28장

리스트와 사전 별명 붙이기와 복사하기

28장 리스트와 사전 별명 붙이기와 복사하기

28.1 객체의 별명 사용하기

28.1.1 불변 객체에 대한 별명

» **변경 가능한 객체를 살펴보기 전에 두 변수에 대해 같은 불변 객체를 가리키게 만드는 대입 연산(등호)을 사용하면 어떤 일이 생기는지를 살펴보자.**

- 콘솔에서 다음 명령을 입력하고 id() 함수를 사용해 a와 b의 위치를 살펴보자.

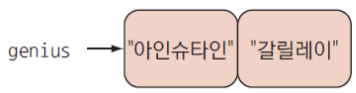
```
a = 1
id(a)
Out[2]: 1906901488

b = a
id(b)
Out[4]: 1906901488
```

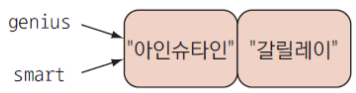
```
a = 2
id(a)
Out[6]: 1906901520

id(b)
Out[7]: 1906901488

a
Out[8]: 2

b
Out[9]: 1
```

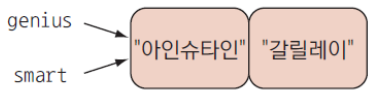
28.1.2 변경 가능한 객체에 대한 별명

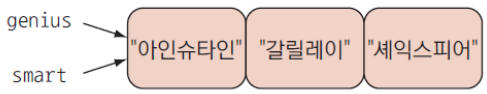

```
genius = ["아인슈타인", "갈릴레이"]
id(genius)
Out[9]: 2899318203976

smart = genius
id(smart)
Out[11]: 2899318203976
```

```
genius.append("셰익스피어")
id(genius)
Out[13]: 2899318203976

id(smart)
Out[14]: 2899318203976

genius
Out[16]: ["아인슈타인", "갈릴레이", "셰익스피어"]

smart
Out[15]: ["아인슈타인", "갈릴레이", "셰익스피어"]
```









28.1.2 변경 가능한 객체에 대한 별명

» 셀프 체크 28.1

» 셀프 체크 28.2

» 셀프 체크 28.3

28.1.3 함수 매개변수로 변경 가능한 객체 전달하기

```
def add_word(d, word, definition): ---- 사전, 문자열(word), 다른 문자열(definition)을 매개변수로 받는 함수
    """ d, 문자열을 문자열의 리스트와 연관시키는 사전(dict)
        word, 문자열
        definition, 문자열
        d를 변경한다(word: definition 쌍을 사전에 추가)
        word가 이미 d에 들어 있다면 definition을 word와 연관된 리스트에 추가한다
        아무 것도 반환하지 않는다
    """
    if word in d: ---- 사전에 들어 있는 word
        d[word].append(definition) ---- definition을 word와 연관된 리스트 끝에 추가
    else: ---- 사전에 들어 있지 않은 word
        d[word] = [definition] ---- 사전에 word를 키로, definition을 유일한 원소로 하는 리스트를 값으로 추가

words = {} ---- 함수 밖, 빈 사전 생성
add_word(words, 'box', 'fight') ---- "words"라는 이름의 사전을 인자로 함수 호출
print(words) ---- {'box': ['fight']} 출력
add_word(words, 'box', 'container') ---- "box"라는 키에 대한 값을 추가하기 위해 함수 호출
print(words) ---- {'box': ['fight', 'container']} 출력
add_word(words, 'ox', 'animal') ---- {'box': ['fight', 'container'], 'ox': ['animal']} 출력
print(words) ---- 다른 항목을 추가하기 위해 함수 호출
```

코드 28-1
사전을 변경하는 함수

28.2 변경 가능한 객체의 복사본 만들기

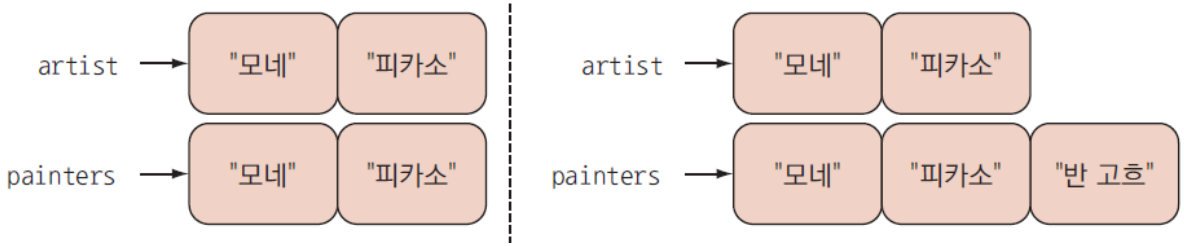
28.2.1 변경 가능한 객체를 복사하는 명령들

1. 원본과 같은 값을 포함하는 리스트 객체를 새로 만드는 것이다.

- 원소가 들어 있는 artists라는 리스트가 있다면 다음 명령을 사용해 새로운 리스트를 만들어 painters라는 변수에 대입할 수 있다.

```
painters = list(artists)
```

```
artists = ["모네", "피카소"]
painters = list(artists)
painters.append("반 고흐")

painters
Out[24]: ["모네", "피카소", "반 고흐"]

artists
Out[25]: ["모네", "피카소"]
```



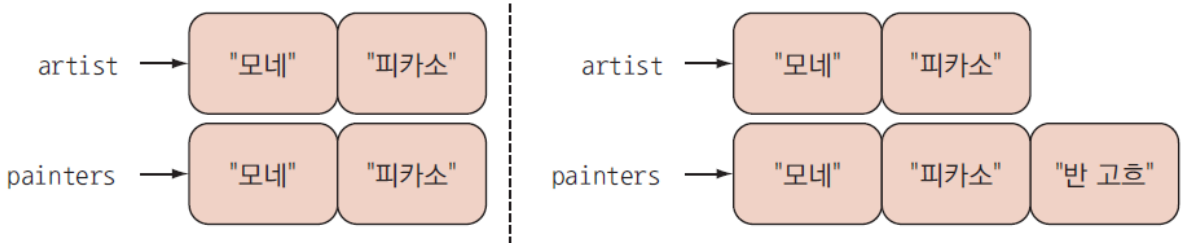
28.2.1 변경 가능한 객체를 복사하는 명령들

2. copy() 함수를 사용하면 된다.

- artists가 리스트라면 다음 명령은 artists와 같은 원소를 포함하는 새 객체를 만들되, 원소들을 복사해서 새 객체에 넣는다

```
painters = artists.copy()
```

```
artists = ["모네", "피카소"]
painters = artists.copy()
painters.append("반 고흐")

painters
Out[24]: ["모네", "피카소", "반 고흐"]

artists
Out[25]: ["모네", "피카소"]
```



28.2.2 리스트 원소를 정렬한 복사본 만들기

» 원본을 그대로 유지하면서 원소가 정렬된 복사본을 얻고 싶은 경우에는 정렬된 버전을 반환하는 함수를 통해 정렬된 복사본을 새 변수에 대입한다.

```
kid_ages = [2,1,4]
sorted_ages = sorted(kid_ages)

sorted_ages
Out[61]: [1, 2, 4]

kid_ages
Out[62]: [2, 1, 4]
```

» kids_ages.sort()를 호출했다면 복사본이 생기지 않고 리스트 자체가 바뀐다.

» 셀프 체크 28.4

28.2.3 변경 가능한 객체에 대한 이터레이션을 수행할 때 조심할 점

» 사전에서 어떤 조건을 만족하는 원소를 제거하고 싶을 때가 있다.

```
songs = {"Wannabe": 1, "Roar": 1, "Let It Be": 5, "Red Corvette": 4} ---- 노래 사전

for s in songs.keys(): ---- 모든 키-값 쌍에 대해 루프 수행
    if songs[s] == 1: ---- 평점이 1이면…
        songs.pop(s) ---- … 그 노래를 제거
```

코드 28-2
사전을 이터레이션하는 도중에 원소를 삭제하려고 시도함

- 이 코드를 실행하면 RuntimeError: dictionary changed size duringiteration라는 오류(사전의 크기가 이터레이션 중에 변경됨이라는 뜻의 오류)와 함께 실패한다.
- 파이썬에서는 이터레이션을 하는 중에 사전의 크기를 바꿀 수 없다. 즉, 이터레이션 하는 중에 사전에 값을 추가하거나 삭제할 수 없다는 뜻이다.

28.2.3 변경 가능한 객체에 대한 이터레이션을 수행할 때 조심할 점

» 같은 작업을 이번에는 사전이 아니라 리스트에 대해 해 보자.

```
songs = [1, 1, 5, 4] ---- 노래 평점 리스트

for s in songs: ---- 모든 평점에 대해 루프를 돎
    if s == 1: ---- 평점이 1이면…
        songs.pop(s) ---- … 그 노래를 제거

print(songs) ---- [1, 5, 4]를 출력
```

코드 28-3
리스트를 이터레이션하는 도중에 원소를 삭제하려고 시도함

- 코드 28-3과 같이 시도해보면 역시 제대로 작동하지 않는다. 하지만 이번에는 예외가 발생하면서 프로그램이 실패하지는 않는다. 다만 우리가 바라는 것과 다른 동작을 할 뿐이다.
- 이 코드는 songs의 값으로 [5, 4]가 아니라 [1, 5, 4]를 남긴다.

28.2.3 변경 가능한 객체에 대한 이터레이션을 수행할 때 조심할 점

» 리스트의 원소를 제거하거나 추가해야 한다면 먼저 복사본을 만들어야 한다. 그리고 원본은 빈 리스트로 만들고, 복사본에 대해 루프를 돌면서 여러분이 계속 남겨두고 싶은 원소를 (새로 값을 추가 중인) 원본에 차례로 추가한다.

```
songs = [1, 1, 5, 4] ---- 원래의 평점 리스트
songs_copy = songs.copy() ---- 복사본을 만듦
songs = [] ---- 평점 리스트를 새로운 빈 리스트로 설정

for s in songs_copy: ---- 모든 평점에 대해 루프 수행
    if s != 1: ---- 평점이 제거 대상이 아닌 경우라면…
        songs.append(s) ---- … 원본 리스트에 그 평점을 추가

print(songs) ---- [5, 4]를 출력
```

코드 28-4
리스트를 이터레이션하면서 원소를 제거하는 올바른 방법

28.3 요약

28.3 요약

- 객체의 타입과 관계 없이 변수에 변수를 대입하면 파이썬은 별명을 만든다.
- 변경 가능한 객체의 별명을 사용하면 예기치 못한 부작용이 생길 수 있다.
- 변경 가능한 객체의 여러 별명 중 하나를 사용해 그 상태를 변경하면, 다른 모든 별명을 통해서도 그 변경을 관찰할 수 있다.
- 변경 가능한 객체를 복사하는 방법은 두 가지다. 하나는 원본에서 모든 원소를 복사하는 것이고, 다른 하나는 원본에 copy()를 사용해 새로운 객체를 만드는 것이다.

28.3 요약

» (Q28.1) invert_dict라는 함수를 작성하라. invert_dict는 사전을 입력으로 받고, 새로운 사전을 반환한다. 새 사전의 값은 원본 사전의 키이며, 새 사전의 키는 원본 사전의 값이다. 이때 원본 사전의 키나 값은 모두 변경 불가능한 객체들이고, 값은 모두 유일하다고 가정하자.

28.3 요약

» (Q28.2) invert_dict_inplace라는 함수를 작성하라. 이 함수는 사전을 입력으로 받고 아무 것도 반환하지 않는다. 다만 전달받은 사전의 키는 값으로 값은 키로 만들되, 별도의 객체를 만들지 않고 원본 사전을 그 자리에서 변경한다. 역시 원본 사전의 키나 값은 모두 변경 불가능한 객체들이고, 값은 모두 유일하다고 가정하자.