CS101 – 자료 구조: 사전

Lecture 18

School of Computing
KAIST

학습 목표:

- 사전 (dictionary) 자료구조를 활용할 수 있다.
- 리스트, 집합, 사전의 장단점을 이해하여
  목적에 적합한 자료구조를 사용할 수 있다.

사전

사전(Dictionary) 또한 Python에서 유용하게 쓰일 수 있는 자료 구조입니다.

사전은 리스트나 집합처럼 여러 객체를 모아서 만든 객체입니다.
다른 자료 구조와의 가장 큰 차이점은, 다양한 종류의 불변 객체를 사용해서 자료에 접근할 수 있다는 점입니다 (예. 숫자, 문자열을 사용해서 자료에 접근이 가능).

사전에서 사용되는 인덱스는 **키(key)** 라고 부르고, 키가 가리키는 대상 객체는 **값(value)** 이라고 부릅니다. 키와 값의 쌍을 key-value pair라 합니다.

사전은 중괄호 `{}`나 `dict()` 함수를 이용해서 만들 수 있습니다.

```
majors = {"CS": "Computer Science",
          "EE": "Electrical Engineering",
          "MAS": "Mathematical Sciences",
          "ME": "Mechanical Engineering"}

d1 = dict() # an empty dictionary
d2 = {} # an empty dictionary
```

사전

사전의 원소들에는 순서가 없습니다.
사전의 원소에는 키로만 접근할 수 있습니다.

```
>>>majors[0]
KeyError: 0
```

다음과 같이 사전에 새로운 키와 값을 넣을 수 있습니다.

```
>>> majors["PH"] = "Physic"
>>> majors["PH"]
'Physic'
```

다음처럼 키가 가리키는 값을 바꿀 수도 있습니다.

```
>>> majors["PH"] = "Physics"
>>> majors["PH"]
'Physics'
```

사전 관련 함수들

사전 객체 d는 다음과 같은 함수들과 연산자들을 사용할 수 있습니다.

- `len(d)`: d의 원소의 수를 반환한다.
- `key in d`: d가 key를 가지고 있으면 **`True`**를, 아니면 **`False`**를 반환한다.
- `d.get(key, default=None)`: d에서 key가 가리키는 값을 반환한다. d가 key를 가지고 있지 않으면 `default` 값을 반환한다.
- `d.keys()`: d의 모든 key 객체 목록을 반환한다.
- `d.values()`: d의 모든 value 객체 목록을 반환한다.
- `d.items()`: d의 모든 key-value pair 목록을 반환한다.
- **`del`** `d[key]`: key에 해당하는 key-value pair를 사전 d에서 제거한다.

`keys()`, `values()`, `items()`로 반환되는 객체는 리스트 객체가 아닙니다.

이 객체들은 리스트처럼 원소들을 가지고 있지만, 객체를 변경할 수는 없습니다.
(예. `append()`와 같이 원소를 추가할 수 없습니다.)

사전 관련 함수들

다음은 함수들의 사용 예시입니다.

```
>>> majors[0]=0.001

>>> majors
{0: 0.001, 'CS': 'Computer Science', 'PH': 'Physics',
'ME': 'Mechanical Engineering', 'EE': 'Electrical Engineering',
'MAS': 'Mathematical Sciences'}
>>> len(majors)
6
>>> del majors[0]
>>> majors
{'CS': 'Computer Science', 'PH': 'Physics',
'ME': 'Mechanical Engineering', 'EE': 'Electrical Engineering',
'MAS': 'Mathematical Sciences'}
>>> len(majors)
5
>>> "CS" in majors
True
>>> "AI" in majors
False
```

사전 관련 함수들

다음은 함수들의 사용 예시입니다.

```
>>> majors.keys()
dict_keys(['CS', 'PH', 'ME', 'EE', 'MAS'])

>>> majors.values()
dict_values(['Computer Science', 'Physics',
'Mechanical Engineering', 'Electrical Engineering',
'Mathematical Sciences'])

>>> majors.items()
dict_items([('CS','Computer Science'), ('PH','Physics'),
('ME','Mechanical Engineering'), ('EE','Electrical Engineering'),
('MAS','Mathematical Sciences')])
```

사전과 반복문

반복문과 **in** 연산자를 사용하면, 사전의 키들에 해당하는 값들을 찾을 수 있습니다.

```
>>> for key in majors:
...     print("%s is %s." % (key, majors[key]))
CS is Computer Science.
PH is Physics.
ME is Mechanical Engineering.
EE is Electrical Engineering.
MAS is Mathematical Sciences.
```

반복문과 `items()` 함수를 사용하면 key-value pair들을 찾을 수 있습니다.

```
>>> for key, value in majors.items():
...     print("%s is %s." % (key, value))
CS is Computer Science.
PH is Physics.
ME is Mechanical Engineering.
EE is Electrical Engineering.
MAS is Mathematical Sciences.
```

리스트, 집합, 사전

언제 어떤 자료 구조를 사용하면 좋을까요?

- 순서가 있는 객체들을 다룰 때
  -> 리스트 자료 구조
- 순서가 없는 객체들을 다룰 때
  -> 집합 자료 구조
- 키를 통해 키가 가리키는 값을 쉽게 찾고 싶을 때
  -> 사전 자료 구조

리스트, 집합, 사전

KAIST

리스트보다 집합에서 원소의 포함 여부를 더 빠르게 계산할 수 있습니다.

```
import time
large_list = list(range(10000000))
large_set = set(large_list)

st = time.time()
for num in range(100000):
    if num not in large_list:
        print("What?!")
print("Running time for list: %f sec" % (time.time() - st))

st = time.time()
for num in range(100000):
    if num not in large_set:
        print("What?!")
print("Running time for set: %f sec" % (time.time() - st))
```

```
Result:
Running time for list: 78.066966 sec
Running time for set: 0.010978 sec
```

본 강의 학습 목표:

- 사전 (dictionary) 자료구조를 활용할 수 있다.
- 리스트, 집합, 사전의 장단점을 이해하여
  목적에 적합한 자료구조를 사용할 수 있다.

다음 강의 학습목표:

- 자료구조를 활용하여, 영상을 합성할 수 있다 (크로마키).
- 자료구조를 활용하여, 사진에 비밀정보를 숨길 수 있다.