25장

리스트 다루기

25장 리스트 다루기

25.1 리스트와 튜플 비교

25.2 리스트를 만들고 특정 위치에 있는 원소 가져오기

25.3 원소 반복 횟수 세기 및 원소 위치 찾기

25.4 리스트에 원소 추가하기: append, insert, extend

25.5 리스트에서 원소 제거하기: pop

25.6 원소 값 변경하기

25.7 요약

25.1 리스트와 튜플 비교

25.1 리스트와 튜플 비교

리스트는 임의의 타입인 객체들을 모아둔 컬렉션이다.

- 튜플과 리스트는 모두 원소의 순서가 정해져 있는 컬렉션으로, 첫 번째 원소는 0번, 두 번째 원소는 1번 하는 식으로 인덱스가 정해진다.

리스트와 튜플의 가장 큰 차이는 리스트가 변경 가능한 객체인 반면 튜플은 그렇지 않다는 점이다.

- 리스트의 경우 같은 리스트 객체에 원소를 추가, 삭제, 변경할 수 있다.
- 튜플의 경우 추가, 삭제, 변경 등을 수행하기 위해서는 변경 내용에 따라 새로운 튜플을 만들어야 한다.

튜플은 일반적으로 개수가 미리 정해진 데이터를 저장할 때 사용한다.

좀 더 동적인 데이터를 저장할 때 리스트를 사용한다.

셀프 체크 25.1

25.2 리스트를 만들고 특정 위치에 있는 원소 가져오기

25.2 리스트를 만들고 특정 위치에 있는 원소 가져오기

» 각괄호([ ])를 사용해 리스트를 만들 수 있다.

- L = [ ]라는 문장은 빈 리스트(원소가 없는 리스트) 객체를 만들고 그 객체를 L이라는 변수에 연결한다.

» 리스트를 만들 때 원소를 함께 넣어 초기화할 수도 있다.

- 다음 코드는 원소가 3개인 리스트를 만들고 grocery라는 변수에 연결한 것이다.

```
grocery = ["우유", "계란", "빵"]
```

» 문자열이나 튜플과 마찬가지로 len()을 사용해 리스트 길이를 알 수 있다.

» 문자열이나 튜플과 마찬가지로 리스트 첫 번째 원소의 인덱스 번호는 0이고, 두 번째 원소의 인덱스는 1이다. 리스트 L의 마지막 원소는 len(L) – 1번 인덱스 상에 위치한다.

```
grocery = ["우유", "계란", "빵"]
print(grocery[0])
print(grocery[1])
print(grocery[2])
```

25.2 리스트를 만들고 특정 위치에 있는 원소 가져오기

리스트의 길이를 넘어서는 인덱스를 사용하면 어떤 일이 벌어질까?

```
grocery = ["우유", "계란", "빵"]
print(grocery[3])
```

코드를 실행하면 다음과 같은 오류를 볼 수 있다.

```
Traceback (most recent call last):
  File "<ipython-input-14-c90317837012>", line 2, in <module>
    print(grocery[3])
IndexError: list index out of range
```

- 이 리스트에는 원소가 3개 밖에 없으므로 첫 번째 원소는 0번 인덱스에 있고, grocery의 마지막 원소는 2번 인덱스 위치에 있기 때문이다.
- 따라서 3이라는 인덱스는 리스트의 범위를 벗어난다

셀프 체크 25.2

25.3 원소 반복 횟수 세기 및 원소 위치 찾기

25.3 원소 반복 횟수 세기 및 원소 위치 찾기

- count() 명령을 사용하면 특정 원소가 리스트에 몇 번 나타났는지 알 수 있다. L.count(e)는 리스트 L에 원소 e가 몇 번이나 나타나는지를 알려준다.

- 어떤 값과 일치하는 원소 중 가장 앞에 있는 것의 위치를 index() 명령을 통해 얻을 수 있다. L.index(e)는 리스트 L에 있는 원소 중에 맨 앞에 오는 e의 위치(0부터 시작)를 돌려준다.

```
years = [1984, 1986, 1988, 1988]
print(len(years)) ---- 리스트에 원소가 4개 있으므로 4를 출력
print(years.count(1988)) ---- 리스트에 1988이 두 번 나타나므로 2를 출력
print(years.count(2017)) ---- 2017이 리스트에 없으므로 0을 출력
print(years.index(1986)) ---- 1986이 1번 인덱스 위치에 있으므로 1을 출력
print(years.index(1988)) ---- 1988 중 맨 앞에 있는 것의 위치가 세 번째이므로 2를 출력
```

코드 25-1
리스트에서 count와 index 사용하기

25.3 원소 반복 횟수 세기 및 원소 위치 찾기

» 리스트에 없는 원소의 인덱스를 구하려고 하면 어떤 일이 벌어질까?

```
L = []
L.index(0)
```

» 이 코드를 실행하면 다음과 같은 오류가 발생한다.

```
Traceback (most recent call last):
  File "<ipython-input-15-b3f3f6d671a3>", line 2, in <module>
    L.index(0)
ValueError: 0 is not in list
```

» 셀프 체크 25.3

25.4 리스트에 원소 추가하기:append, insert, extend

25.4.1 append 사용하기

L.append(e)는 리스트 L의 맨 뒤에 원소 e를 추가한다.

- 한 번에 원소를 하나씩만 추가할 수 있다.
- append를 실행하면 리스트 L은 추가한 원소가 더 들어간 상태로 바뀐다.

```
grocery = []
```

```
grocery.append("빵")
```

L.insert(i, e)는 리스트의 i 인덱스 위치에 e를 삽입한다.

- insert를 사용하면 인덱스로 지정한 i에 원래 있던 원소와 그 뒤에 있던 원소를 모두 하나씩 뒤로 밀어낸다.
- insert를 하고 나면 리스트 L은 삽입한 원소가 더 들어간 상태로 바뀐다.

```
grocery = ["빵", "우유"]
```

```
grocery.insert(1, "계란")
```

세 가지 원소를 모두 포함하는 리스트로 상태가 바뀐다

```
["빵", "계란", "우유"]
```

25.4.3 extend 사용하기

L.extend(M)은 리스트 M에 있는 원소를 모두 리스트 L 뒤에 추가한다.

- 결과적으로 M의 모든 원소가 리스트 L 뒤에 순서대로 덧붙여진다.
- 리스트 L의 상태는 M의 모든 원소를 포함한 상태로 바뀌지만, 리스트 M은 변하지 않고 그대로 남는다.

```
grocery = ["빵", "계란", "우유"]
for_fun = ["드론", "VR 헤드셋", "게임 콘솔"]
```

```
grocery.extend(for_fun)
```

합친 뒤 grocery에 있는 구입 목록은 다음과 같다.

```
["빵", "계란", "우유", "드론", "VR 헤드셋", "게임 콘솔"]
```

25.4.3 extend 사용하기


```
first3letters = [] ---- 빈 리스트
first3letters.append("a") ---- first3letters 리스트 맨 뒤에 'a' 추가
first3letters.append("c") ---- first3letters 리스트 맨 뒤에 'c' 추가
first3letters.insert(1,"b") ---- first3letters 리스트의 1번 인덱스 위치에 'b' 삽입
print(first3letters) ---- ['a', 'b', 'c'] 출력
last3letters = ["x", "y", "z"] ---- 원소가 3개인 리스트 생성
first3letters.extend(last3letters) ---- first3letters 리스트 맨 뒤에 'x', 'y', 'z' 추가
print(first3letters) ---- ['a', 'b', 'c', 'x', 'y', 'z'] 출력
last3letters.extend(first3letters) ------ 상태가 바뀐 first3letters 리스트를 last3letters 뒤에 덧붙임.
                                          'a', 'b', 'c', 'x', 'y', 'z'를 덧붙이는 것과 같은 효과
print(last3letters) ---- ['x', 'y', 'z', 'a', 'b',
                          'c', 'x', 'y', 'z'] 출력
```

코드 25-2
리스트에 원소 추가하기

- 리스트가 변경 가능한 객체이기 때문에 리스트에 수행한 특정 연산은 리스트 자체의 상태를 바꾼다.
- 코드 25-2에서 first3letters라는 리스트는 원소를 추가하거나, 삽입하거나, 다른 리스트로 확장할 때마다 상태가 바뀐다.
- 마찬가지로 last3letters도 first3letters를 뒤에 덧붙여 확장할 때 내부 상태가 바뀐다.

25.4.3 extend 사용하기

» 셀프 체크 25.4 – 직접 코딩해서 공부하기

25.5 리스트에서 원소 제거하기: pop

25.5 extend 사용하기

리스트에서 원소를 제거할 때는 pop() 연산을 쓴다.

- L.pop()은 리스트 L의 맨 끝에 있는 원소를 없앤다.
- L.pop(i)처럼 괄호 사이에 없앨 원소의 인덱스 번호를 넣을 수도 있다.

```
polite = ["please", "and", "thank", "you"] ---- 원소가 4개인 리스트
print(polite.pop()) ---- pop()이 리스트의 맨 마지막 원소를 제거하고 반환하기 때문에 'you'를 출력
print(polite) ---- pop()이 리스트의 맨 마지막 원소를 제거했기 때문에 ['please', 'and', 'thank']를 출력
print(polite.pop(1)) ---- pop(1)이 1번 인덱스에 있는 원소를 제거하고 반환하기 때문에 'and'를 출력
print(polite) ---- pop(1)이 1번 인덱스의 원소(두 번째 원소)를 제거했기 때문에 ['please', 'thank']를 출력
```

코드 25-3
리스트에서 원소 제거하기

셀프 체크 25.5

25.6 원소 값 변경하기

25.6 원소 값 변경하기

```
colors = ["red", "blue", "yellow"] ---- 문자열로 이뤄진 리스트 초기화
colors[0] = "orange" ---- 'red'를 'orange'로 변경
print(colors) ---- 두 번째 줄에서 0번 인덱스에 있는 원소를 바꿨기 때문에 ['orange', 'blue', 'yellow']를 출력
colors[1] = "green" ---- 0번 인덱스에 'orange'가 들어 있는 리스트의 'blue'를 'green'으로 변경
print(colors) ---- 네 번째 줄에서 1번 인덱스의 원소를 바꿨기 때문에 ['orange', 'green', 'yellow']를 출력
colors[2] = "purple" ---- 0번 인덱스에 'orange', 1번 인덱스에 'green'이 들어 있는 리스트의 'yellow'를 'purple'로 변경
print(colors) ---- 여섯 번째 줄에서 2번 인덱스에 있는 원소를 바꿨기 때문에 ['orange', 'green', 'purple']을 출력
```

코드 25-4
원소 값 변경하기

» 셀프 체크 25.6

25.7 요약

25.7 요약

- 리스트는 비어있거나 원소가 들어 있을 수 있다.
- 원소를 리스트의 맨 뒤에 추가하거나, 임의의 인덱스에 추가하거나, 뒤에 다른 리스트를 덧붙일 수 있다.
- 리스트에서 원소를 제거할 수 있다. 맨 뒤에 있는 원소를 제거하거나, 특정 인덱스에 있는 원소를 제거할 수 있다.
- 리스트 원소의 값을 변경할 수 있다.
- 연산을 수행하면 리스트의 상태가 바뀐다. 따라서 변수에 대입했던 리스트에 대해 어떤 연산을 수행한 후, 바뀐 리스트를 다른 변수에 대입할 필요가 없다.

25.7 요약

» (Q25.1) 빈 리스트에서 시작해 식당 메뉴가 들어 있는 리스트를 만들려고 한다.

```
menu = [ ]
```

1. 리스트를 변경하는 명령을 한 번 이상 사용해서 리스트가 ["피자","맥주","프렌치프라이","치킨윙","샐러드"]가 되게 만들라.
2. 1의 결과 리스트에 계속 작업을 수행하여 리스트가 ["샐러드","맥주","프렌치프라이","치킨윙","피자"]가 되게 만들라.
3. 2의 결과 리스트에 계속 작업을 수행하여 리스트가 ["샐러드","퀴노아","스테이크"]가 되게 만들라.

25.7 요약

» (Q25.2) unique라는 함수를 만들어라. unique는 L이라는 이름의 리스트를 인자로 받는다.unique 함수는 L의 상태를 바꾸지 않고, L에 포함된 원소 중 유일한 원소들만 포함하는(즉, 2개 이상 들어 있는 원소는 제외함) 새 리스트를 반환한다.

25.7 요약

» (Q25.3) common이라는 함수를 만들어라. common은 L1이라는 리스트와 L2라는 리스트를 인자로 받는다. common은 L1의 유일한 원소들이 모두 L2에 속하고, L2의 유일한 원소들이 모두 L1에 속하면 True를 반환한다. (힌트: Q25.2에서 만든 함수를 재사용해 보자.)

- 예를 들어
- common([1,2,3], [3,1,2])는 True
- common([1,1,1], [1])는 True
- common([1], [1, 2])는 False 이다.