파이썬 프로그래밍

사전

한국기술교육대학교
온라인평생교육원

▣ 사전 메소드

1. 중요 메소드

- D.keys(): 사전 D에서 키들을 리스트로 반환
- D.values(): 사전 D에서 값들을 리스트로 반환
- D.items(): 사전 D에서 각 아이템을 튜플형태로 가져와 리스트로 반환
- key in D: 사전 D안에 key를 키값을 가진 아이템이 있는지 확인

```
phone = {'jack':9465215, 'jin':1111, 'Joseph':6584321}

print phone.keys() # 키의 리스트 반환
print phone.values() # 값들의 리스트 반환
print phone.items() # (키, 값)의 리스트 반환
print
print 'jack' in phone # 'jack'이 phone의 키에 포함되어 있는가?
print 'lee' in phone
```

```
['jin', 'Joseph', 'jack']
[1111, 6584321, 9465215]
[('jin', 1111), ('Joseph', 6584321), ('jack', 9465215)]

True
False
```

- phone.keys( ) → 리스트 안에 각각의 사전 키들만 반환
- 순서대로 반환 X → 내부에서 정한대로 반환
- 사전은 순서가 없기 때문에 임의의 순서대로 반환됨
- phone.values( ) → 리스트 안에 각각의 사전 값들만 반환
- phone.items( ) → (키, 값)으로 튜플이 원소가 되어 리스트 반환
- 'key' in 사전 이름 → 그 사전 안에 'key'가 있는지?
- in 이라는 키워드는 'key'에 대한 검색만 가능
- Phone.values( ) → 사전의 value을 의미하므로 in으로 value 검색 가능

▣ 사전 메소드

1. 중요 메소드

- D2 = D.copy(): 사전 D를 복사하여 D2 사전에 할당한다.

```
phone = {'jack':9465215, 'jin':1111, 'Joseph':6584321}
p = phone # 사전 레퍼런스 복사. 사전 객체는 공유된다.

phone['jack'] = 1234 # phone을 변경하면
print phone
print p # p도 함께 변경된다.
print

ph = phone.copy() # 사전복사. 별도의 사전 객체가 마련된다.
phone['jack'] = 1111 # phone을 바꿔도
print phone
print ph # ph는 바뀌지 않는다.
```

```
{'jin': 1111, 'Joseph': 6584321, 'jack': 1234}
{'jin': 1111, 'Joseph': 6584321, 'jack': 1234}

{'jin': 1111, 'Joseph': 6584321, 'jack': 1111}
{'jin': 1111, 'Joseph': 6584321, 'jack': 1234}
```

- phone 변수는 사전 객체를 레퍼런스하고 있는 참조 값을 지님
- p = phone → phone 자체가 가지고 있는 참조값을 p에 할당
- p와 phone은 동일한 사전 객체를 가리킴
- phone이 변경되면 p도 동일하게 변경됨
- phone.copy( ) → 기존 phone 을 그대로 복사해서 새로운 객체 생성
- phone과 ph는 서로 다른 객체를 가리킴

▣ 사전 메소드

1. 중요 메소드

- [주의] D.copy()는 Shallow Copy를 수행한다.

```
phone = {'a': [1,2,3], 'b': 4}
phone2 = phone.copy()
print phone
print phone2
print

phone['b'] = 100
print phone
print phone2
print

phone['a'][0] = 100
print phone
print phone2
```

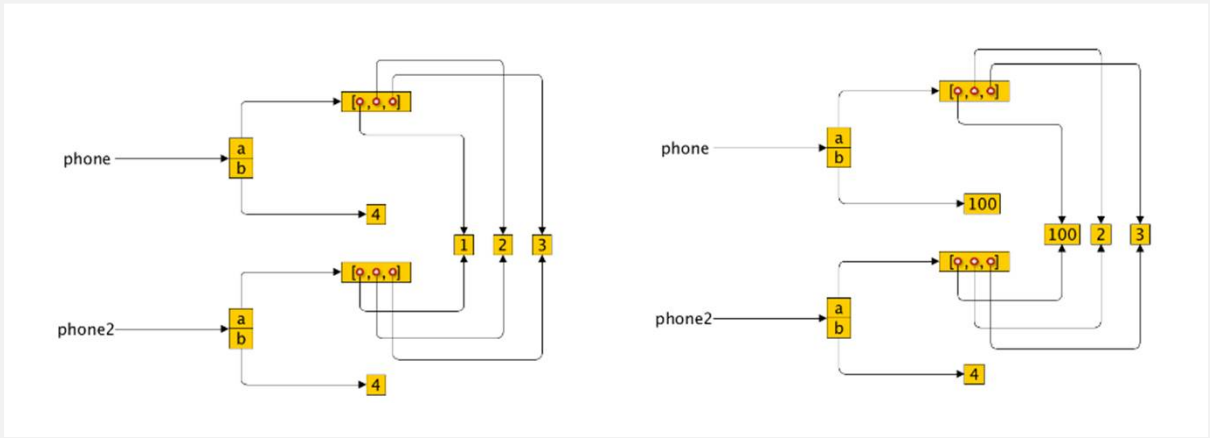
```
{'a': [1, 2, 3], 'b': 4}
{'a': [1, 2, 3], 'b': 4}

{'a': [1, 2, 3], 'b': 100}
{'a': [1, 2, 3], 'b': 4}

{'a': [100, 2, 3], 'b': 100}
{'a': [100, 2, 3], 'b': 4}
```

▣ 사전 메소드

1. 중요 메소드



- phone.copy( ) → shallow copy
- a key → 리스트, b key → 4
- 정수 값은 복사가 되면서 똑같지만 다른 새로운 객체 생성
- 리스트 안 존재하는 1, 2, 3 원소는 공유가 됨
- 1, 2, 3 자체가 copy되어 별도의 1, 2, 3 존재 X
- Shallow copy 반대 → Deep copy
- Deep copy → 공유된 리스트 1, 2, 3을 별도로 만들어 줌
- Shallow copy → 복사하려는 리스트 안 원소까지는 복사 X
- copy( ) → deep copy가 아니라 보통 shallow copy

▣ 사전 메소드

1. 중요 메소드

```
ph = {'jack':9465215, 'jin':1111, 'Joseph':6584321}

print ph.get('jack')  # 'jack'에 대한 값을 얻는다. ph['jack']과 같다.
print ph.get('gslee') # 'gslee'에 대한 값을 얻는다. 값이 없는 경우 None반환
```

```
9465215
None
```

- ph.get('jack') → jack 키에 있는 value 값 가져옴
- get( ) → key를 주면서 value를 검색하는 메소드
- ph['jack'] = ph.get('jack')

```
ph = {'jack':9465215, 'jin':1111, 'Joseph':6584321}
print ph['gslee'] # ph['gslee']는 키가 없는 경우 예외발생
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-11-00dc298a68ec> in <module>()
      1 ph = {'jack':9465215, 'jin':1111, 'Joseph':6584321}
----> 2 print ph['gslee']      # ph['gslee']는 키가 없는 경우 예외발생

KeyError: 'gslee'
```

- ph['gslee'] → 꺽쇠가로는 값이 없으면 error 발생
- 값이 없을 때 ph['a'] = ph.get('a')의 출력결과가 다름

▣ 사전 메소드

1. 중요 메소드

```
ph = {'jack':9465215, 'jin':1111, 'Joseph':6584321}
print ph.get('gslee', 5284) # 인수를 하나 더 제공하면 'gslee'가 없는 경우에 5284 리턴
print ph # 사전에는 변화가 없다
print

print ph.popitem()              # 임의의 아이템을 꺼낸다.
print ph
print

print ph.popitem()              # 임의의 아이템을 꺼낸다.
print ph
print

print ph.pop('jack')            # 키 값을 통해 해당 아이템을 지정하여 꺼낸다.
print ph
```

```
5284
{'jin': 1111, 'Joseph': 6584321, 'jack': 9465215}

('jin', 1111)
{'Joseph': 6584321, 'jack': 9465215}

('Joseph', 6584321)
{'jack': 9465215}

9465215
{}
```

- get 에서 인수를 하나 더 제공하면 gslee가 없을 경우 5284 리턴
- get은 본래 사전에 변화를 주는 메소드 X
- pop.item( ) → 임의의 item을 꺼내는 메소드
- pop('key') → key 값을 통해 해당 item을 지정하여 꺼냄
- pop이 돌려주는 것은 value 값이지만 실제는 item 반환

▣ 사전 메소드

1. 중요 메소드

```
phone = {'jack':9465215, 'jin':1111, 'Joseph':6584321}
ph = {'kim':12312, 'lee': 9090}
phone.update(ph) # 사전 phone의 내용을 ph으로 추가 갱신
print phone
print
phone.clear() # 사전의 모든 입력을 없앤다.
print phone
```

```
{'jin': 1111, 'Joseph': 6584321, 'jack': 9465215, 'kim': 12312, 'lee': 9090}

{}
```

- phone.update(ph) → phone의 내용을 ph로 추가하여 업데이트
- clear 메소드 → 전체 사전 내용을 없앰