

## 4장. 딕셔너리, 튜플, Set



# 딕셔너리(Dictionary)

## ◆ 딕셔너리

리스트 처럼 여러 개의 값을 저장할 수 있고, 키(key)와 값(value)으로 대응시켜 저장하는 자료구조이다.

중괄호{ }를 사용한다.

딕셔너리 이름 = { 키:값, 키:값.... }

{ 'name': '한국민', 'age': 28 }

dictionary

키	키	키
값	값	값

# 딕셔너리(Dictionary)

## ◆ 딕셔너리 생성 및 관리

```
person = {} #빈 딕셔너리
print(person) # 딕셔너리 객체 출력

# 요소 추가
person['name'] = "오상식"
person['age'] = 35
person['phone'] = "010-1234-5678"

# 객체 출력
print(person)
print(type(person)) #자료형

# 특정 요소 출력
print(person['name'])
```

# 딕셔너리(Dictionary)

## ◆ 딕셔너리 생성 및 관리

```
# 특정 요소 수정
person['name'] = "최지능"

# 요소 삭제
del person['age']

# 전체 출력
for key in person:
    print(key, ': ', person[key])
```

```
{  
'name': '오상식', 'age': 35, 'phone': '010-1234-5678'}  
<class 'dict'>  
오상식  
name : 최지능  
phone : 010-1234-5678
```

# 딕셔너리(Dictionary)

## ◆ 딕셔너리 주요 메서드

함수	사용 예
d[key] = value	d = {'Tomas':13, 'Jane':9} d['Mike'] = 10    # 요소 추가 {'Tomas':13, 'Jane':9, 'Mike':10 }
del d[key]	del d['Jane']    #요소 삭제 {'Tomas':13, 'Mike':10 }
d.pop(key)	d.pop('Mike') 10 {'Tomas':13}
clear()	d.clear()    # d={} 빈 딕셔너리
d.keys()	d.keys()    # 모든 키 가져오기 d_keys(['Tomas', 'Mike'])
d.values()	d.Values()    # 모든 값 가져오기 d_values([13, 10])

# 딕셔너리(Dictionary)

## ◆ dictionary 메서드 사용

```
student = {'정우': 13, '유진': 9}
print(student)  #{'정우': 13, '유진': 9}

print(student.keys())
print(student.values())

# 요소 추가
student['민영'] = 11

# 요소 수정 - 키로 검색
student['유진'] = 8

# 요소 삭제 - 키로 삭제
student.pop('정우')

print(student)  #{'유진': 8, '민영': 11}

for st in student:
    print(st, ': ', student[st])
```

# 딕셔너리(Dictionary)

## ● 용어 사전 만들기

♣ 컴퓨터 용어 사전 ♣

검색할 용어를 입력하세요(종료: q or Q): 이진수

컴퓨터가 사용하는 0과 1로 이루어진 수

검색할 용어를 입력하세요(종료: q or Q): 버그

프로그램이 적절하게 동작하는데 실패하거나 오류가 발생하는 코드 조각

검색할 용어를 입력하세요(종료: q or Q): 함수

정의된 단어가 없습니다.

검색할 용어를 입력하세요(종료: q or Q): q

프로그램 종료!

1. Dictionary 자료구조에 컴퓨터 용어와 정의를 저장한다.
2. 용어를 계속 반복해서 검색 할 수 있다.
3. 검색한 용어가 없으면 정의된 단어가 없음을 알려준다.
4. 검색을 종료하려면 'q' 또는 'Q'를 입력한다.

# 딕셔너리(Dictionary)

- 용어 사전 만들기

```
# try ~ except 구문을 사용 - 예외 처리
print("♠ 컴퓨터 용어 사전 ♠")
print()

# 딕셔너리 생성
dic = {
    "이진수" : "컴퓨터가 사용하는 0과 1로 이루어진 수",
    "알고리즘": "어떤 문제를 해결하기 위해 정해진 일련의 절차",
    "버그": "프로그램이 적절하게 동작하는데 실패하거나 \
오류가 발생하는 코드 조각"
}
```



# 딕셔너리(Dictionary)

- 용어 사전 만들기

```
while True:
    try:
        word = input("검색할 용어를 입력하세요(종료: q or Q): ")

        if word == 'q' or word == 'Q':
            print("프로그램 종료!")
            break
        else:
            definition = dic[word] #키로 값을 검색
            print(definition)
    except KeyError:
        print("정의된 단어가 없습니다.")
```

# 학생의 성적 관리

## ● 학생의 성적 통계

학생 4명의 국어, 영어, 수학 과목의 합계 및 평균 계산하기

```
student_list = [  
    {"name": "이대 한", "kor": 80, "eng": 80, "math": 75},  
    {"name": "박민 국", "kor": 70, "eng": 65, "math": 60},  
    {"name": "오상식", "kor": 75, "eng": 70, "math": 50},  
    {"name": "최지 능", "kor": 90, "eng": 95, "math": 90}  
]  
  
# 첫번째 요소 검색  
print(student_list[0])  
  
print("=====  
print(" 이름  국어  영어  수학")  
for student in student_list:  
    print(f'{student["name"]} {student["kor"]} {student["eng"]} {student["math"]}')  
===== 성적표 =====")
```

# 학생의 성적 관리

## ● 학생의 성적 통계

```
# 개인별 총점과 평균
print("== 개인별 총점과 평균 ==")
print(" 이름   총점   평균")
for student in student_list:
    total = student["kor"] + student["eng"] + student["math"]
    avg = total / 3
    print(f'{student["name"]} {total} {avg:.2f}')

# 과목별 총점과 평균
sum_subj = [0, 0, 0]
avg_subj = [0.0, 0.0, 0.0]

# 과목별 총점 계산
for student in student_list:
    sum_subj[0] += student["kor"]
    sum_subj[1] += student["eng"]
    sum_subj[2] += student["math"]
```

```
===== 성적표 =====
이름   국어  영어  수학
이대한  80  80  75
박민국  70  65  60
오상식  75  70  50
최지능  90  95  90
== 개인별 총점과 평균 ==
이름   총점   평균
이대한  235  78.33
박민국  195  65.00
오상식  195  65.00
최지능  275  91.67
최지능  275  91.67
```

# 학생의 성적 관리

## ● 학생의 성적 통계

```
print("== 과목별 총점 ==")
print(f'국어 총점 : {sum_subj[0]}')
print(f'영어 총점 : {sum_subj[1]}')
print(f'수학 총점 : {sum_subj[2]}')

# 과목별 평균 계산
for student in student_list:
    avg_subj[0] = sum_subj[0] / len(student_list)
    avg_subj[1] = sum_subj[1] / len(student_list)
    avg_subj[2] = sum_subj[2] / len(student_list)

print("== 과목별 평균 ==")
print(f'국어 평균 : {avg_subj[0]:.1f}')
print(f'영어 평균 : {avg_subj[1]:.1f}')
print(f'수학 평균 : {avg_subj[2]:.1f}')
```

```
== 과목별 총점 ==
국어 총점 : 315
영어 총점 : 310
영어 총점 : 310
수학 총점 : 275
== 과목별 평균 ==
국어 평균 : 78.8
영어 평균 : 77.5
수학 평균 : 68.8
```

## 실습 문제 – 딕셔너리

다음의 실행 결과가 나오도록 빈 칸을 작성하시오.(파일: member.py)

```
member = {"이름": "신유빈", "나이": 20, "특기": "탁구"}  
result =   
  
print(member)  
print(result)
```

👉 실행 결과

```
{'이름': '신유빈', '특기': '탁구'}  
20
```

# 튜플(tuple)

- 튜플(tuple)

- 튜플의 요소를 변경(추가, 수정, 삭제)할 수 없다.
- 요소 추가는 초기화나 튜플간 합치기를 하면 가능함
- 리스트처럼 동일한 방식으로 인덱싱과 슬라이싱 가능함
- 소괄호( ) 를 사용한다.

튜플 이름 = (요소1, 요소2....)

```
t1 = ()  
t2 = (1, )  
t3 = (1, 2, 3)  
t4 = ('a', 'b', 'c')
```

# 튜플(tuple)

- 튜플 자료형

```
# 빈 튜플 생성
t1 = ()
print(t1)

# 요소 1개 저장
# t2 = (1) #코머를 붙이지 않으면 정수로 인식함
# print(type(t2))

t2 = (1, )
print(t2)
print(type(t2))

# 여러개 요소 저장
t3 = (2, 3, 4)

# t3 인덱싱
print(t3[0])
print(t3[-1])
```

```
()
(1,)
<class 'tuple'>
2
4
(1, 2, 3, 4)
```

# 튜플(tuple)

- 튜플 자료형

```
# t3 수정 및 삭제 불가  
# t3[1] = 10  
# del t3[2]
```

```
# 튜플 합치기  
t4 = t2 + t3  
print(t4)
```

```
# 문자 저장 튜플  
t5 = ('a', 'b', 'c', 'd')  
print(t5)
```

```
# t3 슬라이싱  
print(t5[1:3])  
print(t5[1:])  
print(t5[0:4])  
print(t5[:])
```

튜플의 요소는 변경할 수 없다.

```
Traceback (most recent call last):  
File "d:\korea_IT\pyworks\tuple_set\tuple_ex.py", line 19, in <module>  
    t3[1] = 10  
    ~~~~  
TypeError: 'tuple' object does not support item assignment
```

```
('a', 'b', 'c', 'd')  
( 'b', 'c')  
( 'b', 'c', 'd')  
( 'a', 'b', 'c', 'd')  
( 'a', 'b', 'c', 'd')
```

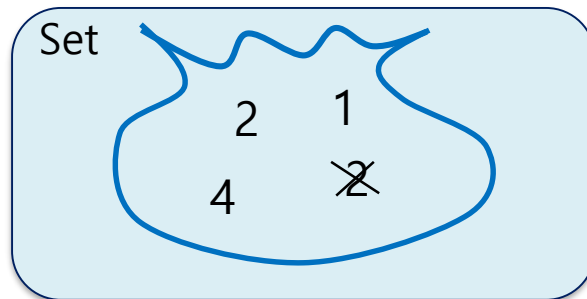


# 집합 자료형(set)

- 집합 자료형

- 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료구조
- 중복을 허용하지 않고 , 순서가 없다.
- 중괄호 { }를 사용한다.

집합 이름 = { 요소1, 요소2, 요소3 }



# 집합 자료형(set)

## ● 집합 자료형 예제

```
s1 = set() #빈 집합 생성
print(s1)
print(type(s1))

# 요소 중복 불가함
s2 = {1, 2, 3, 2, 3}
print(s2)

# 인덱싱 불가함
# print(s2[1])

# set 자료 생성
s3 = {'s', 'k', 'y'}
print(s3) # 순서 없이 출력됨
```

```
set()
<class 'set'>
{1, 2, 3}
{'k', 's', 'y'}
```

# 집합 자료형(set)

## ● 집합 자료형

연산자	집합
&	교집합
	합집합
a-b	차집합

```
a = {1, 2, 3}
b = {2, 3, 4}
```

```
c = a & b    # 교집합
d = a | b    # 합집합
e = a - b    # 차집합
```

```
print(c)
print(d)
print(e)
```

```
{1}
{1, 2, 3, 4}
{1, 3, 4}
```

# 집합 자료형(set)

- 리스트에서 중복 제거

```
print("*** 중복 제거 ***")
a = [1, 1, 1, 2, 3, 3] # 리스트 생성
set_v = set(a)        # 집합 자료

print(set_v)
print(list(set_v))
```

```
{1, 2, 3}
[1, 2, 3]
```

# 집합(set) 관련 메서드

- set 관련 메서드

함수	사용 예
<b>add(x)</b>	s = {1, 2, 3} s.add(4)    # 요소 추가 {1, 2, 3, 4}
<b>remove(x)</b>	s1 = {1, 3, 4} s1.remove(3) {1, 2}
<b>clear()</b>	s1 = {1, 3, 4} s1.clear()    # 모두 지우기 set()
<b>x in s</b>	fruits = {"apple", "banana", "grape"} "apple" in fruits True "grape" not in fruits    #False

# 집합 자료형(set)

- set 관련 메서드

```
numbers = {1, 2, 3}
```

```
# 요소 추가
```

```
numbers.add(4)
```

```
print(numbers)
```

```
# 요소 수정 불가
```

```
# numbers[3] = 10
```

```
# 요소 삭제
```

```
numbers.remove(2)
```

```
print(numbers)
```

```
{1, 2, 3, 4}
```

```
{1, 3, 4}
```

```
{'peach', 'apple', 'banana'}
```

```
True
```

```
False
```

```
False
```

# 집합 자료형(set)

- set 관련 메서드

```
# set 자료 생성
fruits = {"apple", "banana", "grape"}

# 요소 추가
fruits.add("peach")

# 요소 삭제
fruits.remove("grape")
print(fruits)

print("apple" in fruits)
print("banana" not in fruits)
print("strawberry" in fruits)
```