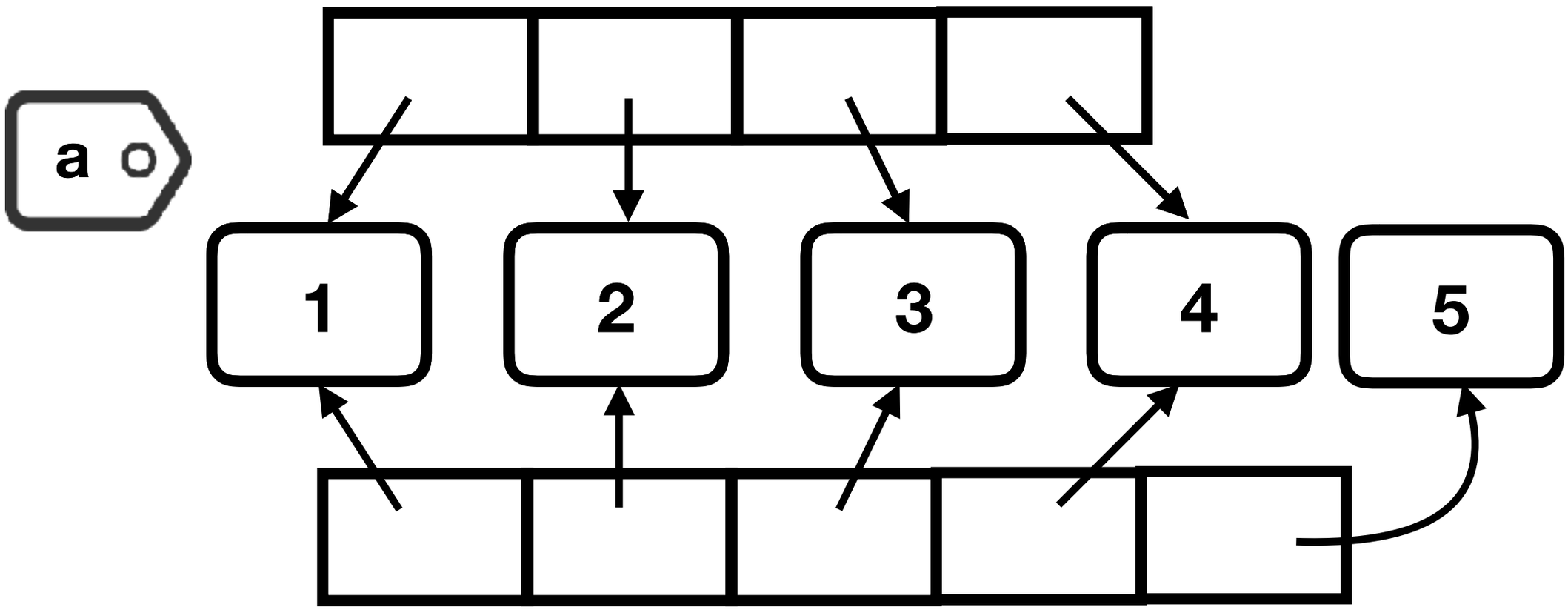
리스트의 덧셈과 재할당

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a = a + [5]   # 리스트 객체의 재바인딩(rebinding)
```



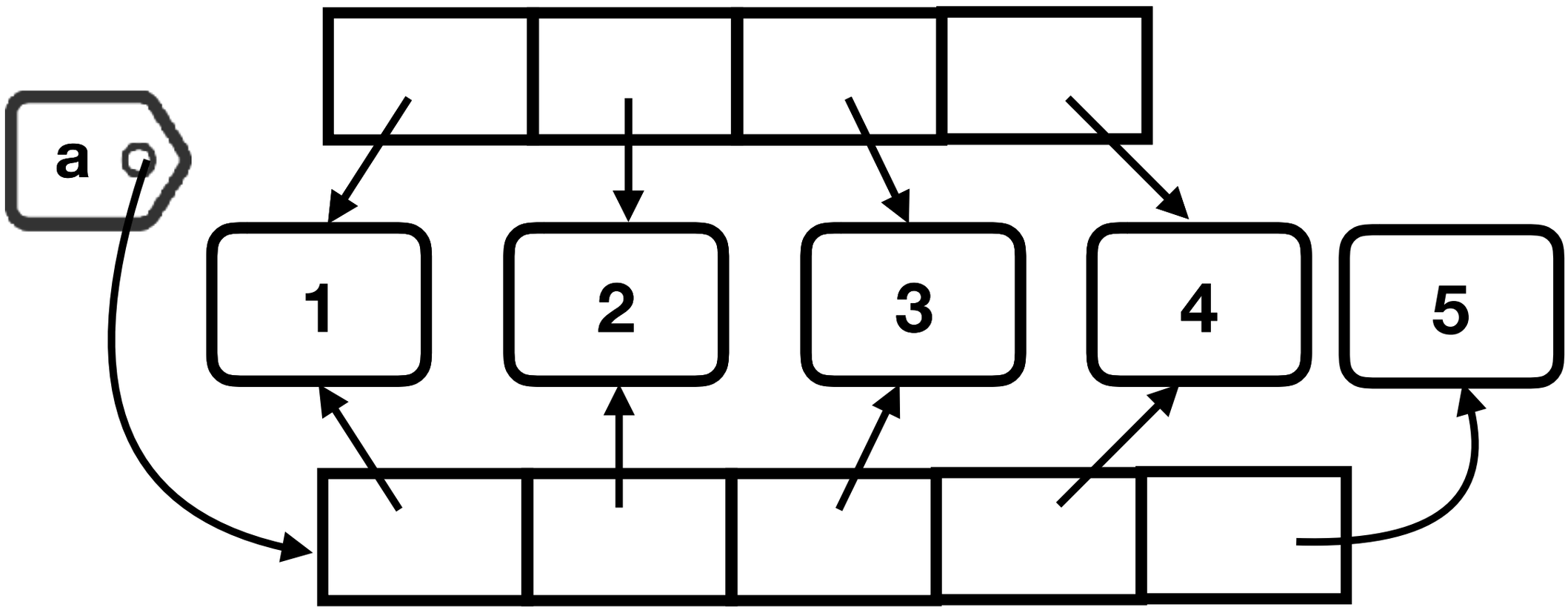
리스트의 덧셈과 재할당

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a = a + [5]   # 리스트 객체의 재바인딩(rebinding)
```



리스트의 덧셈과 재할당

```
# 리스트 객체 생성
a = [1, 2, 3, 4]
a = a + [5]   # 리스트 객체의 재바인딩(rebinding)
```



Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

append() 메소드의 수행 결과

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

append() 메소드의 수행 결과

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
 4509013104
>>> a = a + [5]
>>> id(a)
 4512940232
>>> id(a[0])
 4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

append() 메소드의 수행 결과

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

append() 메소드의 수행 결과

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

Lab

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940232
>>> a.append(5)
>>> id(a)
4512940232
```

append() 메소드의 수행 결과

```
>>> a = [1, 2, 3, 4]
>>> id(a)
4512940552
>>> id(a[0])
 4509013104
>>> a = a + [5]
>>> id(a)
4512940232
>>> id(a[0])
4509013104
```

a = a + [5] 수행결과

Lab

정리

- 리스트 객체는 변경가능(mutable) 객체
  - int 형, tuple 형, str 형 객체는 변경불가능(immutable) 객체
- 리스트의 append() 메소드는 객체의 내용을 변경시킴
- 리스트의 + 연산은 객체로 복사해서 재바인딩 함

감사합니다.