

교수의 파이썬

02_2 리스트 축약2

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 파이썬

02_2 리스트 축약2

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 고급 파이썬

02_2 리스트 축약2

창원대학교 정보통신공학과 교수 박동규

넌넌한 교수의 고급 파이썬

02_2 리스트 축약2

창원대학교 정보통신공학과 교수 박동규

if 조건식을 이용한 필터링

`list()` 함수, `filter()` 함수, 람다 함수를 이용한 필터링

리스트 축약표현을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x>=19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x>=19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```


if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```


if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```



if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```

반복문을 통해 할당된
각각의 x에 대하여
if x >= 19라는
조건식 필터 적용

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```

반복문을 통해 할당된
각각의 x에 대하여
if x >= 19라는
조건식 필터 적용

실행 결과

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```

반복문을 통해 할당된
각각의 x에 대하여
if x >= 19라는
조건식 필터 적용

실행 결과

```
성년 리스트 : [34, 39, 20, 54]
```

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
adult_ages = list(filter(lambda x: x >= 19, ages))
print('성년 리스트 :', adult_ages)
```

리스트 축약표현을 이용한 필터링

```
ages = [34, 39, 20, 18, 13, 54]
print('성년 리스트 :', [x for x in ages if x >= 19])
```

반복문을 통해 할당된
각각의 x에 대하여
if x >= 19라는
조건식 필터 적용

실행 결과

```
성년 리스트 : [34, 39, 20, 54] # 19 이상인 값만 반환됨
```

if 조건식을 이용한 필터링

`list()` 함수, `filter()` 함수, 람다 함수를 이용한 음수 필터링

리스트 축약 표현과 if 조건식을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = [x for x in n_list if x < 0]
print('음수 리스트 :', minus_list)
```

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = [x for x in n_list if x < 0]
print('음수 리스트 :', minus_list)
```

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = [x for x in n_list if x < 0]
print('음수 리스트 :', minus_list)
```

실행 결과

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = [x for x in n_list if x < 0]
print('음수 리스트 :', minus_list)
```

실행 결과

```
음수 리스트 : [-30, -5, -90, -36]
```

if 조건식을 이용한 필터링

list() 함수, filter() 함수, 람다 함수를 이용한 음수 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = list(filter(lambda x: x < 0, n_list))
print('음수 리스트 :', minus_list)
```

리스트 축약 표현과 if 조건식을 이용한 필터링

```
n_list = [-30, 45, -5, -90, 20, 53, 77, -36]
minus_list = [x for x in n_list if x < 0]
print('음수 리스트 :', minus_list)
```

실행 결과

```
음수 리스트 : [-30, -5, -90, -36] # 음수 값만을 가진 리스트가 출력됨
```

Lab

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]  
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
[>>> [x * x for x in range(10)]  
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]  
[>>> [x for x in range(10) if x % 2 == 0]  
[0, 2, 4, 6, 8]  
[>>> [x for x in range(10) if x % 2 == 1]  
[1, 3, 5, 7, 9]  
[>>> [x * x for x in range(10) if x % 2 == 0]  
[0, 4, 16, 36, 64]  
[>>> [x * x for x in range(10) if x % 2 == 1]  
[1, 9, 25, 49, 81]
```


리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]          # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1]
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]  # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1]
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)] # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0]
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1]
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]          # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]      # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1]
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]          # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]      # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1]
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]  # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수 값
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]  # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수 값
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]          # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]      # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수 값
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수의 제곱 값
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```


리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]  # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수 값
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수의 제곱 값
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1]
[1, 9, 25, 49, 81]
```

리스트의 축약 표현 실습

```
[>>> [x for x in range(10)]      # 0에서 9까지 숫자를 포함한 리스트
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[>>> [x * x for x in range(10)]  # 0에서 9까지 숫자의 제곱 값
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[>>> [x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수 값
[0, 2, 4, 6, 8]
[>>> [x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수 값
[1, 3, 5, 7, 9]
[>>> [x * x for x in range(10) if x % 2 == 0] # 0에서 9까지 숫자 중 짝수의 제곱 값
[0, 4, 16, 36, 64]
[>>> [x * x for x in range(10) if x % 2 == 1] # 0에서 9까지 숫자 중 홀수의 제곱 값
[1, 9, 25, 49, 81]
```

리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

리스트 축약을 이용한 두 리스트의 곱하기 기능

리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

```
product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]  
print(product_xy)
```


리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

```
product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]  
print(product_xy)
```



리스트의 축약 표현을 이용한 이중 for 루프 구현


두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

```
product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]  
print(product_xy)
```

한줄의 코딩으로
표현 가능



리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

```
product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]  
print(product_xy)
```

한줄의 코딩으로
표현 가능

실행 결과

리스트의 축약 표현을 이용한 이중 for 루프 구현

두 리스트를 곱하여 새 리스트를 생성하는 코드

```
product_xy = []  
for x in [1, 2, 3]: # 이중 for 루프를 통해 두 리스트 원소의 곱을 모두 구함  
    for y in [2, 4, 6]:  
        product_xy.append(x * y)  
  
print(product_xy)
```

리스트 축약을 이용한 두 리스트의 곱하기 기능

```
product_xy = [x * y for x in [1, 2, 3] for y in [2, 4, 6]]  
print(product_xy)
```

한줄의 코딩으로
표현 가능

실행 결과

```
[2, 4, 6, 4, 8, 12, 6, 12, 18]
```

Lab

리스트의 축약 표현 실습

리스트의 축약 표현을 사용한 2와 3의 배수 구하기

리스트의 축약 표현을 사용한 2와 3과 5의 배수 구하기

리스트의 축약 표현 실습

리스트의 축약 표현을 사용한 2와 3의 배수 구하기

```
>>> [n for n in range(1,31) if n % 2 == 0 if n % 3 ==0]  
[6, 12, 18, 24, 30]
```

리스트의 축약 표현을 사용한 2와 3과 5의 배수 구하기

리스트의 축약 표현 실습

리스트의 축약 표현을 사용한 2와 3의 배수 구하기

```
>>> [n for n in range(1,31) if n % 2 == 0 if n % 3 ==0]  
[6, 12, 18, 24, 30] # 1에서 30까지의 수 중에서 2와 3의 배수가 출력됨
```

리스트의 축약 표현을 사용한 2와 3과 5의 배수 구하기

리스트의 축약 표현 실습

리스트의 축약 표현을 사용한 2와 3의 배수 구하기

```
[>>> [n for n in range(1,31) if n % 2 == 0 if n % 3 == 0]  
[6, 12, 18, 24, 30] # 1에서 30까지의 수 중에서 2와 3의 배수가 출력됨
```

리스트의 축약 표현을 사용한 2와 3과 5의 배수 구하기

```
[>>> [n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]  
[30]
```

리스트의 축약 표현 실습

리스트의 축약 표현을 사용한 2와 3의 배수 구하기

```
[>>> [n for n in range(1,31) if n % 2 == 0 if n % 3 == 0]  
[6, 12, 18, 24, 30] # 1에서 30까지의 수 중에서 2와 3의 배수가 출력됨
```

리스트의 축약 표현을 사용한 2와 3과 5의 배수 구하기

```
[>>> [n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]  
[30]_ # 1에서 30까지의 수 중에서 2와 3과 5의 배수가 출력됨
```

리스트의 축약 표현과 가독성

리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나
지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면
이해하기가 매우 어려워진다.

리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나
지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면
이해하기가 매우 어려워진다.

줄바꿈 **전**

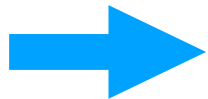
```
[n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```

리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나
지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면
이해하기가 매우 어려워진다.

줄바꿈 전

```
[n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```



리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나
지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면
이해하기가 매우 어려워진다.

줄바꿈 전

```
[n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```


➡ 이러한 문제점은 다음과 같이 줄바꿈을 통해서 가독성을 증가시킬 수 있다.

리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나
지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면
이해하기가 매우 어려워진다.

줄바꿈 전

```
[n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```

 이러한 문제점은 다음과 같이 줄바꿈을 통해서 가독성을 증가시킬 수 있다.

줄바꿈 후

```
[n for n in range(1, 31)
if n % 2 == 0
if n % 3 == 0
if n % 5 == 0]
```

리스트의 축약 표현과 가독성

리스트의 축약 표현이 많은 장점을 가지고 있으나 지나치게 많은 조건을 넣거나 표현식을 복잡하게 만들면 이해하기가 매우 어려워진다.

줄바꿈 전

```
[n for n in range(1, 31) if n % 2 == 0 if n % 3 == 0 if n % 5 == 0]
```

➡ 이러한 문제점은 다음과 같이 줄바꿈을 통해서 가독성을 증가시킬 수 있다.

줄바꿈 후

```
[n for n in range(1, 31)
if n % 2 == 0
if n % 3 == 0
if n % 5 == 0]
```

너무 많은 축약은 가독성을 해칠 수 있다는 점을 항상 이해하고 읽기 좋은 코드를 만들기 위해 노력을 기울여야 한다.

Lab

감사합니다.