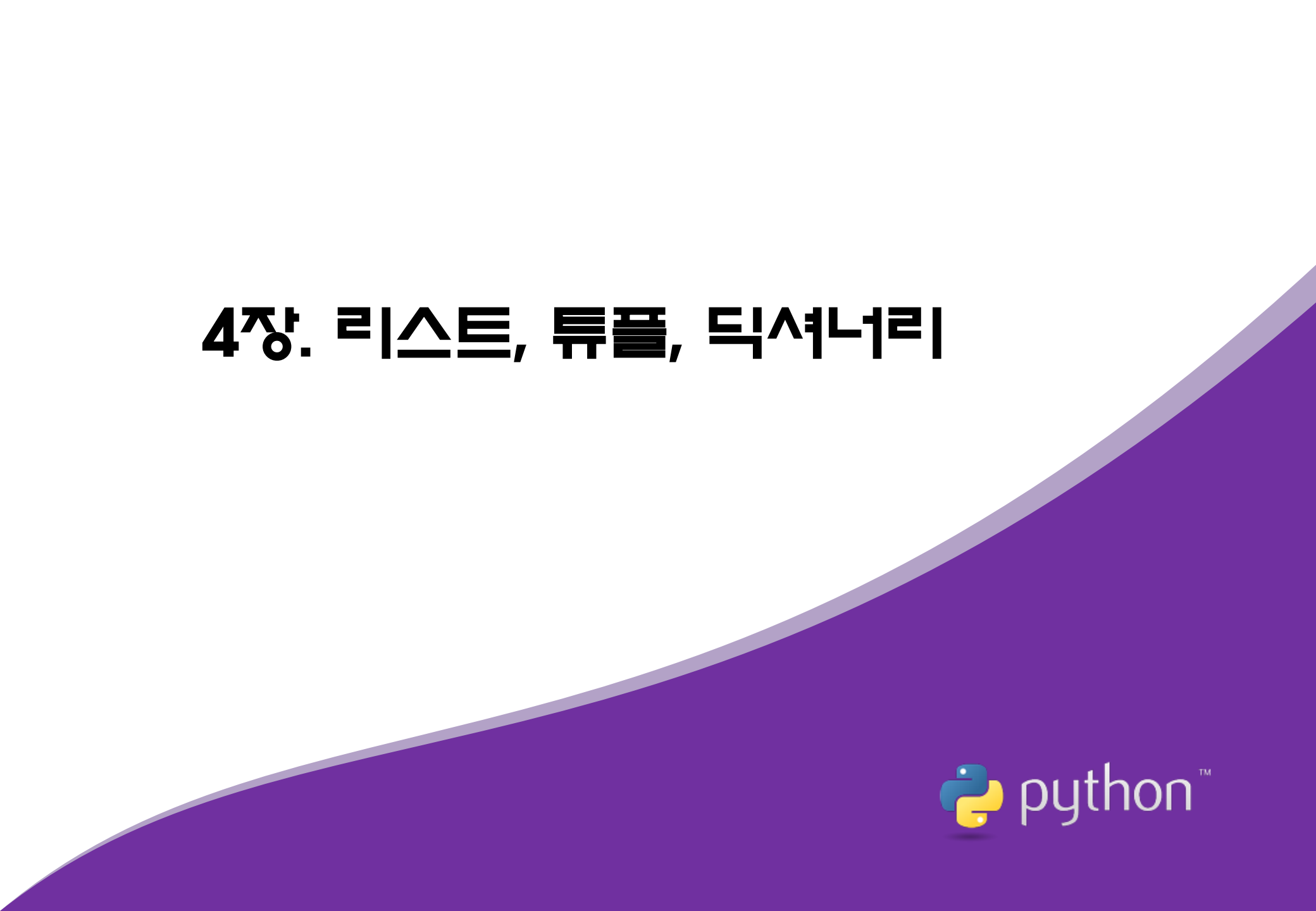
4장. 리스트, 튜플, 딕셔너리



목 차

리스트 사용의 필요성

리스트 사용의 필요성

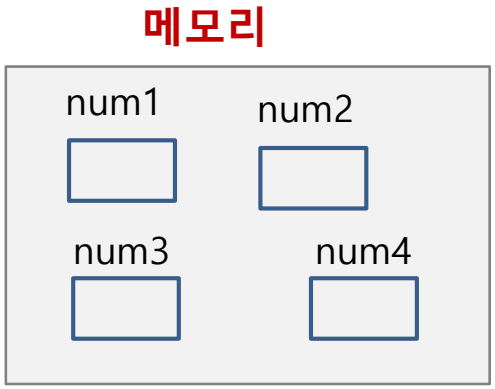
- 정수 10개를 이용한 학생의 성적 프로그램을 만들때

  10개의 변수를 선언

  num1, num2, num3... num10;

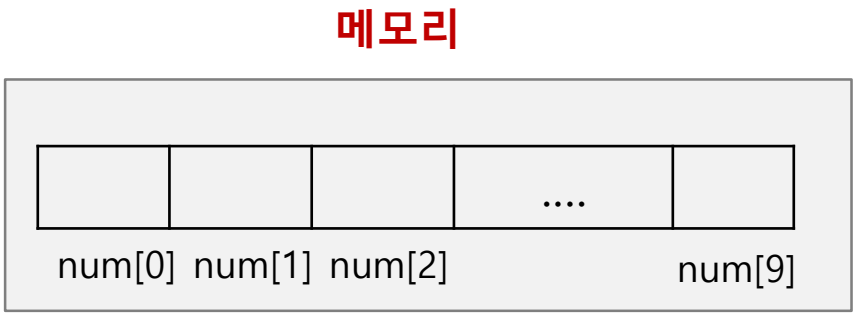
  정보가 흩어진 채 저장되고, 변수 이름이 많아

  비효율적이고 관리하기 어렵다.



리스트 사용의 장점

- 인덱스를 이용하여 순차(순서)적으로 관리할 수 있다 -> 효율적이다.



리스트(list)란?

리스트(list)란?

- 여러 개의 연속적인 값을 저장하고자 할 때 사용하는 자료형이다.

리스트의 생성

리스트 이름 = [요소1, 요소2, 요소3...]

season = [“봄”, “여름”, “가을”, “겨울”]

number = [1. 2, 3, 4, 5]

리스트(list)의 생성

- 리스트의 생성 및 인덱싱

```
# 리스트 생성 및 사용
seasons = ["봄", "여름", "가을", "겨울"]

# 1개 출력(인덱싱)
print(seasons[0])
print(seasons[-1])

# 리스트 출력
print(seasons)
print(type(seasons))

# 슬라이싱
# [0:n] -> n-1 위치
print(seasons[0:3])
print(seasons[:3])
print(seasons[2:4])
print(seasons[2:])

# 전체 요소 출력
for i in seasons:
    print(i)
```

리스트(list)의 주요 함수

- 리스트의 주요 메서드(함수)

함수	기능	사용 예
append()	요소 추가	a = [1, 2, 3] a.append(4) a = [1, 2, 3, 4]
insert()	특정 위치에 추가	a = [2, 4, 5] a.insert(1,3) #1번 위치에 3 삽입 a = [2, 3, 4, 5]

pop()	요소 삭제	a = [1, 2, 3, 4, 5]
		a.pop() # 마지막 위치의 요소 제거
		a = [1, 2, 3, 4]
		a.pop(1) #1 위치의 2 제거
		a = [1, 3, 4]

remove()	특정 요소 삭제	s = ['모닝', 'BMW', 'BENZ', '스포티지']
		s.remove('BMW') #요소 직접 삭제
		s = ['모닝', 'BENZ', '스포티지']

len()	리스트의 개수	s = ['모닝', 'BMW', 'BENZ', '스포티지'] len(s) 4

리스트(list)의 활용

- 리스트의 관리

```
# 리스트 추가, 조회, 수정, 삭제
score = [ 10, 20, 30, 40 ]

# 수정
score[1] = 50
print(score)

# 삭제
del score[2]
print(score)
```

```
# 요소 추가 - append 함수
score.append(60) # 맨 뒤로 추가
print(score)
score.insert(1, 20) # 1번 위치에 추가
print(score)

# 요소 삭제
score.remove(50)
print(score)

# 요소의 개수
print(len(score))
```

리스트(list) 반복 – in 사용

- 〈값〉 in [list]

  리스트 내부에 값이 있으면 True, 없으면 False

```
>>> a = [273, 32, 105, 47, 81]
>>> 47 in a
True
>>> 105 in a
True
>>> for i in a:
        print(i)


273
32
105
47
81
>>> for i in a:
        if i < 100:
            print(i)


32
47
81
```

리스트(list)의 연산

◆ 리스트의 연산 – 개수, 합계, 평균 구하기

```
# 점수 리스트 생성
score = [70, 80, 55, 60, 90, 40]
total = 0

# 합계, 개수, 평균
for i in score:
    total += i
    print(i)

count = len(score) #len(리스트) - 개수
avg = total / count

print("개수 : %d개" % count)
print("합계 : %d점" % total)
print("합계 : %d점" % sum(score)) # sum(리스트)-합계
print("평균 : %.2f점" % avg)
```

실습 문제

실습) 총 5명의 학생의 시험점수가 60점 이상이면 합격, 아니면 불합격

```
==== for ~ in 사용 ====
1번 학생은 합격입니다.
2번 학생은 불합격입니다.
3번 학생은 합격입니다.
4번 학생은 불합격입니다.
5번 학생은 합격입니다.
```

```
print("for ~ range() 사용")
n = len(scores)
for i in range(0, n):
    if scores[i] >= 60:
        print("%d번 학생은 합격입니다." % (i+1))
    else:
        print("%d번 학생은 불합격입니다." % (i+1))
```

```
print("==== for ~ in 사용 ====")
scores = [90, 45, 67, 59, 85]
index = 0
for score in scores:
    index = index + 1
    if score >= 60:
        #print(index, "번 학생은 합격입니다.")
        print("%d번 학생은 합격입니다." % index)
    else:
        #print(index, "번 학생은 불합격입니다.")
        print("%d번 학생은 불합격입니다." % index)
```

리스트(list)의 연산

◆ 리스트의 최대값과 최대값 위치 찾기

```
# 점수 리스트 생성
score = [70, 80, 55, 60, 90, 40]

# 최고 점수
max = score[0]
for i in score:
    if max < i:
        max = i

print("최고 점수 : %d" % max)

# 최고 점수의 위치
max_idx = 0
n = len(score)
for i in range(1, n):
    if score[max_idx] < score[i]:
        max_idx = i

print("최고 점수의 위치 : %d" % max_idx)
```

리스트(list)의 정렬

◆ 리스트의 정렬 – 오름차순, 내림차순 정렬

```
# 점수 리스트 선언 및 생성
score = [70, 80, 50, 60, 90, 40]

# 오름차순 정렬
n = len(score)
for i in range(0, n):
    for j in range(0, n-1):
        if score[j] > score[j+1]:
            score[j], score[j+1] = score[j+1], score[j]

        '''
        # 1행
          70, 50, 60, 80, 40, 90
        # 2행
          50, 60, 70, 40, 80, 90
        # 3행
          50, 60, 40, 70, 80, 90
        # 4행
          50, 40, 60, 70, 80, 90
        # 5행
          40, 50, 60, 70, 80, 90

        '''
for i in score:
    print(i)
```

두 수의 교환

```
>>> x = 1
>>> y = 2
>>> x, y = y, x
>>> x
2
>>> y
1
```

리스트(list) 복사

◆ 리스트의 복사

```
# 배열의 복사
a = [1, 2, 3, 4, 5, 6]
a2 = []

for i in a:
    a2.append(i)

print(a2)

# 홀수를 저장하는 리스트
a3 = []
for i in a:
    if i % 2 == 1:
        a3.append(i)

print(a3)
```

리스트(list) 내포

◆ 리스트 내포 사용하기

[표현식 for 항목(요소) in 리스트]

```
a = [1, 2, 3, 4]
result = []
for num in a:
    result.append(num*3)

print(result)
```

```
result = []
result = [num*3 for num in a]
print(result)
```

[3, 6, 9, 12]

```
result = [num*3 for num in a if num % 2 == 0]
```

[6, 12]

문자열 인덱싱, 슬라이싱

- 문자열 – 1차원 리스트이다.

  ※ 끝번호는 (끝번호 -1)과 같다

```
>>> say = "Have a nice day!"
>>> say[0]
'H'
>>> say[-1]
'!'
>>> say[-2]
'y'
>>> say[0:4]
'Have'
>>> say[:4]
'Have'
>>> say[7:]
'nice day!'
```

문자열 슬라이싱 및 결합

- 슬라이싱

```
>>> s = "20210621Rainy"
>>> year = s[:4]
>>> day = s[4:8]
>>> weather = s[8:]
>>> year
'2021'
>>> day
'0621'
>>> weather
'Rainy'
```

- 문자열 수정하기 - 결합

```
>>> w = 'Pithon'
>>> w[:1]
'P'
>>> w[2:]
'thon'
>>> w[:1] + 'y' + w[2:]
'Python'
```

문자열 출력 방법

- **문자열 포맷 코드**

"문자열 포맷" % 변수(상수)

코드	내 용
%d	정수(decimal)
%f	실수(float)
%s	문자열(string)

```
print("I eat %d apples" % 3)
print("I eat %s apples" % 'five')

number = 5
print("I eat %d apples" % number)
day = 3
print("I ate %d apples. so I was sick for %s days" % (number, day))
```

문자열 – 특별한 1차원 리스트

❖ 문자열 함수(메서드) 정리

메서드	설명

split()	>>> s = 'banana, grape, kiwi'
	>>> s = fruit.split(',') -> 구분기호(구분자)
	>>> s
	['banana', ' grape', ' kiwi']

replace()	>>> s = 'Hello, World'
	>>> s = s.replace('World', 'Korea')
	>>> s
	'Hello, Korea'

format()	>>> sentence = 'My name is {0}. I am {1} years old.'.format('Mario', 40)
	>>> sentence
	'My name is Mario. I am 40 years old.'

문자열 – 특별한 1차원 리스트

❖ 문자열 함수(메서드) 정리

메서드	설명

find()	>>> s = "Hello"
	>>> s.find('H')
	0
	>>> s.find('ll')
	2
	>>> s.find('k')
	-1
	문자열이 존재하는 위치. 없으면 -1반환

lstrip()
rstrip()
strip()
	>>> " hi, soo".lstrip()
	'hi, soo'
	>>> "hi, elsa ".rstrip()
	'hi, elsa'
	문자열의 공백제거

isnumeric()	>>> '123AB'.isnumeric() False

split() 예제

- split() 예제 – 콜론으로 구분하고 리스트로 반환

```
s = "a:b:c:d"
s = s.split(':')
print(s)
```

```
['a', 'b', 'c', 'd']
```

- 두 수를 동시에 입력 받아 더하기

```
# 두 수를 동시에 입력받아 공백으로 구분하기
n1, n2 = input("두 수 입력 : ").split()
x = int(n1)
y = int(n2)
add = x + y
print(add)
```

공백으로 분리하기

split() 예제

- 세 수를 동시에 입력 받아 더하기 – 리스트 사용

```
n = input("세 수 입력 : ")
li = n.split()
print(li)

total = 0
for i in li:
    total += int(i)

print("합계 :", total)
```

문자열 함수

- **format() 함수**

"{ 0 } ..... { 1 }".format(a, b)

```
# format() 함수
print("I eat {0} apples".format(3))

n = 5
print("I eat {0} apples".format(n))
day = 2
print("I ate {0} apples. so I was sick for {1} days".format(n, day))
```

문자열 함수

- stlip() 함수 – 공백 문자 제거

```
# 공백 문자 제거함수 - strip()

# 왼쪽 공백 제거 - lstrip()
str1 = "  hi, soo"
print(str1.strip())
print(str1.lstrip())

# 오른쪽 공백 제거 - rstrip()
str2 = "hi, soo   "
print(str2.strip())
```

```
txt = "     banana     "

x = txt.strip()

print("of all fruits", x, "is my favorite")
```

문자열 함수

- **find() 함수 – 공백 문자 제거**

```
# 문자열을 찾아서 위치를 반환하는 함수 - find()
# 중복될 경우 첫번째 위치를 반환하고 없으면 -1을 반환
str = "Hello"
print(str.find('H'))
print(str.find('ll'))
print(str.find('k'))

txt = "Hello, welcome to my world"
x = txt.find('welcome')
print(x)
```

2차원 리스트(list)

- 2차원 리스트의 선언 및 생성

a[0][0]	a[0][1]
a[1][0]	a[1][1]
a[2][0]	a[2][1]

```
>>> a = [[10,20],[30,40],[50,60]]
>>> a[0][0]
10
>>> a[0][1]
20
>>> a[1][0]
30
>>> a[1][1]
40
>>> a[2][0]
50
>>> a[2][1]
60
>>> for x, y in a:
        print(x, y)


10 20
30 40
50 60
```

2차원 리스트(list)

- 2차원 리스트의 크기 및 출력

```
a = [
      [10, 20],
      [30, 40],
      [50, 60]
    ]

print("리스트의 크기(행) : ", len(a))
print("리스트의 크기(열) : ", len(a[0]))
print("리스트의 크기(열) : ", len(a[1]))

# 리스트 내의 값 출력 - range() 사용
for i in range(len(a)):  #0~3
    for j in range(len(a[i])): #0~2
        print(a[i][j], end=' ')
    print()
```

```
# 리스트 내의 값 출력
for x, y in a:
    print(x, y)
```

2차원 리스트(list)

- 2차원 리스트의 추가 및 출력

```
# 리스트 추가
a.append([70, 80])
print(a)

# 리스트 내의 값 출력
for x, y in a:
    print(x, y)
```

```
[[10, 21], [30, 40], [50, 60], [70, 80]]
10 21
30 40
50 60
70 80
```

2차원 리스트(list)

- 2차원 리스트의 연산

```
# 연산
total = 0
avg = 0.0
count = 0  # 2차원 리스트는 개수를 세야함
for i in range(len(a)):  #0~3
    for j in range(len(a[i])): #0~2
        total += a[i][j]
        count += 1
    print()

avg = total / count # 평균 = 합계 / 개수
print("합계 : %d" % total)
print("평균 : %.2f" % avg)
```

2차원 리스트(list)

- **2차원 리스트 – 문자열과 함께 사용시**

리스트 이름 = [요소1, 요소2, **[요소1, 요소2, 요소3]**]

```
>>> e = [7, 3,['chicken', 'eagle', 'monkey']]
>>> e[0]
7
>>> e[-1]
['chicken', 'eagle', 'monkey']
>>> e[2]
['chicken', 'eagle', 'monkey']
>>> e[2][1]
'eagle'
```

이차원 리스트 요소에 접근

2차원 리스트(list)

이차원 리스트

아래의 a 리스트에서 실행 결과대로 출력해 보세요

☞ 실행 결과

```
['a', 'b']
```

```
a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]

    print(a[3][:2])

    print(a[3][0:2])
```

2차원 리스트(list)

- 2차원 리스트의 연산

```
d1 = [1, 2, 3]
d2 = [
        [80],
        [90],
        [100]
]

d1.append(10)   # 1차원 리스트는 값을 추가
print(d1)
"""
print(d2[0][0])
print(d2[1][0])
print(d2[2][0])

for i in d2:
    print([i][0], end=' ')
```

```
d2.append([70])  # 이차원 리스트는 리스트를 추가
print(d2)
"""
sum = d2[0][0] + d2[1][0] + d2[2][0]
d2.append([sum])  # 합계를 구하여 추가
avg = (d2[0][0] + d2[1][0] + d2[2][0]) / 3
d2.append([avg])  # 평균을 구하여 추가
print(d2)
```

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

  괄호( ) 를 사용한다.

  튜플 이름 = **(요소1, 요소2....)**

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

튜플(tuple)

```
# tuple 생성 및 관리
t1 = ()
print(t1)

t1 = (1, ) # 1개 추가시 콤머를 붙임
print(t1)
print(type(t1))

'''
t1 = (1)
print(t1)
print(type(t1)) - 정수
'''
```

```
t2 = (2, 3, 4)
print(t2)

#요소 추가 - 초기화나 튜플 합하기
t1 = t1 + t2
print(t1)
print(t2)

# 인덱싱과 슬라이싱
print(t2[0])
print(t2[0:])
print(t2[0:-1])
print(t2[1:2])

#수정 불가
#t2[2] = 5

#삭제 불가
#del t2[1]
```

튜플(tuple)

```
()
(1,)
<class 'tuple'>
(2, 3, 4)
(1, 2, 3, 4)
(2, 3, 4)
2
(2, 3, 4)
(2, 3)
(3,)
Traceback (most recent call last):
  File "C:/pyworks/ch04/tuple_ex.py", line 30, in <module>
    t2[2] = 5
TypeError: 'tuple' object does not support item assignment
```

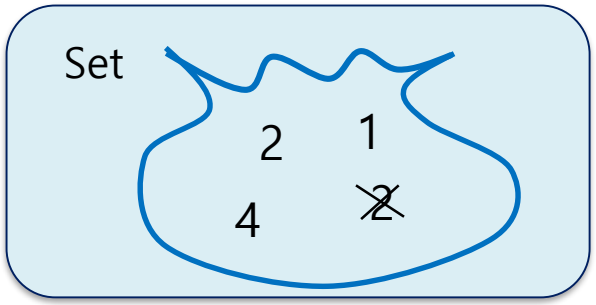
튜플의 요소는 변경할 수 없다.

집합 자료형(set)

- **집합 자료형**

  집합에 관련된 것을 쉽게 처리하기 위해 만든 자료구조로 중복을 허용하지 않고 , 순서가 없다.

집합 이름 = {요소1, 요소2, 요소3}



순서가 없고, 중복이 허용되지 않음

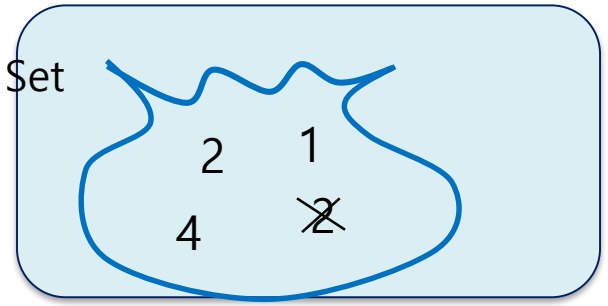
```
>>> s = {2, 4, 6, 8}
>>> s
{8, 2, 4, 6}
>>> say = set('Hello')
>>> say
{'H', 'o', 'e', 'l'}
```

집합 자료형(set)

- **집합 자료형**

  집합에 관련된 것을 쉽게 처리하기 위해 만든 자료구조로 중복을 허용하지 않고 , 순서가 없다.

집합 이름 = {요소1, 요소2, 요소3}




```
>>> s = set()
>>> s
set()
>>> s.add(1)
>>> s
{1}
>>> s.add(2)
>>> s
{1, 2}
>>> s.add(3)
>>> s
{1, 2, 3}
```

집합(set)

- **set 메서드**

함수	사용 예
add(x)	s = {1, 2, 3} s.add(4) # 요소 추가 {1, 2, 3, 4}
remove(x)	s1 = {1, 3, 4} s1.remove(3) {1, 2}

clear()	s1 = {1, 3, 4}
	s.clear() # 모두 지우기
	set()

x in s	fruits = {"apple", "banana", "grape"} "apple" in fruits True "grape" not in fruits #False

집합 자료형(set)

- **집합 자료형**

연산자	집합
&	합집합
|	교집합
a-b	차집합

```
>>> a = {1, 2, 3}
>>> b = {2, 3, 4}
>>> a & b
{2, 3}
>>> a | b
{1, 2, 3, 4}
>>> a - b
```

집합(set)

- set 메서드

```
>>> say = set('Hello')
>>> say
{'e', 'H', 'o', 'l'}
>>> s = { 1, 2, 2, 3, 3}
>>> s
{1, 2, 3}
```

순서가 없다.

중복이 허용되지 않음

리스트의 중복 제거

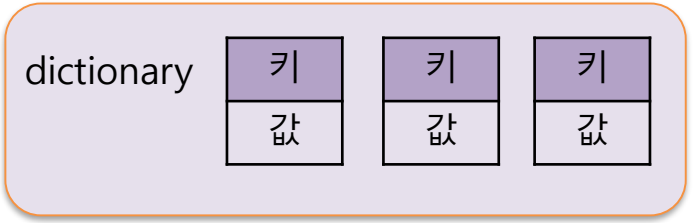
```
a = [1,1,1,2,2,3,3,3,4,4,5]
aSet = set(a)
print(aSet)
b = list(aSet)
print(b)
```

```
{1, 2, 3, 4, 5}
[1, 2, 3, 4, 5]
```

딕셔너리(Dictionary)

◆ 딕셔너리

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



딕셔너리 이름 = { **키:값, 키:값....**}

```
>>> person = {}
>>> person["name"] = "장그래"
>>> person
{'name': '장그래'}
>>> person["age"] = 28
>>> person
{'name': '장그래', 'age': 28}
>>> person["phone"] = "0101234567"
>>> person
{'name': '장그래', 'age': 28, 'phone': '0101234567'}
>>> del person["age"]
>>> person
{'name': '장그래', 'phone': '0101234567'}
```

빈 딕셔너리 생성

요소 추가하기

요소 추가 삭제시 키(key) 값을 사용

요소 삭제하기

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)

```
>>> dic={1:'A', 2:'B', 3:'C'}
>>> dic[4] = 'D'
>>> dic
{1: 'A', 2: 'B', 3: 'C', 4: 'D'}
>>> dic[4] = 'E'
>>> dic
{1: 'A', 2: 'B', 3: 'C', 4: 'E'}
>>> dic.pop(2)
'B'
>>> dic
{1: 'A', 3: 'C', 4: 'E'}
>>> dic.keys()
dict_keys([1, 3, 4])
>>> dic.values()
dict_values(['A', 'C', 'E'])
>>> for key in dic:
        print(key)
        print(dic[key])


1
A
3
C
4
E
```

Key가 같으면 중복 안되고, 값만 수정됨.

Key, Value값 얻는 함수 사용.

딕셔너리(Dictionary)

dictionary와 다른 자료 구조의 연동

list(dic)

tuple(dic)

set(dic)

```
dic = {'kor':90, 'eng':70, 'math':80}
print(dic.keys())

print(list(dic.keys()))  # 키를 리스트로 변환


print(dic.values())
print(list(dic.values())) # 값을 리스트로 변환

print(tuple(dic.values())) # 값을 튜플로 변환

print(set(dic.values())) # 값을 set으로 변환

# 키값만 가져옴
print(list(dic))
print(tuple(dic))
print(set(dic))
```

```
dict_keys(['kor', 'eng', 'math'])
['kor', 'eng', 'math']
dict_values([90, 70, 80])
[90, 70, 80]
(90, 70, 80)
{80, 90, 70}
['kor', 'eng', 'math']
('kor', 'eng', 'math')
{'eng', 'math', 'kor'}
```

학생의 성적 관리

- **학생의 성적 통계**

학생 5명의 국어, 수학, python 과목 성적 합계 및 평균 계산

```
students =[
    {'name': '이대한', 'korean': 88, 'math':77, 'python': 95},
    {'name': '장민국', 'korean': 82, 'math':65, 'python': 85},
    {'name': '오상식', 'korean': 58, 'math':83, 'python': 75},
    {'name': '천선란', 'korean': 99, 'math':91, 'python': 85},
    {'name': '김초엽', 'korean': 94, 'math':87, 'python': 75},
]

# 개인별 총점과 평균
print(' 이름  총점 평균')
for student in students:     # 학생을 1명씩 반복
    total = student['korean'] + student['math'] + student['python']
    avg = total / 3
    print("%s %d %.1f" % (student['name'], total, avg))   # 출력
```

학생의 성적 관리

학생 5명의 과목 별 합계와 평균

```
 이름   총점 평균
이대한 260 86.7
장민국 232 77.3
오상식 216 72.0
천선란 275 91.7
김초엽 256 85.3
국어 합계 : 421점
수학 합계 : 403점
python 합계 : 415점
국어 평균 : 84.2점
수학 평균 : 80.6점
python 평균 : 83.0점
```

```python
# 과목별 총점과 평균
sum_kor = 0
sum_math = 0
sum_py = 0
for student in students:
    sum_kor += student['korean']
    sum_math += student['math']
    sum_py += student['python']

avg_kor = sum_kor / 5
avg_math = sum_math / 5
avg_py = sum_py / 5
print("국어 합계 : %d점" % sum_kor)
print("수학 합계 : %d점" % sum_math)
print("python 합계 : %d점" % sum_py)
print("국어 평균 : %.1f점" % avg_kor)
print("수학 평균 : %.1f점" % avg_math)
print("python 평균 : %.1f점" % avg_py)
```

딕셔너리(Dictionary)

- 용어 사전 만들기

```
dic = {"알고리즘": "컴퓨터로 작업을 수행하기 위해 컴퓨터가 이해할 수 있도록 단계별로 설명해 놓은 것",
       "버그": "프로그램이 적절하게 동작하는데 실패하거나 또는 전혀 동작하지 않는 원인을 제공하는 코드 조각",
       "이진수": "2진법으로 나타낸 숫자, 0과 1로 구성됨",
       "html": "하이퍼 텍스트 마크업 언어로 웹 페이지를 구성하는 언어이다.",
       "CSS": "웹 페이지를 구성하는 요소로 디자인을 담당하는 웹 기술이다.",
       "함수": "재사용 가능한 코드 조각"
}

print(dic['이진수'])
print(dic['html'])
```

```
print("♣ 용어 사전 ♣")
x = input("정의되어 있는 단어를 입력하세요 : ")
try:
    definition = dic[x]
    print(definition)
except:
    print("찾는 단어가 없습니다.")
```