

# 교수의 파이썬

03\_3 반복가능 객체를 위한 내장함수

창원대학교 정보통신공학과 교수 박동규

# 넌넌한 교수의 파이썬

03\_3 반복가능 객체를 위한 내장함수

창원대학교 정보통신공학과 교수 박동규

# 넌넌한 교수의 고급 파이썬

03\_3 반복가능 객체를 위한 내장함수

창원대학교 정보통신공학과 교수 박동규

# 넌넌한 교수의 고급 파이썬

03\_3 반복가능 객체를 위한 내장함수

창원대학교 정보통신공학과 교수 박동규

# 반복가능 객체를 위한 내장함수

# 반복가능 객체를 위한 내장함수

- 반복가능 객체는 다양한 파이썬 내장함수를 사용할 수 있다
- `min()` 이나 `max()`와 같은 함수는 반복가능 객체를 인자로 받아 최솟값과 최댓값을 반환한다.
- 이 외에도 `all()`, `any()`, `ascii()`, `bool()`, `filter()`, `iter()`와 같은 고급 내장함수도 제공되고 있다.



`all()` 함수



## all() 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]
[>>> l2 = [0, 2, 4, 8]
[>>> l3 = [0, 0, 0, 0]
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
[>>> l2 = [0, 2, 4, 8]
[>>> l3 = [0, 0, 0, 0]
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]      # 모든 요소가 0이 아님
[>>> l2 = [0, 2, 4, 8]      # 한 요소가 0임
[>>> l3 = [0, 0, 0, 0]
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
[>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
[>>> l3 = [0, 0, 0, 0]    # 모든 요소가 0임
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
[>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
[>>> l3 = [0, 0, 0, 0]    # 모든 요소가 0임
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

## all( ) 함수

반복 가능한 항목들이 모두 참일 때만, 참을 반환한다.

```
[>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
[>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
[>>> l3 = [0, 0, 0, 0]    # 모든 요소가 0임
[>>> all(l1)
True
[>>> all(l2)
False
[>>> all(l3)
False
```

C언어와 마찬가지로  
0이면 거짓, 0이 아니면 참





`any( )` 함수

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

```
>>> l1 = [1, 2, 3, 4]
>>> l2 = [0, 2, 4, 8]
>>> l3 = [0, 0, 0, 0]
>>> any(l1)
True
>>> any(l2)
True
>>> any(l3)
False
```

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

```
>>> l1 = [1, 2, 3, 4] # 모든 요소가 0이 아님
>>> l2 = [0, 2, 4, 8]
>>> l3 = [0, 0, 0, 0]
>>> any(l1)
True
>>> any(l2)
True
>>> any(l3)
False
```

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

```
>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
>>> l3 = [0, 0, 0, 0]
>>> any(l1)
True
>>> any(l2)
True
>>> any(l3)
False
```

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

```
>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
>>> l3 = [0, 0, 0, 0]    # 모든 요소가 0임
>>> any(l1)
True
>>> any(l2)
True
>>> any(l3)
False
```

## any( ) 함수

임의의 반복 가능한 항목들 중에서  
참이 하나라도 있을 경우 참을 반환한다.

```
>>> l1 = [1, 2, 3, 4]    # 모든 요소가 0이 아님
>>> l2 = [0, 2, 4, 8]    # 한 요소가 0임
>>> l3 = [0, 0, 0, 0]    # 모든 요소가 0임
>>> any(l1)
True
>>> any(l2)
True
>>> any(l3)
False
```





bool( ) 함수

## bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
[>>> l2 = [0, 2, 4, 8]
[>>> l3 = [0, 0, 0, 0]
[>>> bool(l1)
True
[>>> bool(l2)
True
[>>> bool(l3)
True
[>>> l4 = []
[>>> bool(l4)
False
```

# bool( ) 함수

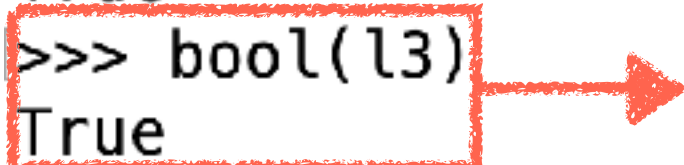
반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
[>>> l2 = [0, 2, 4, 8]
[>>> l3 = [0, 0, 0, 0]
[>>> bool(l1)
True
[>>> bool(l2)
True
[>>> bool(l3)
True
[>>> l4 = []
[>>> bool(l4)
False
```

# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
[>>> l2 = [0, 2, 4, 8]
[>>> l3 = [0, 0, 0, 0]
[>>> bool(l1)
True
[>>> bool(l2)
True
[>>> bool(l3)
True
[>>> l4 = []
[>>> bool(l4)
False
```



# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
```

```
[>>> l2 = [0, 2, 4, 8]
```

```
[>>> l3 = [0, 0, 0, 0]
```

```
[>>> bool(l1)
```

```
True
```

```
[>>> bool(l2)
```

```
True
```

```
[>>> bool(l3)
```

```
True
```

```
[>>> l4 = []
```

```
[>>> bool(l4)
```

```
False
```

요소가 0, 0, 0, 0 이지만  
0이라는 요소가 존재하므로  
True 반환

# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
>>> l2 = [0, 2, 4, 8]
>>> l3 = [0, 0, 0, 0]
>>> bool(l1)
```

True

```
[>>> bool(l2)
```

True

```
[>>> bool(l3)
```

True

```
[>>> l4 = []
```

```
[>>> bool(l4)
```

False

요소가 0, 0, 0, 0 이지만  
0이라는 요소가 존재하므로  
True 반환

# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
```

```
[>>> l2 = [0, 2, 4, 8]
```

```
[>>> l3 = [0, 0, 0, 0]
```

```
[>>> bool(l1)
```

```
True
```

```
[>>> bool(l2)
```

```
True
```

```
[>>> bool(l3)
```

```
True
```

```
[>>> l4 = []
```

```
[>>> bool(l4)
```

```
False
```

요소가 0, 0, 0, 0 이지만  
0이라는 요소가 존재하므로  
True 반환



# bool( ) 함수

반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
```

```
[>>> l2 = [0, 2, 4, 8]
```

```
[>>> l3 = [0, 0, 0, 0]
```

```
[>>> bool(l1)
```

```
True
```

```
[>>> bool(l2)
```

```
True
```

```
[>>> bool(l3)
```

```
True
```

```
[>>> l4 = []
```

```
[>>> bool(l4)
```

```
False
```

요소가 0, 0, 0, 0 이지만  
0이라는 요소가 존재하므로  
True 반환

원소가 전혀 없는 리스트이기 때문에  
False 반환

# bool( ) 함수

전달된 조건이나 숫자를 기반으로 True/False를 반환  
반복가능 객체의 값의 존재 여부를 부울 값으로 변환한다.  
즉, 아래와 같이 리스트의 항목 유무를 True와 False로 알려준다.

```
[>>> l1 = [1, 2, 3, 4]
```

```
[>>> l2 = [0, 2, 4, 8]
```

```
[>>> l3 = [0, 0, 0, 0]
```

```
[>>> bool(l1)
```

```
True
```

```
[>>> bool(l2)
```

```
True
```

```
[>>> bool(l3)
```

```
True
```

```
[>>> l4 = []
```

```
[>>> bool(l4)
```

```
False
```

요소가 0, 0, 0, 0 이지만  
0이라는 요소가 존재하므로  
True 반환

원소가 전혀 없는 리스트이기 때문에  
False 반환

**Lab**

# 반복가능 객체를 위한 내장함수

# 반복가능 객체를 위한 내장함수

`filter()` 함수

# 반복가능 객체를 위한 내장함수

## `filter()` 함수

반복가능한 항목들을 하나하나 꺼내어 함수에 넣는다.  
그리고 그 리턴 값이 참인 것만 묶어서 반환하는 함수이다.

일반적으로 람다 함수를 필터링시 사용함

# 리스트와 문자열

# 리스트와 문자열

list() 함수



# 리스트와 문자열

`list()` 함수

내장함수로서 문자열이나 튜플을 리스트로 변환할 때 사용한다.

# 리스트와 문자열

## list() 함수

내장함수로서 문자열이나 튜플을 리스트로 변환할 때 사용한다.

```
>>> char_list = list('hello')  
>>> char_list  
['h', 'e', 'l', 'l', 'o']
```

# 리스트와 문자열

## list() 함수

내장함수로서 문자열이나 튜플을 리스트로 변환할 때 사용한다.

```
>>> char_list = list('hello')  
>>> char_list  
['h', 'e', 'l', 'l', 'o']
```

```
>>> s = {1, 2, 3, 4}
>>> type(s)
<class 'set'>
>>> s_list = list(s)
>>> s_list
[1, 2, 3, 4]
```

```
>>> t_list = list((1, 2, 3, 4))
>>> t_list
[1, 2, 3, 4]
>>> {1, 2, 3, 4}
```

```
>>> t_list = list(1, 2, 3, 4)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    t_list = list(1, 2, 3, 4)
TypeError: list expected at most 1 argument
```

```
>>> s = {1, 2, 3, 4}
>>> type(s)
<class 'set'>
>>> s_list = list(s)
>>> s_list
[1, 2, 3, 4]
```

집합으로부터 리스트를 만들 수 있다

```
>>> t_list = list((1, 2, 3, 4))
>>> t_list
[1, 2, 3, 4]
>>> {1, 2, 3, 4}
```

```
>>> t_list = list(1, 2, 3, 4)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    t_list = list(1, 2, 3, 4)
TypeError: list expected at most 1 argument
```

```
>>> s = {1, 2, 3, 4}
>>> type(s)
<class 'set'>
>>> s_list = list(s)
>>> s_list
[1, 2, 3, 4]
```

집합으로부터 리스트를 만들 수 있다

```
>>> t_list = list((1, 2, 3, 4))
>>> t_list
[1, 2, 3, 4]
>>> {1, 2, 3, 4}
```

튜플로부터 리스트를 만들 수 있다  
( )에 주의

```
>>> t_list = list(1, 2, 3, 4)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    t_list = list(1, 2, 3, 4)
TypeError: list expected at most 1 argument
```

# 리스트와 문자열

# 리스트와 문자열

split() 메소드



# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.  
그래서 공백을 구분자로 하여 list로 만들어준다.

# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.

그래서 공백을 구분자로 하여 list로 만들어준다.

```
[>>> words = 'Python은 아름다운 언어입니다 .'  
>>> words_list = words.split()  
>>> words_list  
['Python은 ', '아름다운 ', '언어입니다 .']
```

# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.

그래서 공백을 구분자로 하여 list로 만들어준다.

```
[>>> words = 'Python은 아름다운 언어입니다.'
>>> words_list = words.split()
>>> words_list
['Python은 ', '아름다운 ', '언어입니다.']
>>> time_str = '2019.02.20'
>>> time_str.split('.')
['2019', '02', '20']
```

# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.

그래서 공백을 구분자로 하여 list로 만들어준다.

```
[>>> words = 'Python은 아름다운 언어입니다.'
>>> words_list = words.split()
>>> words_list
['Python은 ', '아름다운 ', '언어입니다.']

>>> time_str = '2019.02.20'
>>> time_str.split('.')
['2019', '02', '20']
```


# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.

그래서 공백을 구분자로 하여 list로 만들어준다.

```
[>>> words = 'Python은 아름다운 언어입니다 .'  
>>> words_list = words.split()  
>>> words_list  
['Python은 ', '아름다운 ', '언어입니다 .']  
  
>>> time_str = '2019.02.20'  
>>> time_str.split('.')  
['2019', '02', '20']
```



# 리스트와 문자열

## split() 메소드

디폴트 구분자로 공백을 사용한다.

그래서 공백을 구분자로 하여 list로 만들어준다.

```
[>>> words = 'Python은 아름다운 언어입니다 .'  
>>> words_list = words.split()  
>>> words_list  
['Python은 ', '아름다운 ', '언어입니다 .']  
  
>>> time_str = '2019.02.20'  
>>> time_str.split('.')  
['2019', '02', '20']
```

구분자로 다른 문자를  
사용할 수 있다.

# 리스트와 문자열

# 리스트와 문자열

join() 메소드



# 리스트와 문자열

## join() 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
>>> time_list
['2019', '02', '21']
>>> '.'.join(time_list)
'2019.02.21'
>>> ','.join(time_list)
'2019,02,21'
```

# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

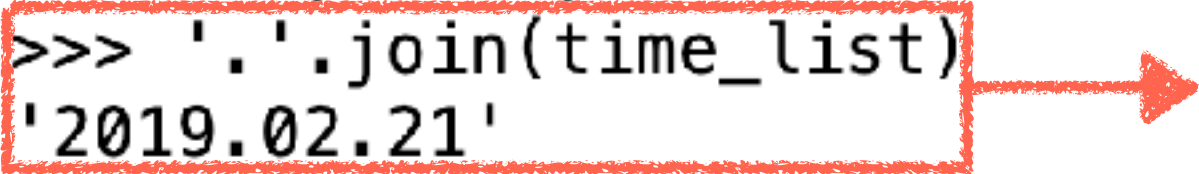
```
>>> time_list = ['2019', '02', '21']  
>>> time_list  
['2019', '02', '21']  
>>> '.'.join(time_list)  
'2019.02.21'  
>>> ','.join(time_list)  
'2019,02,21'
```

# 리스트와 문자열

## join() 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']  
>>> time_list  
['2019', '02', '21']  
>>> '.'.join(time_list)  
'2019.02.21'  
>>> ','.join(time_list)  
'2019,02,21'
```



# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
```

```
>>> time_list
```

```
['2019', '02', '21']
```

```
>>> '.'.join(time_list)
```

```
'2019.02.21'
```

time\_list 내의 인자들을  
'.'으로 연결해준다.

```
>>> ','.join(time_list)
```

```
'2019,02,21'
```

# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
```

```
>>> time_list
```

```
['2019', '02', '21']
```

```
>>> '.'.join(time_list)  
'2019.02.21'
```

```
>>> ', '.join(time_list)  
'2019, 02, 21'
```

time\_list 내의 인자들을  
'.'으로 연결해준다.

# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
```

```
>>> time_list
```

```
['2019', '02', '21']
```

```
>>> '.'.join(time_list)
```

```
'2019.02.21'
```

```
>>> ','.join(time_list)
```

```
'2019,02,21'
```

time\_list 내의 인자들을  
'.'으로 연결해준다.

# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
```

```
>>> time_list
```

```
['2019', '02', '21']
```

```
>>> '.'.join(time_list)  
'2019.02.21'
```

time\_list 내의 인자들을  
'.'으로 연결해준다.

```
>>> ','.join(time_list)  
'2019,02,21'
```

time\_list 내의 인자들을  
','으로 연결해준다.



# 리스트와 문자열

## join( ) 메소드

리스트의 항목들을 하나의 문자열로 연결할 때 사용

```
>>> time_list = ['2019', '02', '21']
```

```
>>> time_list
```

```
['2019', '02', '21']
```

```
>>> '.'.join(time_list)
```

```
'2019.02.21'
```

```
>>> ','.join(time_list)
```

```
'2019,02,21'
```

time\_list 내의 인자들을  
'.'으로 연결해준다.

time\_list 내의 인자들을  
','으로 연결해준다.

이와같이 `split( )` 메소드와 `join( )` 메소드를 이용하여  
문자열과 리스트를 구분하거나 연결하는 것도 가능하다.

**Lab**

# 정리

- 반복가능 자료형에는 `any()`, `all()`, `bool()`, `iter()`, `filter()`, `map()` 등의 내장함수를 사용하여 필요한 연산을 할 수 있다.
- `list()` 함수를 사용하여 집합, 튜플, 문자열을 리스트 객체로 만들 수 있다
- `join()`, `split()`과 같은 많은 메소드가 있다



감사합니다.