딕셔너리와_집합

- description : 딕셔너리와_집합
- author : 도봉산핵주먹
- email : hylee@repia.com
- lastupdate : 2020-06-22

딕셔너리와_집합

예제 코드

```
# Section04-4
# 파이썬 데이터 타입(자료형)
# 딕셔너리, 집합 자료형

# 딕셔너리 자료형(순서X, 중복X, 수정O, 삭제O)
# Key, Value (Json) -> MongoDB

# 선언
print('#===== 딕셔너리 =====#')
a = {'name': 'Kim', 'phone': '01012345678', 'birth': 870124}
b = {0: 'Hello python!'}
c = {'arr': [1, 2, 3, 4]}

print('#=== type, 값 출력 ===#')
print('a - ', type(a), a)
print('b - ', type(b), b)
print('c - ', type(c), c)

print()

# 출력
print('#=== 값 출력 방법 ===#')
print('a - ', a['name'])  # 존재X -> 에러 발생
print('a - ', a.get('name'))  # 존재X -> None 처리
print('b - ', b[0])
print('b - ', b.get(0))
print('c - ', c['arr'])
print('c - ', c['arr'][3])
print('c - ', c.get('arr'))

print()

# 딕셔너리 추가
print('#=== 딕셔너리 추가 ===#')
a['address'] = 'seoul'
```

```
print('a - ', a)
a['rank'] = [1, 2, 3]
print('a - ', a)

print()

# dict_keys, dict_values, dict_items : 반복문(iterate) 사용 가능
# dict_items -> dict 전체
print('#===  dict_keys, dict_values, dict_items 알아보기 ===#')
print('a - ', a.keys())
print('b - ', b.keys())
print('c - ', c.keys())

print()

print('a - ', list(a.keys()))
print('b - ', list(b.keys()))
print('c - ', list(c.keys()))

print()

print('a - ', a.values())
print('b - ', b.values())
print('c - ', c.values())

print()

print('a - ', list(a.values()))
print('b - ', list(b.values()))
print('c - ', list(c.values()))

print()

print('a - ', a.items())
print('b - ', b.items())
print('c - ', c.items())

print()

print('a - ', list(a.items()))
print('b - ', list(b.items()))
print('c - ', list(c.items()))

print()

print('a - ', 'name' in a)
print('a - ', 'addr' in a)

print()
# Key만 가져오고 index로 접근 안됨.
```

```
# index로 접근하려변 변수화 해서 가져와야됨.
print('#== dict_keys 나 values는 변수화 해서 index값으로 접근하기 ==#')
temp = list(a.keys())
print(temp[1:3])
# Key만 가져오고 index로 접근 안됨.

print()
print()

print('#==== 집합 ====#')
# 집합(Sets) 자료형(순서X, 중복X)

# 선언
a = set()
b = set([1, 2, 3, 4])
c = set([1, 4, 5, 6])
d = set([1, 2, 'Pen', 'Cap', 'Plate'])

print('#=== type, 값 출력 ===#')
print('a - ', type(a), a)
print('b - ', type(b), b)
print('c - ', type(c), c)
print('d - ', type(d), d)

print()

# 튜플 변환
print('#=== 튜플로 변환 ===#')
t = tuple(b)
print('t - ', type(t), t)
print('t - ', t[0], t[1:3])

print()

# 리스트 변환
print('#=== 리스트로 변환 ===#')
l = list(c)
print('l - ', type(l), l)
print('l - ', l[0], l[1:3])

print()

print("#=== 집합 자료형 활용 ===")
# 집합 자료형 활용
s1 = set([1, 2, 3, 4, 5, 6])
s2 = set([4, 5, 6, 7, 8, 9])


print("#== 교집합 ==")
print('l - ', s1 & s2)
print('l - ', s1.intersection(s2))
```

```
print("#== 합집합 ==")
print('l - ', s1 | s2)
print('l - ', s1.union(s2))

print("#== 차집합 ==")
print('l - ', s1 - s2)
print('l - ', s1.difference(s2))

print()

print("#=== 집합 추가 / 제거 ===")
# 추가 & 제거
s1 = set([1, 2, 3, 4])
s1.add(5)
print('s1 - ', s1)

s1.remove(2)
print('s1 - ', s1)
```

실행 콘솔

```
#==== 딕셔너리 ====#
#=== type, 값 출력 ===#
a -  <class 'dict'> {'name': 'Kim', 'phone': '01012345678', 'birth': 870124}
b -  <class 'dict'> {0: 'Hello python!'}
c -  <class 'dict'> {'arr': [1, 2, 3, 4]}

#=== 값 출력 방법 ===#
a -  Kim
a -  Kim
b -  Hello python!
b -  Hello python!
c -  [1, 2, 3, 4]
c -  4
c -  [1, 2, 3, 4]

#=== 딕셔너리 추가 ===#
a -  {'name': 'Kim', 'phone': '01012345678', 'birth': 870124, 'address':
'seoul'}
a -  {'name': 'Kim', 'phone': '01012345678', 'birth': 870124, 'address':
'seoul', 'rank': [1, 2, 3]}

#===  dict_keys, dict_values, dict_items 알아보기 ===#
a -  dict_keys(['name', 'phone', 'birth', 'address', 'rank'])
b -  dict_keys([0])
c -  dict_keys(['arr'])

a -  ['name', 'phone', 'birth', 'address', 'rank']
b -  [0]
```

```
c -  ['arr']

a -  dict_values(['Kim', '01012345678', 870124, 'seoul', [1, 2, 3]])
b -  dict_values(['Hello python!'])
c -  dict_values([[1, 2, 3, 4]])

a -  ['Kim', '01012345678', 870124, 'seoul', [1, 2, 3]]
b -  ['Hello python!']
c -  [[1, 2, 3, 4]]

a -  dict_items([('name', 'Kim'), ('phone', '01012345678'), ('birth',
870124), ('address', 'seoul'), ('rank', [1, 2, 3])])
b -  dict_items([(0, 'Hello python!')])
c -  dict_items([('arr', [1, 2, 3, 4])])

a -  [('name', 'Kim'), ('phone', '01012345678'), ('birth', 870124),
('address', 'seoul'), ('rank', [1, 2, 3])]
b -  [(0, 'Hello python!')]
c -  [('arr', [1, 2, 3, 4])]

a -  True
a -  False

#== dict_keys 나 values는 변수화 해서 index값으로 접근하기 ==#
['phone', 'birth']


#===== 집합 =====#
#=== type, 값 출력 ===#
a -  <class 'set'> set()
b -  <class 'set'> {1, 2, 3, 4}
c -  <class 'set'> {1, 4, 5, 6}
d -  <class 'set'> {1, 2, 'Plate', 'Cap', 'Pen'}

#=== 튜플로 변환 ===#
t -  <class 'tuple'> (1, 2, 3, 4)
t -  1 (2, 3)

#=== 리스트로 변환 ===#
l -  <class 'list'> [1, 4, 5, 6]
l -  1 [4, 5]

#=== 집합 자료형 활용 ===
#== 교집합 ==
l -  {4, 5, 6}
l -  {4, 5, 6}
#== 합집합 ==
l -  {1, 2, 3, 4, 5, 6, 7, 8, 9}
l -  {1, 2, 3, 4, 5, 6, 7, 8, 9}
#== 차집합 ==
l -  {1, 2, 3}
l -  {1, 2, 3}
```

```
# === 집합 추가 / 제거 ===
s1 -  {1, 2, 3, 4, 5}
s1 -  {1, 3, 4, 5}
```

Tip

도봉산핵주먹, python, 딕셔너리, 집합