

# Lesson 8

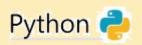
Comprehension(컴프리헨션)



## 목차

- 1. comprehension 개요
- 2. list comprehension
- 3. zip() 함수
- 4. dictionary comprehension

#### Comprehension이란?



- iterable(반복 가능한) 객체를 다룰 때 반복문을 한 줄로 작성하는 문법.
- 기존의 객체를 확장 변경하는데 주로 사용됨.

<ul><li>List Comprehension</li></ul>	[	]
Set Comprehension	{	}
<ul><li>Dict Comprehension</li></ul>	{	}

- 사용법:
  - 변수=[출력값 for 변수 in 반복 가능한 객체]
  - 변수=[출력값 for 변수 in 반복 가능한 객체 if 조건식]
- Comprehension(컴프리헨션)의 장점
  - 속도가 빠르다
  - 간결하고 직관적이다

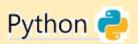
comprehension : 이해, 이해력 컴퓨터 과학 분야에서 어떤 데이터 구조나 알고리즘을 더 쉽게 이해하고 읽을 수 있게 하는 기술을 의미하는 용어.긴 코드를 짧게 작성하기 위한 문법.

#### Comprehension의 속도

end time=time.time()

print("생성 시간:",end time-start time)

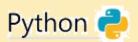
\*백만개의 숫자를 리스트로 생성하기



```
import time
    li=[]
    start time=time.time()
    for i in range(1000000):
        li.append(i)
                                             0.26357102394104004
    end time=time.time()
    print("생성 시간:",end time-start time)
코드
    *백만개의 숫자를 리스트로 생성하기(comprehension 사용)
    import time
    li=[]
                                                        0.11393332481384277
    start time=time.time()
    li=[i for i in range(1000000)]
```

리스트 컴프리헨션

#### 리스트 Comprehension



```
# 기존 리스트 값에 2를 곱한 값을 새로운 리스트로 생성

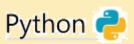
nums=[1,3,5]

new_nums=[]
for num in nums:
 new_nums.append(num*2)

print(new nums)
```

```
# Comprehension 사용
nums=[1,3,5]
new_nums=[num*2 for num in nums]
print(new_nums)
```

#### 리스트 Comprehension - 조건 지정

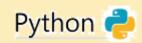


```
# 多个만 2를 곱한 결과 표시
nums=[]
for num in [1,2,3,4,5]:
    if num%2==1:
        nums.append(num*2)
print(nums)
```

# Comprehension 사용
nums=[num\*2 for num in [1,2,3,4,5] if num%2==1]
print(new nums)

```
# Comprehension 사용
nums=[num*2 if num%2==1 else num for num in [1,2,3,4,5]]
print(nums)
```

#### 리스트 Comprehension



```
# 리스트 a에서 1을 모두 삭제

nums=[1,2,1,1,3,1]

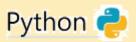
while 1 in nums:

nums.remove(1)

print(nums)
```

```
# Comprehension 사용
nums=[1,2,1,1,3,1]
nums=[num for num in nums if num!=1]
print(nums)
```

#### 리스트 Comprehension



```
코드
```

```
mans=["영희","철수"]
cards=["가위","바위","보"]
datas=[]
for man in mans:
    for card in cards:
        datas.append((man,card))
print(datas)
```

```
# Comprehension 사용
mans=["영희","철수"]
cards=["가위","바위","보"]
datas=[(man,card) for man in mans for card in cards]
print(datas)
```

## 도전!

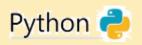
[문제] 아래 리스트에서 이름의 길이가 5자 이상인 것만 새로운 리스트로 생성하기

names=["Elsa", "Alex", "Dave", "Maria", "Anna",

"Elizabeth"]

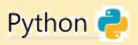
zip() 팩킹과 언팩킹

## zip() 함수



- 전달된 반복 가능한 객체의 각 요소를 튜플로 묶어서 반환하는 함수.
- 길이가 다르면 짧은 객체를 기준으로 생성됨.
- 사용법: zip(seq1, seq2,...)

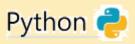
## zip() 함수



names=["코스모스", "총균쇠", "연금술사"] prices=[100,150,90] # packing for book in zip(names, prices): ("코스모스", 100) print(book) ("총균쇠", 150) ("연금술사", 90) # unpacking for name, price in zip(names, prices): print(f"{name}의 가격은 {price}입니다.")

> 코스모스의 가격은 100입니다. 총균쇠의 가격은 150입니다. 연금술사의 가격은 90입니다.

## zip() 함수



```
코드
```

```
# 두 리스트의 합계
li1=[1,2,3]
li2=[5,6,7]
for i,j in zip(li1,li2):
print(i+j)
6
8
10
```

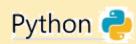
코드

```
# Comprehension 사용
li1=[1,2,3]
li2=[5,6,7]
li3=[i+j for i,j in zip(li1,li2)]
print(li3)
```

[6,8,10]

딕셔너리 컴프리헨션

### 딕셔너리 Comprehension



```
food_key=["라면","피자","맥주","치킨","삼겹살"]
food_value=["김치","피클","땅콩","치킨무","상추"]
food={}
for i in range(len(food_value)):
    food[food_key[i]]=food_value[i]

print(food)
{"라면": "김치", "피자": "피클", "맥주": "땅콩", "치킨": "치킨무", "삼겹살": "상추"}
```

```
# Comprehension 사용
food_key=["라면","피자","맥주","치킨","삼겹살"]
food_value=["김치","피클","땅콩","치킨무","상추"]
food={k:v for (k,v) in zip(food_key,food_value)}
print(food)
```