

# 2018학년도 2학기 언어와 컴퓨터

## 제4강 반복 가능한 자료형

박수지

서울대학교 인문대학 언어학과

2018년 9월 12일 수요일

## 오늘의 목표

- 1 객체의 구성 요소를 설명할 수 있다.
- 2 동일성과 동등성을 구별할 수 있다.
- 3 반복 가능한 자료형으로서 문자열, 리스트, 튜플의 공통점과 차이점을 말할 수 있다.

## 의문

0과 0.0은 같은가?

## 관찰

- 값은 같다.
- 그런데 뭔가 다르다.

## 확인하기

```
>>> zero_int = 0
>>> zero_float = 0.0
>>> zero_int == zero_float
True
>>> zero_int is zero_float
False
```

**==** 동등성 (equality) 비교 연산자

**is** 동일성 (identity) 비교 연산자

## 같은 값을 가진 다른 객체

```
>>> a = '파이썬'
>>> b = '파이썬'
>>> a == b
True
>>> a is b
False
>>> id(a)
4339513456
>>> id(b)
4339513552
```

## 파이썬에서의 객체

데이터를 추상적으로 나타낸 것

⇒ 거의 모든 것이 객체다.

## 객체가 가지는 것

- 식별성 (identity)
- 유형 (type)
- 값 (value)

`id()` 객체의 식별성을 돌려주는 함수

# 문자열: 문자의 열

## 문자열과 문자의 관계

`len()` 길이  
`str.find()` 위치  
`str.count()` 출현 횟수  
`in` 연산자 소속 여부

## 예시

```
>>> len('Python')
6
>>> 'Python'.find('p')
-1
>>> 'Python'.count('p')
0
>>> 'p' in 'Python'
False
```

# 문자열: 단어의 목록

## 예시

```
>>> words = '달아 달아 밝은 달아'.split()
>>> words
['달아', '달아', '밝은', '달아']
>>> words[2]
'밝은'
>>> words.index('밝은')
2
>>> words.count('달아')
3
>>> '달' not in words
True
```

# 리스트

내장된 가변열 (Built-in mutable sequence)

<https://docs.python.org/ko/3/library/stdtypes.html#list>

형식

[항목1, 항목2, ....]

형변환 예시

```
>>> list()
[]
>>> list('Python')
['P', 'y', 't', 'h', 'o', 'n']
>>> list(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

# 열 (sequence) 유형

## 공통 연산

- 소속 (in), 연결 (+), 반복 (\*), 비교 (<)
- 인덱싱, 슬라이싱
- len(), min(), max() 함수
- s.index(), s.count() 메소드

## 해당 자료형

- str
- list
- tuple
- range

## 주의

비교 연산자와 min(), max() 함수는 특정 조건하에서만 사용할 수 있다.



# 가변열

## 리스트 연산 확인하기

```
>>> dir(list)
[...., 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

# 리스트 메소드

## 늘리기

### 추가

```
>>> basket = ['egg', 'milk']  
>>> basket.append('cheese')  
>>> basket  
['egg', 'milk', 'cheese']
```

### 확장

```
>>> basket = ['egg', 'milk']  
>>> basket.extend(['cheese'])  
>>> basket  
['egg', 'milk', 'cheese']
```

`list.append()` 리스트 뒤에 항목을 추가하는 메소드

`list.extend()` 리스트 뒤에 다른 열을 붙이는 메소드

## 연결 및 할당 연산자를 사용한 확장

```
>>> basket = ['egg']  
>>> basket = basket + ['milk', 'cheese']  
>>> basket  
['egg', 'milk', 'cheese']
```

## 복합 할당 연산자

`basket = basket + ['milk', 'cheese']`를 줄여서  
`basket += ['milk', 'cheese']`로 쓸 수 있다.

# 리스트 메소드

## 줄이기

### 끝에서 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> fi.pop()
5
>>> fi
[1, 1, 2, 3]
```

### 값으로 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> fi.remove(1)
>>> fi
[1, 2, 3, 5]
```

## del과 인덱스로 항목 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> del fi[3]
>>> fi
[1, 1, 2, 5]
```

## del과 슬라이스로 항목 제거

```
>>> fi = [1, 1, 2, 3, 5]
>>> del fi[1::2]
>>> fi
[1, 2, 5]
```

# 리스트 메소드

## 순서 바꾸기

### 정렬

```
>>> py = list('Python')
>>> py.sort()
>>> py
['P', 'h', 'n', 'o', 't', 'y']
```

### 뒤집기

```
>>> py = list('Python')
>>> py.reverse()
>>> py
['n', 'o', 'h', 't', 'y', 'P']
```

# 파괴적 함수 對 비파괴적 함수

원본에 영향을 주는가?

`list.sort()` 메소드:  
파괴적

```
>>> pi = [3, 1, 4, 1, 5]
>>> pi.sort()
>>> pi
[1, 1, 3, 4, 5]
```

`sorted()` 함수:  
비파괴적

```
>>> pi = [3, 1, 4, 1, 5]
>>> sorted(pi)
[1, 1, 3, 4, 5]
>>> pi
[3, 1, 4, 1, 5]
```

# 튜플

## 내장된 불변열 (Built-in immutable sequence)

<https://docs.python.org/ko/3/library/stdtypes.html#tuple>

### 형식

(항목1, 항목2, ....)

### 괄호 생략하기

```
>>> a = 1, 2
>>> a
(1, 2)
>>> type(a)
<class 'tuple'>
```

### 메소드 탐색

```
>>> dir(tuple)
[...., 'count', 'index']
```



# 가변 vs. 불변

## 리스트: 가변열

```
>>> basket = ['bacon', 'egg', 'milk']  
>>> basket[0] = 'ham'  
>>> basket  
['ham', 'egg', 'milk']
```

# 가변 vs. 불변

## 튜플: 불변열

```
>>> food = ('milk', 200, 800)
>>> food[1] = 500
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

# 리스트 복사

## 같은 객체의 다른 이름

```
>>> basket1 = ['bacon', 'milk', ['cheese', 'cheese']]
>>> basket2 = basket1
>>> basket1 is basket2
True

>>> basket2[0] = 'yoghurt'
>>> basket1
['yoghurt', 'milk', ['cheese', 'cheese']]
```

Python Tutor(<http://pythontutor.com>)에서 시각화해 보기

# 리스트 복사

## 값을 복사한 새로운 개체

```
>>> basket1 = ['bacon', 'milk', ['cheese', 'cheese']]
>>> basket2 = basket1.copy()
>>> basket1 is basket2
False

>>> basket2[0] = 'yoghurt'
>>> basket1 # 0번째 항목은 바뀌지 않는데
['bacon', 'milk', ['cheese', 'cheese']]
>>> basket2[2].pop()
'cheese'
>>> basket1 # 2번째 항목은 바뀜
['bacon', 'milk', ['cheese']]
```

# 리스트 복사

## 중첩된 리스트 안의 값까지 복사한 새로운 개체

```
>>> from copy import deepcopy
>>> basket1 = ['bacon', 'milk', ['cheese', 'cheese']]
>>> basket2 = deepcopy(basket1)
>>> basket1 is basket2
False

>>> basket2[0] = 'yoghurt'
>>> basket1 # 0번째 항목도 바뀌지 않고
['bacon', 'milk', ['cheese', 'cheese']]
>>> basket2[2].pop()
'cheese'
>>> basket1 # 2번째 항목도 바뀌지 않음
['bacon', 'milk', ['cheese', 'cheese']]
```

# 범위

## 형식

`range(start, end, step)`

- 슬라이스의 `[start:end:step]`과 사용법이 거의 같음

## 예시

```
>>> list(range(5)) # 기본적으로 0부터 시작함
[0, 1, 2, 3, 4]
>>> list(range(-5, 5))
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
>>> list(range(10, 0, -2))
[10, 8, 6, 4, 2]
```

# 딕셔너리

## 형식

```
{키1: 값1, 키2: 값2, ....}
```

- 인덱스 대신 키를 사용하여 값을 찾는다.

## 예시

```
>>> numbers = {'one': 1, 'two': 2, 'three': 3}
>>> numbers['two']
2
>>> numbers[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 3
```

# 딕셔너리 연산

## 늘리기

### 예시

```
>>> hansol = {'HP': 80}
>>> hansol['MP'] = 90 # 새로운 키와 값 할당
>>> hansol
{'HP': 80, 'MP': 90}
>>> hansol.update({'class': 'mage'})
>>> hansol
{'HP': 80, 'MP': 90, 'class': 'mage'}
```

>>> `dir(dict)`로 메소드를 더 확인해 보시다.



## count\_char.py

```
1 # -*- coding: utf-8 -*-  
2 # 문장 내에서 특정 문자의 개수를 세는 프로그램  
3 sentence = input('문장을 입력하세요: ')  
4 char = input('찾고 싶은 문자를 입력하세요: ')  
5 n = sentence.lower().count(char.lower())  
6 print('{}의 출현 횟수: {}'.format(char, n))
```

## sort\_words.py

```
1 # -*- coding: utf-8 -*-  
2 # 문장 내 단어를 가나다순으로 정렬하는 프로그램  
3 sentence = input('문장을 입력하세요: ')  
4 words = sentence.split()  
5 words.sort()  
6 print(words)
```

# 요약

## 객체

- 1 데이터의 추상적 표현
- 2 식별성, 유형, 값을 가짐

## 객체의 같음

동등성 값이 같음

■ == 연산자

동일성 식별성이 같음

■ is 연산자

## 반복 가능한 자료형

- 1 열
  - 가변: list
  - 불변: str, range
- 2 매핑(mapping)
  - dict

## 다음 시간 예고

조건문

## 더 생각할 문제

- 1 비교, `min()`, `max()`는 어떤 조건에서 사용 가능한가?

예시 `>>> min([1, '2'])`

- 2 문자열은 변경 가능한 자료형인가?

- 3 딕셔너리는 변경 가능한 자료형인가?

- 확인하는 방법

- 1 요소의 값을 새로 할당할 수 있는가?
- 2 메소드 중에서 파괴적 함수가 있는가?