

23강. 문제풀이(딕셔너리와 반복문)

혼자 공부하는 파이썬 - 윤인성 -

<https://www.youtube.com/watch?v=WaBQz5Gnk-s&list=PLBXuLgInP-5kr0PclHz1ubNZgESmliuB7&index=25>



TABLE OF CONTENTS



1. 딕셔너리 내부에 키가 있는지 확인하기

1. 딕셔너리 내부에 키가 있는지 확인하기(in 키워드)

딕셔너리에 존재하지 않는 키에 접근하면 KeyError가 발생합니다.
크게 **in 키워드**와 **get()함수**를 사용하여 확인할 수 있습니다.

```
dictionary = {  
    "name": "7D 건조 망고",  
    "type": "당절임",  
    "ingredient": ["망고", "설탕", "메타중아황산나트륨", "치자황색소"],  
    "origin": "필리핀"  
}
```

```
key = input("접근하고자 하는 키를 입력하세요:")
```

```
if key in dictionary:  
    print(key, ":", dictionary[key])  
else:  
    print("존재하지 않는 키입니다.")
```

```
PS C:\Users\User\Desktop\작업\Python>  
접근하고자 하는 키를 입력하세요:name  
name : 7D 건조 망고  
PS C:\Users\User\Desktop\작업\Python>  
접근하고자 하는 키를 입력하세요:zz  
존재하지 않는 키입니다.
```

2. 딕셔너리 내부에 키가 있는지 확인하기(get함수)

딕셔너리[키]를 입력할 때와 같은 기능을 수행하지만, 존재하지 않는 키에 접근할 경우 `KeyError`를 발생하지 않고 `None`을 출력합니다.

```
dictionary = {  
    "name": "7D 건조 망고",  
    "type": "당절임",  
    "ingredient": ["망고", "설탕", "메타중아황산나트륨", "치자황색소"],  
    "origin": "필리핀"  
}  
  
print(dictionary.get("age"))
```

〈실행결과〉
None

확인문제 1번

1. 다음 표에서 dict_a의 결과가 나오도록 빈칸을 채워보세요.

| dict_a의 값 | dict_a에 적용할 코드 | dict_a의 결과 |
|----------------|----------------------|----------------|
| {} | <input type="text"/> | {"name": "구름"} |
| {"name": "구름"} | <input type="text"/> | {} |

<정답>

```
dict_a["name"] = "구름"  
del dict_a["name"]
```

확인문제 2번

2. 딕셔너리와 리스트를 조합하면 다음 코드의 변수 `pets`처럼 다양한 정보를 축적할 수 있습니다. 이를 실행결과처럼 출력되도록 빈칸에 반복문과 `print()` 함수를 조합해 보세요.

```
# 딕셔너리를 선언합니다.
```

```
pets = [  
    {"name": "구름", "age": 5},  
    {"name": "초코", "age": 3},  
    {"name": "아지", "age": 1},  
    {"name": "호랑이", "age": 1}  
]
```

실행결과

```
# 우리 동네 애완 동물들  
구름 5살  
초코 3살  
아지 1살  
호랑이 1살
```

↓
숫자와 문자열 사이에
빈 칸이 없게 출력하세요.

확인문제 2번 (정답)

```
pets= [  
    {"name": "구름", "age": 5},  
    {"name": "초코", "age": 3},  
    {"name": "아지", "age": 1},  
    {"name": "호랑이", "age": 1}  
]  
print("# 우리 동네 애완 동물들")  
for pet in pets:  
    print("{} {}살".format(pet["name"], pet["age"]))
```

일반적으로 데이터 배열할 때 pets처럼 뒤에 -s를 붙이는 경우가 많음.
반복문 변수에는 -s를 떼줌.

확인문제 3번

3. 다음 빈칸을 채워서 numbers 내부에 들어 있는 숫자가 몇 번 등장하는지를 출력하는 코드를 작성해 보세요.

숫자는 무작위로 입력해도 상관 없습니다.

```
numbers = [1,2,6,8,4,3,2,1,9,5,4,9,7,2,1,3,5,4,8,9,7,2,3]
```

```
counter = {}
```

```
for number in numbers:
```

최종 출력

```
print(counter)
```

실행결과

7:33 / 15:18

{1: 3, 2: 4, 6: 1, 8: 2, 4: 3, 3: 3, 9: 3, 5: 2, 7: 2}

확인문제 3번 (정답)

```
numbers = [1,2,6,8,4,3,2,1,9,5,4,9,7,2,1,3,5,4,8,9,7,2,3]
counter = {}

for number in numbers:
    if number in counter:
        counter[number] += 1
    else:
        counter[number] = 1

print(counter)
```

counter... 이해 될 때까지 분석하고 외워두기

확인문제 4번

이를 활용해 다음 빈칸을 채워 실행결과와 같이 출력되게 만들어 보세요.

딕셔너리를 선언합니다.

```
character = {  
    "name": "기사",  
    "level": 12,  
    "items": {  
        "sword": "불꽃의 검",  
        "armor": "풀플레이트"  
    },  
    "skill": ["베기", "세게 베기", "아주 세게 베기"]  
}
```

for 반복문을 사용합니다.

```
for key in character:
```

실행결과

```
name : 기사  
level : 12  
sword : 불꽃의 검  
armor : 풀플레이트  
skill : 베기  
skill : 세게 베기  
skill : 아주 세게 베기
```

확인문제 4번 (정답)

```
character = {  
    "name": "기사",  
    "level": 12,  
    "items": {  
        "sword": "불꽃의 검",  
        "armor": "풀플레이트"  
    },  
    "skill": ["베기", "세게 베기", "아주 세게 베기"]  
}  
  
for key in character:  
    if type(character[key]) is dict:  
        for k in character[key]:  
            print("{} : {}".format(k, character[key][k]))  
    elif type(character[key]) is list:  
        for a in character[key]:  
            print("{} : {}".format(key, a))  
    else:  
        print("{} : {}".format(key, character[key]))
```