데이터처리프로그래밍

Collection module

강원대학교 교육혁신원 송혜정

<hjsong@kangwon.ac.kr>

Collection module

✓ 학습목표
- Collection module을 이해하고 활용한다.

✓ 학습내용
- Collection module
- Deque(double-ended queue)
- OrderedDict
- Defaultdict
- Counter

강의에 앞서서..

- 본 강의자료는 아래의 자료들을 참고하여 만들어 졌음을 알립니다

1. 데이터과학을 위한 파이썬 프로그래밍, 최성철, 한빛아카데미,2019
2. Python (https://docs.python.org)
3. 실용 파이썬 프로그래밍(Practical Python Programming)
   (https://wikidocs.net/84426)

Collections module

- collections module
  - 다양한 자료구조인 list, tuple, set, dict 등을 구현한 특수 컨테이너 데이터 형
  - 파이썬의 내장 모듈

- Collections의 자료구조 객체

namedtuple()	이름 붙은 필드를 갖는 튜플 서브 클래스를 만들기 위한 팩토리 함수
deque	양쪽 끝에서 빠르게 추가와 삭제를 할 수 있는 리스트류 컨테이너
ChainMap	여러 매핑의 단일 뷰를 만드는 딕셔너리류 클래스
Counter	해시 가능한 객체를 세는 데 사용하는 딕셔너리 서브 클래스
OrderedDict	항목이 추가된 순서를 기억하는 딕셔너리 서브 클래스
defaultdict	누락된 값을 제공하기 위해 팩토리 함수를 호출하는 딕셔너리 서브 클래스
UserDict	더 쉬운 딕셔너리 서브 클래싱을 위해 딕셔너리 객체를 감싸는 래퍼
UserList	더 쉬운 리스트 서브 클래싱을 위해 리스트 객체를 감싸는 래퍼
UserString	더 쉬운 문자열 서브 클래싱을 위해 문자열 객체를 감싸는 래퍼

https://docs.python.org/ko/3/library/collections.html#module-collections

Collection module

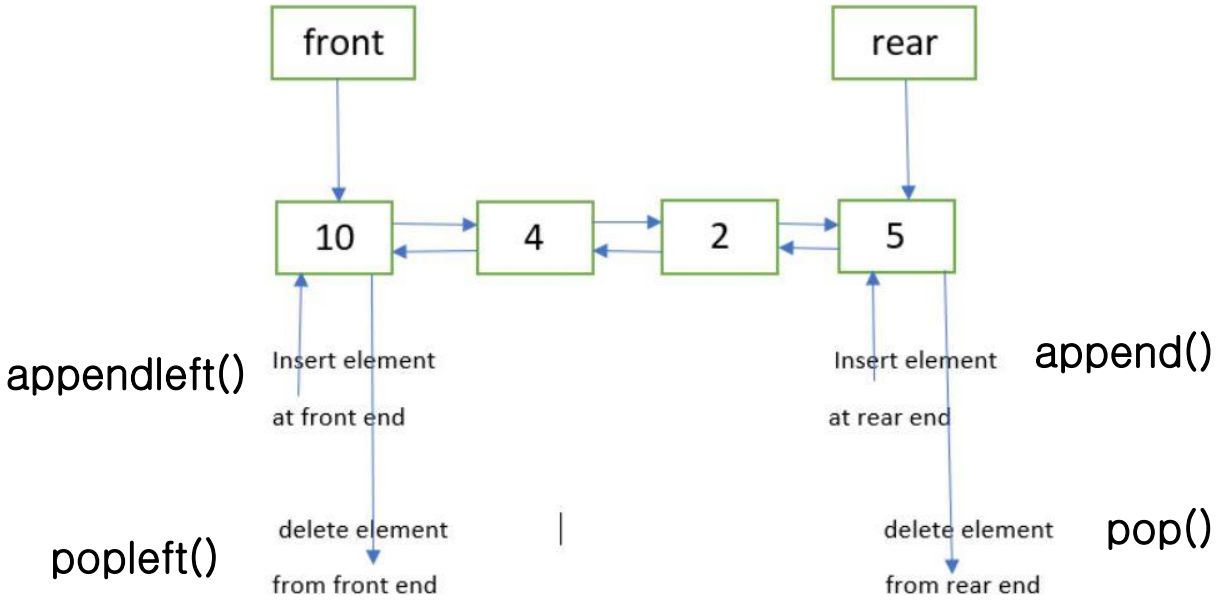
- collections module import
  - Import를 이용하여 collections module에 포함된 각 자료구조를 사용하기 위하여 가져오기

```
from collections import deque
from collections import OrderedDict
from collections import defaultdict
from collections import Counter
```

```
from collections import *
```

Deque

- Deque(double-ended queue)
  - 양방향 입출력이 가능한 큐
  - 리스트와 유사한 형태의 자료구조 객체
  - 스택과 큐를 일반화 한 객체
  - class collections.deque([iterable[, maxlen]])
- 구조



https://www.geeksforgeeks.org/implementation-deque-using-doubly-linked-list/

Deque

- Deque 함수(function, method)
  - append(x) : 데크의 오른쪽에 x를 추가
  - appendleft(x) : 데크의 왼쪽에 x를 추가
  - pop() : 데크의 오른쪽에서 요소를 제거하고 반환
  - popleft(): 데크의 왼쪽에서 요소를 제거하고 반환
  - clear() : 데크에서 모든 요소를 제거하고 길이가 0인 상태로 초기화
  - copy() : 데크의 얕은 복사본 생성
  - count(x) : x 와 같은 데크 요소의 수 카운팅
  - extend(iterable) : iterable 인자에서 온 요소를 추가하여 데크의 오른쪽을 확장
  - extendleft(iterable) : iterable에서 온 요소를 추가하여 데크의 왼쪽을 확장
  - index(x[, start[, stop]]): 데크에 있는 x의 위치를 반환
  - insert(i, x) : x를 데크의 i 위치에 삽입
  - remove(value): value의 첫 번째 요소을 제거
  - reverse() : 데크의 요소들을 제자리에서 순서를 뒤집고 None을 반환
  - rotate(n=1) : 데크를 n 단계 오른쪽으로 회전, n이 음수이면, 왼쪽으로 회전
  - maxlen : 데크의 최대 크기

Deque

- 생성 및 요소 접근

```
#deque 사용을 위해 모듈 가져오기
from collections import deque

#deque 생성
nd = deque([1,2,3])   #리스트로 생성
print (nd)
sd= deque("123")      #문자열로 생성
print (sd)

#요소 접근
print('sd[0] =', sd[0])    #왼쪽 요소 접근
print('sd[-1] =', sd[-1])  #오른쪽 요소 접근

#search
print("'h' in sd = ", 'h' in sd)     # 'h'를 deque에서 검색, 찾으면True
```

```
deque([1, 2, 3])
deque(['1', '2', '3'])
sd[0] = 1
sd[-1] = 3
'h' in sd =  False
```

Deque

- append & extend & pop

```
sd= deque()
#추가
sd.append(2)                    # 오른쪽에 요소 추가
print ('sd.append(2) = ', sd)
sd.appendleft('l')                # 왼쪽에 요소 추가
print ("sd.appendleft('l') = ", sd)

#확장
sd.extend('xyz')                        # 여러 요소을 오른쪽에 추가하여 확장
print ("sd.extend('xyz') = ", sd)
sd.extendleft('abc')                    # 여러 요소을 왼쪽에 추가하여 확장
print ("sd.extendleft('abc') = ", sd)

#삭제
r = sd.pop()                            #오른쪽 요소 삭제
print('sd.pop() =', r)
print ('sd.pop() =', sd)

r = sd.popleft()                        #왼쪽 요소 삭제
print('sd.popleft() =', r)
print('sd.popleft() =', sd)
```

```
sd.append(2) =  deque([2])
sd.appendleft('l') =  deque(['l', 2])
sd.extend('xyz') =  deque(['l', 2, 'x', 'y', 'z'])
sd.extendleft('abc') =  deque(['c', 'b', 'a', 'l', 2, 'x', 'y', 'z'])
sd.pop() = z
sd.pop() = deque(['c', 'b', 'a', 'l', 2, 'x', 'y'])
sd.popleft() = c
sd.popleft() = deque(['b', 'a', 'l', 2, 'x', 'y'])
```

Deque

- reverse & rotate

```
#reverse
sd.reverse()
print ('sd.reverse() = ', sd)

#rotate
sd.rotate(1)                        # right rotation
print ("sd.rotate(1) = ", sd)
sd.rotate(-1)                       # left rotation
print ("sd.rotate(-1) = ", sd)

#empty
sd.clear()                          # empty the deque
print ('sd.clear()  = ', sd)
sd.pop()                            # cannot pop from an empty deque
```

```
sd.reverse() =  deque(['y', 'x', 2, 'l', 'a', 'b'])
sd.rotate(1) =  deque(['b', 'y', 'x', 2, 'l', 'a'])
sd.rotate(-1) =  deque(['y', 'x', 2, 'l', 'a', 'b'])
sd.clear()  =  deque([])

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-14-5a3597ff78ef> in <module>
     12 sd.clear()                          # empty the deque
     13 print ('sd.clear()  = ', sd)
---> 14 sd.pop()                           # cannot pop from an empty deque

IndexError: pop from an empty deque
```

OrderedDict

- OrderedDictionary
  - 순서를 가진 딕셔너리 객체
  - 입력순서로 저장되는 구조
  - class collections.OrderedDict([items])
  - 파이썬 3.6 부터는 기본 사전(dict)도 OrderedDict 클래스와 동일하게 동작

```
from collections import OrderedDict

#파이썬 내장 dict 생성
dic = {}
dic['X'] = 1
dic['B'] = 2
dic['C'] = 3
print(dic)

#collections 모듈내의 OrderedDict 객체 생성
od = OrderedDict()
od['X'] = 1
od['B'] = 2
od['C'] = 3
print(od)

#dict로 rderedDict 객체 생성
od2 = OrderedDict(dic)
print(od2)
```

```
{'X': 1, 'B': 2, 'C': 3}
OrderedDict([('X', 1), ('B', 2), ('C', 3)])
OrderedDict([('X', 1), ('B', 2), ('C', 3)])
```

OrderedDict

- OrderedDictionary 생성 및 접근

```
#튜플리스트로 딕셔너리 생성
ages = [('kim',22), ('lee', 30), ('min', 40)]  #튜플리스트
agedic = dict(ages)                  #파이썬 내장 dict 생성
ageodic = OrderedDict(ages)          #OrderedDict 객체 생성
print(agedic)
print(ageodic)

#key, value 추출
items = ageodic.items()      #키-값 쌍을 튜플의 리스트로 반환
keys = ageodic.keys()        #키들만 추출하여 리스트로 반환
values = ageodic.values()    #값들만 추출하여 리스트로 반환

print('items = ', items)
print('keys = ', keys)
print('values = ', values)
```

```
{'kim': 22, 'lee': 30, 'min': 40}
OrderedDict([('kim', 22), ('lee', 30), ('min', 40)])
items =  odict_items([('kim', 22), ('lee', 30), ('min', 40)])
keys =  odict_keys(['kim', 'lee', 'min'])
values =  odict_values([22, 30, 40])
```

OrderedDict

예제1. 성적처리 예

성명과 3개의 점수로 구성된 성적 튜플리스트로 초기화된 값을 이용하여

성적 딕셔너리를 생성하고 성적 디셔너리를 이용하여 평균점수 디셔너리 생성

```
#성명과 3개의 점수로 구성된 성적튜플리스트
scores = [('kim',[100,50,70]), ('lee', [90,80,88]), ('min', [60,70,8])]

#성적튜플리스트로 성적 딕셔너리 생성
sdic = OrderedDict(scores)       #OrderedDict 객체 생성
#성명과 평균 점수로 구성된 평균딕셔너리 생성
adic = OrderedDict()

#성적 디셔너리의 아이템들을 추출하여 평균을 계산하여 평균 디셔너리에 추가
for k, v in sdic.items():
    a = sum(v) / len(v)
    adic[k] = a          #평균 디셔너리에 성명키에 해당하는 평균점수 추가
#결과 출력
print("성적 dict = ", sdic)
print("평균 dict = ", adic)
```

```
성적 dict =  OrderedDict([('kim', [100, 50, 70]), ('lee', [90, 80, 88]), ('min', [60, 70, 8])])
평균 dict =  OrderedDict([('kim', 73.33333333333333), ('lee', 86.0), ('min', 46.0)])
```