

## Ch2. 파이썬 기본 문법 익히기

---



1. 변수
2. 자료형(1)
3. 자료형(2)
4. 조건문
5. 반복문
6. 입출력
7. 함수, 람다
8. 모듈, 패키지, 클래스

▪ code: [https://github.com/zzhining/python\\_data\\_basic](https://github.com/zzhining/python_data_basic)

# 01 변수

## ■ 변수는 바구니

변수이름 = 데이터

```
In [1]: a = 5    # a라는 바구니에 값 5를 넣는다. [Enter]  
a       # a 안에 있는 값을 확인한다.    [▶Run 또는 Shift + Enter]
```

Out [1]: 5

```
In [2]: b = 3    # b라는 바구니에 값 3을 넣는다. [Enter]  
b       # b 안에 있는 값을 확인한다.    [▶Run 또는 Shift + Enter]
```

Out [2]: 3

```
In [3]: a + b    # a 바구니에 있는 값과 b 바구니에 있는 값을 더한다.    [▶Run 또는 Shift + Enter]
```

Out [3]: 8

```
In [4]: c = '가나다'  
c
```

Out [4]: '가나다'

```
In [5]: radio_freq = 107.9  
radio_freq
```

Out [5]: 107.9

## ■ 잘못된 변수명

- 숫자로 시작하는 변수
- 공백이 포함된 변수
- 기호가 포함된 변수(밑줄 제외)
- 예약어

```
: # 잘못된 변수명: 숫자로 시작하는 변수  
2var = "행복"
```

```
File "<ipython-input-6-f08ab471a553>", line 2
```

```
2var = "행복"
```

```
SyntaxError: invalid syntax
```

```
: # 잘못된 변수명: 공백이 포함된 변수  
happy var = "행복"
```

```
File "<ipython-input-7-57ed49f07157>", line 2
```

```
happy var = "행복"
```

```
SyntaxError: invalid syntax
```

## ■ 잘못된 변수명

- 숫자로 시작하는 변수
- 공백이 포함된 변수
- 기호가 포함된 변수(밑줄 제외)
- 예약어

```
: # 잘못된 변수명: 특수기호(!)가 포함된 변수  
happyvar! = "행복"
```

```
File "<ipython-input-5-cbb9d68a6486>", line 2
```

```
happyvar! = "행복"
```

```
SyntaxError: invalid syntax
```

```
: # 잘못된 변수명: 파이썬 예약어로 만든 변수  
def = "행복"
```

```
File "<ipython-input-9-4ffb7deb5118>", line 2
```

```
def = "행복"
```

```
SyntaxError: invalid syntax
```

## ■ 파이썬 변수명 특징

- 대소문자 구분
- 한 번에 여러 변수 선언 가능
- 하나의 값을 여러 변수에 담을 수 있음

```
abc = 5  
abc
```

5

```
ABC = "Apple"  
ABC
```

'Apple'

```
In [9]: x, y, z = "Apple", "Banana", "Carrot"
```

```
In [10]: x
```

Out [10]: 'Apple'

```
In [11]: y
```

Out [11]: 'Banana'

```
In [15]: z
```

Out [15]: 'Carrot'

```
In [12]: _, var = "Not Use", "Use"  
-
```

Out [12]: 'Not Use'

```
In [18]: var
```

Out [18]: 'Use'

```
In [13]: x = y = z = "Dog"  
x
```

Out [13]: 'Dog'

```
In [21]: y
```

Out [21]: 'Dog'

```
In [22]: z
```

Out [22]: 'Dog'

## 02 자료형(1) – 기본 데이터 타입



- type(변수이름)
  - 변수의 자료형을 반환하는 함수
- 파이썬의 기본 자료형
  - 숫자(numeric): 정수(integer), 실수(float)
  - 불리언(boolean): True, False
  - 문자형(String)

```
# 따옴표로 감싼 문자, 'Hello'와 "Hello"는 동일
a = 'Hello'
type(a)
```

str

```
# 따옴표로 감싼 숫자는 문자
b = '123'
type(b)
```

str

```
# 띄어쓰기가 포함된 문자열
c = "How are you?"
type(c)
```

```
x = """Twinkle, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky."""
x
```

'Twinkle, twinkle, little star,\nHow I wonder what you are!\nUp above the world so high,\nLike a diamond in the sky.'

## ■ 데이터 타입의 형 변환

- `int()`: 정수형으로 변환
- `float()`: 실수형으로 변환
- `bool()`: 불리언형으로 변환
- `str()`: 문자열로 변환

```
temperature = '20'  
humidity = '50'  
  
temperature + humidity  
  
'2050'
```

```
int(temperature) + int(humidity)  
  
70
```

```
a = 0  
b = 500  
c = ''  
d = '하하호호'
```

```
bool(a)
```

False

```
bool(b)
```

True

```
bool(c)
```

False

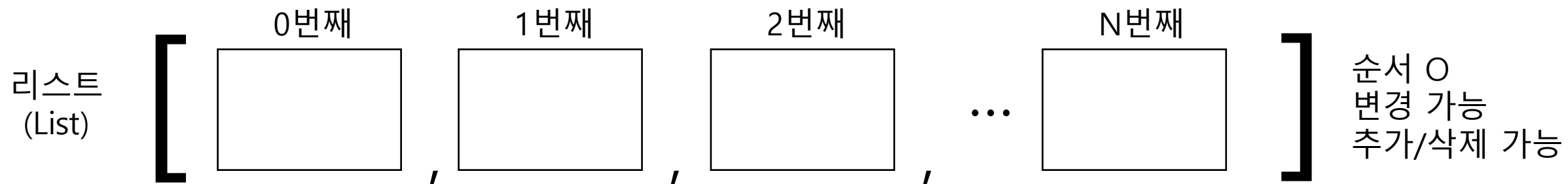
```
bool(d)
```

True

## 03 자료형(2) – 컨테이너 타입

- 컨테이너 타입: 여러 개의 값을 다루는 자료형
  - 리스트(list) : 데이터를 연속적으로 관리, 값을 바꿀 수 있음
  - 튜플(tuple) : 데이터를 연속적으로 관리, 값을 바꿀 수 없음
  - 세트(set) : 집합과 같은 속성
  - 딕셔너리 : key-value 쌍

리스트	['가', '나', '다', '라']
튜플	('가', '나', '다', '라')
세트	{ '가', '나', '다', '라' }
딕셔너리	{ '가':123, '나':456, '다':789 }

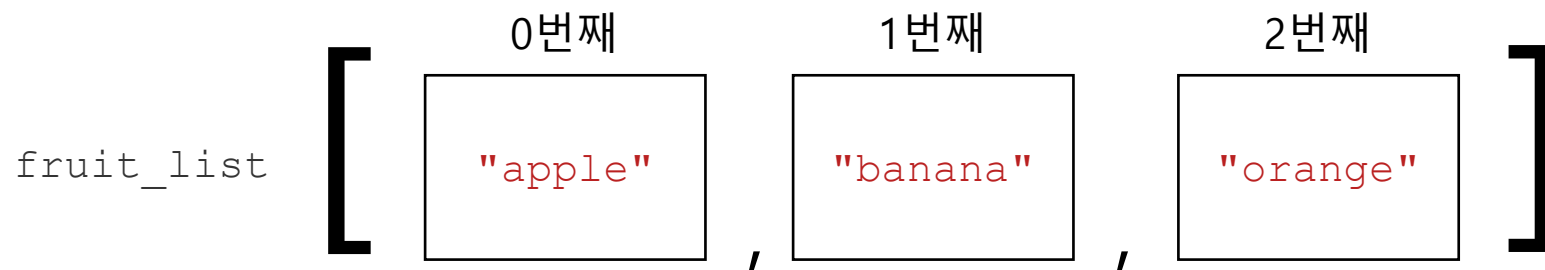


```
fruit_list = ["apple", "banana", "orange"]
```

```
# 중복허용
```

```
fruit_list = ["apple", "banana", "orange", "apple", "banana"]  
fruit_list
```

```
['apple', 'banana', 'orange', 'apple', 'banana']
```



```
: fruit_list = ["apple", "banana", "orange"]  
fruit_list[0]
```

```
: 'apple'
```

```
: fruit_list[1]
```

```
: 'banana'
```

```
: fruit_list[2]
```

```
: 'orange'
```

```
: fruit_list[3]
```

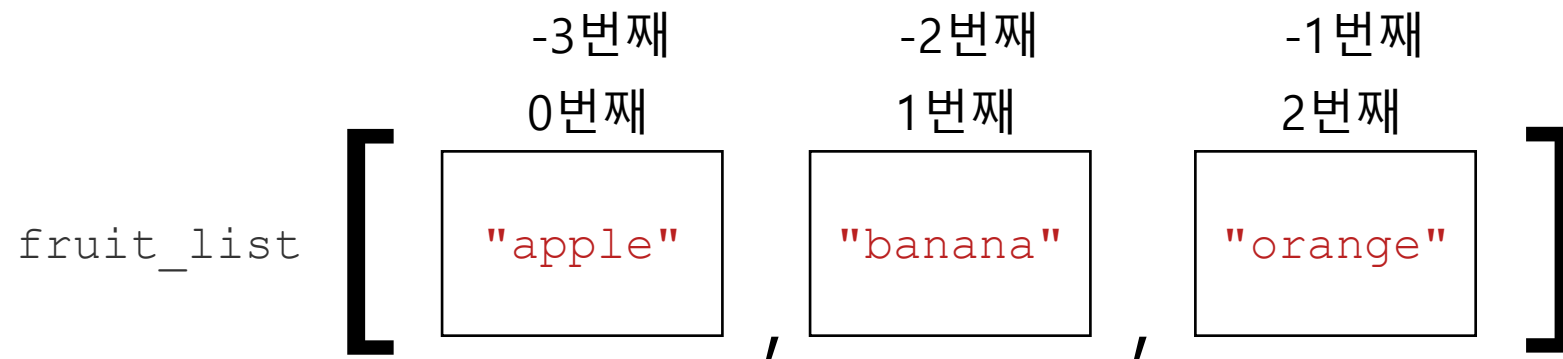
```
IndexError                                Traceback (most recent call last)  
<ipython-input-8-6f9a1405e6b2> in <module>  
----> 1 fruit_list[3]
```

**IndexError**: list index out of range

[Tip] 리스트 길이 확인하는 방법

```
fruit_list = ["apple", "banana", "orange"]  
len(fruit_list)
```

3



```
fruit_list[-1]
```

```
'orange'
```

```
fruit_list[-2]
```

```
'banana'
```

포함                  미포함  
[ 1 : 3 ] → 1번째, 2번째 인덱스의 아이템 반환

```
fruit_list = ["apple", "banana", "orange"]
```

```
fruit_list[1:3]
```

```
['banana', 'orange']
```

```
fruit_list[:3]    # 시작 위치를 지정하지 않을
```

```
['apple', 'banana', 'orange']
```

```
fruit_list[1:]    # 종료 위치를 지정하지 않을
```

```
['banana', 'orange']
```

```
fruit_list[:]    # 시작, 종료 위치를 지정하지 않을 = 전체 선택
```

```
['apple', 'banana', 'orange']
```

```
fruit_list[1] = "kiwi" # 1번 위치의 아이템에 새로운 값 할당  
fruit_list
```

```
['apple', 'kiwi', 'orange']
```

```
fruit_list[1:3] = ["strawberry", "blueberry"]  
fruit_list
```

```
['apple', 'strawberry', 'blueberry']
```



## ■ 아이템 추가

---

<b>insert()</b>	지정한 위치에 아이템을 추가
-----------------	-----------------

<b>append()</b>	가장 마지막 위치에 아이템을 추가
-----------------	--------------------

<b>extend()</b>	다른 리스트에 있는 아이템을 합침
-----------------	--------------------

<b>+</b>	여러 리스트에 있는 아이템을 합친 새로운 리스트 생성
----------	-------------------------------

## ■ 아이템 추가

### [문제]

현재 fruit\_list에는 ['apple', 'strawberry', 'blueberry'] 세 개의 값이 들어 있습니다.  
2번 인덱스(즉, 세 번째) 위치에 “mango” 아이템을 추가하세요.

### [답]

```
fruit_list.insert(2, "mango")  
fruit_list
```

```
['apple', 'strawberry', 'mango', 'blueberry']
```

## ■ 아이템 추가

[문제]

현재 fruit\_list에는 ['apple', 'strawberry', 'mango', 'blueberry']  
네 개의 값이 들어 있습니다. 마지막 위치에 “watermelon” 아이템을 추가하세요.

[답]

```
fruit_list.append("watermelon")  
fruit_list
```

```
['apple', 'strawberry', 'mango', 'blueberry', 'watermelon']
```

## ■ 아이템 추가

### [문제]

현재 fruit\_list에는 ['apple', 'strawberry', 'mango', 'blueberry', 'watermelon'] 다섯 개의 아이템이 들어 있습니다.  
그리고 vegetable\_list에는 ["carrot", "tomato", "onion"] 세 개의 아이템이 들어 있습니다.  
fruit\_list에 vegetable\_list에 있는 아이템을 추가하세요.

### [답]

```
vegetable_list = ["carrot", "tomato", "onion"]  
fruit_list.extend(vegetable_list)  
fruit_list
```

```
['apple',  
'strawberry',  
'mango',  
'blueberry',  
'watermelon',  
'carrot',  
'tomato',  
'onion']
```

## ■ 아이템 삭제

---

<code>remove()</code>	지정한 값을 가지는 아이템을 삭제
-----------------------	--------------------

<code>del</code>	지정한 아이템(특정 위치에 있는 아이템, 리스트 자체)을 삭제
------------------	------------------------------------

<code>clear()</code>	리스트에 있는 모든 아이템을 삭제
----------------------	--------------------

---

## ■ 아이템 삭제

[문제]

현재 fruit\_list에는 ["apple", "tomato", "banana", "orange"] 네 개의 값이 들어 있습니다. “tomato” 값을 삭제하세요.

[답]

```
fruit_list = ["apple", "tomato", "banana", "orange"]  
fruit_list.remove("tomato")  
fruit_list
```

```
['apple', 'banana', 'orange']
```

## ■ 아이템 삭제

[문제]

현재 fruit\_list에는 ['apple', 'banana', 'orange'] 세 개의 값이 들어 있습니다. 리스트 가장 마지막에 위치한 아이템을 삭제하세요.

[답]

```
del fruit_list[-1]  
fruit_list
```

```
['apple', 'banana']
```

## ■ 아이템 삭제

[문제]

현재 fruit\_list에는 ['apple', 'banana'] 두 개의 값이 들어 있습니다. fruit\_list리스트를 삭제하세요.

[답]

```
del fruit_list  
fruit_list
```

```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-24-959c847a65c7> in <module>  
      1 del fruit_list  
----> 2 fruit_list
```

```
NameError: name 'fruit_list' is not defined
```



## ■ 아이템 삭제

[문제]

현재 fruit\_list에는 ["apple", "banana", "orange"] 세 개의 값이 들어 있습니다. fruit\_list리스트 안에 있는 아이템을 모두 삭제하세요.

[답]

```
fruit_list = ["apple", "banana", "orange"]  
fruit_list.clear()  
fruit_list
```

```
[]
```

## ■ 아이템 정렬

---

sort()	리스트 내의 아이템을 정렬(알파벳순 또는 크기순)
--------	-----------------------------

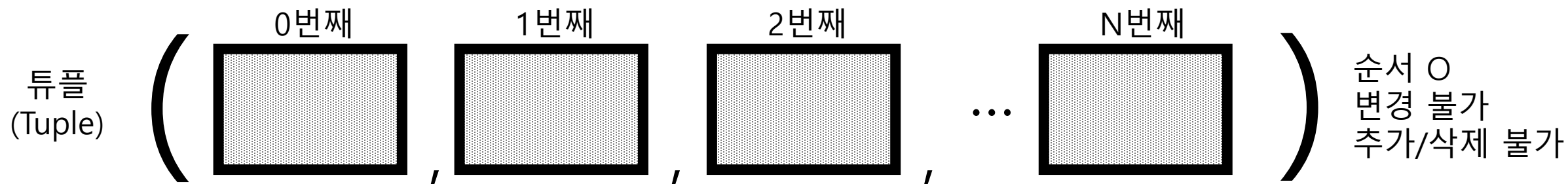
---

```
fruit_list = ['strawberry', 'mango', 'blueberry', 'watermelon', "apple", "banana", "orange"]
fruit_list.sort()
fruit_list
```

```
['apple', 'banana', 'blueberry', 'mango', 'orange', 'strawberry', 'watermelon']
```

```
fruit_list.sort(reverse = True)
fruit_list
```

```
['watermelon', 'strawberry', 'orange', 'mango', 'blueberry', 'banana', 'apple']
```



```
fruit_tuple = ("apple", "banana", "orange")
```

```
fruit_tuple = ("apple", "banana", "orange", "apple", "banana")
```

```
fruit_tuple
```

```
('apple', 'banana', 'orange', 'apple', 'banana')
```

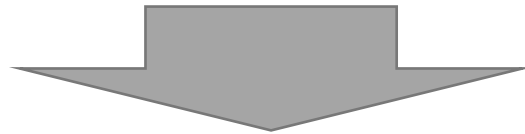
```
fruit_tuple = ("apple", "banana", "orange")
```

```
fruit_tuple[1]
```

```
'banana'
```

## ■ 튜플값의 변경

fruit\_tuple ( 0번째  
"apple" , 1번째  
"banana" , 2번째  
"orange" ) 편집 불가



```
fruite_list = list(fruit_tuple)
```

fruit\_list [ "apple" , "banana" , "orange" ] 편집 가능

## ■ 튜플값의 변경

```
fruit_tuple = ("apple", "banana", "orange")  
fruit_tuple[1]
```

```
'banana'
```

```
# 아이템 변경 안됨  
fruit_tuple[1] = "kiwi"
```

```
TypeError                                Traceback (most recent call last)  
<ipython-input-31-a09fd5403afe> in <module>  
    1 # 아이템 변경 안됨  
----> 2 fruit_tuple[1] = "kiwi"
```

```
TypeError: 'tuple' object does not support item assignment
```

```
fruit_tuple.append("watermelon")
```

```
AttributeError                            Traceback (most recent call last)  
<ipython-input-32-40d379ebc83b> in <module>  
----> 1 fruit_tuple.append("watermelon")
```

```
AttributeError: 'tuple' object has no attribute 'append'
```

```
fruit_tuple.remove("apple")
```

```
AttributeError                            Traceback (most recent call last)  
<ipython-input-33-4689ee50c2cb> in <module>  
----> 1 fruit_tuple.remove("apple")
```

```
AttributeError: 'tuple' object has no attribute 'remove'
```

## ■ 튜플값의 변경

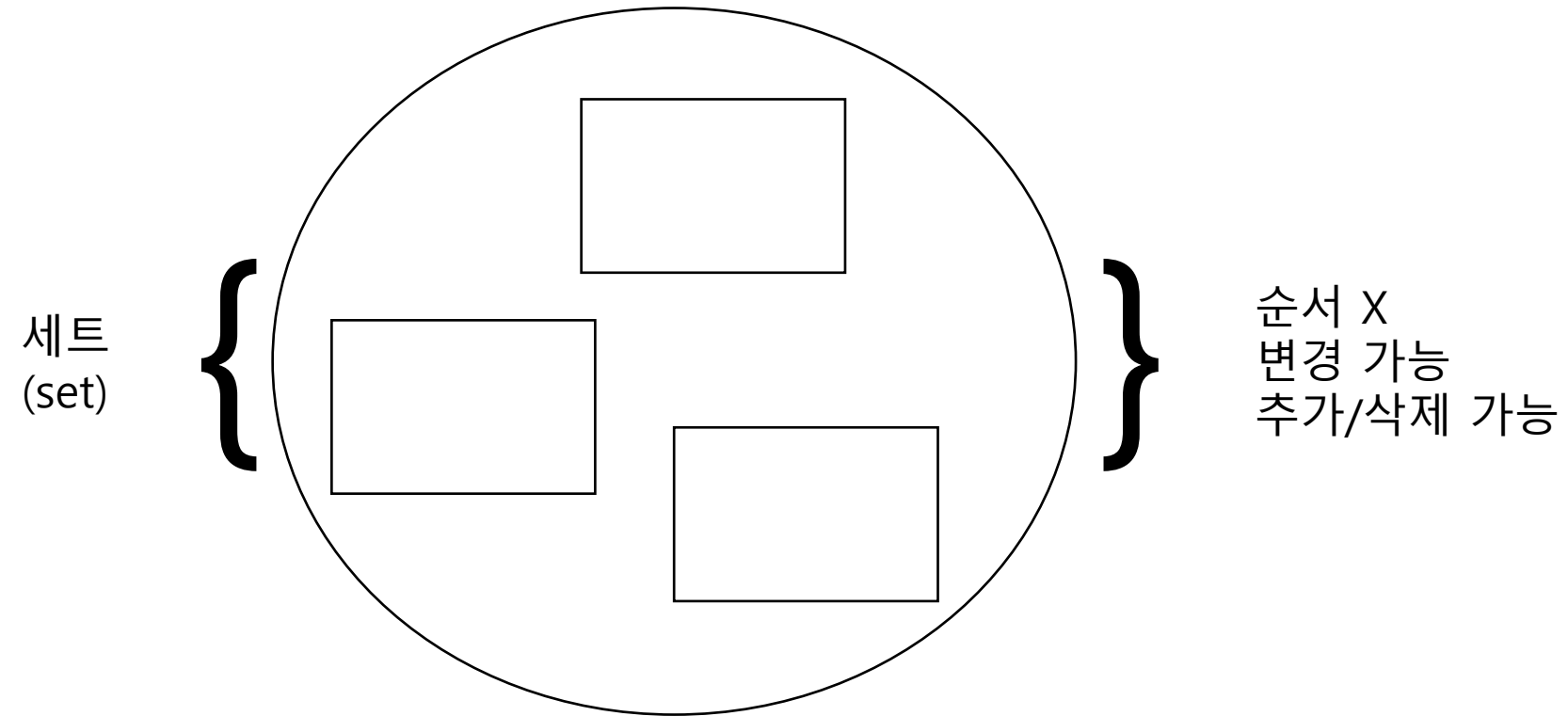
```
# 튜플을 리스트로 형 변환  
fruite_list = list(fruit_tuple)
```

```
# 리스트의 값 편집, 추가 및 삭제  
fruite_list.append("watermelon")  
fruite_list.remove("apple")
```

```
# 리스트를 다시 튜플로 형 변환  
fruit_tuple = tuple(fruite_list)  
fruit_tuple
```

```
('banana', 'orange', 'watermelon')
```

```
# del로 튜플 삭제  
del fruit_tuple
```



## ■ 세트 생성

```
fruit_set = {"apple", "banana", "orange"}
```

```
fruit_set = {"apple", "banana", "orange", "apple", "banana"}  
fruit_set # 중복 허용하지 않음  
{'apple', 'banana', 'orange'}
```

## ■ 세트는 인덱스가 없음

```
# Set는 순서가 없기 때문에, 특정 아이템을 선택할 수 없음  
fruit_set[1]
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-38-14fd45b2ed30> in <module>  
    1 # Set는 순서가 없기 때문에, 특정 아이템을 선택할 수 없음  
----> 2 fruit_set[1]
```

```
TypeError: 'set' object is not subscriptable
```



## ■ 아이템 추가

---

<code>add()</code>	신규 아이템을 추가
--------------------	------------

<code>update()</code>	다른 세트에 있는 아이템을 추가
-----------------------	-------------------

## ■ 아이템 추가

[문제]

현재 fruit\_set에는 {'apple', 'banana', 'orange'} 세 개의 값이 들어 있습니다. “kiwi” 아이템을 추가하세요.

[답]

```
fruit_set.add("kiwi")  
fruit_set
```

```
{'apple', 'banana', 'kiwi', 'orange'}
```

## ■ 아이템 추가

[문제]

현재 fruit\_set에는 {'apple', 'banana', 'kiwi', 'orange'} 네 개의 값이 들어 있습니다.  
vegetable\_set에는 ("carrot", "tomato", "onion") 세 개의 아이템이 들어 있습니다.  
fruit\_set에 vegetable\_set에 담긴 아이템을 넣어보세요.

[답]

```
vegetable_set = ("carrot", "tomato", "onion")  
fruit_set.update(vegetable_set)  
fruit_set
```

```
{'apple', 'banana', 'carrot', 'kiwi', 'onion', 'orange', 'tomato'}
```

## ■ 아이템 삭제

---

**remove()**      지정한 값을 가지는 아이템을 삭제

**del**      지정한 아이템(특정 위치에 있는 아이템, 세트 자체)을 삭제

**clear()**      세트에 있는 모든 아이템을 삭제

---

```
fruit_set.remove("onion")  
fruit_set
```

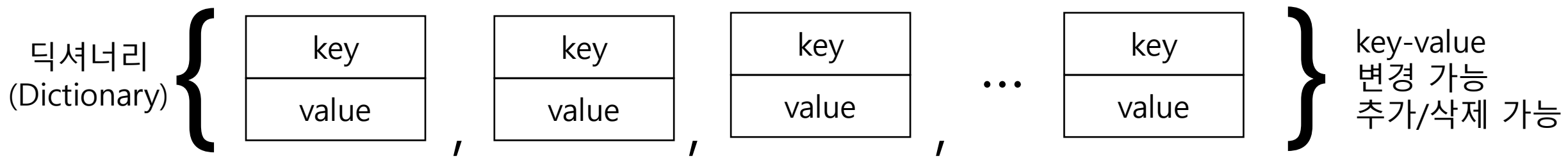
```
{'apple', 'banana', 'carrot', 'kiwi', 'orange', 'tomato'}
```

```
fruit_set.clear()  
fruit_set
```

```
set()
```

## ■ (참고)기타 세트 연산

difference()	차집합(두 세트 간의 차이를 반환)
intersection()	교집합(두 세트 모두 포함한 아이템을 반환)
isdisjoint()	두 세트에 교집합이 있는지 여부 반환
issubset()	이 세트가 다른 세트에 포함되는지 여부 반환
issuperset()	이 세트가 다른 세트를 포함하는지 여부 반환
symmetric_difference()	두 세트의 합집합에서 교집합 부분을 제외한 부분을 반환 (두 세트의 대칭 차이가 있는 세트를 반환)
union()	합집합



```
my_dict = {  
    "name": "Harry",  
    "age": 27,  
    "height" : 190,  
    "weight" : 99.9  
}  
my_dict
```

```
{'name': 'Harry', 'age': 27, 'height': 190, 'weight': 99.9}
```

## ■ 아이템 선택 및 추가

- 각 아이템의 키(key)값을 사용하여 선택

```
my_dict = {  
    "name": "Harry",  
    "age": 27,  
    "height": 190,  
    "weight": 99.9  
}  
my_dict
```

```
{'name': 'Harry', 'age': 27, 'height': 190, 'weight': 99.9}
```

```
my_dict.keys()
```

```
dict_keys(['name', 'age', 'height', 'weight'])
```

```
my_dict["age"]
```

```
27
```

```
my_dict["age"] = 28  
my_dict
```

```
{'name': 'Harry', 'age': 28, 'height': 190, 'weight': 99.9}
```

```
my_dict.update({"weight": 100})  
my_dict
```

```
{'name': 'Harry', 'age': 28, 'height': 190, 'weight': 100}
```

```
my_dict.update({"address": "Busan"})  
my_dict
```

```
{'name': 'Harry', 'age': 28, 'height': 190, 'weight': 100, 'address': 'Busan'}
```

## ■ 아이템 삭제

**popitem()**      마지막 아이템 삭제

**pop(키값)**      인자로 넘긴 키값과 값의 쌍을 삭제

**clear()**      딕셔너리에 있는 모든 아이템을 삭제

```
my_dict.popitem()  
( 'address', 'Busan' )
```

```
my_dict  
{ 'name': 'Harry', 'age': 28, 'height': 190, 'weight': 100 }
```

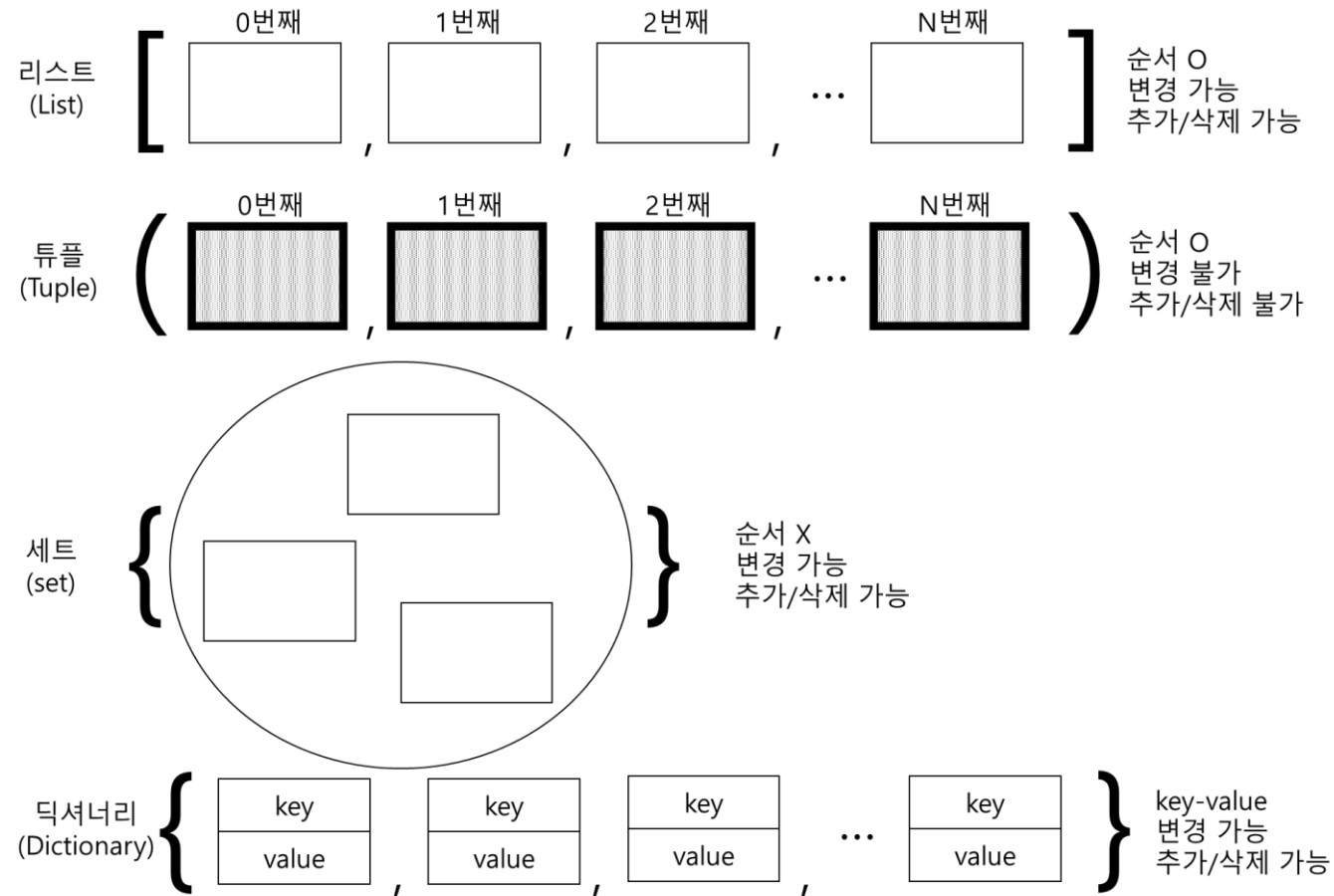
```
my_dict.pop("age")  
28
```

```
my_dict  
{ 'name': 'Harry', 'height': 190, 'weight': 100 }
```

```
my_dict.clear()  
my_dict  
{ }
```

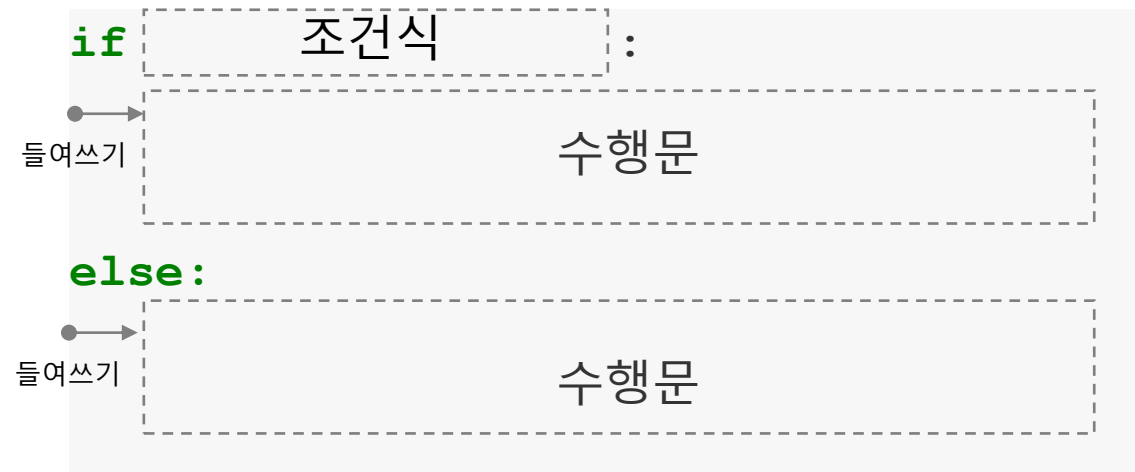
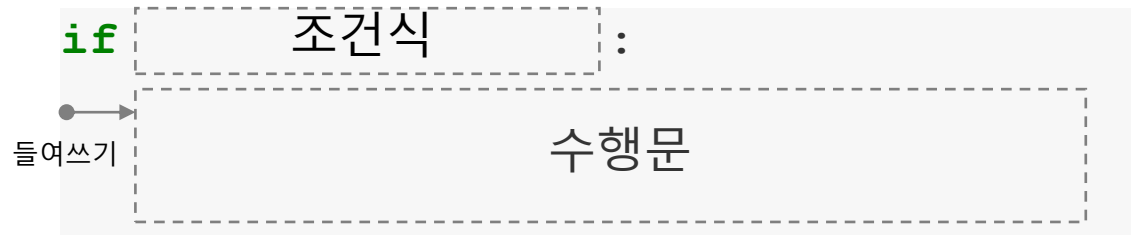


## ■ 컨테이너 타입 정리



## 04 조건문

- if
  - 조건 지정(만약 ~ 라면)
- else
  - if 조건에 해당하지 않는 나머지 조건을 처리(그것이 아니면)
- elif
  - if 조건에 해당하지 않는 나머지 경우 중에서 새로운 조건을 지정하여 처리(그것이 아니고 ~라면)

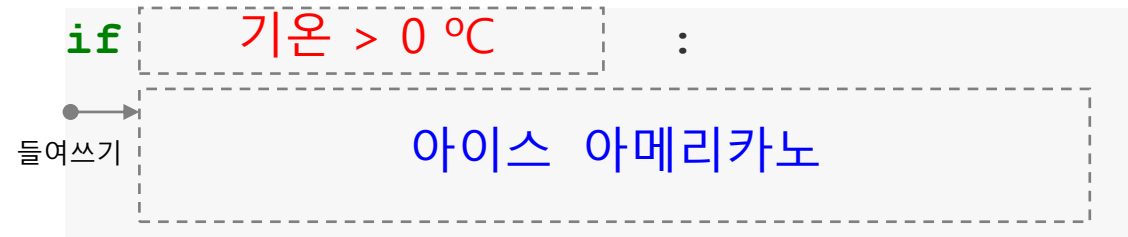


## ■ 다양한 조건식

$a == b$	a와 b가 같은지?
$a != b$	a와 b가 다른지?
$a < b$	a가 b보다 작은지?
$a <= b$	a가 b보다 작거나 같은지
$a > b$	a가 b보다 큰지?
$a >= b$	a가 b보다 크거나 같은지?
$a \text{ and } b$	a 조건을 만족하면서 b 조건을 만족하는 경우 참을 반환
$a \text{ or } b$	a 조건을 만족하거나 b 조건을 만족하는 경우 참을 반환

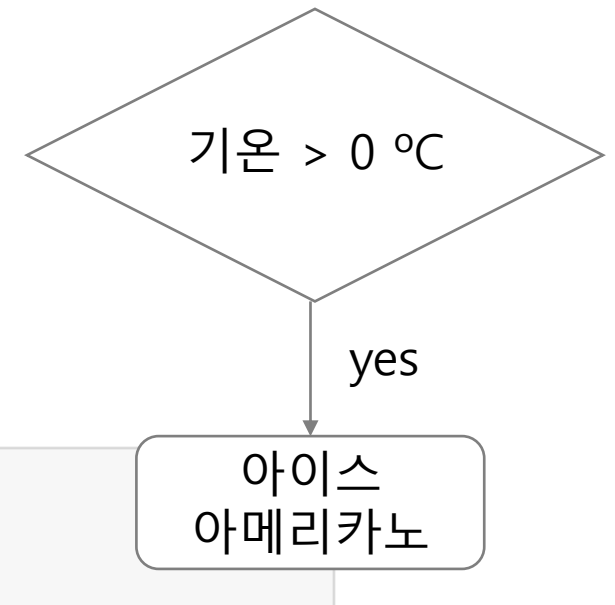
**[문제]**

기온이 0℃보다 높으면 '아이스 아메리카노'를 출력하는 프로그램을 만들어 보세요.

**[답]**

```
today_temp = 30  
  
if today_temp > 0:  
    print("아이스 아메리카노")
```

아이스 아메리카노



## ■ 주의

### • 정상 동작코드

```
today_temp = 30  
  
if today_temp > 0:  
    print("아이스 아메리카노")
```

아이스 아메리카노

### • 에러 발생 코드

```
today_temp = 30  
  
if today_temp > 0:  
print("아이스 아메리카노")
```

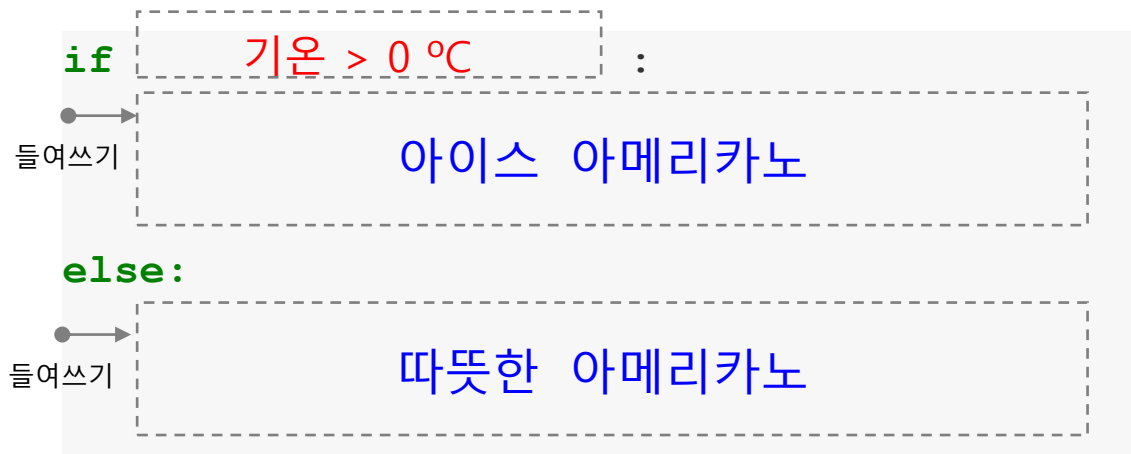
```
File "<ipython-input-2-c8a3b76ff21a>", line 4  
    print("아이스 아메리카노")  
    ^
```

**IndentationError:** expected an indented block

## [문제]

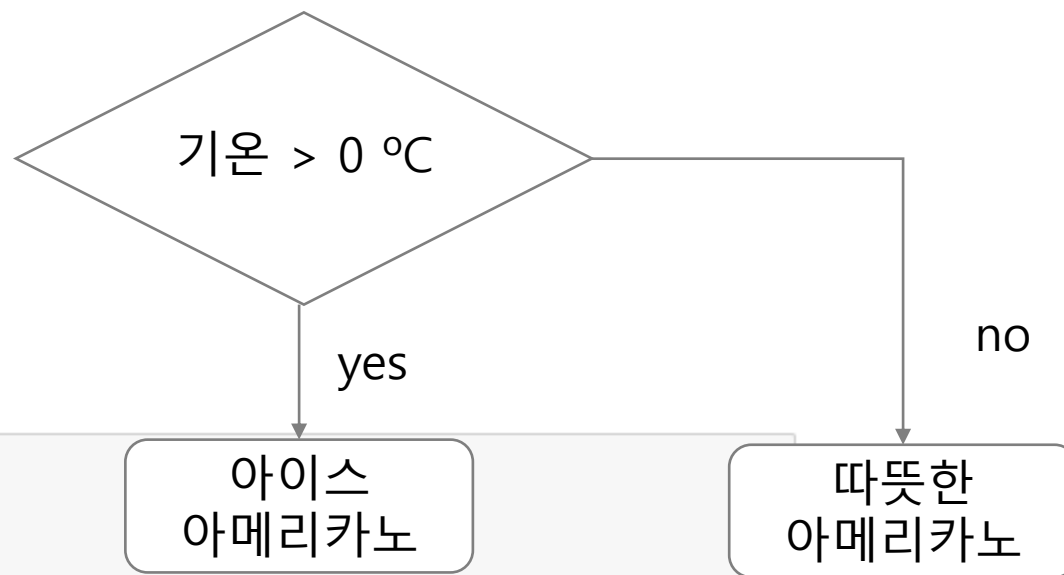
기온이 0℃보다 높으면 '아이스 아메리카노'를 출력하고, 그렇지 않은 경우에는 '따뜻한 아메리카노'를 출력하는 프로그램을 만들어 보세요.

## [답]



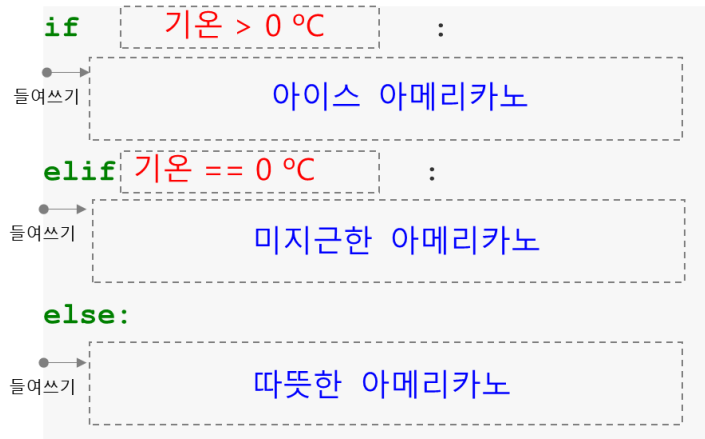
```
today_temp = 30  
  
if today_temp > 0:  
    print("아이스 아메리카노")  
else:  
    print("따뜻한 아메리카노")
```

아이스 아메리카노



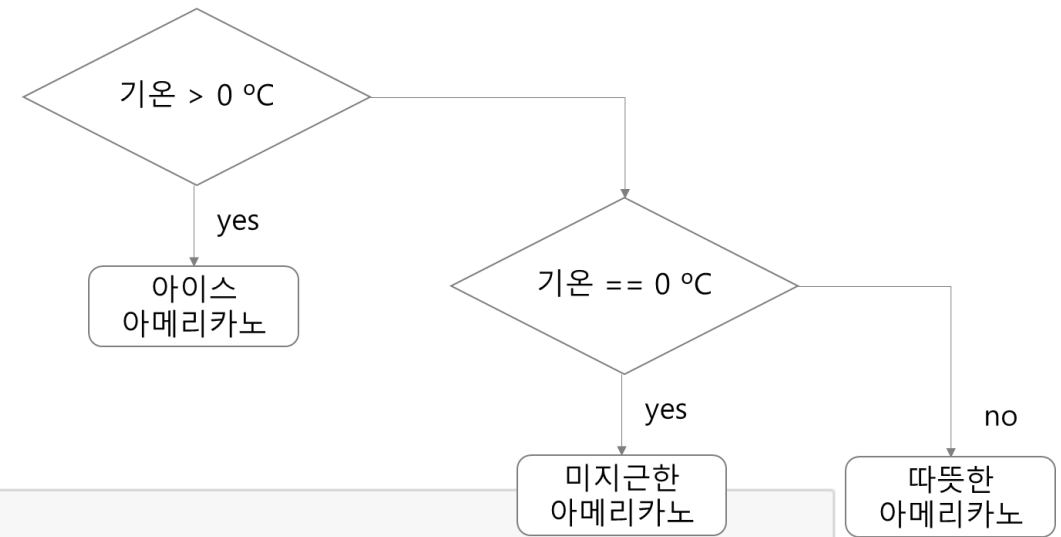
**[문제]**

기온이 0℃보다 높으면 ‘아이스 아메리카노’, 기온이 0℃이면 ‘미지근한 아메리카노’, 나머지 경우에는 ‘따뜻한 아메리카노’를 출력하는 프로그램을 만들어 보세요.

**[답]**

```
today_temp = 30  
  
if 0 < today_temp < 10:  
    print("아이스 아메리카노")  
elif today_temp == 10:  
    print("미지근한 아메리카노")  
else:  
    print("따뜻한 아메리카노")
```

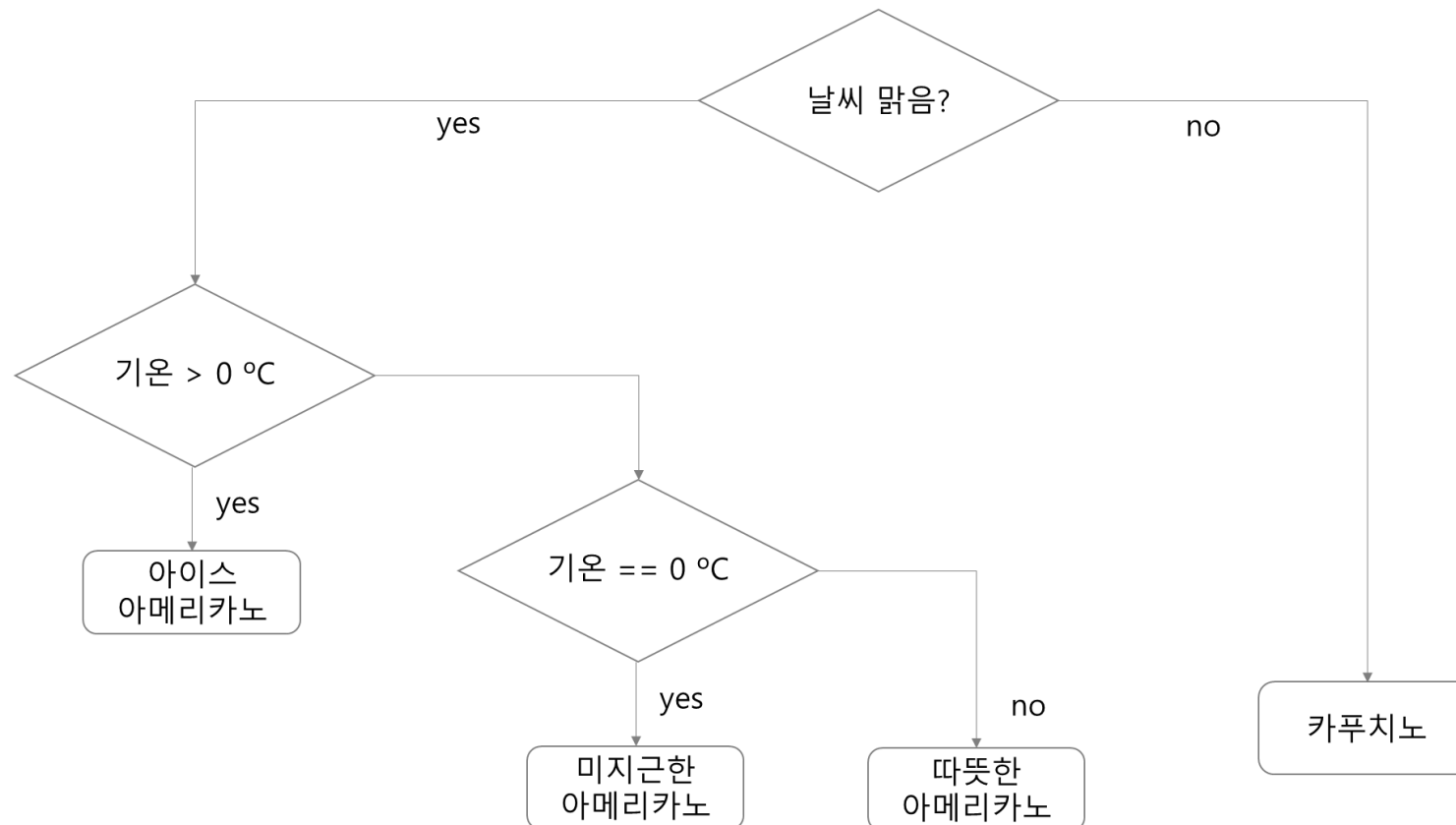
아이스 아메리카노





## [문제]

날씨가 맑은 날인 경우, 기온이 0℃보다 높으면 '아이스 아메리카노', 기온이 0℃이면 '미지근한 아메리카노', 나머지 경우에는 '따뜻한 아메리카노'를 출력하고  
날씨가 맑지 않은 경우, '카푸치노'를 출력하는 프로그램을 만들어 보세요.



## [문제]

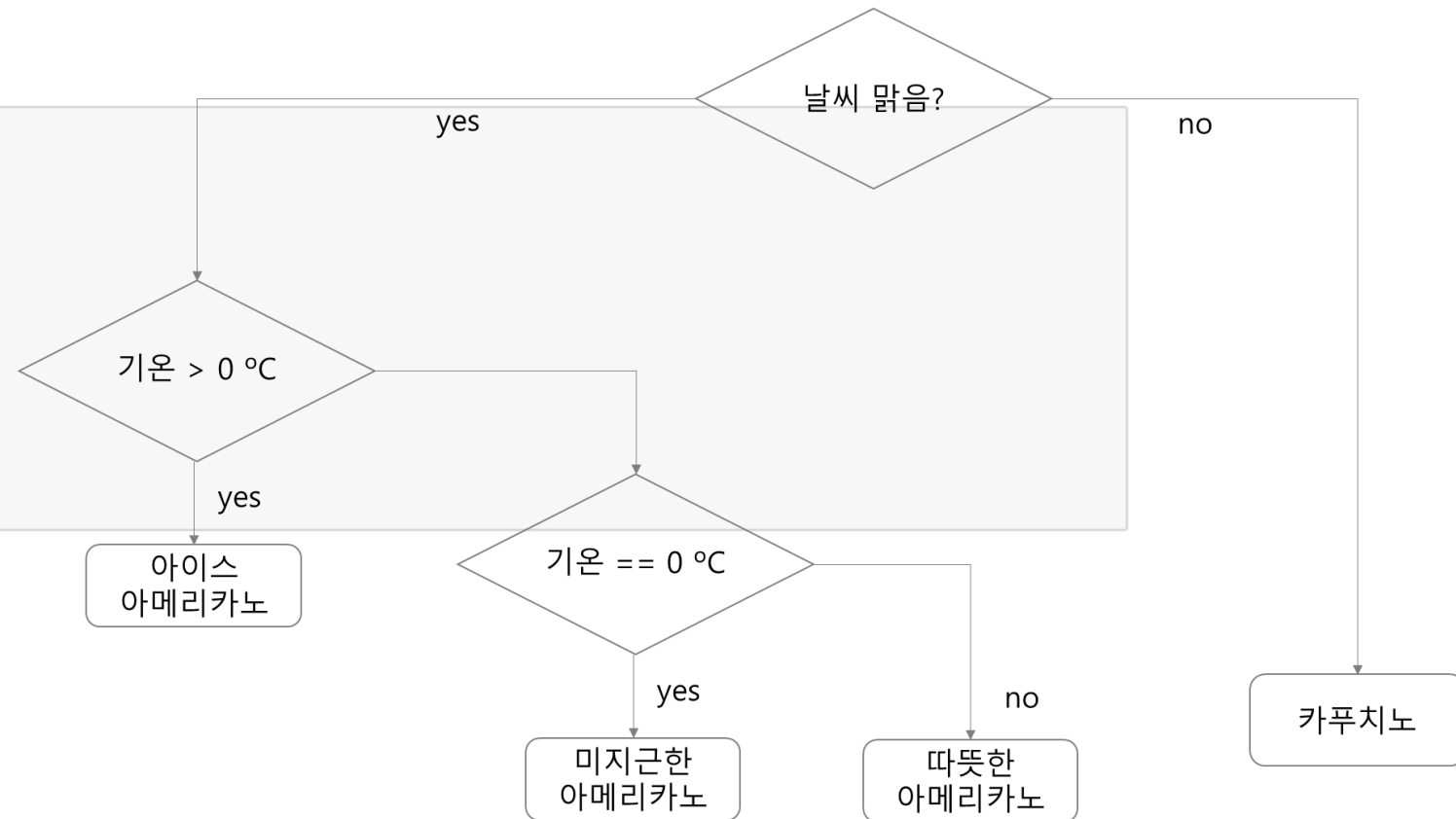
날씨가 맑은 날인 경우, 기온이 0℃보다 높으면 '아이스 아메리카노', 기온이 0℃이면 '미지근한 아메리카노', 나머지 경우에는 '따뜻한 아메리카노'를 출력하고  
날씨가 맑지 않은 경우, '카푸치노'를 출력하는 프로그램을 만들어 보세요.

## [답]

```
weather = "비"
today_temp = 30

if weather == "맑음":
    if 0 < today_temp:
        print("아이스 아메리카노")
    elif 0 == today_temp:
        print("미지근한 아메리카노")
    else:
        print("따뜻한 아메리카노")
else:
    print("카푸치노")
```

카푸치노

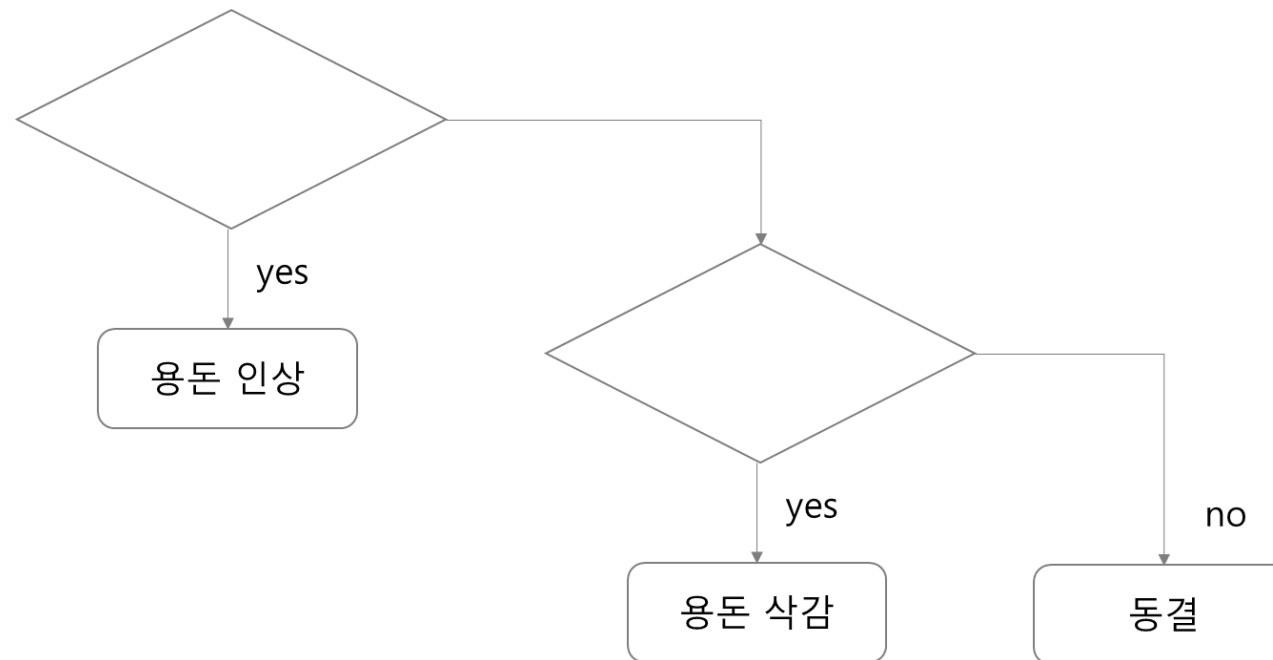


**[문제]**

이번 기말고사 시험성적에 따라 용돈의 운명이 바뀝니다. 엄마가 제시한 조건은 다음과 같습니다.

- 영어 90점 이상, 수학 90점 이상(두 조건 모두 만족) : 용돈 인상
- 영어 80점 이하, 수학 80점 이하(두 조건 모두 만족) : 용돈 삭감
- 기타 : 동결

수학성적과 영어성적을 담는 변수를 만들고, 점수에 따라 용돈의 운명을 출력하는 프로그램을 만들어 보세요.



**[문제]**

이번 기말고사 시험성적에 따라 용돈의 운명이 바뀝니다. 엄마가 제시한 조건은 다음과 같습니다.

- 영어 90점 이상, 수학 90점 이상(두 조건 모두 만족) : 용돈 인상
- 영어 80점 이하, 수학 80점 이하(두 조건 모두 만족) : 용돈 삭감
- 기타 : 동결

수학성적과 영어성적을 담은 변수를 만들고, 점수에 따라 용돈의 운명을 출력하는 프로그램을 만들어 보세요.

**[답]**

```
# 영어 90점 이상, 수학 90점 이상: 용돈 인상
# 영어 80점 이하, 수학 80점 이하: 용돈 삭감

math_score = 80
eng_score = 100

if eng_score >= 90 and math_score >= 90:
    print("YAY! 용돈 인상")
elif eng_score <= 80 and math_score <= 80:
    print("용돈 삭감 ㅠㅠ")
else:
    print("동결")
```

동결

**[문제]**

엄마가 제시한 조건은 달성하기 너무 힘들 것 같습니다. 아래와 같이 타협하려고 합니다.

- 영어 90점 이상 또는 수학 90점 이상 : 용돈 인상
- 영어 80점 이하 또는 수학 80점 이하 : 용돈 삭감
- 기타 : 동결

수학성적과 영어성적을 담은 변수를 만들고, 점수에 따라 용돈의 운명을 출력하는 프로그램을 만들어 보세요.

**[답]**

```
# 영어 90점 이상 또는 수학 90점 이상 : 용돈 인상
# 영어 80점 이하 또는 수학 80점 이하 : 용돈 삭감

math_score = 80
eng_score = 100

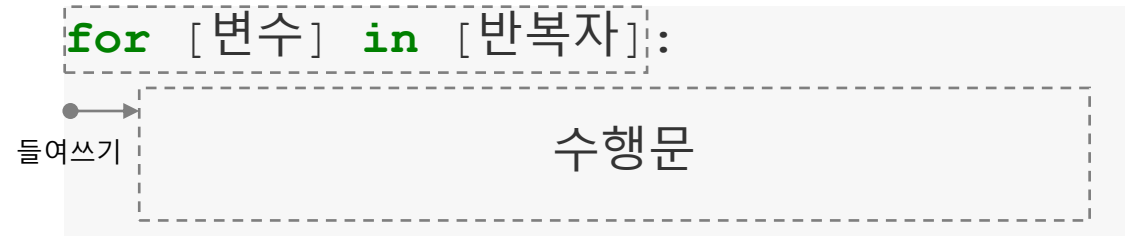
if eng_score >= 90 or math_score >= 90:
    print("YAY! 용돈 인상")
elif eng_score <= 80 or math_score <= 80:
    print("용돈 삭감 ππ")
else:
    print("동결")
```

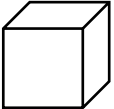
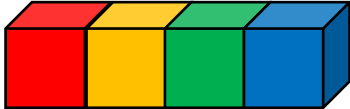
YAY! 용돈 인상

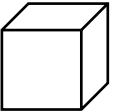
## 05 반복문

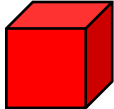
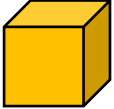
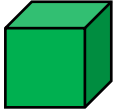
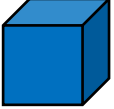
## ■ for 문

- 반복자(iterator): 보유한 아이템을 순회할 수 있는 특징



for  in  :

print(  )

첫번째 시행	 출력
두번째 시행	 출력
세번째 시행	 출력
네번째 시행	 출력

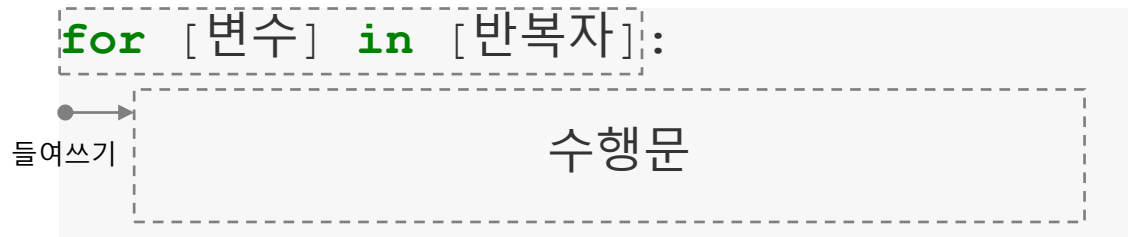
**[문제]**

1학년 2반의 시험 성적은 다음과 같습니다.

[80, 90, 70, 65, 85, 95, 90, 80, 75, 80]

시험 문제 중, 한 문제가 잘못 출제되어 모두 5점 씩 추가 점수를 받도록 조치를 취했습니다.

변경 후 점수를 출력하세요.



- 변수: s
- 반복자: scores
- 수행문: s+5

**[답]**

```
scores = [80, 90, 70, 65, 85, 95, 90, 80, 75, 80]
new_scores = []

for s in scores:
    new = s + 5
    new_scores.append(new)

print(new_scores)
```

[85, 95, 75, 70, 90, 100, 95, 85, 80, 85]



**[문제]**

1학년 3반의 시험 성적은 다음과 같습니다.

[80, 90, 70, 65, 95, 100, 90, 80, 75, 80]

시험 문제 중, 한 문제가 잘못 출제되어 모두 5점씩 추가 점수를 받도록 조치를 취했습니다. 변경 후 점수를 출력하세요.  
(현재 100점인 학생은 변경 후 점수도 100점)

**[답]**

```
scores = [80, 90, 70, 65, 95, 100, 90, 80, 75, 80]
new_score = []

for s in scores:
    if s < 100:
        new = s + 5
    else:
        new = s
    new_score.append(new)

print(new_score)
```

[85, 95, 75, 70, 100, 100, 95, 85, 80, 85]

## ■ 리스트 컴프리헨션

```
[수행문1 if 조건문 else 수행문2 for 변수 in 리스트]
```

↑  
if 조건문 만족할 경우의 수행문

↑  
else 조건을 만족할 경우의 수행문

## ■ 리스트 컴프리헨션

```
[수행문1 if 조건문 else 수행문2 for 변수 in 리스트]
```

```
new_score = []
```

```
for s in scores:
```

```
    if s < 100:
```

```
        new = s + 5
```

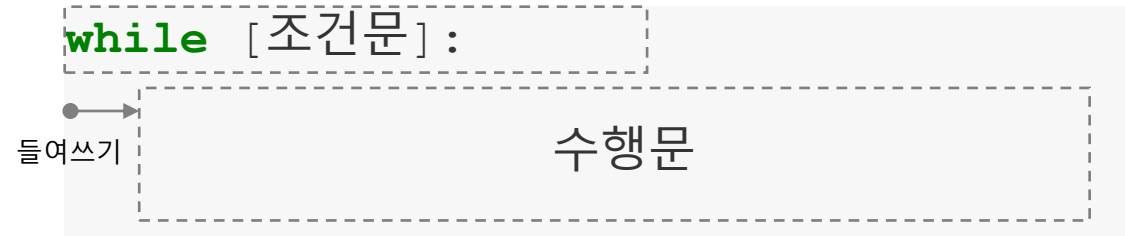
```
    else:
```

```
        new = s
```

```
    new_score.append(new)
```

```
new_score = [s + 5 if s < 100 else s for s in scores]
```

## ■ while문



**[문제]**

1학년 3반의 시험 성적은 다음과 같습니다.

[80, 90, 70, 65, 95, 100, 90, 80, 75, 80]

시험 문제 중, 한 문제가 잘못 출제되어 모두 5점씩 추가 점수를 받도록 조치를 취했습니다. 변경 후 점수를 출력하세요.  
(현재 100점인 학생은 변경 후 점수도 100점)

**[답] – for문**

```
scores = [80, 90, 70, 65, 95, 100, 90, 80, 75, 80]
new_score = []

for s in scores:
    if s < 100:
        new = s + 5
    else:
        new = s
    new_score.append(new)

print(new_score)
```

[85, 95, 75, 70, 100, 100, 95, 85, 80, 85]

**[답] – while문**

```
scores = [80, 90, 70, 65, 95, 100, 90, 80, 75, 80]
new_score = []
index = 0

while(index < len(scores)):
    if scores[index] < 100:
        new = scores[index] + 5
    else:
        new = scores[index]
    new_score.append(new)
    index = index + 1

print(new_score)
```

[85, 95, 75, 70, 100, 100, 95, 85, 80, 85]

## ■ 흐름제어

- break: 반복문 수행을 중단하고 그 다음 구문 수행
- pass: 반복문 수행은 유지한 상태에서 아무 동작도 수행하지 않음

**[문제]**

이번에 개발할 프로그램은 '스마트폰 사용량 감시 프로그램'입니다.  
사용자가 스마트폰 사용 허용시간을 300분으로 지정해 두었습니다. 이 사용자는 스마트폰을 한 번 사용할 때마다 50분씩 사용한  
다고 할 때, 지정시간에 도달하였을 경우 핸드폰 사용을 중단하도록 하는 프로그램을 만들어 보세요.

**[답]**

```
time = 0 # 누적 사용 시간
while(True):
    print('현재 사용량: {}'.format(time))
    if(time >= 300):
        print('[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.')
        break
    else:
        time = time + 50
```

현재 사용량: 0  
현재 사용량: 50  
현재 사용량: 100  
현재 사용량: 150  
현재 사용량: 200  
현재 사용량: 250  
현재 사용량: 300  
[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.

print('현재 사용량: {}'.format(time))



**[문제]**

‘스마트폰 사용량 감시 프로그램’의 신규 버전에서는 사용자에게 현재 사용량이 150분 미만일 경우에는 ‘안전’하다는 메시지를 보여주는 컨셉이 추가되었습니다. 기존 작성한 코드에서 메시지를 보여주는 컨셉만 추가해 보세요.

**[답]**

```
time = 0 # 누적 사용 시간
while(True):
    print('현재 사용량: {}'.format(time))
    if(time < 150):
        print('안전')
    if(time >= 300):
        print('[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.')
        break
    else:
        time = time + 50
```

현재 사용량: 0

안전

현재 사용량: 50

안전

현재 사용량: 100

안전

현재 사용량: 150

현재 사용량: 200

현재 사용량: 250

현재 사용량: 300

[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.



**[문제]**

‘스마트폰 사용량 감시 프로그램’의 신규 버전에서는 이전 버전에 추가되었던 ‘안전’ 메시지를 보여주는 컨셉을 삭제하기로 결정되었습니다. 기존 작성한 코드에서 메시지를 보여주는 부분이 동작하지 않도록 코드를 수정해보세요.

**[답]**

```
time = 0 # 누적 사용 시간
while(True):
    print('현재 사용량: {}'.format(time))
    if(time < 150):
        pass
    if(time >= 300):
        print('[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.')
        break
    else:
        time = time + 50
```

현재 사용량: 0

현재 사용량: 50

현재 사용량: 100

현재 사용량: 150

현재 사용량: 200

현재 사용량: 250

현재 사용량: 300

[사용 중단] 하루 사용 권장량에 도달 또는 초과하였습니다.

## 06 입출력

## ■ 시스템 입출력

- 화면에 데이터를 출력하거나 사용자로부터 직접 데이터를 입력 받을 수 있음
- 입력: `input()`
- 출력: `print()`
  - 포맷 스트링: 값을 출력하는 패턴을 지정 (예: 소수점 2자리까지만 출력)

## ■ 파일 입출력

- 파일에 쓰여있는 데이터를 읽어들이거나 파일에 데이터를 쓸 수 있음
- `open()`, `read()`, `write()`, `close()`
- `with()`문
  - 파일 읽기 쓰기 작업이 완료된 후 `close()`를 호출하지 않아도 자동으로 리소스 반환

## ■ 시스템 입력

```
In [*]: input()
```

```
In [*]: input()
```

```
In [1]: input()
```

홍길동

```
Out[1]: '홍길동'
```

## ■ 시스템 입력

[문제] 사용자로부터 이름을 입력 받고, 이름과 함께 인사말을 출력하세요.

예:

- 이름을 입력하세요: 홍길동
- 홍길동님, 안녕하세요

[답]

```
name = input("이름을 입력하세요:")  
print(name + "님, 안녕하세요")
```

이름을 입력하세요:아이리포  
아이리포님, 안녕하세요

## ■ 시스템 출력

```
print("apple", "peach", "mango")
```

apple peach mango

```
print("apple", "peach", "mango", sep=",")
```

apple,peach,mango

```
print("원숭이 엉덩이는 빨개 ")  
print("빨가면 사과")
```

원숭이 엉덩이는 빨개  
빨가면 사과

```
print("원숭이 엉덩이는 빨개 ", end="")  
print("빨가면 사과")
```

원숭이 엉덩이는 빨개 빨가면 사과

```
print("나는" + " 빵을 " + "먹고싶다")
```

나는 빵을 먹고싶다

```
print("산토끼 토끼야\n어디를 가느냐\n강총강총 뛰어서\n어디를 가느냐")
```

산토끼 토끼야  
어디를 가느냐  
강총강총 뛰어서  
어디를 가느냐

## ■ 시스템 출력

```
food = "치킨"  
text = "나는 {}을 먹고 싶다"  
print(text.format(food))
```

나는 치킨을 먹고 싶다

```
print("나는 {}을 먹고 싶다".format("치킨"))
```

나는 치킨을 먹고 싶다

```
food1 = "피자"  
food2 = "치킨"  
text = "나는 {}, {}을 먹고 싶다"  
print(text.format("피자", "치킨"))
```


나는 피자, 치킨을 먹고 싶다

```
print("나는 {0}, {1}을 먹고 싶다. 우리집엔 {1}이 배달되지 않아 슬프다.".format("피자", "치킨"))
```

나는 피자, 치킨을 먹고 싶다. 우리집엔 치킨이 배달되지 않아 슬프다.

```
text = "{name}님, 반갑습니다. 적립금은 {money}원 입니다..  
print(text.format(name = "홍길동", money = 500))
```

홍길동님, 반갑습니다. 적립금은 500원 입니다..



```
print('현재 사용량: {}'.format(time))
```

## ■ 시스템 출력

[Tip] 문자열에 %s 를 작성하여, 치환할 문자를 지정

```
food = "치킨"  
print("나는 %s을 먹고 싶다." %food)
```

나는 치킨을 먹고 싶다.

```
print("{:.2f}% 확신합니다.".format(95.1234567))
```

95.12% 확신합니다.

```
print("한 달 휴대폰 요금은 {:,}원 입니다.".format(100000))
```

한 달 휴대폰 요금은 100,000원 입니다.



## ■ 파일 입출력

```
f = open("abc.txt", "w") # 쓰기모드로 파일 열기
f.write("A B C D E F G ")
f.close()
```

```
f = open("abc.txt", "r")
print(f.read())
f.close()
```

A B C D E F G

```
f = open("abc.txt", "w")
f.write("a b c d e f g ")
f.close()
```

```
f = open("abc.txt", "r")
print(f.read())
f.close()
```

a b c d e f g

```
# 추가(append)모드로 파일 열기
f = open("abc.txt", "a")
f.write("H I J K L M N O P Q R S T U V W X Y Z")
f.close()
```

```
# 읽기모드로 파일 열기
f = open("abc.txt", 'r')
lines = f.readlines()
for line in lines:
    print(line)
f.close()
```

a b c d e f g H I J K L M N O P Q R S T U V W X Y Z

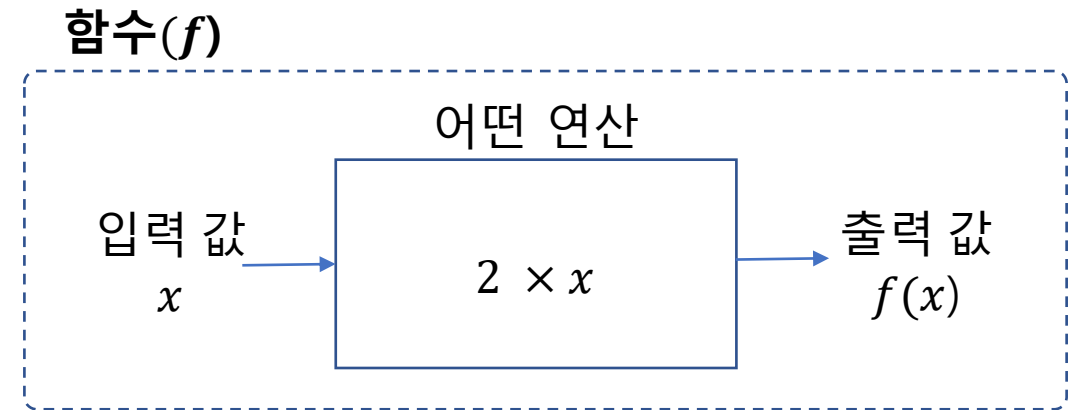
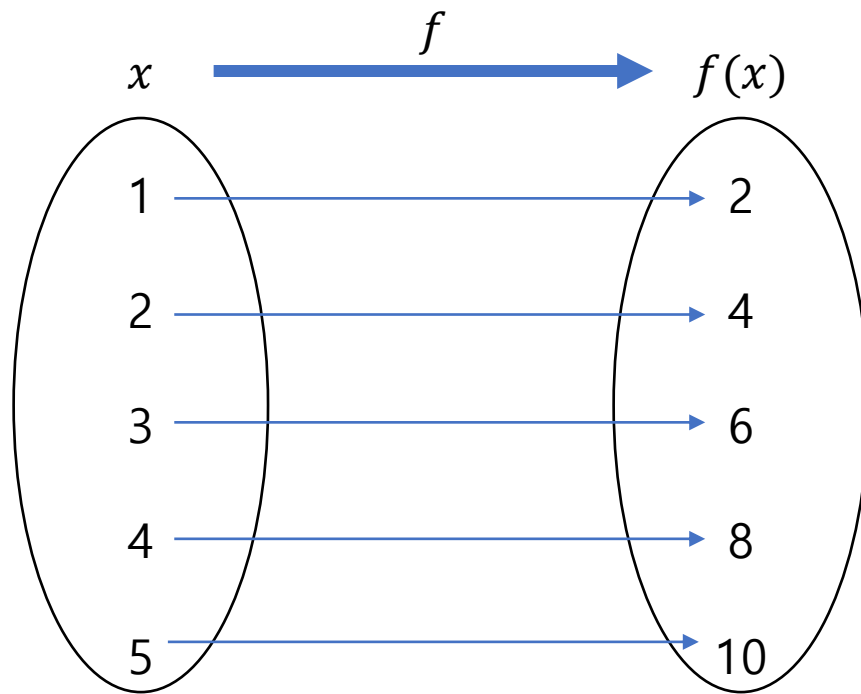
## ■ 파일 입출력

```
with open("일기.txt", "w") as f:  
    f.write("2020년 3월 12일 금요일\n")  
  
with open("일기.txt", "a") as f:  
    f.write("날씨 맑음")  
  
with open("일기.txt", "r") as f:  
    print(f.read())
```

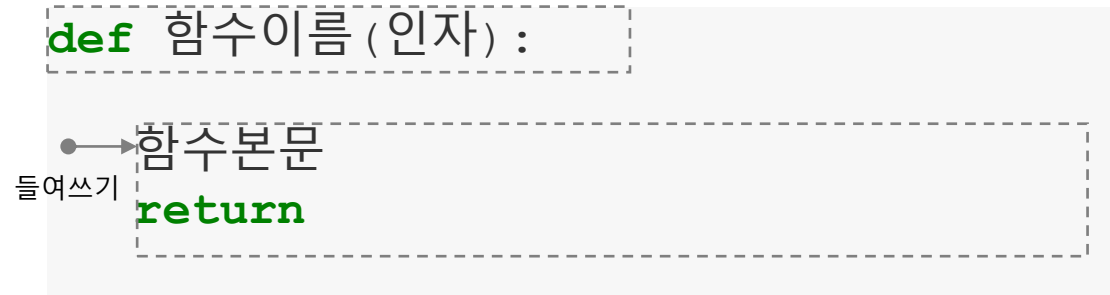
2020년 3월 12일 금요일  
날씨 맑음

## 07 함수, 람다

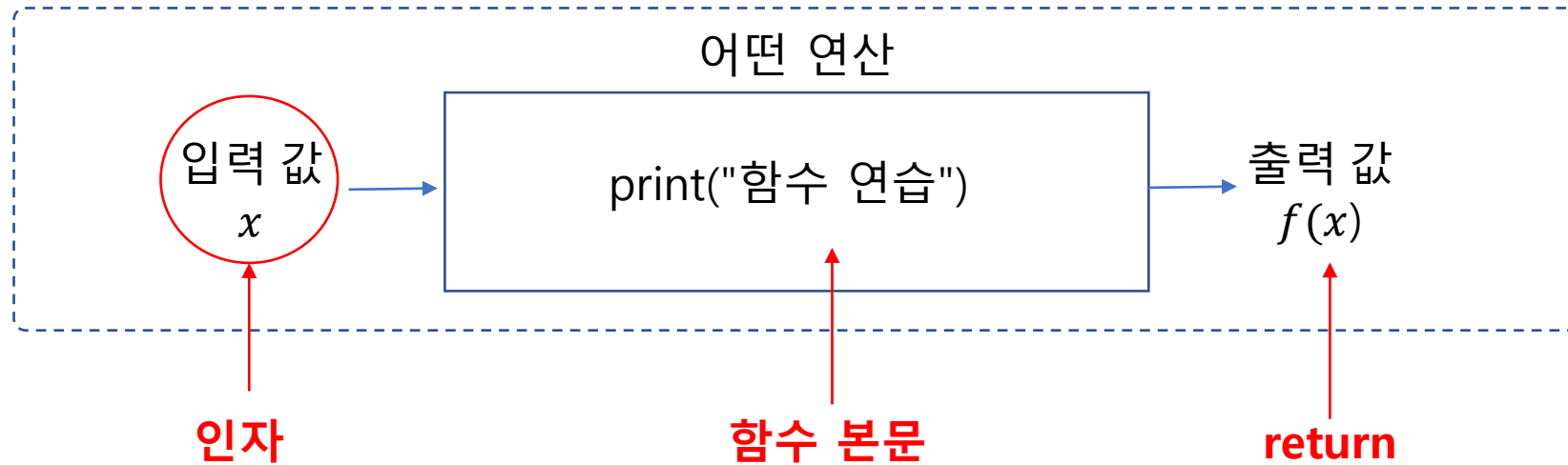
## ■ 함수란?



## ■ 함수



### 함수( $f$ ) 함수이름



**[문제]**

기온이 0℃보다 높으면 '아이스 아메리카노'를 출력하고, 그렇지 않은 경우에는 '따뜻한 아메리카노'를 출력하는 함수를 만들어 보세요.

**[답]**

```
def coffee(temp):  
    if temp > 0 :  
        print("아이스 아메리카노")  
    else:  
        print("따뜻한 아메리카노")
```

```
coffee(30)
```

아이스 아메리카노

```
coffee(-10)
```

따뜻한 아메리카노

**[문제]**

기온이 0°C보다 높으면 '아이스 아메리카노'를 출력하고, 그렇지 않은 경우에는 '따뜻한 아메리카노'를 출력하는 함수를 만들어 보세요.

**[답] – 화면에 출력하는 유형**

```
def coffee(temp):  
    if temp > 0 :  
        print("아이스 아메리카노")  
    else:  
        print("따뜻한 아메리카노")
```

```
coffee(30)
```

아이스 아메리카노

```
coffee(-10)
```

따뜻한 아메리카노

**[답] – 값을 반환하는 유형**

```
def coffee(temp):  
    result = ''  
    if temp > 0 :  
        result = '아이스 아메리카노'  
    else:  
        result = '따뜻한 아메리카노'  
    return result
```

```
c = coffee(30)  
print('추천 커피는 {}입니다'.format(c))
```

```
c = coffee(-10)  
print('추천 커피는 {}입니다'.format(c))
```

**[문제]**

사용자로부터 키와 성별을 입력 받아서 권장 체중을 화면에 출력하는 함수를 작성하세요.  
권장 체중은 다음과 같이 계산합니다.

- 남성 권장 체중 = (키 - 100)
- 여성 권장 체중 = (키 - 100) \* 0.9

**[답]**

```
def print_weight(height, man=True):  
    weight = 0  
    if(man):  
        weight = (height - 100) * 0.9  
    else:  
        weight = height - 100  
    print("권장 체중은 {}kg 입니다".format(weight))
```

```
print_weight(170)
```

```
print_weight(170, True)
```

권장 체중은 63.0kg 입니다

```
print_weight(170, False)
```

권장 체중은 70kg 입니다



**[문제]**

입력 받은 인자를 모두 더하는 함수를 작성하세요

**[답]**

```
def sum(*args):  
    result = 0  
    for num in args:  
        result = result + num  
    return result
```

```
sum(1,2,3)
```

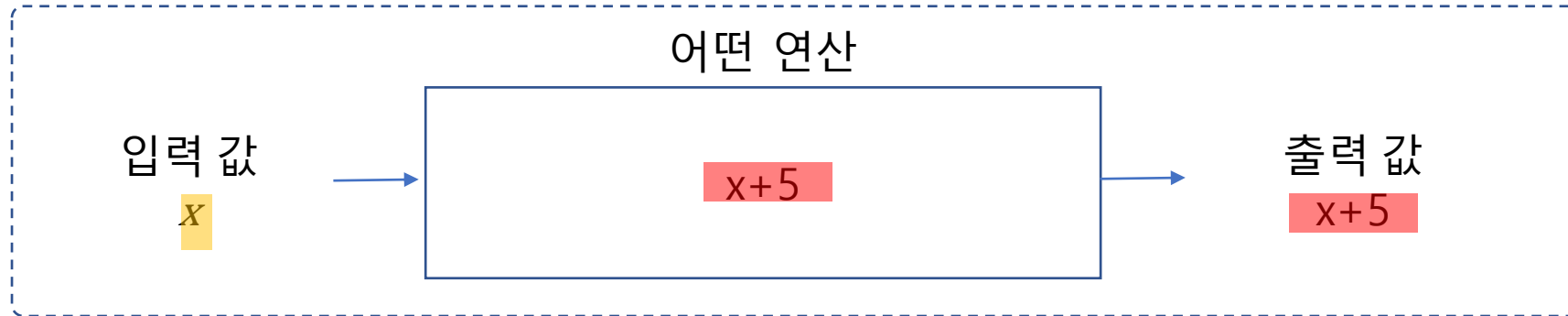
6

```
sum(2,4,6,8,10)
```

30

## ■ 람다

함수( $f$ ) = `sum5`



함수(def)	람다(lambda)
<pre>def sum5(x):     return x + 5</pre>	<pre>sum5 = lambda x: x + 5</pre>

## ■ 람다

함수(def)	람다(lambda)
<pre>def sum5(x):     return x + 5</pre>	<pre>sum5 = lambda x: x + 5</pre>

```
(lambda x : x + 5)(100)
```

105

```
sum5 = lambda x : x + 5  
print(sum5(100))
```

105

```
multiply = lambda x, y : x * y  
print(multiply(4, 5))
```

20

```
(lambda x, y : x * y)(4, 5)
```

20

**[문제]**

사용자로부터 키와 성별을 입력 받아서 권장 체중을 화면에 출력하는 람다식을 작성하세요.

권장 체중은 다음과 같이 계산합니다.

- 남성 권장 체중 = (키 - 100)
- 여성 권장 체중 = (키 - 100) \* 0.9

**[답]**

```
weight2 = lambda man, height : (height - 100) * 0.9 if (man) else (height - 100)
```

```
weight2(False, 170)
```

```
70
```

```
weight2(True, 170)
```

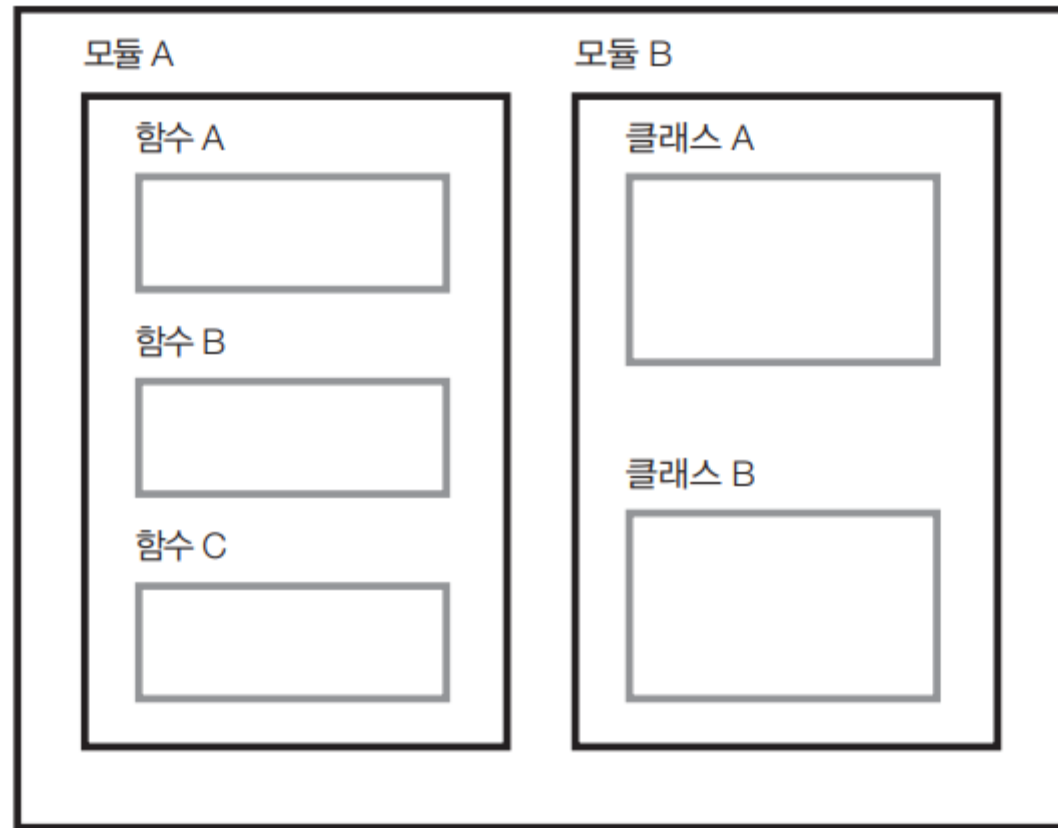
```
63.0
```

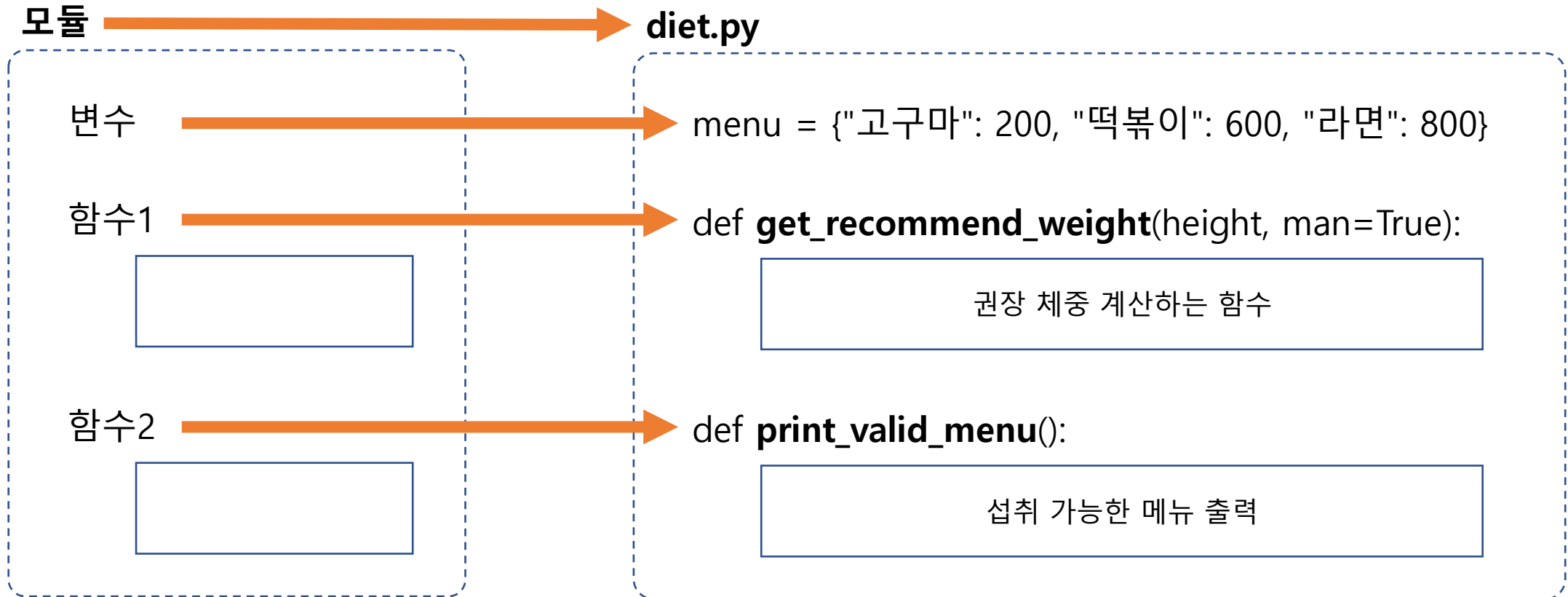
함수(def)	<pre>def weight(man, height):     if (man):         return (height - 100) * 0.9     else:         return (height - 100)</pre>
람다(lambda)	<pre>weight = lambda man, height : (height - 100) * 0.9 if (man) else (height - 100)</pre>

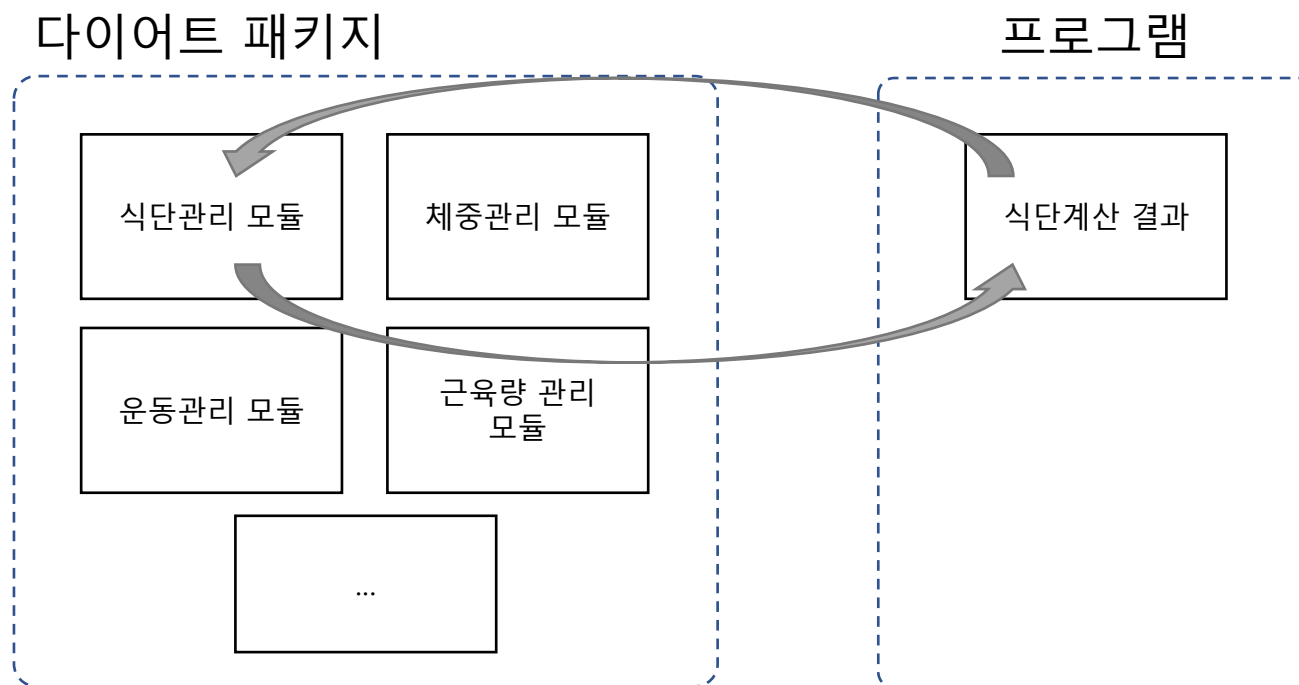
## 08 모듈, 패키지, 클래스

## ■ 패키지, 모듈, 함수, 클래스

패키지









```
In [4]: import pandas as pd
```

```
In [ ]: pd.DataFrame()
```

```
Init signature:
pd.DataFrame(
    data=None,
    index: Union[Collection, NoneType] = None,
    columns: Union[Collection, NoneType] = None,
    dtype: Union[ForwardRef('ExtensionDtype'), str, numpy.dtype, Type[Union[str, float, int, complex, bool]], NoneType] = None,
    copy: bool = False,
)

Docstring:
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

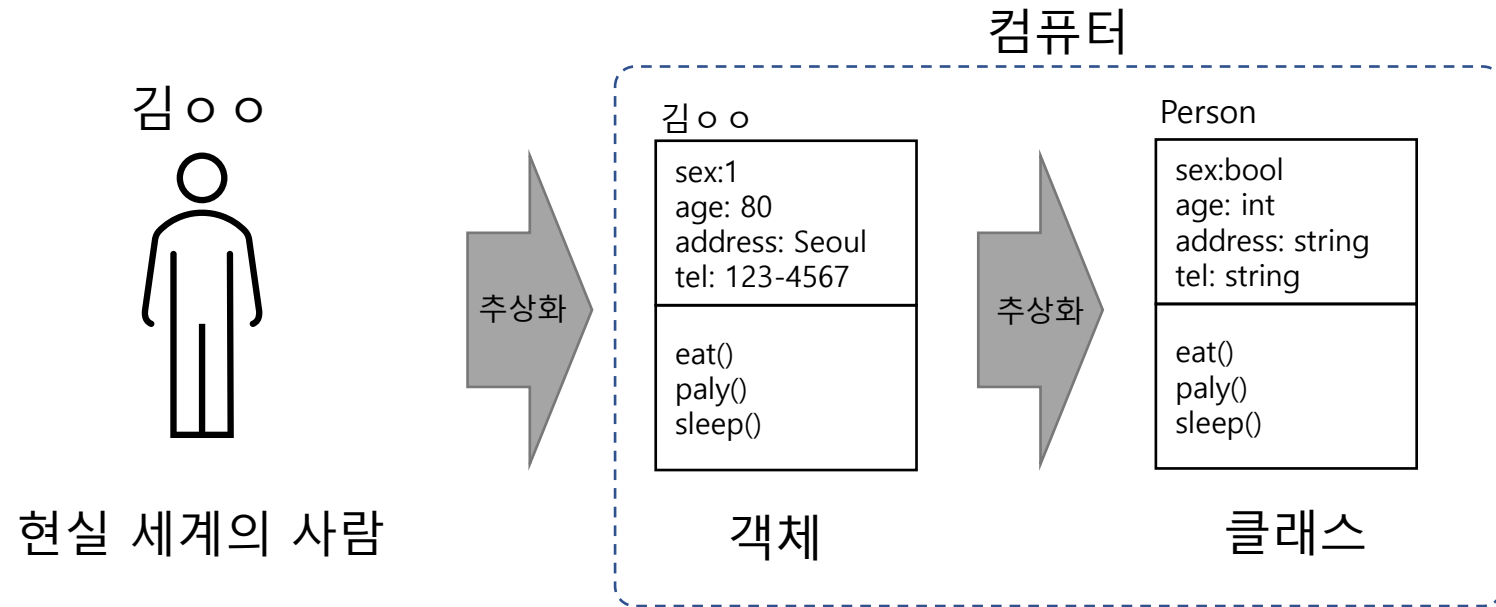
Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
pandas data structure.

Parameters
-----
data : ndarray (structured or homogeneous), Iterable, dict, or DataFrame
    Dict can contain Series, arrays, constants, or list-like objects.

    .. versionchanged:: 0.23.0
        If data is a dict, column order follows insertion-order for
        Python 3.6 and later.

    .. versionchanged:: 0.25.0
        If data is a list of dicts, column order follows insertion-order
        for Python 3.6 and later.

index : Index or array-like
    Index to use for resulting frame. Will default to RangeIndex if
    no indexing information part of input data and no index provided.
columns : Index or array-like
    Column labels to use for resulting frame. Will default to
    RangeIndex (0, 1, 2, ..., n) if no column labels are provided
```



# Q&A

---