Lecture #08 | 집합(set), 사전(dict)

SE213 프로그래밍 (2019)

Written by Mingyu Cho
Edited by Donghoon Shin

공지

- 2nd assignment
  - 4/24 ~ 5/7 (midnight)
  - 3 problems at dgist.elice.io

오늘 다룰 내용

- 파일 입출력 (바이너리 파일)
- 자료구조
  - 집합(`set`)
  - 사전(`dict`)

파일

- 파일
  - 문자, 숫자 등으로 이루어진 정보의 집합체
  - HDD, SSD, USB drive, 클라우드 등의 저장장치에 저장됨
  - 파일이 저장되어 있는 공간(파일 경로 혹은 path)과 이름(파일명)으로 구분됨
- 텍스트 파일: 여러 줄의 사람들이 인지할 수 있는 문자들로 이루어진 파일
  - 문자인코딩(예: ASCII, UTF-8, CP-949)에 따라 표현할 수 있는 문자가 달라짐
  - 참고: 파이썬 소스 파일(.py), html 파일 (.html) 등도 텍스트 파일의 일종
- 바이너리 파일: 텍스트 파일이 아닌 파일
  - 예: 아래아한글 문서파일 (.hwp), 워드파일 (.doc), 이미지파일 (.jpg, .png) 등

바이너리 파일입출력

- 파일 입출력 전후로 파일을 열고, 닫는 단계가 필요함 (컴퓨터로 문서를 편집하기 위해서 파일을 여는 것과 유사함)
  - 파일 열기 (open)
  - 파일 읽기 혹은 쓰기 (read or write)
  - 파일 닫기 (close): `with`를 사용한 경우 생략
- python에서 제일 간단한 바이너리 파일 입출력 방법 (입출력 모드 “b”)
  - 파일입력
    - `read()` 함수 이용
  - 파일출력
    - `write()` 함수 이용

바이너리 파일 읽기/쓰기

```
f = open('workfile', 'rb+')
print(f.write(b'0123456789abcdef') )
f.seek(5) # Go to the 6th byte in the file
print(f.read(1) )
f.seek(-3, 2) # Go to the 3rd byte before the end
print(f.read(1))
print(f.read())
f.close()
```

```
16
b'5‘
b'd'
b'ef'
```

집합(set)

- 집합: 순서가 없고 중복이 없는 원소들을 포함 (수학에서 집합과 유사)
  - 원소의 자료형: 불변하는(immutable) 형(부울, 정수,실수, 튜플, 문자열)만 가능
    - 일반적으로 문자열 혹은 정수가 많이 사용됨

- 정의
  - {}를 이용하고, 원소들을 ,로 구분하여 정의
  - 원소들을 포함하는 리스트, 튜플 등을 `set()` 함수의 인자로 하여 형변환
  - 단, 빈 집합(empty set)을 정의하고 싶을 때는 원소 없이 `set()` 함수만 사용함

- 출력: {}를 사용

예시: 집합의 생성

```
s1 = {42, 1024, 23}
print(s1)
s2 = set() # empty set
print(s2)
basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
print(basket)
str_set = set('abc')
print(str_set)
t1 = (42, 1024, 23)
tuple_set = set(t1)
print(tuple_set)
l1 = [42, 1024, 23, 1024, 23, 1]
list_set = set(l1)
print(list_set)
print(list(list_set), sorted(list(list_set)))
```

```
{1024, 42, 23}
set()
{'pear', 'banana', 'apple', 'orange'}
{'c', 'a', 'b'}
{1024, 42, 23}
{1024, 1, 42, 23}
[1024, 1, 42, 23] [1, 23, 42, 1024]
```

예시: 집합의 추가/삭제

```
s1 = set() # empty set
print(s1)
s1.add(6)
print(s1)
s1.add(23)
print(s1)
s1.add(6)
print(s1)
s1.update([6, 1024, 4096])
print(s1)
s1.remove(6) # KeyError if 6 not in s1
print(s1)
print(42 in s1)
print(len(s1)) # number of elements in s1
s1.clear()
print(s1)
```

```
set()
{6}
{6, 23}
{6, 23}
{1024, 4096, 6, 23}
{1024, 4096, 23}
False
3
set()
```

예시: 집합의 연산자 (1/2)

```
s1 = {6, 23}
s2 = {42, 1024, 23}
print(s1 | s2) # union
print(s1.union(s2))
print(s1 - s2) # difference
print(s1.difference(s2))
print(s2 - s1) # difference
print(s2.difference(s1))
print(s1 & s2) # intersection
print(s1.intersection(s2))
print(s1 ^ s2) # symmetric difference
print(s1.symmetric_difference(s2))
```

```
{1024, 6, 23, 42}
{1024, 6, 23, 42}
{6}
{6}
{1024, 42}
{1024, 42}
{23}
{23}
{1024, 42, 6}
{1024, 42, 6}
```

예시: 집합의 연산자 (2/2)

```
s1 = {6, 23}
s2 = {42, 1024, 23}
print(s1.isdisjoint(s2)) # disjoint
print(s1 == s2)
print(s1 <= s2) # subset
print(s1.issubset(s2))
s1.remove(6)
print(s1 <= s2)
print(s1 < s2)
print(s1 >= s2) # superset
print(s1.issuperset(s2))
print(s1 > s2)
s3 = (s1|s2).copy()
print(s3)
```

```
False
False
False
False
True
True
False
False
False
{1024, 42, 23}
```

set comprehension

- 집합 내포

```
s = {x for x in 'abracadabra' if x not in 'abc'}
print(s)
```

```
{'d', 'r'}
```

사전(dict)

- 리스트, 튜플
  - 여러 아이템을 하나의 변수 이름에 저장
  - 숫자를 이용한 인덱스로 아이템에 접근 가능
- 사전(dict): '키(key)'와 '값(value)'의 쌍들로 구성된 mutable mapping
  - 키
    - 불변하는 타입(부울, 정수,실수, 튜플, 문자열)을 사용
    - 일반적으로 문자열 혹은 정수가 많이 사용됨
  - 값: 파이썬의 어떠한 데이터 형태(리스트, 튜플, 사전 등도 포함)도 가능
- 참고: 다른 언어에서는 associative memory, associative array, hash, hashmap 등으로 불림

예시: 사전의 정의와 사용

```
bts_position = {
    '랩몬스터': '리더, 메인래퍼',
    '진': '서브보컬',
    '슈가': '리드래퍼',
    '제이홉': '서브래퍼, 메인댄서',
    '지민': '리드보컬, 리드댄서',
    '뷔': '서브보컬',
    '정국': '메인보컬, 서브래퍼, 리드댄서'
}

print(bts_position)
print(bts_position['뷔'])
print(bts_position['정국'])
```

```
{'랩몬스터': '리더, 메인래퍼', '진': '서브보컬', '슈가': '리드래퍼', '제이홉': '서브래퍼, 메인댄서', '지민': '리드보컬, 리드댄서', '뷔': '서브보컬', '정국': '메인보컬, 서브래퍼, 리드댄서'}
서브보컬
메인보컬, 서브래퍼, 리드댄서
```

사전 생성하기

```
d1 = {} # or dict()
d2 = {'one': 1, 'two': 2, 'three': 3}
print(d1)
print(d2)
# using keywords
d3 = dict(one=1, two=2, three=3)
print(d3)
d4 = dict(d2)
print(d4)
d5 = dict([('one', 1), ('two',2), ('three',3)])
print(d5)
d6 = dict(zip(['one', 'two', 'three'], [1, 2, 3]))
print(d6)
```

```
{}
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
{'one': 1, 'two': 2, 'three': 3}
```

사전으로 형변환: dict() 함수 사용

- 두 개의 원소를 가진 시퀀스를 사전으로 형변환 가능함
  - 각 시퀀스의 첫 번째 항목은 키(key), 두 번째 항목을 값(value)으로 사용됨

```
t1 = [['answer', 42], ['pi', 3.14] , ['e', 2.718]]
d1 = dict(t1)
print(d1)
t2 = (('answer', 42), ('pi', 3.14), ('e', 2.718))
d2 = dict(t2)
print(d2)
```

```
{'answer': 42, 'pi': 3.14, 'e': 2.718}
{'answer': 42, 'pi': 3.14, 'e': 2.718}
```

항목 추가하기

- 새로운 키/값을 이용하여 원소를 추가할 수 있음
  - 리스트와 다르게 초기화되지 않은 키(리스트의 인덱스에 해당)를 사용할 수 있음
- 참고: 값을 읽을 때는 미리 정의된 키만 사용할 수 있음
  - 정의되지 않은 키를 사용하면 KeyError가 발생

```
numbers = {'pi': 3.14, 'e': 2.718}
numbers['golden_ratio'] = 1.618 # add a new item
numbers.update(pi = 3.1416) # {‘pi’:3.1416}
print(numbers)
print(numbers['answer']) # KeyError: 'answer'
```

```
{'pi': 3.1416, 'e': 2.718, 'golden_ratio': 1.618}
```

항목 삭제하기: del, clear()

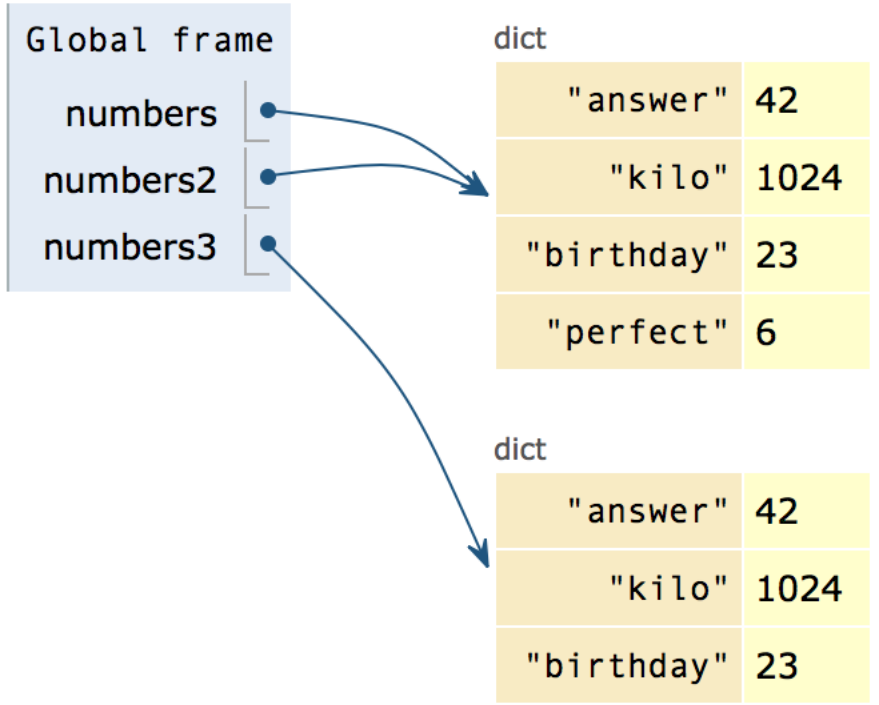
- `del d[key]`: Remove `d[key]` from d. Raises a `KeyError` if `key` is not in the map.
- `dict.clear()`: Remove all items from the dictionary.

```
numbers = {'answer': 42, 'kilo': 1024, 'birthday': 23}
print(numbers)
del numbers['kilo']
print(numbers)
numbers.clear()
print(numbers)
```

```
{'answer': 42, 'kilo': 1024, 'birthday': 23}
{'answer': 42, 'birthday': 23}
{}
```

예시: 사전의 복사, copy() 함수

```
numbers = {'answer': 42, 'kilo':
1024, 'birthday': 23}
numbers2 = numbers
numbers3 = numbers.copy()
numbers2['perfect'] = 6
print(numbers)
print(numbers2)
print(numbers3)
```




```
{'answer': 42, 'kilo': 1024, 'birthday': 23, 'perfect': 6}
{'answer': 42, 'kilo': 1024, 'birthday': 23, 'perfect': 6}
{'answer': 42, 'kilo': 1024, 'birthday': 23}
```

* 오른쪽은 4행까지 실행한 후의 상태

항목 얻기: get()

- `dict.get(key[, default])`: Return the value for `key` if `key` is in the dictionary, else `default`. If `default` is not given, it defaults to `None`, so that this method **never** raises a `KeyError`.

```
numbers = {'answer': 42, 'kilo': 1024, 'birthday': 23}
numbers['perfect'] = 6 # new item is added
#print(numbers['prime']) # KeyError: 'perfect'
print(numbers.get('answer', 0))
print(numbers.get('prime', 0))
```

```
42
0
```

키 멤버십 테스트: in, not in

- 특정한 키가 사전에 속해있는지 확인 (참고: 키만 확인하고 값은 확인하지 않음)

```
bts_position = {
    '랩몬스터': '리더, 메인래퍼',
    '진': '서브보컬',
    '슈가': '리드래퍼',
    '제이홉': '서브래퍼, 메인댄서',
    '지민': '리드보컬, 리드댄서',
    '뷔': '서브보컬',
    '정국': '메인보컬, 서브래퍼, 리드댄서'}
print('진' in bts_position)
print('슈가' not in bts_position)
print('서브보컬' in bts_position)
```

```
True
False
False
```

keys(), values(), items()

- 각각 키, 값, 키와 값의 쌍을 튜플로 반환함
  - 인덱싱이 필요한 경우, list() 함수를 이용하여 리스트로 형변환

```
numbers = {'pi': 3.14, 'e': 2.718, 'gr': 1.618}
print(numbers.keys())
print(numbers.values())
print(numbers.items())
print(list(numbers.items())[2])
```

```
dict_keys(['pi', 'e', 'gr'])
dict_values([3.14, 2.718, 1.618])
dict_items([('pi', 3.14), ('e', 2.718), ('gr', 1.618)])
('gr', 1.618)
```

예시: keys(), values(), items()를 for문과 사용

```
bts_position = {
    '랩몬스터': '리더, 메인래퍼',
    '진': '서브보컬',
    '슈가': '리드래퍼',
    '제이홉': '서브래퍼, 메인댄서',
    '지민': '리드보컬, 리드댄서',
    '뷔': '서브보컬',
    '정국': '메인보컬, 서브래퍼, 리드댄서'}
for nickname in bts_position.keys():
    print(nickname)
for nickname, position in
bts_position.items():
    print(nickname, ':', position)
```

```
랩몬스터
진
슈가
제이홉
지민
뷔
정국
랩몬스터 : 리더, 메인래퍼
진 : 서브보컬
슈가 : 리드래퍼
제이홉 : 서브래퍼, 메인댄서
지민 : 리드보컬, 리드댄서
뷔 : 서브보컬
정국 : 메인보컬, 서브래퍼, 리드댄서
```

dict comprehension

- 사전 내포

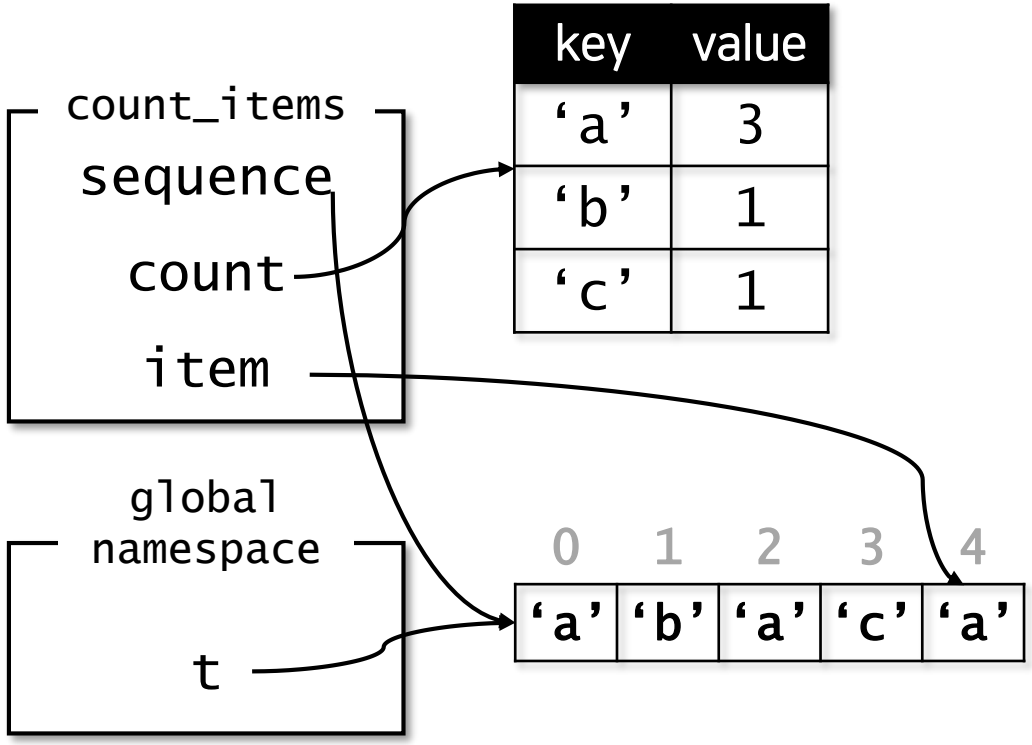
```
d = {x: x**2 for x in (2, 4, 6)}
print(d)
```

```
{2: 4, 4: 16, 6: 36}
```

예시: sequence의 아이템 수 세기

```
01: def count_items(sequence):
02:     count = {}
03:     for item in sequence:
04:         if item in count:
05:             count[item] += 1
06:         else:
07:             count[item] = 1
08:     return count
09:
10: t = ['a', 'b', 'a', 'c', 'a']
11: print(count_items(t))
```

```
{'a': 3, 'b': 1, 'c': 1}
```



예시: sequence의 아이템 수 세기 - get() 함수 이용

```
01: def count_items(sequence):
02:     count = {}
03:     for item in sequence:
04:         count[item] = count.get(item, 0) + 1
05:     return count
06:
07: t = ['a', 'b', 'a', 'c', 'a']
08: print(count_items(t))
```

```
{'a': 3, 'b': 1, 'c': 1}
```

리스트, 튜플, 사전, 집합 비교

	리스트(list)	튜플(tuple)	사전(dict)	집합(set)
정의	`t = [1, 2, 3]`	`t = (1, 2, 3)` `t = 1, 2, 3`	`t = {'1': 3, '2': 4, '3': -1}`	`t = {1, 2, 3}`
인덱스 방식	정수 인덱스	정수 인덱스	(immutable인) 키 예: 정수, 문자열, 튜플	n/a
인덱스 예시	`t[2]`	`t[2]`	`t['2']`	n/a
변경 가능	가능	불가능	가능	가능

ANY QUESTIONS?