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
  - 사용자로부터 이름과 나이를 입력받아 딕셔너리에 저장하고, 저장된 딕셔너리를 출력하는 프로그램을 작성하세요.

- 요구사항

실습문제 2

- 최종 코드

```
def store_in_dictionary():
    name = input("이름을 입력하세요: ")
    age = input("나이를 입력하세요: ")
    person = {"name": name, "age": age}
    print(person)
    print(type(person))

# 함수 호출
store_in_dictionary()
```

실습문제 3

- 문제- 딕셔너리 항목 접근
  - 사용자로부터 키 값을 입력받고, 해당 키에 대한 값을 딕셔너리에서 찾아 출력하세요. 키가 없는 경우 적절한 메시지를 출력하세요.

- 요구사항

실습문제 3

- 최종 코드

```
def access_dictionary():
    data = {'name': 'Alice', 'age': 30}
    key = input("찾고 싶은 키를 입력하세요: ")
    try:
        print(f"{key}의 값: {data[key]}")
    except KeyError:
        print(f"오류: {key}는 존재하지 않는 키입니다.")

# 함수 호출
access_dictionary()
```

집합(set) 자료구조

- 집합(set) 자료구조
  - 수학에서의 집합을 나타내는 자료구조
  - 파이썬은 집합을 표현하는 세트(set)라는 자료형을 제공함
  - 세트는 합집합, 교집합, 차집합 등의 연산이 가능함

집합(set) 자료구조

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

  - `{값1, 값2, 값3, … }`
  - 집합에는 리스트는 포함 못함(튜플은 가능)
  - 집합을 리스트로 변환

  - `lst = list(집합)`

  `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

- 집합(set) 자료구조
  - 세트를 출력해보면 요소의 순서가 랜덤하게 나옴

```
>>> fruit ={'a','b','c','d'}
>>> fruit
{'b', 'c', 'a', 'd'}
```

  - 세트에 들어가는 요소는 중복될 수 없음

```
>>> fruit ={'a','b','c','d','d'}
>>> fruit
{'b', 'c', 'a', 'd'}
```

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 4

- 문제
  - 사용자로부터 여러 개의 숫자를 입력받아 집합을 생성하고 출력하는 프로그램을 작성하세요
- 요구사항

실습문제 4

- 최종 코드

```
def create_and_print_set():
    numbers = input("숫자들을 공백으로 구분하여 입력하세요:
").split()
    number_set = set(numbers)
    print("생성된 집합:", number_set)

# 함수 호출
create_and_print_set()
```

실습문제 5

- 문제
  - **딕셔너리 업데이트**
  - 사용자로부터 딕셔너리에 추가할 키와 값을 입력받고, 기존 딕셔너리에 추가한 후 최종 딕셔너리를 출력하세요.

- 요구사항

실습문제 5

- 최종 코드

```
def update_dictionary():
    data = {'name': 'Alice', 'age': 30}
    key = input("추가할 키를 입력하세요: ")
    value = input("추가할 값을 입력하세요: ")
    data[key] = value
    print("업데이트된 딕셔너리:", data)

# 함수 호출
update_dictionary()
```

실습문제 6

- 문제
  - **딕셔너리 삭제**
  - 사용자로부터 삭제할 키를 입력받고, 해당 키를 딕셔너리에서 삭제한 후 결과를 출력하세요.
- 요구사항
  - 키가 없는 경우 오류 메시지를 출력하세요.

실습문제 6

- 최종 코드

```
def delete_from_dictionary():
    data = {'name': 'Alice', 'age': 30, 'city': 'New York'}
    key = input("삭제할 키를 입력하세요: ")
    if key in data:
        del data[key]
        print("삭제 후 딕셔너리:", data)
    else:
        print(f"오류: {key}는 존재하지 않는 키입니다.")

# 함수 호출
delete_from_dictionary()
```

실습문제 7

- 문제
  - **딕셔너리 키 리스트**
  - 주어진 딕셔너리에서 모든 키를 리스트로 만들어 출력하세요.

- 요구사항
  - 추가로 값을 얻어오기 위해 values() 활용해 보자.

실습문제 7

- 최종 코드

```
def list_dictionary_keys():
    data = {'name': 'Alice', 'age': 30, 'city': 'New York'}
    keys = list(data.keys())
    print("딕셔너리의 키들:", keys)

# 함수 호출
list_dictionary_keys()
```

실습문제 8

- 문제
  - 딕셔너리의 모든 아이템을 순회하면서 키와 값을 출력하는 프로그램을 작성하세요
- 요구사항

실습문제 8

- 최종 코드

```
def iterate_dictionary_items():
    data = {'name': 'Alice', 'age': 30, 'city': 'New York'}
    for key, value in data.items():
        print(f"{key}: {value}")

# 함수 호출
iterate_dictionary_items()
```

실습문제 9

- 문제:병합
  - 두 개의 딕셔너리를 병합하고 결과를 출력하는 프로그램을 작성

- 요구사항
  - dict1 = {'name': 'Alice', 'age': 30}
  - dict2 = {'city': 'New York', 'country': 'USA'}

실습문제 9

- 최종 코드

```
def merge_dictionaries():
    dict1 = {'name': 'Alice', 'age': 30}
    dict2 = {'city': 'New York', 'country': 'USA'}
    dict1.update(dict2)
    print("병합된 딕셔너리:", dict1)

# 함수 호출
merge_dictionaries()
```

실습문제 10

- 문제 - 기본값 설정
  - 사용자로부터 키를 입력받고, 해당 키로 딕셔너리에서 값을 조회합니다.

- 요구사항
  - 키가 없는 경우 'Not Found'를 기본값으로 출력하세요.

실습문제 10

- 최종 코드

```
def get_with_default():
    data = {'name': 'Alice', 'age': 30}
    key = input("값을 조회할 키를 입력하세요: ")
    value = data.get(key, 'Not Found')
    print(f"{key}: {value}")

# 함수 호출
get_with_default()
```

실습문제 11

- 문제 – 조건 적용
  - 주어진 딕셔너리에서 값이 20보다 큰 아이템만 포함하는 새 딕셔너리를 생성하여 출력하세요.

- 요구사항
  - data = {'Alice': 25, 'Bob': 19, 'Cathy': 34, 'Dan': 20}

실습문제 11

- 최종 코드

```
def filter_dictionary():
    data = {'Alice': 25, 'Bob': 19, 'Cathy': 34, 'Dan': 20}

    filtered_data = {k: v for k, v in data.items() if v > 20}

    print("값이 20보다 큰 아이템:", filtered_data)

# 함수 호출
filter_dictionary()
```

실습문제 12

- 문제-요소추가
  - 사용자로부터 숫자 요소를 입력받아, 집합에 추가한 후, 집합을 출력하는 프로그램을 작성하세요.

- 요구사항
  - Set – 1,2,3

실습문제 12

- 최종 코드

```
def add_to_set():
    my_set = {1, 2, 3}
    new_element = int(input("추가할 요소를 입력하세요: "))
    my_set.add(new_element)
    print("업데이트된 집합:", my_set)

# 함수 호출
add_to_set()
```

실습문제 13

- 문제 – 집합 요소 제거하기
  - 사용자로부터 요소를 입력받아 집합에서 해당 요소를 제거한 후, 집합을 출력하는 프로그램을 작성하세요. 요소가 집합에 없는 경우 적절한 메시지를 출력하세요.

- 요구사항
  - 집합 1,2,3,4,5

실습문제 13

- 최종 코드

```
def remove_from_set():
    my_set = set([1, 2, 3, 4, 5])
    element = int(input("제거할 요소를 입력하세요: "))
    try:
        my_set.remove(element)
    except KeyError:
        print("해당 요소가 집합에 없습니다.")
    print("업데이트된 집합:", my_set)

# 함수 호출
remove_from_set()
```

실습문제 14

- 문제
  - 두 집합을 생성하고, 두 집합의 교집합을 계산하여 출력하는 프로그램을 작성하세요

- 요구사항
  - {1, 2, 3, 4, 5} , {4, 5, 6, 7, 8} 활용

실습문제 14

- 최종 코드

```
def intersection_of_sets():
    set1 = {1, 2, 3, 4, 5}
    set2 = {4, 5, 6, 7, 8}
    intersection = set1 & set2
    print("두 집합의 교집합:", intersection)

# 함수 호출
intersection_of_sets()
```

실습문제 15

- 문제 - 합
  - 두 집합의 합집합을 계산하여 출력하는 프로그램을 작성하세요.

- 요구사항
  - {1, 2, 3}
  - {3, 4, 5}

실습문제 15

- 최종 코드

```
def union_of_sets():
    set1 = {1, 2, 3}
    set2 = {3, 4, 5}
    union = set1 | set2
    print("두 집합의 합집합:", union)

# 함수 호출
union_of_sets()
```

실습문제 16

- 문제 – 차
  - 한 집합에서 다른 집합을 빼고 결과를 출력하는 프로그램을 작성하세요.

- 요구사항
  - {1, 2, 3, 4, 5}
  - {4, 5, 6, 7}

실습문제 16

- 최종 코드

```
def difference_of_sets():
    set1 = {1, 2, 3, 4, 5}
    set2 = {4, 5, 6, 7}
    difference = set1 - set2
    print("집합1에서 집합2를 뺀 차집합:", difference)

# 함수 호출
difference_of_sets()
```

실습문제 17

- 문제 – 대칭 차 구하기
  - 두 집합의 대칭 차집합(양쪽 집합 중 어느 한쪽에만 속하는 요소들의 집합)을 계산하여 출력하는 프로그램을 작성하세요.

- 요구사항
  - {1, 2, 3, 4}
  - {3, 4, 5, 6}

실습문제 17

- 최종 코드

```
def symmetric_difference_of_sets():
    set1 = {1, 2, 3, 4}
    set2 = {3, 4, 5, 6}
    symmetric_difference = set1 ^ set2
    print("두 집합의 대칭 차집합:", symmetric_difference)

# 함수 호출
symmetric_difference_of_sets()
```

실습문제 18

- 문제
  - 사용자로부터 여러 개의 아이템을 입력받아, 각 아이템이 몇 번 등장하는지 딕셔너리로 만들어 출력하세요.

- 요구사항
  - 입력값을 Dict 로 변환 출력

실습문제 18

- 최종 코드

```
def count_items():
    items = input("아이템을 공백으로 구분하여 입력하세요:
").split()
    count_dict = {}
    for item in items:
        if item in count_dict:
            count_dict[item] += 1
        else:
            count_dict[item] = 1
    print("아이템 카운트:", count_dict)

# 함수 호출
count_items()
```

실습문제 19

- 문제
  - 사용자로부터 여러 개의 숫자를 입력받고, 중복을 제거한 후, 각 숫자가 몇 번 입력되었는지 출력하세요.

- 요구사항
  - Dict , set 활용

실습문제 19

- 최종 코드

```
def remove_duplicates_and_count():
    numbers = map(int, input("숫자들을 공백으로 구분하여 입력하세요:
").split())
    number_dict = {}
    for number in numbers:
        if number in number_dict:
            number_dict[number] += 1
        else:
            number_dict[number] = 1
    unique_numbers = set(number_dict.keys())
    print("고유 숫자:", unique_numbers)
    print("각 숫자의 카운트:", number_dict)

# 함수 호출
remove_duplicates_and_count()
```

실습문제 20

- 문제
  - 두 문자열을 입력받고, 두 문자열에 공통으로 포함된 문자들을 출력하세요.

- 요구사항

실습문제 20

- 최종 코드

```
def common_characters():
    string1 = input("첫 번째 문자열을 입력하세요: ")
    string2 = input("두 번째 문자열을 입력하세요: ")
    set1 = set(string1)
    set2 = set(string2)
    common_chars = set1 & set2
    common_chars = sorted(common_chars)
    print("두 문자열에 공통으로 포함된 문자:", common_chars)

# 함수 호출
common_characters()
```


CNN