파이썬 프로그래밍
강의 노트 #11

자료 구조 2 (Dictionary, Set)

딕셔너리(Dictionary)

- 키(key)와 값(value)으로 구성된 한 쌍의 데이터를 담을 수 있는 자료구조 (map, hash라고 부르기도 함)
- 중복된 키가 포함될 수 없음
- 키는 수정 불가능(immutable)한 것만 사용 가능
- 값은 변경 가능
- 중괄호로 표현

```
{ 키1 : 값1, 키2 : 값2 }
```

키(key)	값(value)
키1	값1
키2	값2

- 요소에 대한 접근은 [] 사용
  - 딕셔너리_이름[키] 또는 get(키)함수 사용
- 값 추가 또는 변경
  - 딕셔너리_이름[키] = 값
- 삭제는 del() 사용

딕셔너리(Dictionary)

- 딕셔너리 예

```
{} # 빈 딕셔너리
{ "name" : "홍길동", "age" : 20,
  "취미" : ["영화 감상", "게임", "독서"] }
```

- 키
  - 수정 불가능한 것들만 사용할 수 있음
  - 문자열, 숫자, 불린 값, 튜플 등

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
print(d[1])
print(d[False])
print(d[(1,2)])
```

딕셔너리(Dictionary)

- 딕셔너리의 요소 개수 확인하기
  - len() 함수 사용

```
d = {}   # empty dictionary
d1 = {'a':1, 'b':2, 'c':3 }
len({})   # 0
len(d)
len(d1)
len({'a':1, 'b':2, 'c':3 })
```

딕셔너리(Dictionary)

- 요소 접근 예 (딕셔너리_이름[키] 사용)

```
d = { "name" : "홍길동", "age" : 20,
  "취미" : ["영화 감상", "게임", "독서"] }
print(d["name"]) # 홍길동
print(d["취미"]) # ["영화 감상", "게임", "독서"]
print(d["이름"]) # 오류 발생 (KeyError)
```

- 요소 접근 예 (get()함수 사용)

```
d = { "name" : "홍길동", "age" : 20,
  "취미" : ["영화 감상", "게임", "독서"] }
print(d.get("name")) # 홍길동
print(d.get("취미")) # ["영화 감상", "게임", "독서"]
print(d.get("이름")) # None
```

딕셔너리(Dictionary)

- 키가 중복되면 나중에 나오는 값으로 지정됨

```
d = { 1 : 2, True : 20, (1, 2) : "튜플" }
print(d[1])
print(d[True])
print(d[(1,2)])

d = { "name" : "홍길동", "age" : 20,
   "취미" : ["영화 감상", "게임", "독서"],
       "name" : "김길동" }
print(d["name"]) # 김길동
print(d["취미"]) # ["영화 감상", "게임", "독서"]
```

딕셔너리(Dictionary)

- 요소 값 추가 또는 수정
  - 단일 요소 수정

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
d[1] = 3
d[False] = "불린 값"
d[(1, 2)] = [1, 2]
d["key"] = "value"   # 새로운 키와 값 추가
print(d[1])
print(d[False])
print(d[(1,2)])
print(d["key"])
```

딕셔너리(Dictionary)

- 요소 값 추가 또는 수정
  - 여러 요소를 한꺼번에 수정
    - update() 함수 사용

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
d.update({1:3, False:"불린 값", (1,2):[1,2],
          "key":"value" })
print(d[1])
print(d[False])
print(d[(1,2)])
print(d["key"])
```

딕셔너리

- for 문과 딕셔너리
  - 딕셔너리는 순서가 있는 것은 아니지만, for문과 사용할 수 있음
  - 기본적으로 for문과 함께 사용되면 key를 순차적으로 변수에 저장(단 순서는 정해진 것이 없음)

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
for key in d:
    print(f"{key}:{d[key]}")
```

  - keys() 함수를 이용해도 같은 결과를 얻을 수 있음

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
for key in d.keys():
    print(f"{key}:{d[key]}")
```

딕셔너리

- values() 함수를 이용하면 값을 한 개씩 변수에 저장

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
for n in d.values():
    print(n)
```

- 키와 값을 동시에 접근하고 싶으면 items() 함수 사용

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
for k, v in d.items():
    print(k, v)
```

딕셔너리

- 키가 딕셔너리에 있는지 확인
  - in 사용

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
if 1 in d:
    print(d[1])
else:
    print("1은 d의 키가 아닙니다")
```

- 삭제
  - del(딕셔너리_이름[키])

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
del(d[1])
print(d) # { False : 20, (1, 2) : "튜플" }
```

딕셔너리

- 딕셔너리 전체 삭제
  - clear() 함수 사용

```
d = { 1 : 2, False : 20, (1, 2) : "튜플" }
d.clear()
print(d) # {}
```

딕셔너리

- 주의할 점
  - 키 값으로 리스트 사용 불가

실습문제 1

- 문제
  - 다음 문자열 sentences에서 각 알파벳 문자들의 빈도수를 구하는 프로그램 작성
    - sentences = """Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

      Maecenas porttitor congue massa.

      Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet

      commodo magna eros quis urna.

      Nunc viverra imperdiet enim.

      """

실습문제 1

- 요구사항
  - 공백 문자, 탭 문자, 줄바꿈 문자, 마침표나 따옴표, 콤마 같은 특수 문자의 개수도 각각 센다
  - 대문자와 소문자를 구별하지 않고 글자 수를 센다
  - 영문 알파벳은 소문자로 출력하고 개수 출력
  - 화면에 출력할 수 없는 공백 문자는 SPACE, 탭 문자는 TAB, 줄바꿈 문자는 NEWLINE으로 출력하고 개수 출력

실습문제 1

- 최종 코드

```
sentences = """Lorem ipsum dolor sit amet,
consectetuer adipiscing elit.
Maecenas porttitor congue massa.
Fusce posuere, magna sed pulvinar ultricies, purus
lectus malesuada libero, sit
amet commodo magna eros quis urna.
Nunc viverra imperdiet enim.
"""

# 빈 딕셔너리 생성
d = {}

lowerSentences = sentences.lower()
```

실습문제 1

```
# 각 글자들이 딕셔너리에 있는지 확인하고 1을
# 증가시키거나 1로 초기화
for s in lowerSentences:
    if s in d:
        d[s] += 1
    else:
        d[s] = 1

for k, v in d.items():
    if k == ' ':
        k = "SPACE"
    elif k == '\t':
        k = "TAB"
    elif k == '\n':
        k = "NEWLINE"
print(f"{k}:{v}")
```

실습문제 2

- 문제
  - 서울시에 살고 있는 구별 인구 중 10개를 가나다순으로 뽑아서 문자열(guPopulation)을 만들었다.
    - guPopulation = "강남구,233363,강동구,199088,강북구,144410,강서구,267442,관악구,273736,광진구,166638,구로구,180027,금천구,114402,노원구,217272,도봉구,138120"
  - 이 문자열을 이용해서 구:인구 형태의 딕셔너리를 구성
  - 사용자로 부터 정수 한 개를 입력 받고, 입력한 값보다 인구가 적은 구를 찾아 구의 이름과 인구를 화면에 출력하는 프로그램을 작성
- 요구사항
  - 사용자는 정수를 입력할 것이라고 가정
  - 만약 사용자가 입력한 정수값(num)보다 작은 구가 없으면 "인구가 num보다 작은 구가 없습니다 " 를 출력
  - 구:인구 형태로 출력

실습문제 2

- 최종 코드

```
guPopulation =
"강남구,233363,강동구,199088,강북구,144410,강서구,
267442,관악구,273736,광진
구,166638,구로구,180027,금천구,114402,노원구,21727
2,도봉구,138120"

# 문자열에서 리스트 구성. 리스트의 요소는 구, 인구,
# 구 인구, ...
lst = guPopulation.split(',')
```

실습문제 2

```
# 리스트로부터 딕셔너리 생성
d = {}
for i in range(0, len(lst), 2):
    d[lst[i]] = int(lst[i + 1])

# 인구 입력 받기
num = int(input("비교할 인구를 입력하세요: "))

# 딕셔너리에 있는 구/인구 정보와 인구 비교
count = 0
for gu, population in d.items():
    if population < num:
        print(f"{gu}:{population}")
        count += 1 # 출력된 구가 있음
# count == 0이면 출력된 구가 없음
if count == 0:
    print(f"인구가 {num}보다 작은 구가 없습니다")
```

집합(set) 자료구조

- 집합(set) 자료구조
  - 수학에서의 집합을 나타내는 자료구조
  - 순서가 없고, 동일한 데이터가 두 개 이상 존재할 수 없음
  - 집합 관련 연산(교집합, 합집합, 차집합) 지원
  - 중괄호로 표시

```
{값1, 값2, 값3, … }
```

  - 특정 요소에 접근 불가
  - 집합에는 리스트는 포함 못함(튜플은 가능)
  - 집합을 리스트로 변환

```
lst = list(집합)
```

  - 집합의 요소 개수 확인

```
num = len(집합)
```

집합(set) 자료구조

- 집합 구성 방법
  - 하드 코딩

```
{ 1, 2, 3, 'a', (1, 2, 3) }
```

  - 빈 집합 생성

```
s = set()
```

  - set()함수에 iterable 객체 전달(range객체, 문자열, 리스트, 튜플 등) 사용

```
set([1, 2, 3])
set("string")
set("yellow")
```

집합(set) 자료구조

- 요소 추가
  - add()함수 – 단일 요소 추가

```
a = set([1, 2, 3])
a.add("string")
```

  - update()함수 – 여러 개 요소 추가

```
a = set([1, 2, 3])
a.update("string")
```

집합(set) 자료구조

- 집합 연산
  - 교집합
    - & 연산자 이용

```
s1 = { 1, 2, 20, (1, 2), "문자열" }
s2 = { 1, 2, 3 }
ints = s1 & s2
```

    - intersection() 함수 이용

```
ints = s1.intersection(s2)
```

  - 합집합
    - | 연산자 이용

```
unions = s1 | s2
```

    - union() 함수 이용

```
unions = s1.union(s2)
```

집합(set) 자료구조

- 차집합
  - - 연산자 이용

```
diffs = s1 - s2
```

  - difference() 함수 이용

```
diffs = s1.difference(s2)
```

집합(set) 자료구조

- 요소가 집합에 있는지 확인
  - in 사용

```
s = { 1, 2, 20, (1, 2), "문자열" }
if 1 in s:
    print("1은 집합 s에 포함되어 있습니다")
else:
    print("1은 집합 s의 요소가 아닙니다")
```

실습문제 3

- 문제
  - 1~20 사이의 숫자 중에서 무작위로 중복되지 않는 숫자 10개를 뽑아서 출력
- 요구사항
  - 무작위 숫자 생성기로 1~20 사이의 정수를 생성한 후 저장
  - 새로운 무작위 숫자를 생성했을 때 기존에 저장되어 있던 숫자들과 비교해서 중복되지 않는 숫자이면 새로 저장하고, 중복된다면 버리고 새로운 숫자를 생성
  - 10개의 중복되지 않는 숫자들이 생성될 때까지 반복. 즉, 무한 반복하면서 10개가 채워지면 반복을 종료해야 함

실습문제 3

- 최종 코드

```
import random
numSet = set() # 빈 집합 생성

# 무작위로 10개의 중복되지 않는 숫자 생성
while True:
    n = random.randint(1, 20)
    numSet.add(n)
    if len(numSet) == 10:
        break

# 리스트로 변환 후 출력
numList = list(numSet)
for n in numList:
    print(n, end=' ')
```