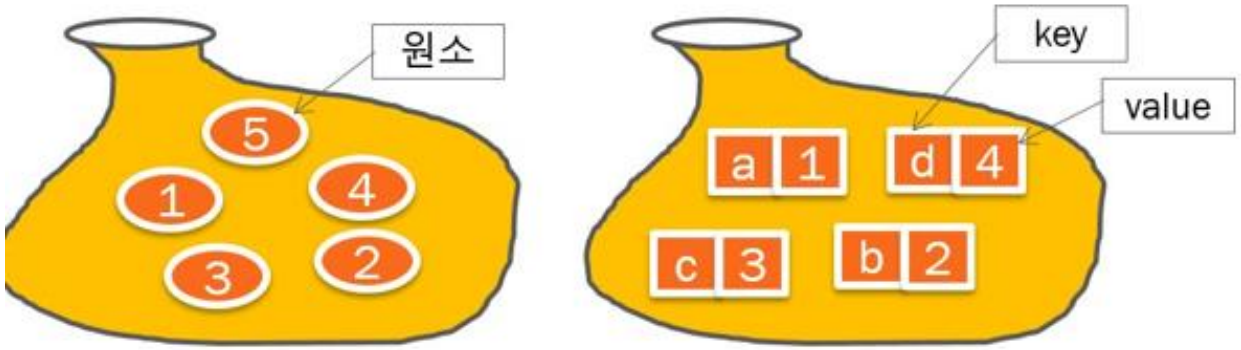
UNISTUDY

파이썬

13강. 자료구조(리스트내포, 튜플)

1. 비순서 자료구조

- 비순서 자료구조((None-Sequence Data Structure)는 리스트처럼 칸막이로 구분되지 않고, 공통 의 영역에 값들이 적재된다.
- 순서 자료구조의 예를 나타내고 있다.
- 파이썬에서는 비순서 자료구조를 생성하기 위해서 set, dict 등의 클래스를 제공한다.
- 비순서 자료구 조의 객체를 생성하는 관련 클래스들에 대해서 알아본다.



2. 셋(set)

- set 클래스는 여러 개의 자료를 비 순서로 적재하는 가변 길이 비순차 자료구조를 생성하는 클래스이 다.
- 즉 리스트처럼 칸막이로 구분되지 않고, 공통의 영역에 값들이 적재된다.
- 다음은 set 클래스의 도 움말을 확인한 결과이다.

```
>>>help(set)
Help on class set in module builtins:

class set(object)
 |  set() -> new empty set object
 |  set(iterable) -> new set object
```

2. 셋(set)

- 가장 첫 번째 ‘class range in module builtins’ 문장에서 set 클래스는 builtins 모듈에 포함된 클래스임을 나타내고 있다.
- ‘인수가 없는 빈 set 객체를 만든다.
- 반복가능 인수로 set 객체를 만든 다.’라고 셋 클래스를 설명하고 있다.

3. 셋 객체 특징

- 비순서 자료구조를 갖는 열거형객체를 생성할 수 있다.
- 다음 형식과 같이 중괄호({ })안에 콤마(,)를 이용하여 원소를 구분한다.

```
변수 =  {값1, 값2, ..... 값n}
```

- 중복을 허용하지 않는다.
- 순서가 없기 때문을 색인(index)을 사용할 수 없다.
- 객체에서 제공하는 함수를 이용하여 추가, 삭제 및 집합 연산 등이 가능하다.

3. 셋 객체 특징

chapter04.lecture.step04_set.py

```
# (1) 중복 불가
s = {1, 3, 5, 3, 1}
print(len(s))
print(s)

# (2) 요소 반복


for d in s :
    print(d, end=' ')  # 1 3 5

print()


# (3) 집합관련 함수
s2 = {3, 6}
print(s.union(s2)) # 합집합
print(s.difference(s2)) # 차집합
print(s.intersection(s2)) # 교집합 :

# (4) 추가, 삭제 함수
s3 = {1, 3, 5}
print(s3)

s3.add(7) # 원소 추가
print(s3) #

s3.discard(3) # 원소 삭제
print(s3) #
```

Python Console

```
3
{1, 3, 5}
{1, 3, 5, 6}
{1, 5}
{3}
{1, 3, 5}
{1, 3, 5, 7}
{1, 5, 7}
```

3. 셋 객체 특징

- 해설 셋 객체 예
- (1) 중복 불가
- 변수 s는 5개의 원소 중에서 4번째 원소 3은 이미지 2번째에서 출현하기 때문에 중복되어 저장되지 않는다. 길이와 전체 원소를 출력하면 확인할 수 있다.
- (2) 요소 반복
- 셋은 색인을 사용할 수 없지만 for문에서 요소 반복으로 사용할 수 있다. 이때 꺼내지는 원소들의 순서는 무순서이다.
- (3) 집합관련 함수
- 셋은 집합의 의미를 갖고 있다. 따라서 집합관련 함수를 객체에서 지원한다. 변수 s와 s2 간의 합집합(union), 차집합(difference), 교집합(intersection) 연산이 관련 함수들에 의해서 수행된다.
- (4) 추가, 삭제 함수
- 셋 객체에서 지원하는 함수를 이용하여 집합의 원소를 추가 및 삭제할 수 있다. 추가는 add(원소) 함수를 이용하고, 삭제는 discard(원소) 함수를 이용한다.

3. 셋 중복 제거

- 중복을 허용하지 않는다는 셋의 특징을 이용하여 리스트의 중복 원소를 제거하는데 셋을 이용할 수 있다

chapter04.lecture.step04_set.py

```
# 중복 원소를 갖는 리스트
gender = ['남','여','남','여']

# 중복 원소 제거
sgender = set(gender) # list -> set
lgender = list(sgender) # set -> list
print(lgender)

print(lgender[1])
```

Python Console

```
['남', '여']
여
```

3. 셋 중복 제거

- 중복 제거 예
- 리스트 객체를 참조하는 gender 변수를 대상으로 set() 함수를 이용하여 셋 자료형으로 변환하면 중복되지 않은 원소만 남는다.
- 이를 다시 list() 함수를 이용하여 리스트 자료형으로 변환하면 순서 자료구조로 변환되기 때문에 색인으로 원소를 참조할 수 있다.

4. 딕트(dict)

- dict 클래스는 사전(dictionary)형으로 여러 개의 자료를 비 순서로 적재하는 가변 길이 비순차 자료 구조를 생성하는 클래스이다. 즉 set 클래스와 동일하게 공통의 영역에 원소들이 적재된다. set과의 차이점은 '키(key)에 '값(value)'을 저장하고, 키를 통해서 값을 참조하는 형식이다.
- 일반적으로 사전 을 연상해 볼 때 특정 단어를 찾기 위해서는 첫 자를 색인으로 빠르게 단어를 찾을 수 있도록 되어있다.
- 딕트 자료형도 마찬가지로 키를 색인으로 하여 값을 찾는데 최고의 성능을 낼 수 있도록 최적화되어 있다.
- 다음은 dict 클래스의 도움말을 확인한 결과이다.

4. 딕트(dict)

- 가장 첫 번째 'class range in module builtins' 문장에서 dict 클래스는 builtins 모듈에 포함된 클래스임을 나타내고 있다. '인수가 없는 빈 dict 객체를 만든다. (키, 값)을 한쌍으로 새로운 사전형 객체를 만든다. 반복 가능 인수로 dict 객체를 만든다.'라고 dict 클래스를 설명하고 있다..

```
>>>help(dict)
Help on class dict in module builtins:

class dict(object)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a mapping object's
 |      (key, value) pairs

 |  dict(iterable) -> new dictionary initialized as if via:
```

5. 딕트 객체 특징

- 사전 형식으로 비순서 자료구조를 갖는 열거형객체를 생성할 수 있다.
- 다음 형식과 같이 중괄호 안에 {'키' : '값'}의 쌍으로 원소를 입력하고, 콤마(,)를 이용하여 원소를 구분 한다.

```
변수 =  {'키' : '값', '키' : '값', .... '키' : '값'}
```

- '키'는 중복이 허용되지 않고, '값'은 중복이 허용된다.
- 색인(index) 대신에 키(key)를 이용해서 '값'을 참조한다.
- 키를 색인으로 이용할 수 있기 때문에 원소 수정, 삭제, 추가 등이 가능하다..

5. 딕트 객체 특징

chapter04.lecture.step05_dict.py

```
# (1) dict 생성 방법1
dic = dict(key1 = 100, key2 = 200, key3 = 300)
print(dic)

# (2) dict 생성 방법2
person = {'name':'홍길동', 'age': 35, 'address' : '서울시'}
print(person)
print( person['name'] )
print( type(dic), type(person) )

# (3) 원소 수정, 삭제, 추가
# dict 원소 수정
person['age'] = 45
print(person)

# dict 원소 삭제
del person['address']

print(person) # {'name': '홍길동', 'age': 45}


# dict 원소 추가
person['pay'] = 350
print(person) # {'name': '홍길동', 'pay': 350, 'age': 45}
```

Python Console

```
{'key1': 100, 'key2': 200, 'key3': 300}
{'name': '홍길동', 'age': 35, 'address': '서울시'}
홍길동
<class 'dict'> <class 'dict'>
{'name': '홍길동', 'age': 45, 'address': '서울시'}
{'name': '홍길동', 'age': 45}
{'name': '홍길동', 'age': 45, 'pay': 350}
```

5. 딕트 객체 특징

- 딕트 객체 예
- (1) dict 생성 방법1
- dict 클래스에서 '키=값' 형식의 인수를 이용하여 딕트 객체를 생성할 수 있다.
- (2) dict 생성 방법2
- 중괄호 기호 안에 '{키 : 값}' 형식으로 딕트 객체를 생성할 수 있다. person 변수는 3개의 원소를 갖는 딕트 객체이다. person['name']는 name 키를 색인으로 하여 '홍길동' 값을 참조하는 형식이 다. 딕트 객체의 자료형은 <class 'dict'>으로 출력된다.
- (3) 원소 수정, 삭제, 추가
- 딕트는 키를 색인으로 이용할 수 있기 때문에 원소의 수정, 삭제, 추가가 가능하다. 'person['age'] = 45' 명령문은 age를 35에서 45로 수정한다. 'del person['address']' 명령문은 address 키를 삭제한다. 'person['pay'] = 350' 명령문은 pay 키에 350값을 할당하 여 새로운 원소로 추가한다.

6. 요소 검사와 반복

- 딕트는 셋과 같이 비순서 자료구조이지만 열거형 객체이기 때문에 요소 검사와 요소 반복 기능을 지원 한다.

6. 요소 검사와 반복

chapter04.lecture.step05_dict.py

```
# (1) 요소 검사
print(person['age']) # 45
print('age' in person) # True

# (2) 요소 반복
for key in person.keys() : # key 넘김
    print(key) # key 출력
for v in person.values() : # value 넘김

    print(v) # value 출력

for i in person.items() : # (key, value) 넘김
    print(i) # ('name', '홍길동')
```

Python Console

```
45
True


name
age
pay


홍길동
45
350


('name', '홍길동')
('age', 45)
('pay', 350)
```

6. 요소 검사와 반복

- 해설 요소 검사와 반복 예
- (1) 요소 검사
- 딕트의 요소를 검사하기 위해서 '요소 in 딕트객체' 형식으로 in 앞부분에 검사할 요소 넣는다. 딕트객체에 해당 '요소'가 있으면 True를 반환하고, 그렇지 않으면 False를 반환한다.
- (2) 요소 반복
- 딕트 객체는 for문에서 요소 반복으로 사용할 수 있다. 딕트 객체에서 제공하는 keys() 함수를 이용하면 딕트 객체에서 '키'만 요소로 반복하고, values() 함수를 이용하면 '값'만 요소로 반복된 다. 만약 '키'와 '값'을 한쌍으로 반복하려면 items() 함수를 이용하면 된다.

7. 단어 출현빈도수 구하기

- 딕트 객체의 특징과 객체에서 제공하는 get() 함수를 이용하여 단어 빈도수(word count)9)를 계산 할 수 있다.

7. 단어 출현빈도수 구하기

chapter04.lecture.step05_dict.py

```
# (1) 단어 데이터 셋
charset = ['abc', 'code', 'band', 'band',
'abc']
wc = {} # 빈 셋

# (2) get() 함수 이용 : key 이용 value 가져오기
for key in charset :
    wc[key] = wc.get(key, 0) + 1 # get() 이
용
print(wc)
```

Python Console

```
{'abc': 2, 'code': 1, 'band
': 2}
```

7. 단어 출현빈도수 구하기

- 해설 단어 빈도수 구하기 예
- (1) 단어 데이터 셋
- charset 리스트 객체는 중복된 5개의 단어를 원소로 갖는다.
- (2) get() 함수 이용
- charset 리스트 객체를 for문의 요소 반복으로 사용하여 순차적으로 단어를 하나씩 key 변수로 넘겨받는다. 넘겨받은 단어를 wc의 '키'로 지정하고, get()함수를 이용하여 '키'에 해당하는 '값' 을 꺼내온다. 이때 값이 없는 경우(최초로 발견된 단어) 0을 초기화하고 1를 더해서 '값'을 만든다. 만약 '값'이 있는 경우(2회 이상 발견된 단어)에는 꺼내온 '값'+1를 더해서 단어를 카운트 한다. 결국 빈 셋으로 시작했지만 {'단어' : '빈도수'} 형태인 딕트 객체로 만들어진 것을 확인할 수 있다.

THANK YOU