

# 자료구조



사전

---

# 사전 (Dictionary)



- 사전

- 우리가 일상 생활에서 사용하는 사전처럼, 이름(label)과 그 이름에 대응되는 내용(detail)이 하나의 항목으로 연결되어 있는 자료 구조이다.

**python** 미국식 [-θaɪn]  영국식 ['paɪθən]  [+ 단어장 추가](#)



[명사] 비단뱀

**Python** 미국식  영국식  [+ 단어장 추가](#)

[명사] (그리스 신화) 피톤(Apollo가 Delphi에서 퇴치한 거대한 뱀).

# 사전 (Dictionary)

- 사전
  - 하나의 항목은 이름(label)과 그 이름에 대응되는 내용(detail)이 한 쌍(pair)으로 연결되어 있다.
  - 이 때, 이름을 **키(key)**, 대응되는 내용을 **값(value)**이라고 한다.
  - 따라서 사전에 있는 각각의 항목은 키-값의 쌍(**key-value pair**)이 된다.

사전 “홍길동의 신상명세”

키(key)	값(value)
이름	홍길동
생일	9월 18일
직업	의적

사전 “나의 연락처”

키(key)	값(value)
홍길동	02-987-6543
성춘향	010-1234-5678

# 사전 (Dictionary)

- 사전 만들기

사전의 이름 = { 키1:값1, 키2:값2, ... }

- 사전은 여러 항목을 쉼표 ‘,’ 기호로 구분하여 기재하고 그 항목들의 양 쪽 끝을 중괄호 ‘{ }’ 기호로 씌워서 만든다.
- 각각의 항목은 키-값의 쌍으로서, 키와 값 사이에 콜론 ‘:’ 기호를 넣어서 키:값의 형태로 기재한다.

d = {“name”:“홍길동”, “birth”:“9월 18일”}

d

“name”	“홍길동”
“birth”	“9월 18일”

※ 이 때, 키는 변하지 않는(immutable) 값이어야 하고 사전 내에서 각각 유일한(unique) 값이어야 한다.

# 사전 (Dictionary)

- 사전 만들기

사전의 이름 = { 키1:값1, 키2:값2, ... }

```
1 d1 = {"name": "홍길동", "birth": "9월 18일", "job": "의적"}
2
3 d2 = {"홍길동": "02-987-6543", "성춘향": "010-1234-5678"}
4
5 d3 = {2016: "원숭이", 2017: "닭", 2018: "개", 2019: "돼지", 2020: "쥐"}
6
7 d4 = {("홍길동", "950918"): [91, 79, 56], ("성춘향", "991231"): [80, 92, 62],
8       ("김철수", "780123"): [73, 66, 81], ("이순신", "450428"): [100, 85, 43]}
9
10 d5 = {}
11
12 d6 = dict()
```

# 사전 (Dictionary)

- 사전 내의 1개의 값 접근하기

## ① 사전의 이름[항목의 키]

- 사전에 있는 항목의 값(value)을 특정하려면 사전의 이름 뒤에 그 항목의 키(key)를 기재하고 양 쪽 끝을 대괄호 '[' ]' 기호로 씌워서 접근할 수 있다.

d	"name"	"홍길동"
	"birth"	"9월 18일"

d["name"] : 문자열 "홍길동"

d["job"] : 오류

- ※ 사전의 항목들에는 순서가 없으며, 따라서 인덱스가 존재하지 않는다. (즉, 가장 처음에 기재한 키-값의 쌍이 0번 항목이라고 말할 수 없다.)

# 사전 (Dictionary)

- 사전 내의 1개의 값 접근하기

- ② 사전의 이름.get(항목의 키)

- 또는, 명령어 get을 이용하여 항목의 키(key)에 대응하는 값(value)을 가져올 수 있다.
    - get을 사용하는 경우, 키가 존재하지 않으면 None이라는 값을 되돌려 준다.

d

"name"	"홍길동"
"birth"	"9월 18일"

d.get("name") : 문자열 "홍길동"      d.get("job") : None

※ None은 "아무 것도 없음", "아무 것도 아님"을 의미하는 특수한 자료 값이다.

# 사전 (Dictionary)

- 사전 내의 1개의 값 접근하기

사전의 이름[항목의 키] 또는 사전의 이름.get(항목의 키)

d3 = {2016: “원숭이”, 2017: “닭”, 2018: “개”,  
2019: “돼지”, 2020: “쥐”}

d3

2016	“원숭이”
2017	“닭”
2018	“개”
2019	“돼지”
2020	“쥐”

```
1 print(d3[2019])
2
3 print(d3.get(2020))
4
5 print(d3.get(2021))
6
7 print(d3[2022])
```

돼지  
쥐  
None

-----  
KeyError



# 사전 (Dictionary)

- 사전의 특정 값 수정하기

**사전의 이름[항목의 키] = 수정할 값 또는 연산**

– 특정 키에 대응하는 값을 변경할 수 있다.

```
1 d3[2019] = "pig"
2 print(d3)
```

{2016: '원숭이', 2017: '닭', 2018: '개', 2019: 'pig', 2020: '쥐'}

```
1 d4[("홍길동", "950918")] = [91, 79, 56, 99]
2 print(d4)
```

{('홍길동', '950918'): [91, 79, 56, 99], ('성춘향', '991231'): [80, 92, 62], ('김철수', '780123'): [73, 66, 81], ('이순신', '450428'): [100, 85, 43]}

```
1 d4[("홍길동", "950918")][-1] = 100
2 print(d4)
```

{('홍길동', '950918'): [91, 79, 56, 100], ('성춘향', '991231'): [80, 92, 62], ('김철수', '780123'): [73, 66, 81], ('이순신', '450428'): [100, 85, 43]}

# 사전 (Dictionary)

- 사전에 새로운 항목 1개 추가하기

**사전의 이름[새로운 키] = 추가할 값 또는 연산**

– 사전에 새로운 키-값의 쌍을 추가한다.

```
1 d3[2021] = "소"  
2 print(d3)
```

{2016: '원숭이', 2017: '닭', 2018: '개', 2019: 'pig', 2020: '쥐', 2021: '소'}

```
1 d4[("홍길동", "950919")] = [100] * 4  
2 print(d4)
```

{('홍길동', '950918'): [91, 79, 56, 100], ('성춘향', '991231'): [80, 92, 62], ('김철수', '780123'): [73, 66, 81], ('이순신', '450428'): [100, 85, 43], ('홍길동', '950919'): [100, 100, 100, 100]}

# 사전 (Dictionary)

- 사전에서 항목 삭제하기

**del** 사전의 이름[삭제하려는 항목의 키]

– 명령어 del을 이용하여 특정 항목을 삭제한다.

```
1 print(d3)
2 del d3[2016]
3 print(d3)
```

```
{2016: '원숭이', 2017: '닭', 2018: '개', 2019: 'pig', 2020: '쥐', 2021: '소'}
{2017: '닭', 2018: '개', 2019: 'pig', 2020: '쥐', 2021: '소'}
```

```
1 print(d4)
2 del d4[("홍길동", "950919")]
3 print(d4)
```

```
{('홍길동', '950918'): [91, 79, 56, 100], ('성춘향', '991231'): [80, 92, 62],
('김철수', '780123'): [73, 66, 81], ('이순신', '450428'): [100, 85, 43], ('홍
길동', '950919'): [100, 100, 100, 100]}
{('홍길동', '950918'): [91, 79, 56, 100], ('성춘향', '991231'): [80, 92, 62],
('김철수', '780123'): [73, 66, 81], ('이순신', '450428'): [100, 85, 43]}
```

# 사전 (Dictionary)

- 사전에서 키 모음 가져오기

**사전의 이름**.**keys()**

– 대상 사전 내의 모든 항목들의 키들을 모아서 가져온다.

```
1 k1 = d3.keys()
2 print(k1)
```

dict\_keys([2017, 2018, 2019, 2020, 2021])

```
1 t1 = list(k1)
2 print(t1)
```

[2017, 2018, 2019, 2020, 2021]

```
1 k2 = d4.keys()
2 print(k2)
```

dict\_keys([('홍길동', '950918'), ('성춘향', '991231'), ('김철수', '780123'), ('이순신', '450428')])

※ 자료 구조 dict\_keys는 리스트로 변환할 수 있다.

# 사전 (Dictionary)

- 사전에서 값 모음 가져오기

**사전의 이름.values( )**

– 대상 사전 내의 모든 항목들의 값들을 모아서 가져온다.

```
1 v1 = d3.values()  
2 print(v1)
```

dict\_values(['닭', '개', 'pig', '쥐', '소'])

```
1 t1 = list(v1)  
2 print(t1)
```

['닭', '개', 'pig', '쥐', '소']

```
1 v2 = d4.values()  
2 print(v2)
```

dict\_values([[91, 79, 56, 100], [80, 92, 62], [73, 66, 81], [100, 85, 43]])

※ 자료 구조 dict\_values는 리스트로 변환할 수 있다.

# 사전 (Dictionary)

- 사전에서 키-값의 쌍 모음 가져오기

## 사전의 이름.items()

- 대상 사전 내의 각 항목, 즉 키와 값의 쌍 1개를 하나의 튜플 형태로 만든 뒤, 이 튜플들을 모아서 가져온다.

```
1 i1 = d3.items()
2 print(i1)
```

```
dict_items([(2017, '닭'), (2018, '개'), (2019, 'pig'),
(2020, '쥐'), (2021, '소')])
```

```
1 i2 = d4.items()
2 print(i2)
```

```
dict_items([(('홍길동', '950918'), [91, 79, 56, 100]),
((('성춘향', '991231'), [80, 92, 62]), (('김철수', '78012
3'), [73, 66, 81])), (('이순신', '450428'), [100, 85, 43])])
```

※ 자료 구조 dict\_items는 리스트로 변환할 수 있다.

# 사전 (Dictionary)

- 사전에서 키-값의 쌍 모음 가져오기

사전의 이름.items()

- 사전은 결국 내부적으로 항목이 2개인 튜플들이 모여 있는 리스트와 마찬가지로이다.

```
1 t1 = list(i1)
2 print(t1)
```

[(2017, '닭'), (2018, '개'), (2019, 'pig'), (2020, '쥐'), (2021, '소')]

```
1 t2 = list(i2)
2 print(t2)
```

[(('홍길동', '950918'), [91, 79, 56, 100]), (('성춘향', '991231'), [80, 92, 62]),  
 (('김철수', '780123'), [73, 66, 81]), (('이순신', '450428'), [100, 85, 43])]

# 사전 (Dictionary)

- 사전에 특정 키가 존재하는지 확인하기

**찾으려는 키** in **사전의 이름** 또는 **사전의 이름**.keys( )

- 사전에 특정 키가 있는지 검사하여, 키가 그 사전에 들어 있다면 결과로 참(True)을 되돌려 준다.

```
1 2019 in d3
```

True

```
1 ("홍길동", "950918") in d4.keys()
```

True

```
1 2022 in d3.keys()
```

False

```
1 "홍길동" in d4
```

False

**찾으려는 키** not in **사전의 이름** 또는 **사전의 이름**.keys( )

- 사전에 특정 키가 있는지 검사하여, 키가 그 사전에 들어 있다면 결과로 거짓(False)을 되돌려 준다.



# 사전 (Dictionary)

- 사전에 특정 값이 존재하는지 확인하기

**찾으려는 키** in **사전의 이름**.values( )

- 사전에 특정 값이 있는지 검사하여, 값이 그 사전에 들어 있다면 결과로 참(True)을 되돌려 준다.

```
1 "쥬" in d3.values()
```

True

```
1 [73, 66, 81] in d4.values()
```

True

```
1 2019 in d3.values()
```

False

```
1 73 in d4.values()
```

False

**찾으려는 키** not in **사전의 이름**.values( )

- 사전에 특정 값이 있는지 검사하여, 값이 그 사전에 들어 있다면 결과로 거짓(False)을 되돌려 준다.

# 사전 (Dictionary)

- 사전에 특정 키-값의 쌍이 존재하는지 확인하기

**찾으려는 키** in **사전의 이름.items()**

- 사전에 특정 항목이 있는지 검사하여, 항목이 그 사전에 들어 있다면 결과로 참(True)을 되돌려 준다.

```
1 (2020, "쥐") in d3.items()
```

True

```
1 2019 in d3.items()
```

False

```
1 (2019, "돼지") in d3.items()
```

False

```
1 2020, "쥐" in d3.items()
```

(2020, False)

**찾으려는 키** not in **사전의 이름.items()**

- 사전에 특정 항목이 있는지 검사하여, 항목이 그 사전에 들어 있다면 결과로 거짓(False)을 되돌려 준다.

# 사전 (Dictionary)

- 사전의 길이 구하기

**len(사전의 이름)**

- 대상 사전에서 항목이 모두 몇 개 존재하는지 확인하여 그 개수를 되돌려 준다.

```
1 d1 = {"name": "홍길동", "birth": "9월 18일", "job": "의적"}
2 print(len(d1))
3
4 d3 = {2016: "원숭이", 2017: "닭", 2018: "개", 2019: "돼지", 2020: "쥐"}
5 print(len(d3))
6
7 d4 = {("홍길동", "950918"): [91, 79, 56], ("성춘향", "991231"): [80, 92, 62],
8       ("김철수", "780123"): [73, 66, 81], ("이순신", "450428"): [100, 85, 43]}
9 print(len(d4))
10
11 d5 = {}
12 print(len(d5))
```

3  
5  
4  
0

# 사전 (Dictionary)

- 사전 자료형 확인하기

**type(자료형을 확인하고 싶은 1개의 자료)**

- 명령어 type을 이용하여 값 또는 변수가 어떤 자료형인지 확인할 수 있다.

```
1 d3 = {2016: "원숭이", 2017: "닭", 2018: "개", 2019: "돼지", 2020: "쥐"}
2 print(type(d3))
3 print(type(d3[2019]))
4
5 d4 = {("홍길동", "950918"): [91, 79, 56], ("성춘향", "991231"): [80, 92, 62],
6       ("김철수", "780123"): [73, 66, 81], ("이순신", "450428"): [100, 85, 43]}
7 print(type(d4))
8 print(type(d4[("홍길동", "950918")]))
9
10 d5 = {}
11 print(type(d5))
```

```
<class 'dict'>
<class 'str'>
<class 'dict'>
<class 'list'>
<class 'dict'>
```

# 사전 (Dictionary)

- 명령어를 이용하여 사전 만들기

사전의 이름 = `dict(키1=값1, 키2=값2, ...)`

- 명령어 `dict`를 이용하여 새로운 사전을 만든다.
- 이렇게 만들 경우, 키는 변수명 형태로 기재해야 한다.

```
1 dl_new = dict(name="홍길동", birth="9월 18일", job="의적")
2 print(dl_new)
```

```
{'name': '홍길동', 'birth': '9월 18일', 'job': '의적'}
```

```
1 d3_new = dict(Y2016="원숭이", Y2017="닭", Y2018="개", Y2019="돼지", Y2020="쥐")
2 print(d3_new)
```

```
{'Y2016': '원숭이', 'Y2017': '닭', 'Y2018': '개', 'Y2019': '돼지', 'Y2020': '쥐'}
```

```
1 d3_error = dict(2016="원숭이", 2017="닭", 2018="개", 2019="돼지", 2020="쥐")
2 print(d3_error)
```

File "<ipython-input-106-fd0076cac712>", line 1

`d3_error = dict(2016="원숭이", 2017="닭", 2018="개", 2019="돼지", 2020="쥐")`

**SyntaxError:** keyword can't be an expression

# 사전 (Dictionary)

- 항목이 튜플인 리스트를 사전 자료형으로 변환하기  
사전의 이름 = `dict(변환하려는 리스트)`
  - 명령어 `dict`를 이용하여 튜플들의 리스트를 새로운 사전으로 변환한다.

```
1 name = "홍길동"
2 birth = "9월 18일"
3 job = "의적"
4
5 t1 = "name", name
6 t2 = "birth", birth
7 t3 = "job", job
8
9 data = [t1, t2, t3]
10 print(data)
```

```
[('name', '홍길동'), ('birth', '9월 18일'), ('job', '의적')]
```

```
1 dl_from_list = dict(data)
2 print(dl_from_list)
```

```
{'name': '홍길동', 'birth': '9월 18일', 'job': '의적'}
```

# 사전 (Dictionary)

- 사전에 다른 사전 결합하기

사전의 이름.update(결합할 1개의 사전)

- 대상 사전에 다른 사전을 결합하여 확장한다.
- 동일한 키가 존재하는 경우, 원래 사전에 있는 값은 결합할 사전의 값으로 대체된다.

```
1 d1_other = {"birth": "09.18", "company": "활빈당", "position": "CEO"}
2
3 d1.update(d1_other)
4
5 print(d1)
```

```
{'name': '홍길동', 'birth': '09.18', 'job': '의적', 'company': '활빈당',  
'position': 'CEO'}
```

# 사전 (Dictionary)

- 사전을 만들 때 주의할 점
  - 키는 변하지 않는 값이어야 하기 때문에, 리스트 또는 사전을 사전의 키로 사용할 수 없다.

```
1 d = {[1, 2, 3]: "one, two, three"}
```

```
-----  
TypeError                                Traceback  
<i python-input-130-6134f0c736bf> in <module>  
--> 1 d = {[1, 2, 3]: "one, two, three"}
```

```
TypeError: unhashable type: 'list'
```

- 키는 사전 내에서 유일한 값이어야 한다. 그렇지 않을 경우, 정상적으로 동작한다는 것을 보장하지 않는다.

```
1 d = {"name": "홍길동", "name": "Hong, Kil-Dong", "birth": "9월 18일"}  
2 print(d)
```

```
{'name': 'Hong, Kil-Dong', 'birth': '9월 18일'}
```