제 5 강 Dictionary

학습 목차

- **Dictionary 정의**
- **Dictionary 연산**
- **Set**
- **Complex data type 정리**

Dictionary

- **Key 와 Value 짝 들의 집합**
- **파이썬의 내장 자료구조**

```
>>> a_student = {'name': 'daehyun', 'grade': 'A+'}
>>> a_student['name']
'daehyun'
>>> a_student['grade']
'A+'
>>> a_student['age']
Traceback (most recent call last):
  File "C:\Program Files\JetBrains\PyCharm Community Edition 2022.3.2\plugi
    coro = func()
           ^^^^^^
  File "<input>", line 1, in <module>
KeyError: 'age'
>>> a_student['age'] = 51
>>> a_student['age']
51
```

Dictionary vs. List

- **List : 순서와 값이 다 같아야, 같음.**
- **Dictionary: 순서에 상관없이 key-value 짝이 모두 같으면, 같음.**

```
>>> spam = ['cats', 'dogs', 'moose']
>>> bacon = ['dogs', 'moose', 'cats']
>>> spam == bacon
False
>>> eggs = {'name': 'Zophie', 'species': 'cat', 'age': '8'}
>>> ham = {'species': 'cat', 'age': '8', 'name': 'Zophie'}
>>> eggs == ham
True
```

keys(), values(), items()

```
>>> data = {'color': 'red', 'age': 42}
>>> for key in data.keys():
        print(key)

color
age
>>> for value in data.values():
        print(value)

red
42
>>> for item in data.items():
        print(item)

('color', 'red')
('age', 42)
```

```
>>> data.keys()
dict_keys(['color', 'age'])

>>> list(data.keys())
['color', 'age']
>>>
```

Key와 value의 존재 확인

```
>>> spam = {'name': 'Zophie', 'age': 7}
>>> 'name' in spam
True
>>> 'name' in spam.keys()
True
>>> 'Zophie' in spam.values()
True
>>> 'color' in spam.keys()
False
>>> 'color' not in spam.keys()
True
>>> 'color' in spam
False
```

get()

- Key가 없을 경우, 기본 value 를 대신 선택.

```
>>>
>>>
>>> a_student.get('name')
'daehyun'
>>> a_student.get('height')
>>> a_student.get('height', 176)
176

>>>
```

setdefault()

- **Key가 없으면, value를 지정.**
- **기존 key가 이미 있으면, 무시됨.**

```
>>> spam = {'name': 'Pooka', 'age': 5}
>>> spam.setdefault('color', 'black')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
>>> spam.setdefault('color', 'white')
'black'
>>> spam
{'color': 'black', 'age': 5, 'name': 'Pooka'}
```

© 이대현

characterCounter.py v.1

```python
text = 'It was a bright cold day in April, and the clocks were striking thirteen.'
count = {}
for c in text:
    count.setdefault(c, 0)
    count[c] = count[c] + 1

print(count)
```
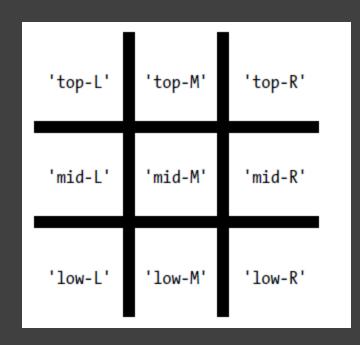
characterCounter.py v.2

```python
text = 'It was a bright cold day in April, and the clocks were
striking thirteen.'
from collections import defaultdict

count = defaultdict(lambda: 0)
for c in text:
    count[c] = count[c] + 1

print(count)
print(type(count))
```

TicTacToe.py

```python
theBoard = {'top-L': ' ', 'top-M': ' ', 'top-R': ' ', 'mid-L': ' ', 'mid-M': ' ', 'mid-R': ' ', 'low-L': ' ', 'low-M': ' ', 'low-R': ' '}

def printBoard(board):
    print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
    print('-+-+-')
    print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
    print('-+-+-')
    print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])
turn = 'X'
for i in range(9):
    printBoard(theBoard)
    print('Turn for ' + turn + '. Move on which space?')
    move = input()
    theBoard[move] = turn
    if turn == 'X':
        turn = 'O'
    else:
        turn = 'X'
printBoard(theBoard)
```

set

- 집합 자료형. 리스트와 달리, 중복을 허용하지 않고, 순서가 없음.

```
Python 3.7.0 Shell                                          —  □  ✕
File  Edit  Shell  Debug  Options  Window  Help
>>> s1 = {1,2,3}
>>> type(s1)
<class 'set'>
>>> s1 = {1,2,2,4}
>>> s1
{1, 2, 4}
>>> l1 = [1,2,2,2,2,3,3,3,3,5,5,5,5,5]
>>> s1 = set(l1)
>>> s1
{1, 2, 3, 5}
>>> s2 = {3,5,6,7}
>>> s1 + s2
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    s1 + s2
TypeError: unsupported operand type(s) for +: 'set' and 'set'
>>> s1 | s2
{1, 2, 3, 5, 6, 7}
>>> s1 & s2
{3, 5}
>>> s2 - s1
{6, 7}
>>> s1 - s2
{1, 2}
>>> s1.add(8)
>>> s1
{1, 2, 3, 5, 8}
>>> s2.remove(6)
>>> s2
{3, 5, 7}
                                                        Ln: 86  Col: 4
```

Complex Data Type

- **List – list**
  - 순서가 있는, 중복을 허용하는 데이타들의 집합.
  - 원하는 데이터를 찾기 위해, 순서 index 를 이용.

  [val1, val2, ···]

- **Dictionary – dict**
  - 검색을 위한 키를 갖는 데이타들의 집합
  - key – value 쌍 들의 집합

  { key1 : val1, key2 : val2, ··· }

- **Tuple – tuple**
  - 순서가 있는, 중복을 허용하는 데이타들의 집합
  - 다만, 데이타값을 변경하는 것은 불가

  (val1, val2, ···)

- **Set – set**
  - 중복을 허용하지 않는, 순서에 상관없는 데이타들의 집합

  { val1, val2, ··· }