UNIT 22

리스트와 튜플 응용하기

22 리스트와 튜플 응용하기

리스트와 튜플 응용하기

- 파이썬의 리스트는 생각보다 기능이 많은데, 요소를 추가/삭제하거나, 정보를 조회하는 메서드(함수)도 제공함
- for 반복문과 결합하면 연속적이고 반복되는 값을 손쉽게 처리할 수 있음

22.1 리스트 조작하기

리스트 조작하기

- 리스트를 조작하는 메서드(method)임(메서드는 객체에 속한 함수를 뜻함)

리스트에 요소 추가하기

- append: 요소 하나를 추가
- extend: 리스트를 연결하여 확장
- insert: 특정 인덱스에 요소 추가

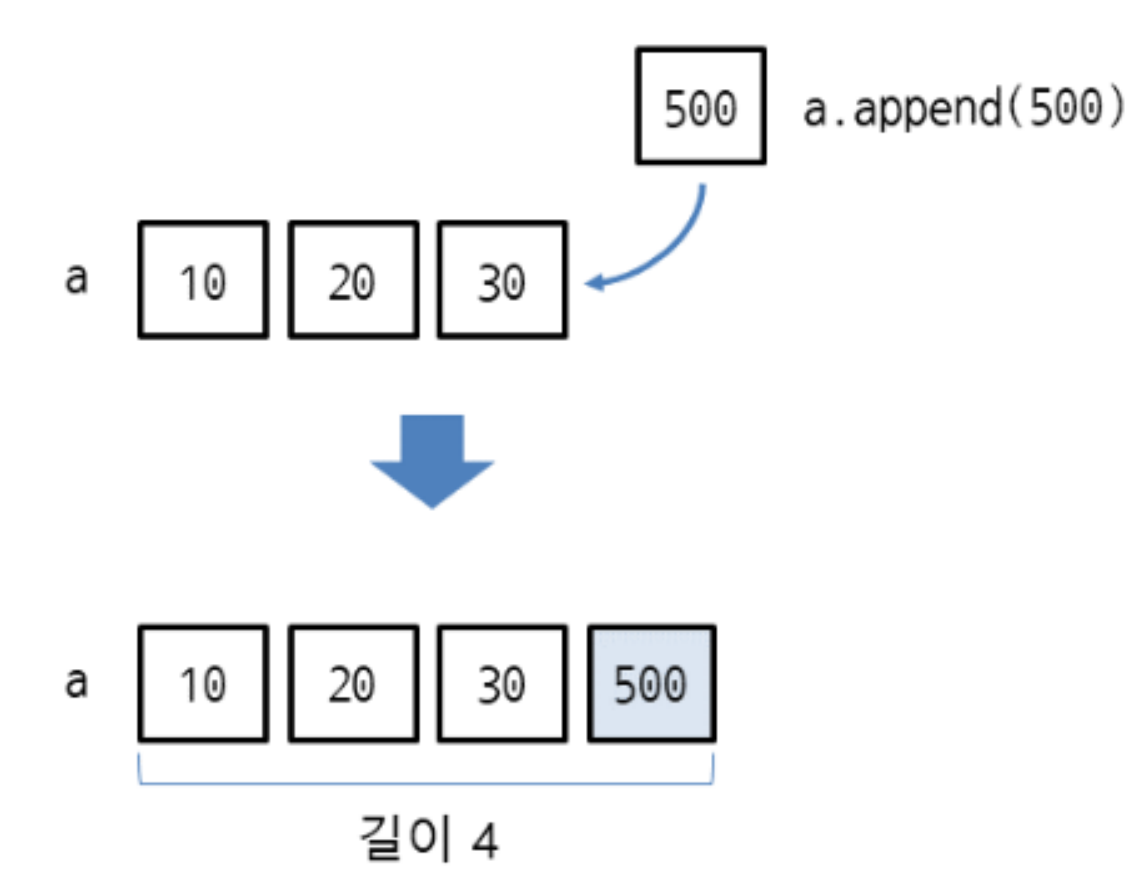
리스트에 요소 하나 추가하기

- 다음은 리스트 [10, 20, 30]에 500을 추가하여 리스트는 [10, 20, 30, 500]이 됨(메서드를 호출한 리스트가 변경되며 새 리스트는 생성되지 않음)

```
>>> a = [10, 20, 30]
>>> a.append(500)
>>> a
[10, 20, 30, 500]
>>> len(a)
4
```

22.1 리스트 조작하기

▼ 그림 append로 리스트 끝에 요소 하나 추가



22.1 리스트 조작하기

리스트에 요소 하나 추가하기

- 물론 빈 리스트에 값을 추가할 수도 있음

```
>>> a = []
>>> a.append(10)
>>> a
[10]
```

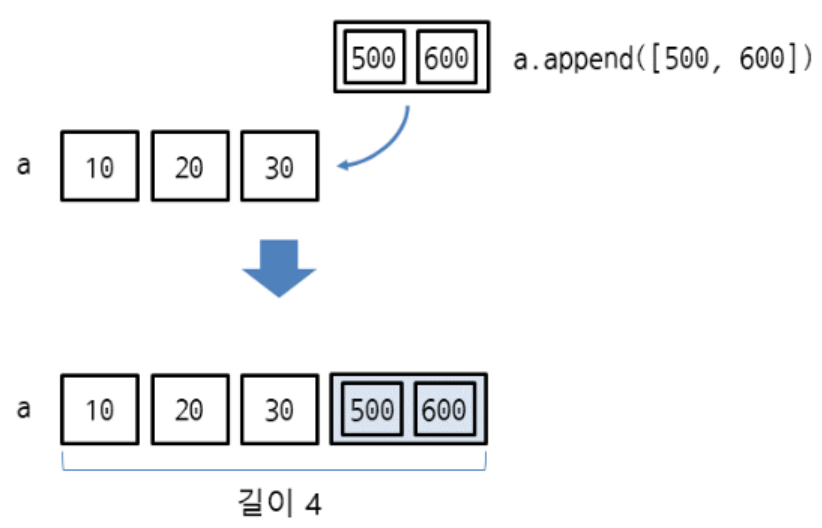
22.1 리스트 조작하기

리스트 안에 리스트 추가하기

- append는 append(리스트)처럼 리스트를 넣으면 리스트 안에 리스트가 들어감

```
>>> a = [10, 20, 30]
>>> a.append([500, 600])
>>> a
[10, 20, 30, [500, 600]]
>>> len(a)
4
```

▼ 그림 append로 리스트 안에 리스트 추가하기



22.1 리스트 조작하기

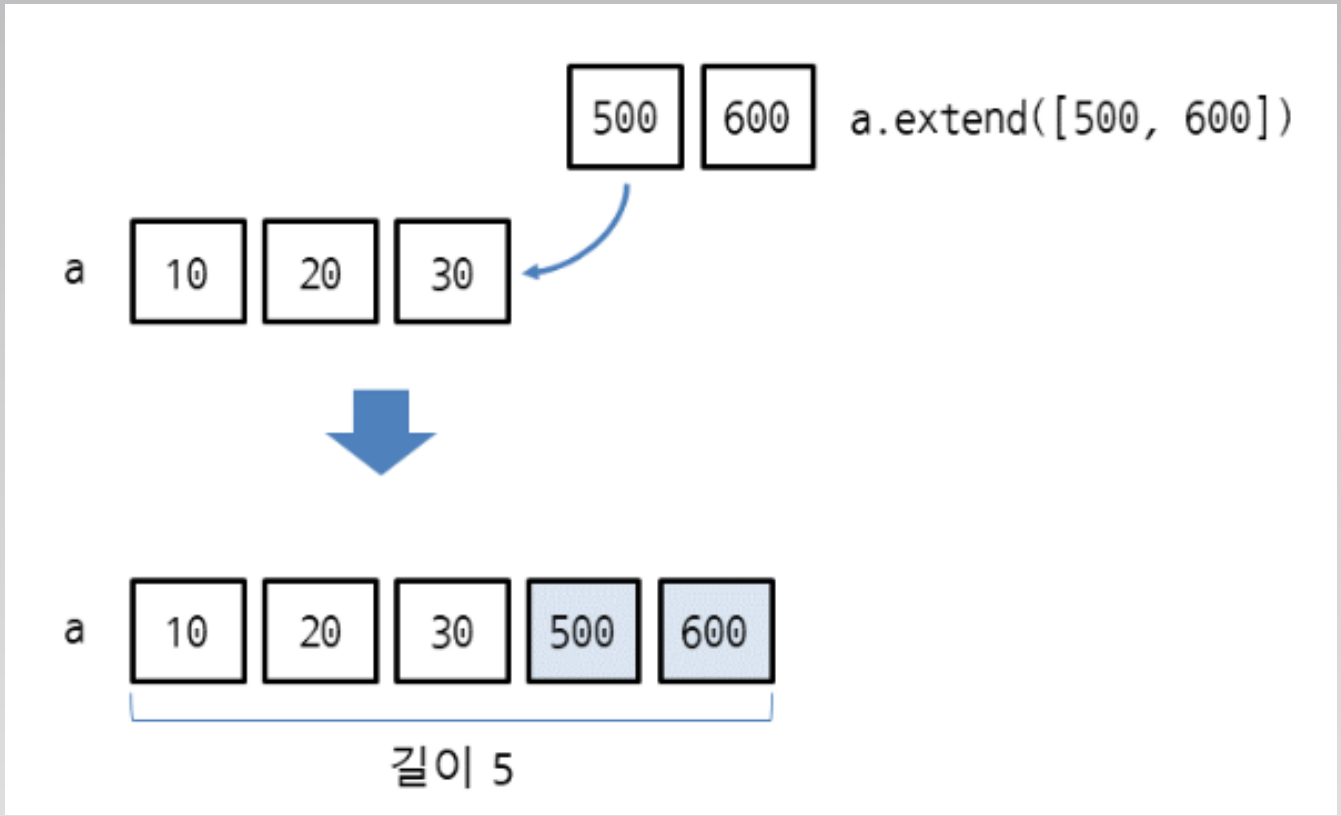
리스트 확장하기

- extend(리스트)는 리스트 끝에 다른 리스트를 연결하여 리스트를 확장함

```
>>> a = [10, 20, 30]
>>> a.extend([500, 600])
>>> a
[10, 20, 30, 500, 600]
>>> len(a)
5
```

22.1 리스트 조작하기

▼ 그림 extend로 리스트 확장하기



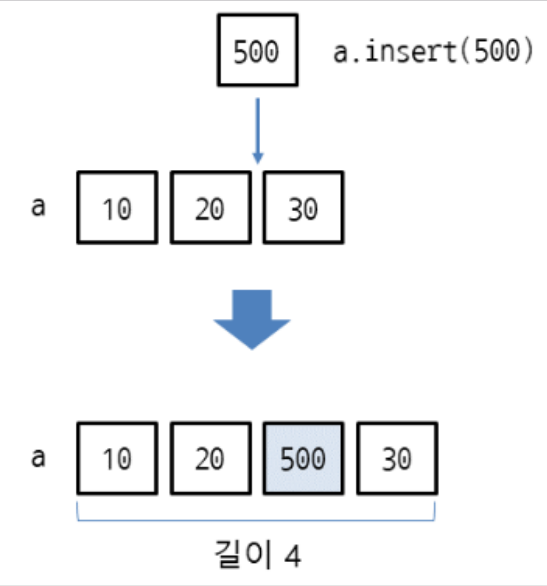
22.1 리스트 조작하기

리스트의 특정 인덱스에 요소 추가하기

- insert(인덱스, 요소)는 리스트의 특정 인덱스에 요소 하나를 추가함

```
>>> a = [10, 20, 30]
>>> a.insert(2, 500)
>>> a
[10, 20, 500, 30]
>>> len(a)
4
```

▼ 그림 insert로 특정 인덱스에 요소 추가



22.1 리스트 조작하기

» 리스트의 특정 인덱스에 요소 추가하기

- **insert(0, 요소)**: 리스트의 맨 처음에 요소를 추가
- **insert(len(리스트), 요소)**: 리스트 끝에 요소를 추가

```
>>> a = [10, 20, 30]
>>> a.insert(0, 500)
>>> a
[500, 10, 20, 30]
```

- insert에 마지막 인덱스보다 큰 값을 지정하면 리스트 끝에 요소 하나를 추가할 수 있음

```
>>> a = [10, 20, 30]
>>> a.insert(len(a), 500)
>>> a
[10, 20, 30, 500]
```

22.1 리스트 조작하기

리스트의 특정 인덱스에 요소 추가하기

- len(리스트)는 마지막 인덱스보다 1이 더 크기 때문에 리스트 끝에 값을 추가할 때 자주 활용함
- insert는 요소 하나를 추가하므로 insert에 리스트를 넣으면 append처럼 리스트 안에 리스트가 들어감

```
>>> a = [10, 20, 30]
>>> a.insert(1, [500, 600])
>>> a
[10, [500, 600], 20, 30]
```

```
>>> a = [10, 20, 30]
>>> a[1:1] = [500, 600]
>>> a
[10, 500, 600, 20, 30]
```

22.1 리스트 조작하기

리스트에서 요소 삭제하기

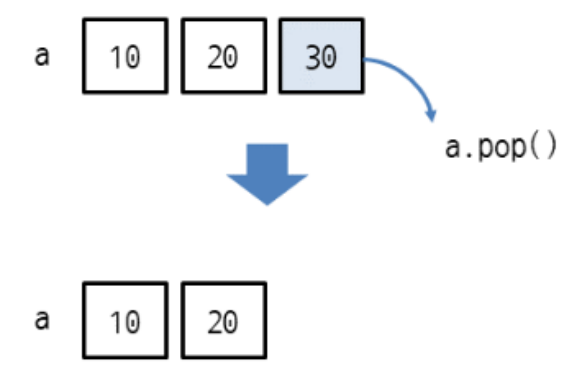
- pop: 마지막 요소 또는 특정 인덱스의 요소를 삭제
- remove: 특정 값을 찾아서 삭제

리스트에서 특정 인덱스의 요소를 삭제하기

- pop()은 리스트의 마지막 요소를 삭제한 뒤 삭제한 요소를 반환함

```
>>> a = [10, 20, 30]
>>> a.pop()
30
>>> a
[10, 20]
```

▼ 그림 pop으로 리스트의 마지막 요소 삭제



22.1 리스트 조작하기

리스트에서 특정 인덱스의 요소를 삭제하기

- pop(인덱스)는 해당 인덱스의 요소를 삭제한 뒤 삭제한 요소를 반환함

```
>>> a = [10, 20, 30]
>>> a.pop(1)
20
>>> a
[10, 30]
```

- 사실 pop 대신 del을 사용해도 상관없음

```
>>> a = [10, 20, 30]
>>> del a[1]
>>> a
[10, 30]
```

22.1 리스트 조작하기

리스트에서 특정 값을 찾아서 삭제하기

- remove(값)은 리스트에서 특정 값을 찾아서 삭제함

```
>>> a = [10, 20, 30]
>>> a.remove(20)
>>> a
[10, 30]
```

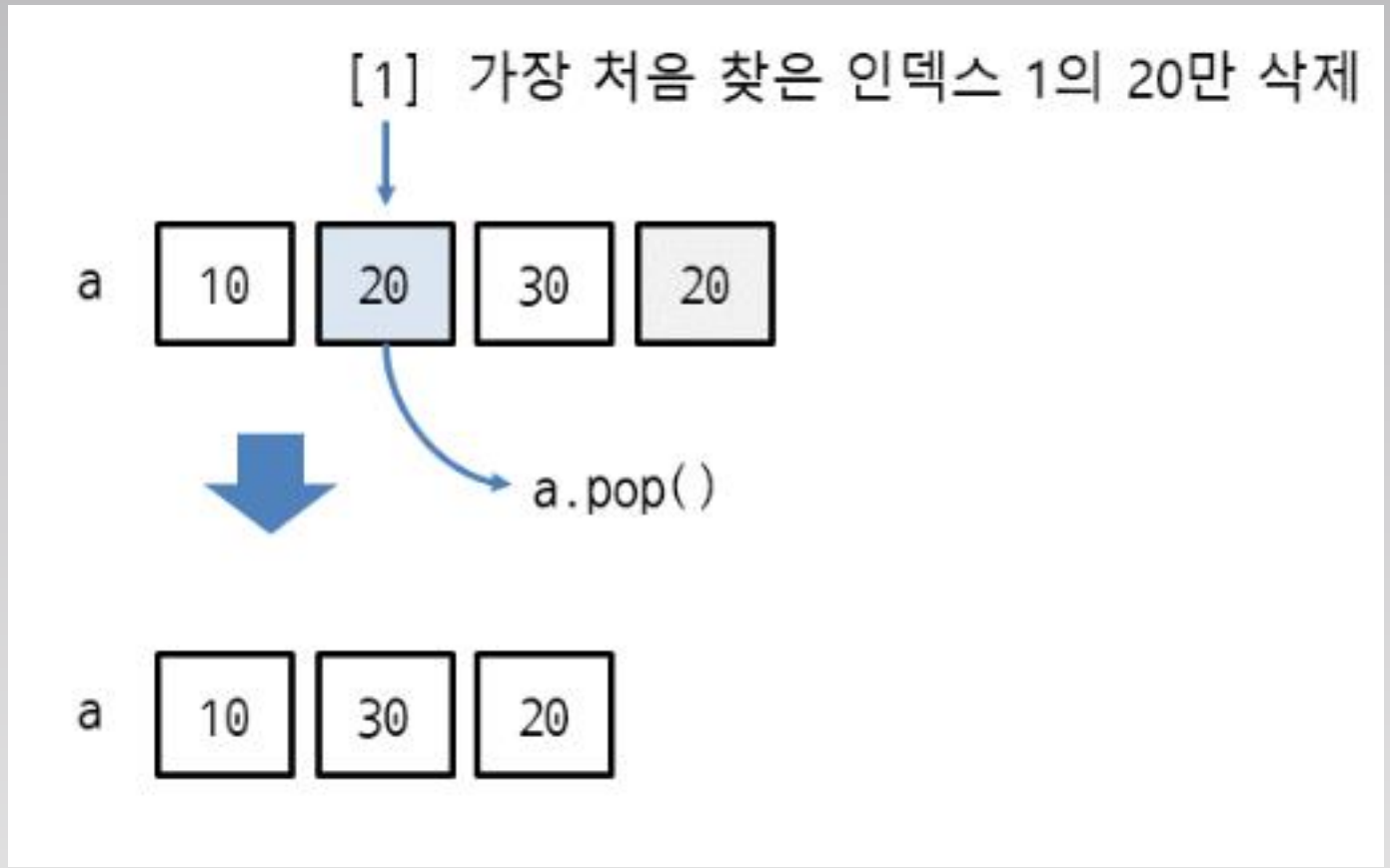
- 리스트에 같은 값이 여러 개 있을 경우 처음 찾은 값을 삭제함

```
>>> a = [10, 20, 30, 20]
>>> a.remove(20)
>>> a
[10, 30, 20]
```

- 리스트 a에 20이 2개 있지만 가장 처음 찾은 인덱스 1의 20만 삭제함

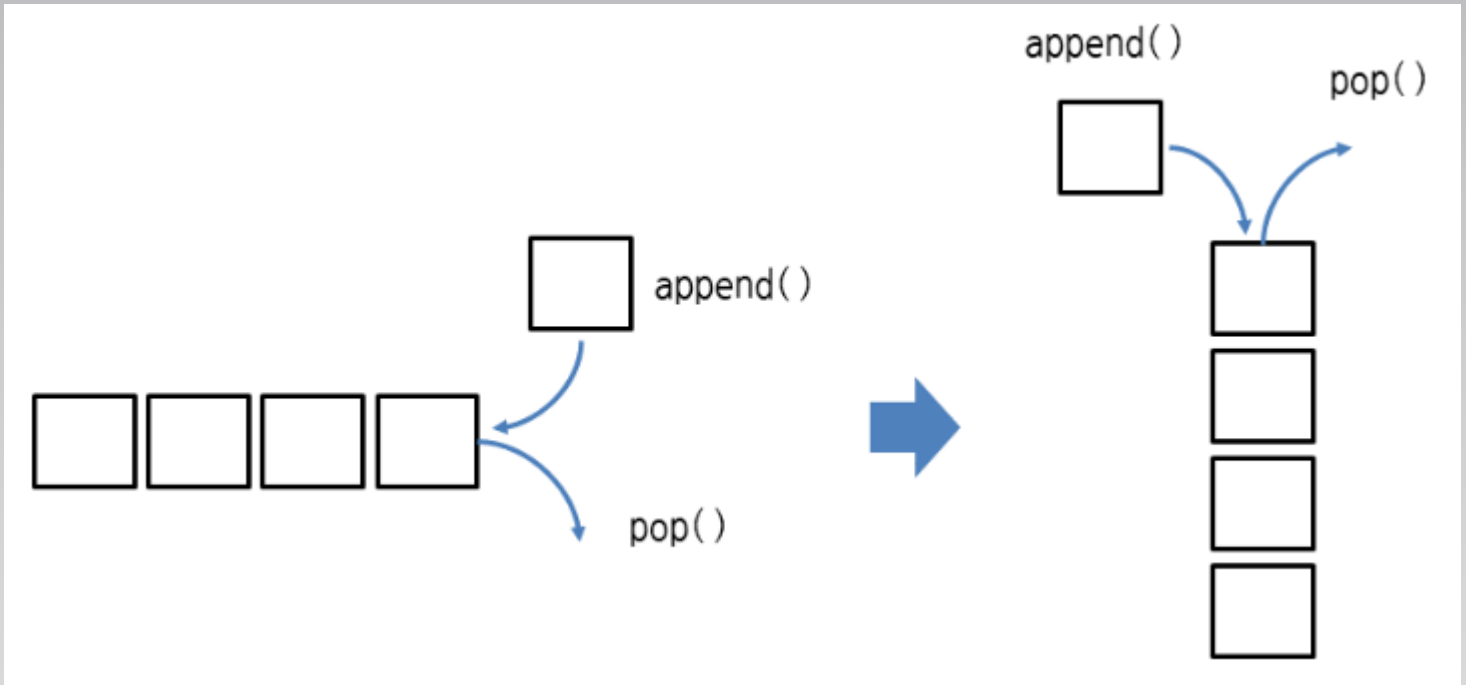
22.1 리스트 조작하기

▼ 그림 remove로 특정 값을 찾아서 삭제



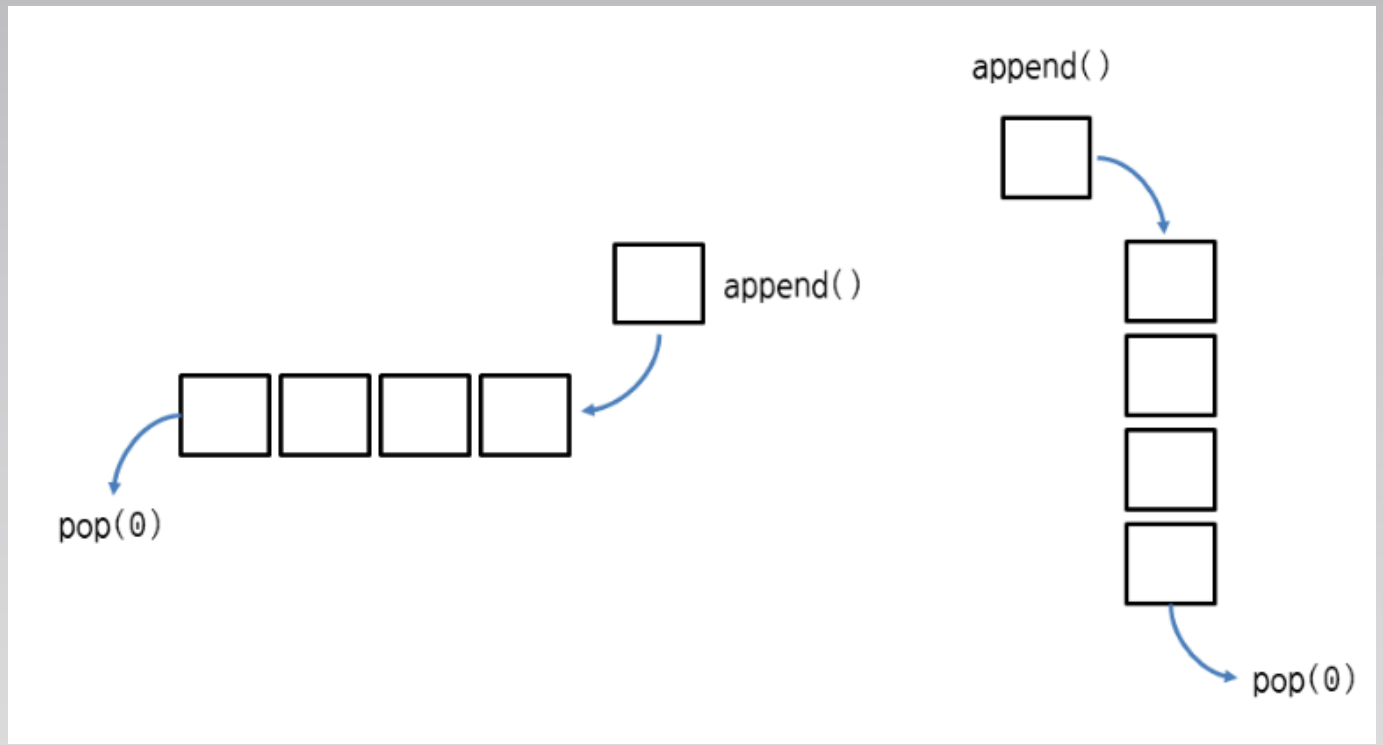
22.1 리스트 조작하기

▼ 그림 리스트로 스택 만들기



22.1 리스트 조작하기

▼ 그림 리스트로 큐 만들기



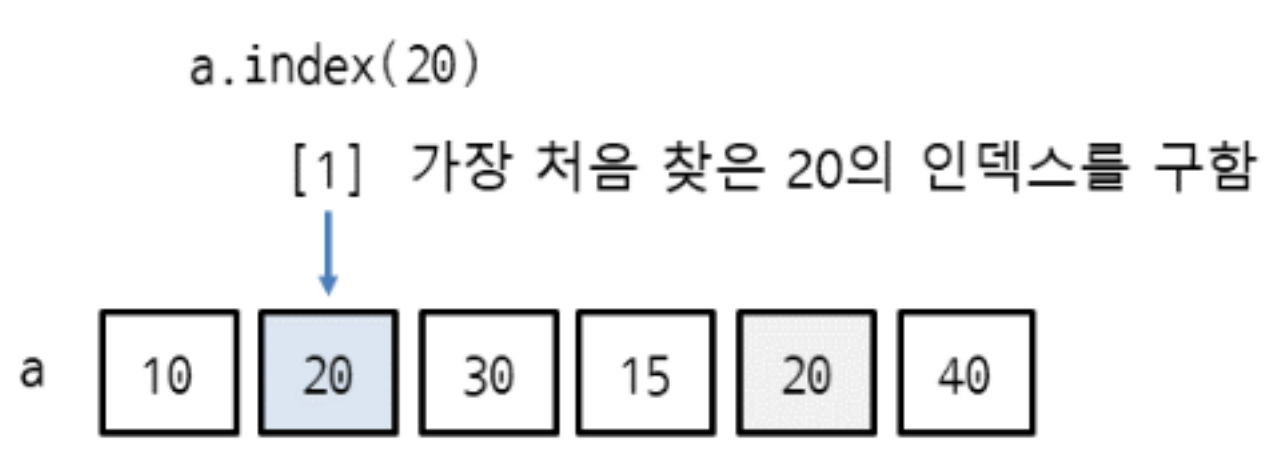
22.1 리스트 조작하기

» 리스트에서 특정 값의 인덱스 구하기

- index(값)은 리스트에서 특정 값의 인덱스를 구함
- 같은 값이 여러 개일 경우 처음 찾은 인덱스를 구함

```
>>> a = [10, 20, 30, 15, 20, 40]
>>> a.index(20)
1
```

▼ 그림 index로 특정 값의 인덱스 구하기



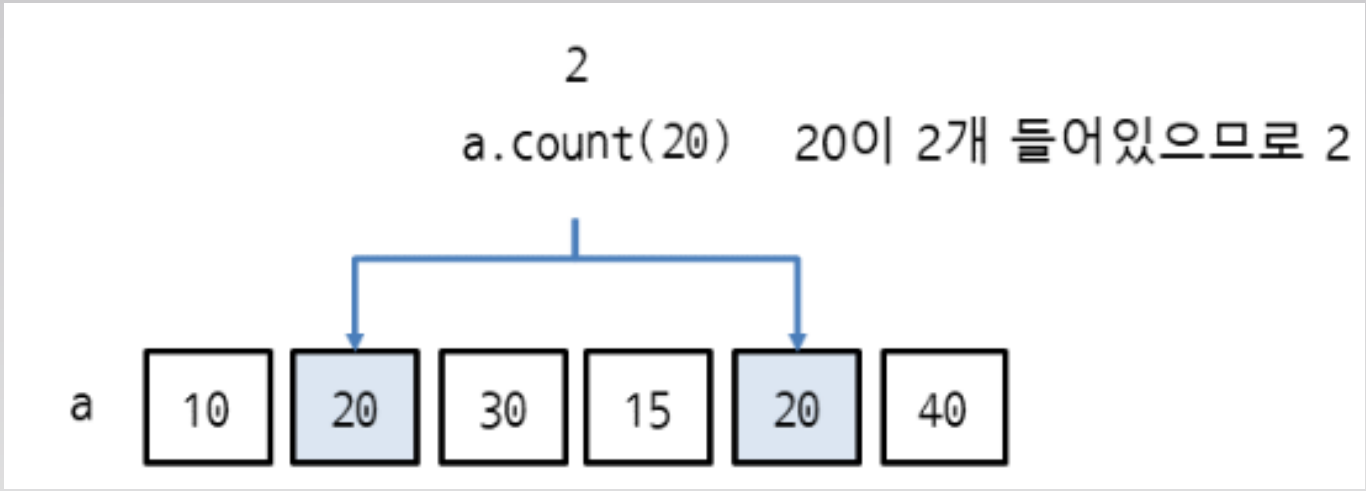
22.1 리스트 조작하기

특정 값의 개수 구하기

- count(값)은 리스트에서 특정 값의 개수를 구함

```
>>> a = [10, 20, 30, 15, 20, 40]
>>> a.count(20)
2
```

▼ 그림 count로 특정 값의 개수 구하기



22.1 리스트 조작하기

리스트의 순서를 뒤집기

- reverse()는 리스트에서 요소의 순서를 반대로 뒤집음

```
>>> a = [10, 20, 30, 15, 20, 40]
>>> a.reverse()
>>> a
[40, 20, 15, 30, 20, 10]
```

22.1 리스트 조작하기

리스트의 요소를 정렬하기

- sort()는 리스트의 요소을 작은 순서대로 정렬함(오름차순)

- **sort() 또는 sort(reverse=False): 리스트의 값을 작은 순서대로 정렬(오름차순)**
- **sort(reverse=True)**: 리스트의 값을 큰 순서대로 정렬(내림차순)

```
>>> a = [10, 20, 30, 15, 20, 40]
>>> a.sort()
>>> a
[10, 15, 20, 20, 30, 40]
```

22.1 리스트 조작하기

리스트의 모든 요소를 삭제하기

- clear()는 리스트의 모든 요소를 삭제함

```
>>> a = [10, 20, 30]
>>> a.clear()
>>> a
[]
```

- clear 대신 del a[:]와 같이 시작, 끝 인덱스를 생략하여 리스트의 모든 요소를 삭제할 수도 있음

```
>>> a = [10, 20, 30]
>>> del a[:]
>>> a
[]
```

22.1 리스트 조작하기

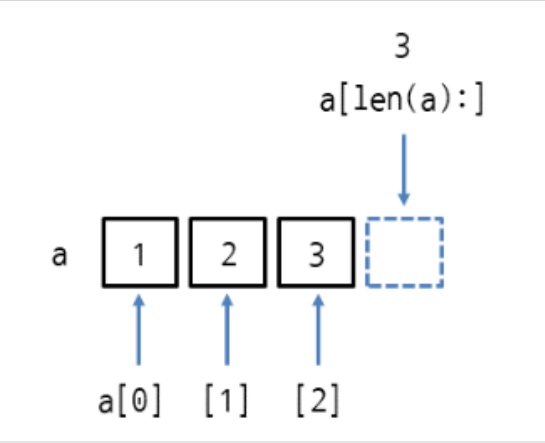
리스트를 슬라이스로 조작하기

- 다음은 리스트 끝에 값이 한 개 들어있는 리스트를 추가함

```
>>> a = [10, 20, 30]
>>> a[len(a):] = [500]
>>> a
[10, 20, 30, 500]
```

- a[len(a):]는 시작 인덱스를 len(a)로 지정해서 리스트의 마지막 인덱스보다 1이 더 큰 상태임
- 그림과 같이 리스트 끝에서부터 시작하겠다는 뜻임(이때는 리스트의 범위를 벗어난 인덱스를 사용할 수 있음)

▼ 그림 a[len(a):]의 뜻



22.1 리스트 조작하기

리스트를 슬라이스로 조작하기

- a[len(a):] = [500]과 같이 값이 한 개 들어있는 리스트를 할당하면 리스트 a 끝에 값을 한 개 추가하며 a.append(500)과 같음
- a[len(a):] = [500, 600]과 같이 요소가 여러 개 들어있는 리스트를 할당하면 리스트 a 끝에 다른 리스트를 연결하며 a.extend([500, 600])과 같음

```
>>> a = [10, 20, 30]
>>> a[len(a):] = [500, 600]
>>> a
[10, 20, 30, 500, 600]
```

22.2 리스트의 할당과 복사 알아보기

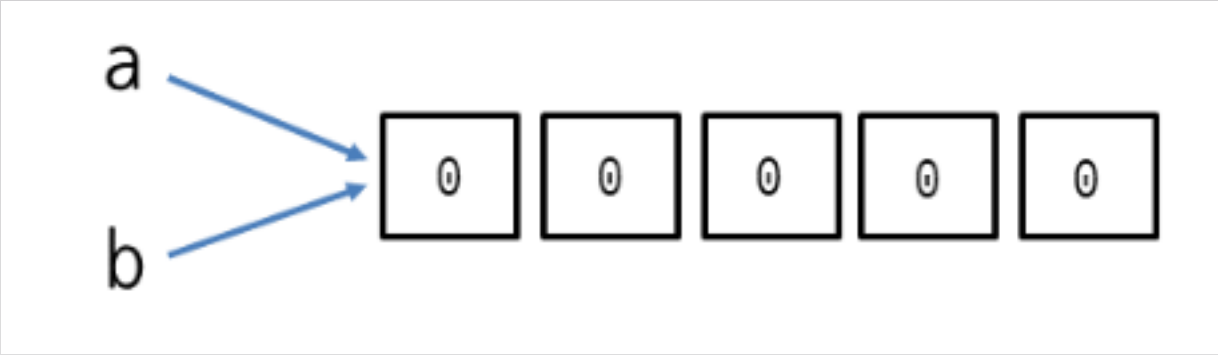
» 리스트의 할당과 복사 알아보기

- 먼저 다음과 같이 리스트를 만든 뒤 다른 변수에 할당함

```
>>> a = [0, 0, 0, 0, 0]
>>> b = a
```

- b = a와 같이 리스트를 다른 변수에 할당하면 리스트는 두 개가 될 것 같지만 실제로는 리스트가 한 개임

▼ 그림 리스트의 할당 b = a



22.2 리스트의 할당과 복사 알아보기

» 리스트의 할당과 복사 알아보기

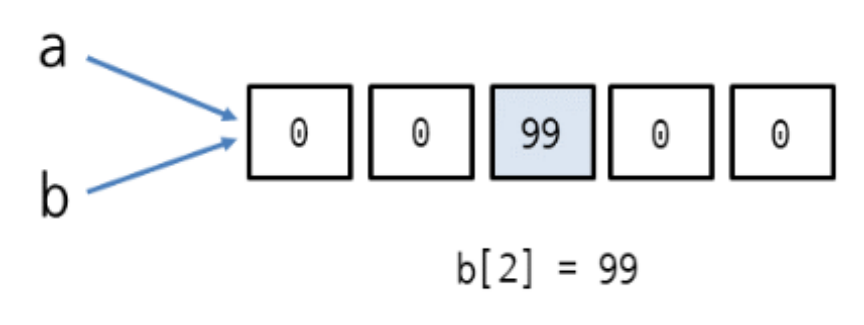
- 변수 이름만 다를 뿐 리스트 a와 b는 같은 객체임

```
>>> a is b
True
```

- a와 b는 같으므로 b[2] = 99와 같이 리스트 b의 요소를 변경하면 리스트 a와 b에 모두 반영됨

```
>>> b[2] = 99
>>> a
[0, 0, 99, 0, 0]
>>> b
[0, 0, 99, 0, 0]
```

▼ 그림 리스트를 할당한 뒤 b의 요소를 변경했을 때



22.2 리스트의 할당과 복사 알아보기

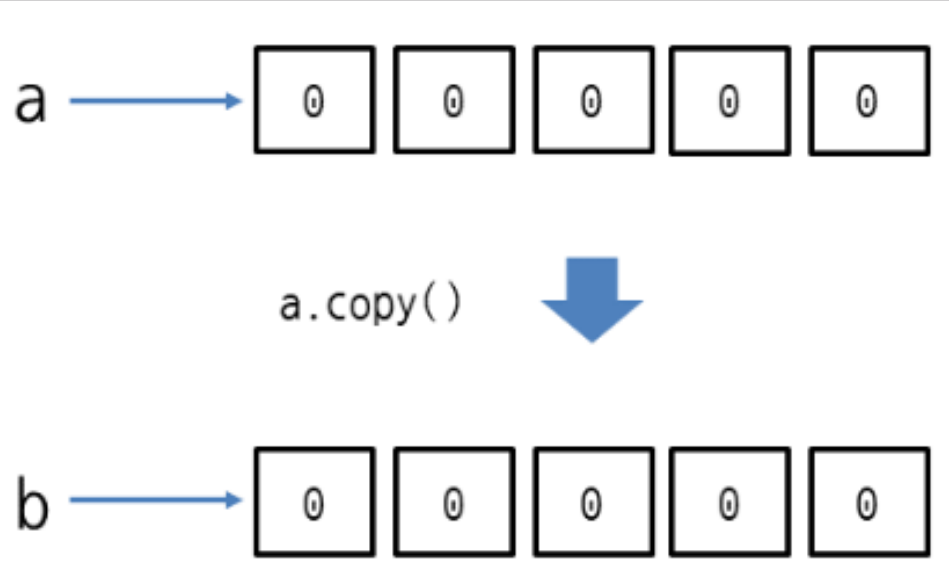
» 리스트의 할당과 복사 알아보기

- 리스트 a와 b를 완전히 두 개로 만들려면 copy 메서드로 모든 요소를 복사해야 함

```
>>> a = [0, 0, 0, 0, 0]
>>> b = a.copy()
```

- b = a.copy()와 같이 copy를 사용한 뒤 b에 할당해주면 리스트 a의 요소가 모두 b에 복사됨

▼ 그림 리스트를 복사했을 때



22.2 리스트의 할당과 복사 알아보기

리스트의 할당과 복사 알아보기

- a와 b를 is 연산자로 비교해보면 False가 나온 두 리스트는 다른 객체임
- 복사된 요소는 모두 같으므로 ==로 비교하면 True가 나옴

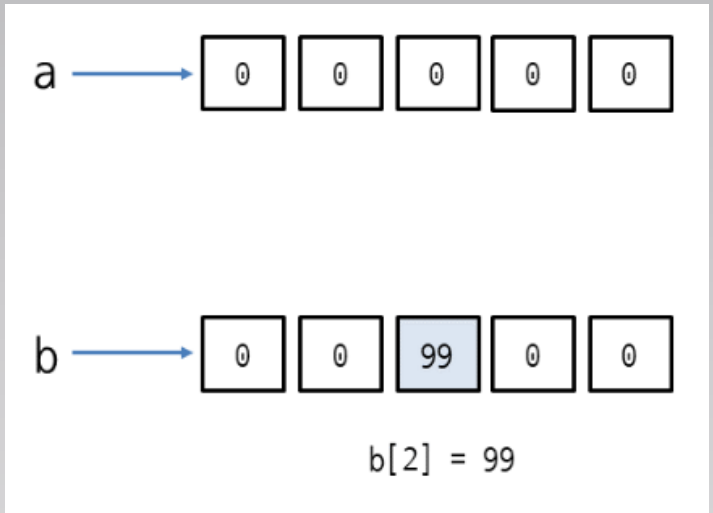
```
>>> a is b
False
>>> a == b
True
```

- 리스트 a와 b는 별개이므로 한쪽의 값을 변경해도 다른 리스트에 영향을 미치지 않음

```
>>> b[2] = 99
>>> a
[0, 0, 0, 0, 0]
>>> b
[0, 0, 99, 0, 0]
```

22.2 리스트의 할당과 복사 알아보기

▼ 그림 리스트를 복사한 뒤 b의 요소를 변경했을 때



22.3 반복문으로 리스트의 요소를 모두 출력하기

for 반복문으로 요소 출력하기

```
for 변수 in 리스트:
    반복할 코드
```

```
>>> a = [38, 21, 53, 62, 19]
>>> for i in a:
...     print(i)
...
38
21
53
62
19
```

- print로 i를 출력하면 모든 요소를 순서대로 출력할 수 있음
- 물론 in 다음에 리스트를 직접 지정해도 상관 없음

```
for i in [38, 21, 53, 62, 19]:
    print(i)
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

인덱스와 요소를 함께 출력하기

- for 인덱스, 요소 in enumerate(리스트):

```
>>> a = [38, 21, 53, 62, 19]
>>> for index, value in enumerate(a):
...     print(index, value)
...
0 38
1 21
2 53
3 62
4 19
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

인덱스와 요소를 함께 출력하기

- 앞의 코드는 인덱스를 0부터 출력하는데 1부터 출력하고 싶을 수도 있음

```
>>> for index, value in enumerate(a):
...     print(index + 1, value)
...
1 38
2 21
3 53
4 62
5 19
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

인덱스와 요소를 함께 출력하기

- 파이썬 다운 방법

• for 인덱스, 요소 in enumerate(리스트, start=숫자):

```
>>> for index, value in enumerate(a, start=1):
...     print(index, value)
...
1 38
2 21
3 53
4 62
5 19
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

while반복문으로 요소 출력하기

```
>>> a = [38, 21, 53, 62, 19]
>>> i = 0
>>> while i < len(a):
...     print(a[i])
...     i += 1
...
38
21
53
62
19
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

while반복문으로 요소 출력하기

- i <= len(a)처럼 <=을 사용하면 리스트의 범위를 벗어나게 되므로 주의해야 함

```
>>> a = [38, 21, 53, 62, 19]
>>> i = 0
>>> while i <= len(a):
...     print(a[i])
...     i += 1
...
38
21
53
62
19
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
IndexError: list index out of range
```

22.3 반복문으로 리스트의 요소를 모두 출력하기

while반복문으로 요소 출력하기

```
while i < len(a):
    print(a[i])
    i += 1
```

22.4 리스트의 가장 작은 수, 가장 큰 수, 합계 구하기

가장 작은 수와 가장 큰 수 구하기

- 요소를 모두 반복하면서 숫자를 찾아내 보자

```
>>> a = [38, 21, 53, 62, 19]
>>> smallest = a[0]
>>> for i in a:
...     if i < smallest:
...         smallest = i
...
>>> smallest
19
```

- 숫자를 계속 비교해서 숫자가 작으면 smallest를 바꾸는 방식임

22.4 리스트의 가장 작은 수, 가장 큰 수, 합계 구하기

가장 작은 수와 가장 큰 수 구하기

- 가장 큰 수는 부등호를 반대로 만들면 됨

```
>>> a = [38, 21, 53, 62, 19]
>>> largest = a[0]
>>> for i in a:
...     if i > largest:
...         largest = i
...
>>> largest
62
```

- 리스트의 숫자를 계속 비교해서 숫자가 크면 largest를 바꾸는 방식임

22.4 리스트의 가장 작은 수, 가장 큰 수, 합계 구하기

가장 작은 수와 가장 큰 수 구하기

- 리스트를 작은 순서대로 정렬(오름차순)하면 첫 번째 요소가 가장 작은 수임
- 반대로 큰 순서대로 정렬(내림차순)하면 첫 번째 요소가 가장 큰 수가 됨

```
>>> a = [38, 21, 53, 62, 19]
>>> a.sort()
>>> a[0]
19
>>> a.sort(reverse=True)
>>> a[0]
62
```

22.4 리스트의 가장 작은 수, 가장 큰 수, 합계 구하기

가장 작은 수와 가장 큰 수 구하기

- 간단한 방법은 파이썬에서 제공하는 min, max 함수를 사용하면 됨

```
>>> a = [38, 21, 53, 62, 19]
>>> min(a)
19
>>> max(a)
62
```

- min은 리스트에서 가장 작은 값을 구하고, max는 가장 큰 값을 구함

22.4 리스트의 가장 작은 수, 가장 큰 수, 합계 구하기

요소의 합계 구하기

- 합계를 구할 때도 반복문을 사용할 수 있음

```
>>> a = [10, 10, 10, 10, 10]
>>> x = 0
>>> for i in a:
...     x += i
...
>>> x
50
```

- 파이썬에서는 합계를 구하는 sum 함수를 제공함

```
>>> a = [10, 10, 10, 10, 10]
>>> sum(a)
50
```

- min, max, sum에는 리스트뿐만 아니라 모든 반복 가능한 객체(iterable)를 넣을 수 있음

22.5 리스트 표현식 사용하기

» 리스트 표현식 사용하기

- 파이썬의 리스트가 특이한 점은 리스트 안에 for 반복문과 if 조건문을 사용할 수 있다는 점
- 리스트 안에 식, for 반복문, if 조건문 등을 지정하여 리스트를 생성하는 것을 리스트 컴프리헨션(list comprehension)이라고 함
- 책이나 인터넷에서도 리스트 내포, 리스트 내장, 리스트 축약, 리스트 해석 등으로 씀
- 컴프리헨션은 능력, 이해력, 시험 등의 뜻도 있지만, 어떤 것을 잡아서 담아둔다는 뜻이 있음
- 식으로 지정해서 생성된 것을 리스트로 잡아두는 것이 리스트 컴프리헨션임

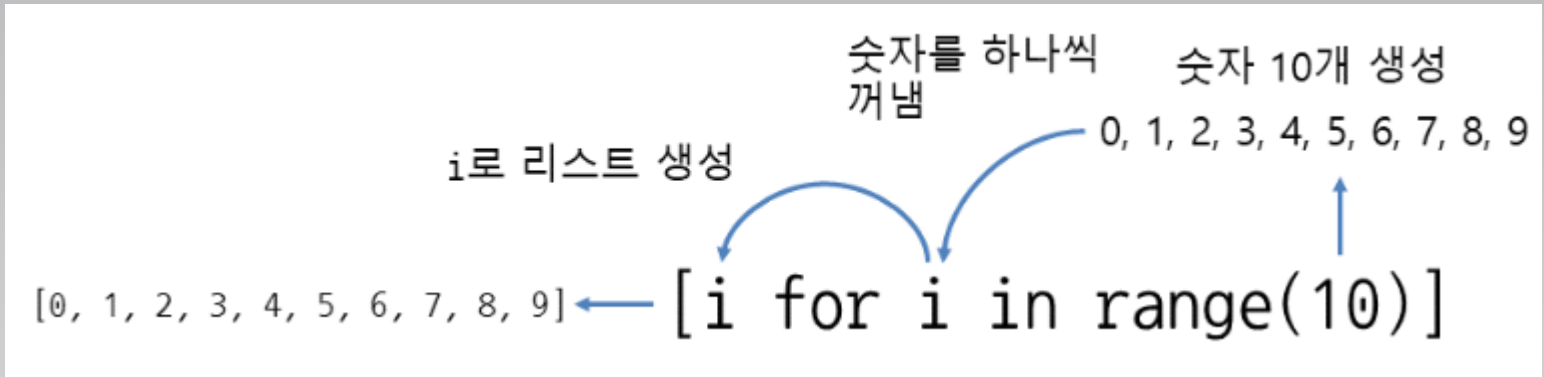
- `[식 for 변수 in 리스트]`
- `list(식 for 변수 in 리스트)`

```
>>> a = [i for i in range(10)]        # 0부터 9까지 숫자를 생성하여 리스트 생성
>>> a
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> b = list(i for i in range(10))    # 0부터 9까지 숫자를 생성하여 리스트 생성
>>> b
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- range(10)으로 0부터 9까지 생성하여 변수 i에 숫자를 꺼내고, 최종적으로 i를 이용하여 리스트를 만든다는 뜻

22.5 리스트 표현식 사용하기

▼ 그림 리스트 표현식의 동작 순서



22.5 리스트 표현식 사용하기

리스트 표현식 사용하기

- [i for i in range(10)]는 변수 i를 그대로 사용하지만, 다음과 같이 식 부분에서 i를 다른 값과 연산하면 각 연산의 결과를 리스트로 생성함

```
>>> c = [i + 5 for i in range(10)]    # 0부터 9까지 숫자를 생성하면서 값에 5를 더하여 리스트 생성
>>> c
[5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
>>> d = [i * 2 for i in range(10)]    # 0부터 9까지 숫자를 생성하면서 값에 2를 곱하여 리스트 생성
>>> d
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

22.5 리스트 표현식 사용하기

리스트 표현식에서 if 조건문 사용하기

- 다음과 같이 if 조건문은 for 반복문 뒤에 지정함

- [식 for 변수 in 리스트 if 조건식]
- list(식 for 변수 in 리스트 if 조건식)

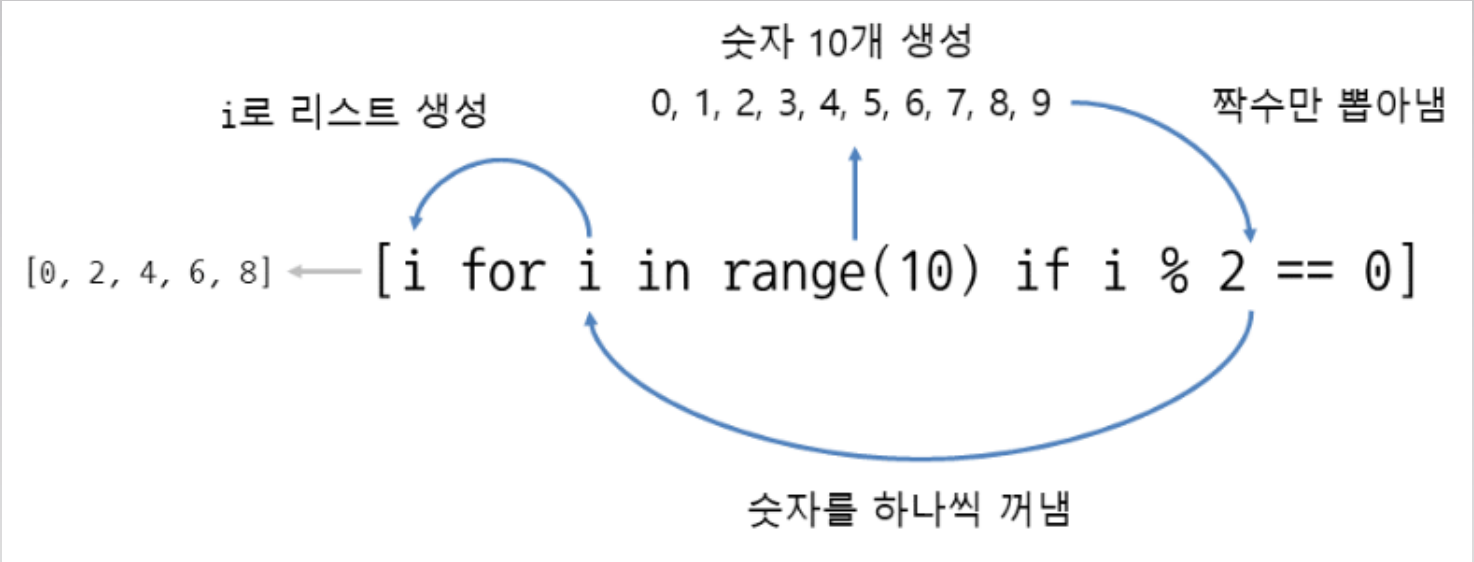
```
>>> a = [i for i in range(10) if i % 2 == 0]    # 0~9 숫자 중 2의 배수인 숫자(짝수)로 리스트 생성
>>> a
[0, 2, 4, 6, 8]
```

22.5 리스트 표현식 사용하기

» 리스트 표현식에서 if 조건문 사용하기

- 다음과 같이 for 반복문 뒤에 if 조건문을 지정하면 숫자를 생성한 뒤 if 조건문에서 특정 숫자만 뽑아내서 리스트를 생성함

▼ 그림 리스트 표현식에서 if 조건문 사용하기



22.5 리스트 표현식 사용하기

리스트 표현식에서 if 조건문 사용하기

- 다음과 같이 i를 다른 값과 연산해서 리스트를 만들어도 됨

```
>>> b = [i + 5 for i in range(10) if i % 2 == 1]    # 0~9 숫자 중 홀수에 5를 더하여 리스트 생성
>>> b
[6, 8, 10, 12, 14]
```

22.5 리스트 표현식 사용하기

for 반복문과 if 조건문을 여러 번 사용하기

```
[식 for 변수1 in 리스트1 if 조건식1      for 변수2 in 리스트2 if 조건식2      ...      for 변수n in 리스트n if 조건식
n]

list(식 for 변수1 in 리스트1 if 조건식1          for 변수2 in 리스트2 if 조건식2          ...          for 변수n in 리
스트n if 조건식n)
```

- 다음은 2단부터 9단까지 구구단을 리스트 생성함

```
>>> a = [i * j for j in range(2, 10) for i in range(1, 10)]
>>> a
[2, 4, 6, 8, 10, 12, 14, 16, 18, 3, 6, 9, 12, 15, 18, 21, 24, 27, 4, 8, 12, 16, 20, 24, 28, 32, 36, 5, 10, 15,
20, 25, 30, 35, 40, 45, 6, 12, 18, 24, 30, 36, 42, 48, 54, 7, 14, 21, 28, 35, 42, 49, 56, 63, 8, 16, 24, 32, 4
0, 48, 56, 64, 72, 9, 18, 27, 36, 45, 54, 63, 72, 81]
```

22.5 리스트 표현식 사용하기

for 반복문과 if 조건문을 여러 번 사용하기

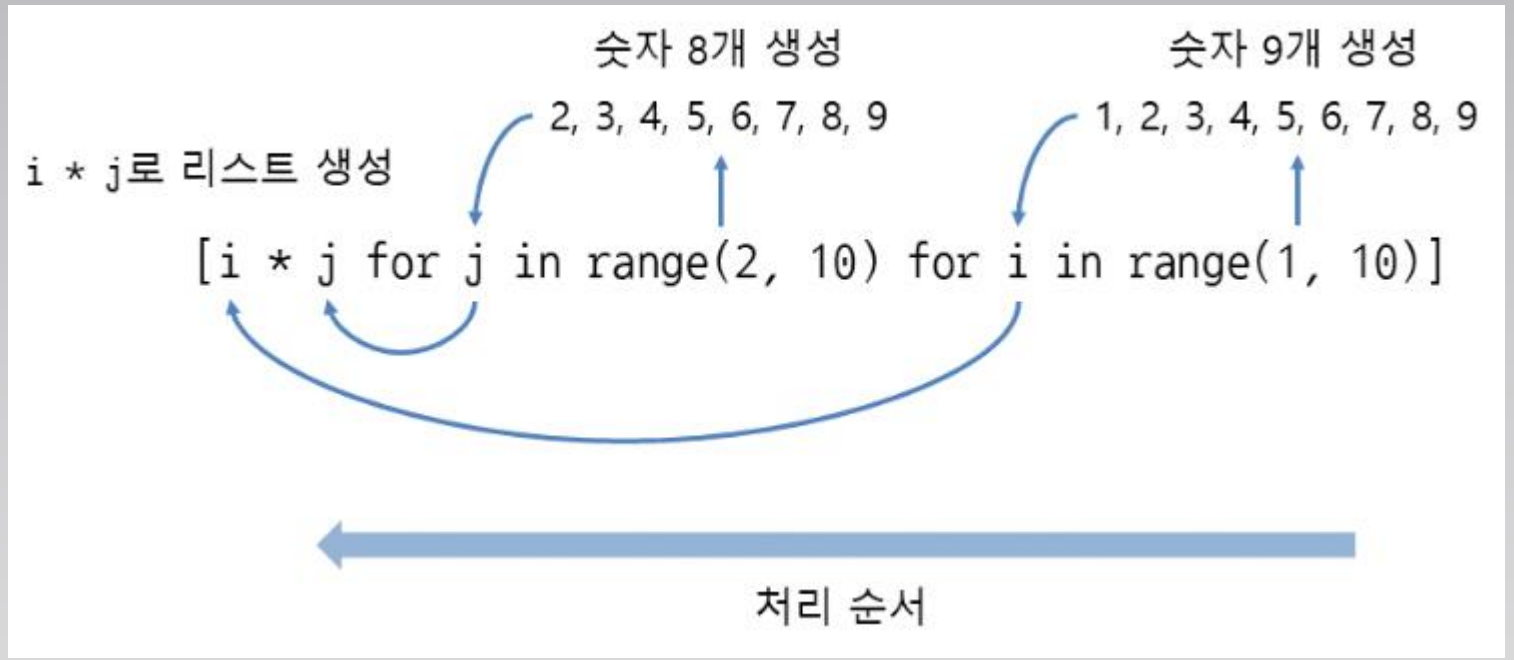
- 코드를 한 줄로 입력했지만 다음과 같이 여러 줄로 입력해도 됨
- 들여쓰기는 해도 되고 하지 않아도 됨
- 가독성을 위해 들여쓰기를 해주는 것이 좋음

```
a = [i * j for j in range(2, 10)
           for i in range(1, 10)]
```

- 리스트 표현식에 for가 여러 개일 때 처리 순서는 뒤에서 앞으로 순임

22.5 리스트 표현식 사용하기

▼ 그림 리스트 표현식에 for가 여러 개일 때 처리 순서



22.6 리스트에 map 사용하기

리스트에 map 사용하기

- map은 리스트의 요소를 지정된 함수로 처리해주는 함수임(map은 원본 리스트를 변경하지 않고 새 리스트를 생성)

  - `list(map(함수, 리스트))`
  - `tuple(map(함수, 튜플))`

- 먼저 for 반복문을 사용해서 변환해보자

```
>>> a = [1.2, 2.5, 3.7, 4.6]
>>> for i in range(len(a)):
...     a[i] = int(a[i])
...
>>> a
[1, 2, 3, 4]
```

22.6 리스트에 map 사용하기

리스트에 map 사용하기

- 매번 for 반복문으로 반복하면서 요소를 변환하려니 조금 번거로우니 map을 사용하면 편리함

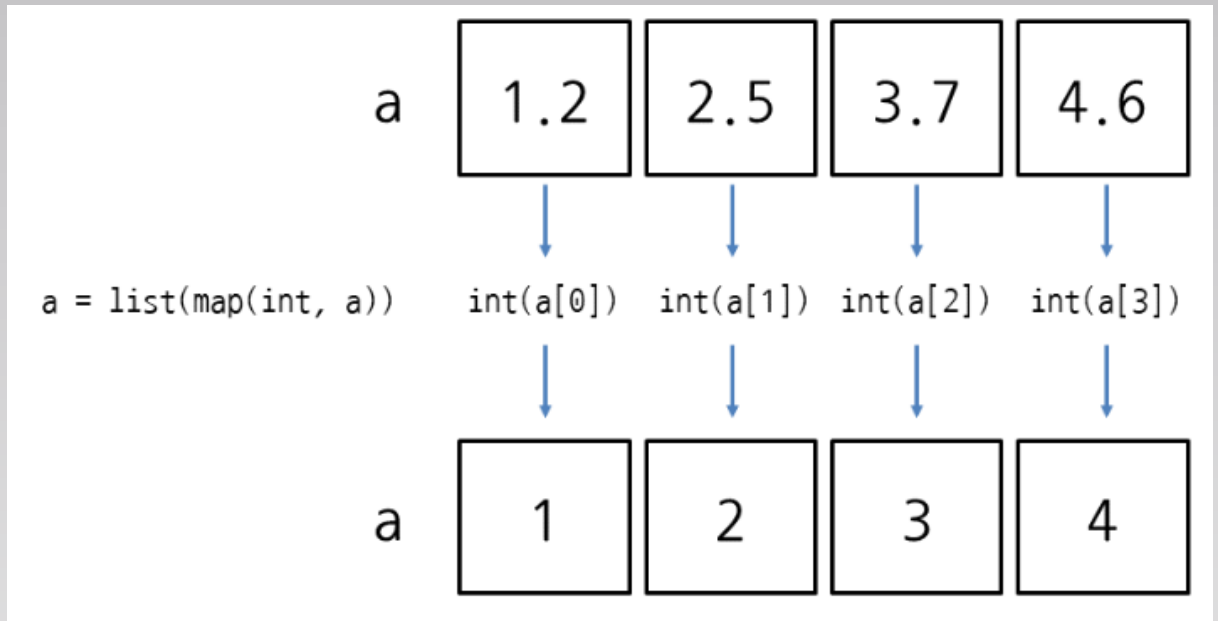
```
>>> a = [1.2, 2.5, 3.7, 4.6]
>>> a = list(map(int, a))
>>> a
[1, 2, 3, 4]
```

22.6 리스트에 map 사용하기

리스트에 map 사용하기

- list를 사용해서 map의 결과를 다시 리스트로 만들어주자

▼ 그림 리스트에 map 함수 사용



22.6 리스트에 map 사용하기

» 리스트에 map 사용하기

- range를 사용해서 숫자를 만든 뒤 숫자를 문자열로 변환해보자

```
>>> a = list(map(str, range(10)))
>>> a
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

- 리스트를 출력해보면 각 요소가 ' '(작은따옴표)로 묶인 것을 볼 수 있음

22.6 리스트에 map 사용하기

input().split()과 map

- 다음과 같이 input().split()을 사용한 뒤에 변수 한 개에 저장해보면 리스트인지 확인할 수 있음

```
>>> a = input().split()
10 20 (입력)
>>> a
['10', '20']
```

- 이제 map을 사용해서 정수로 변환해보자

```
>>> a = map(int, input().split())
10 20 (입력)
>>> a
<map object at 0x03DFB0D0>
>>> list(a)
[10, 20]
```

22.6 리스트에 map 사용하기

input().split()과 map

- 이 리스트 [10, 20]을 변수 두 개에 저장하면 지금까지 사용한 a, b = map(int, input().split())와 같은 동작이 됨

```
>>> a, b = [10, 20]
>>> a
10
>>> b
20
```

- 사실 map이 반환하는 맵 객체는 이터레이터라서 변수 여러 개에 저장하는 언패킹(unpacking)이 가능함
- a, b = map(int, input().split())처럼 list를 생략한 것임

22.6 리스트에 map 사용하기

input().split()과 map

- a, b = map(int, input().split())을 풀어서 쓰면 다음과 같은 코드가 됨

```
x = input().split()     # input().split()의 결과는 문자열 리스트
m = map(int, x)         # 리스트의 요소를 int로 변환, 결과는 맵 객체
a, b = m                # 맵 객체는 변수 여러 개에 저장할 수 있음
```

22.7 튜플 응용하기

튜플 응용하기

- 튜플은 리스트와는 달리 내용을 변경할 수 없음(불변, immutable)
- 내용을 변경하는 append 같은 메서드는 사용할 수 없고, 요소의 정보를 구하는 메서드만 사용할 수 있음

22.7 튜플 응용하기

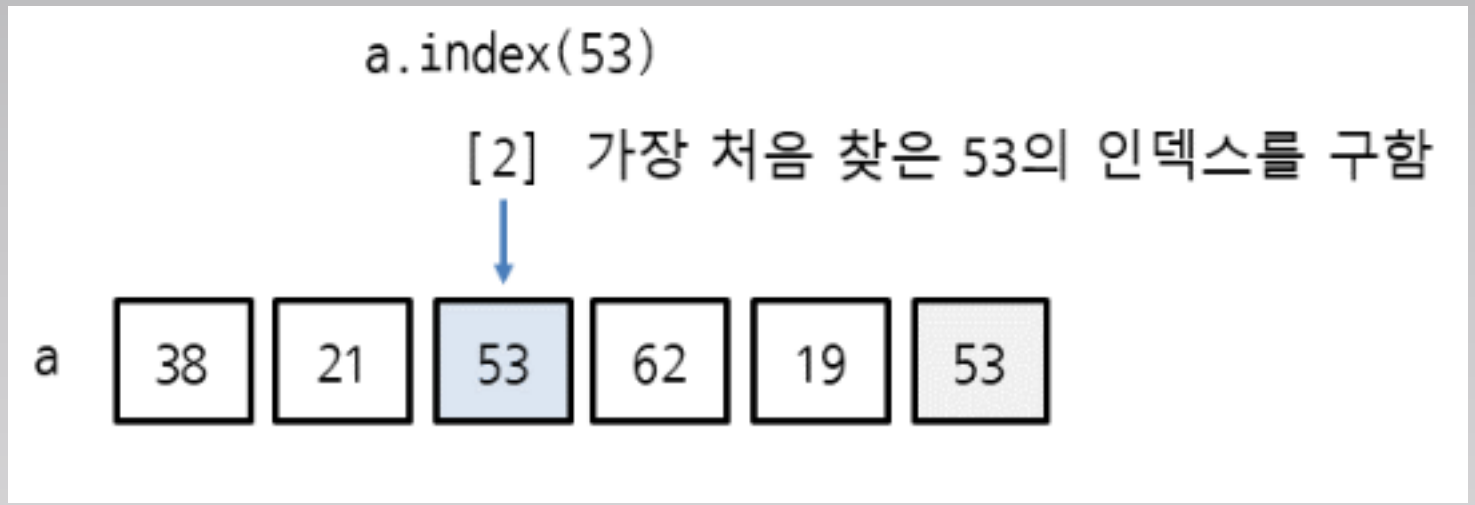
튜플에서 특정 값의 인덱스 구하기

- 같은 값이 여러 개일 경우 처음 찾은 인덱스를 구함(가장 작은 인덱스)

```
>>> a = (38, 21, 53, 62, 19, 53)
>>> a.index(53)
2
```

22.7 튜플 응용하기

▼ 그림 index로 특정 값의 인덱스 구하기

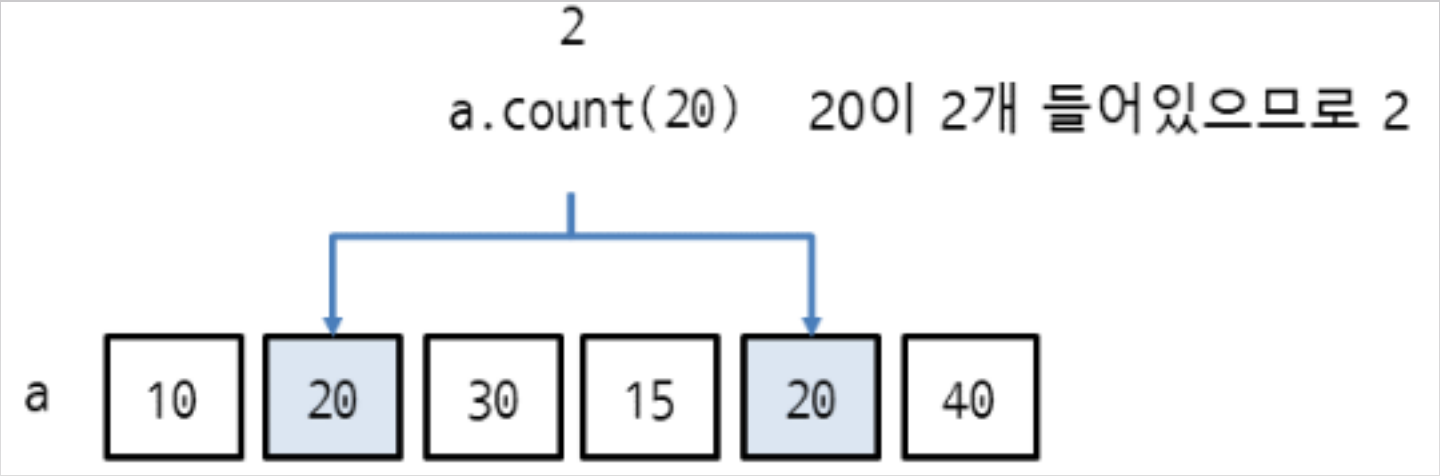


22.7 튜플 응용하기

» 특정 값의 개수 구하기

```
>>> a = (10, 20, 30, 15, 20, 40)
>>> a.count(20)
2
```

▼ 그림 count로 특정 값의 개수 구하기



22.7 튜플 응용하기

for 반복문으로 요소 출력하기

```
>>> a = (38, 21, 53, 62, 19)
>>> for i in a:
...     print(i, end=' ')
...
38 21 53 62 19
```

22.7 튜플 응용하기

튜플 표현식 사용하기

- tuple(식 for 변수 in 리스트 if 조건식)

```
>>> a = tuple(i for i in range(10) if i % 2 == 0)
>>> a
(0, 2, 4, 6, 8)
```

- 참고로 ( )(괄호) 안에 표현식을 넣으면 튜플이 아니라 제너레이터 표현식이 됨

```
>>> (i for i in range(10) if i % 2 == 0)
<generator object <genexpr> at 0x050FE420>
```

22.7 튜플 응용하기

tuple에 map 사용하기

```
>>> a = (1.2, 2.5, 3.7, 4.6)
>>> a = tuple(map(int, a))
>>> a
(1, 2, 3, 4)
```

22.7 튜플 응용하기

» 튜플에서 가장 작은 수, 가장 큰 수, 합계 구하기

```
>>> a = (38, 21, 53, 62, 19)
>>> min(a)
19
>>> max(a)
62
>>> sum(a)
193
```