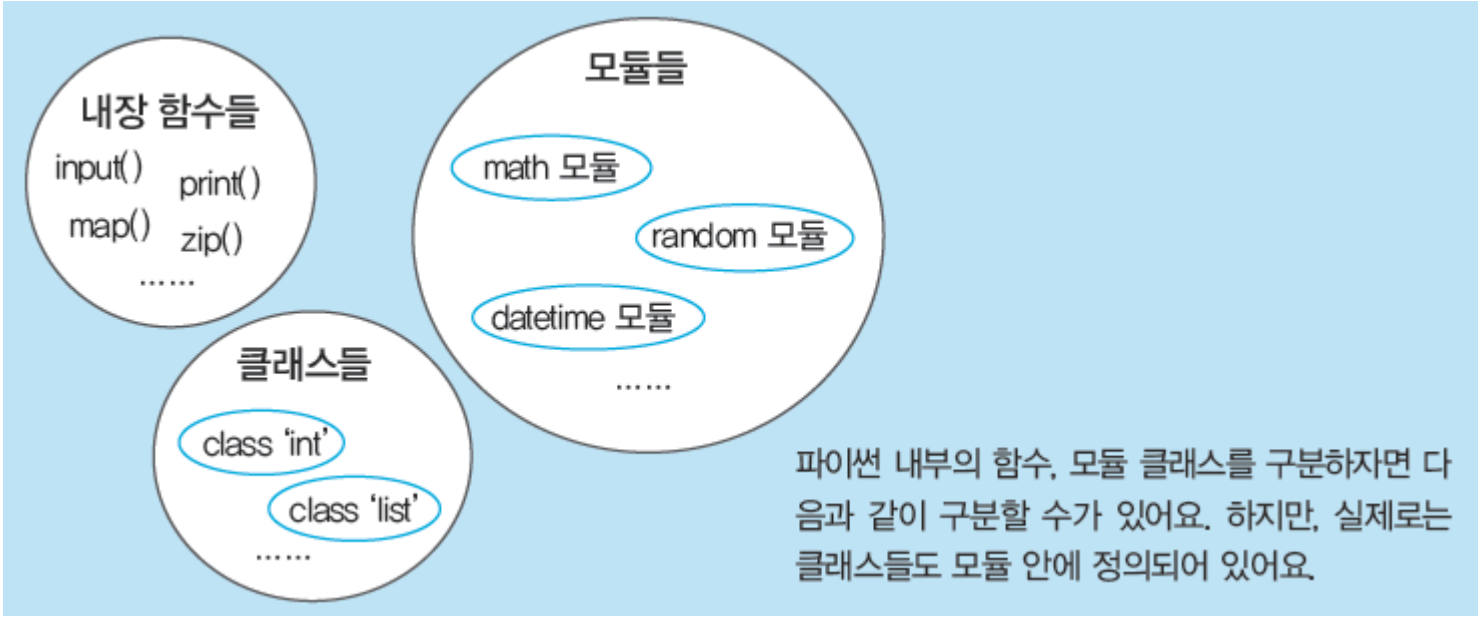
15. 모듈

1. 모듈은 무엇일까요
2. 모듈 사용하기
3. 모듈 만들기
4. random 모듈
5. itertools 모듈
6. keyword 모듈
7. sys 모듈
8. copy 모듈
9. 정리

1. 모듈은 무엇일까요

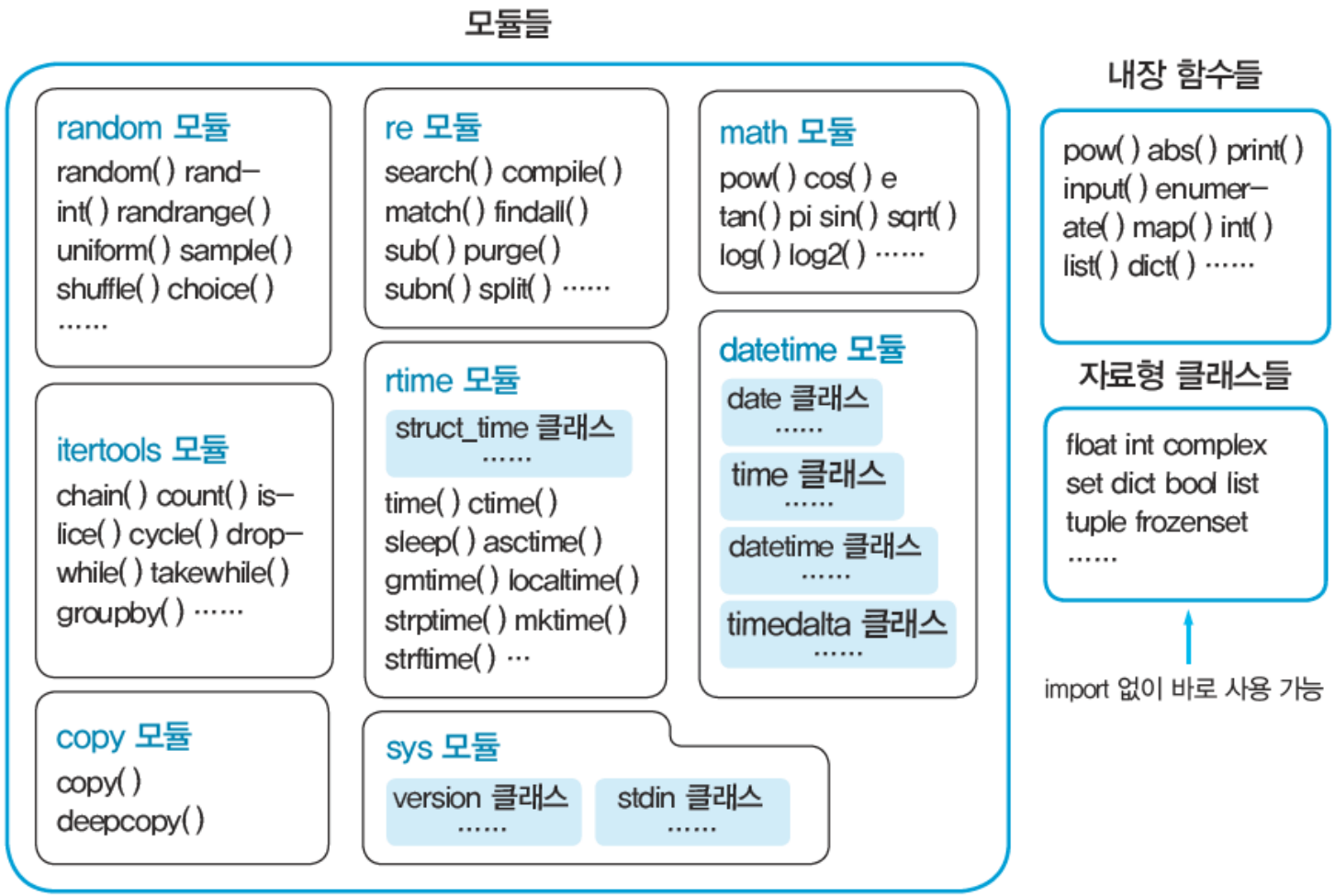
◆ 파이썬 구성

- 비슷한 종류의 일을 하는 함수들 끼리 따로 묶어서 '모듈'이라는 묶음으로 제공
- 내장 함수들은 그냥 사용할 수 있는데, 모듈 안에 있는 함수들은 사용하기 전에 관련 모듈을 먼저 import 해야 함.



1. 모듈은 무엇일까요

◆ 파이썬 구성



파이썬의 모듈, 내장 함수, 클래스

1. 모듈은 무엇일까요

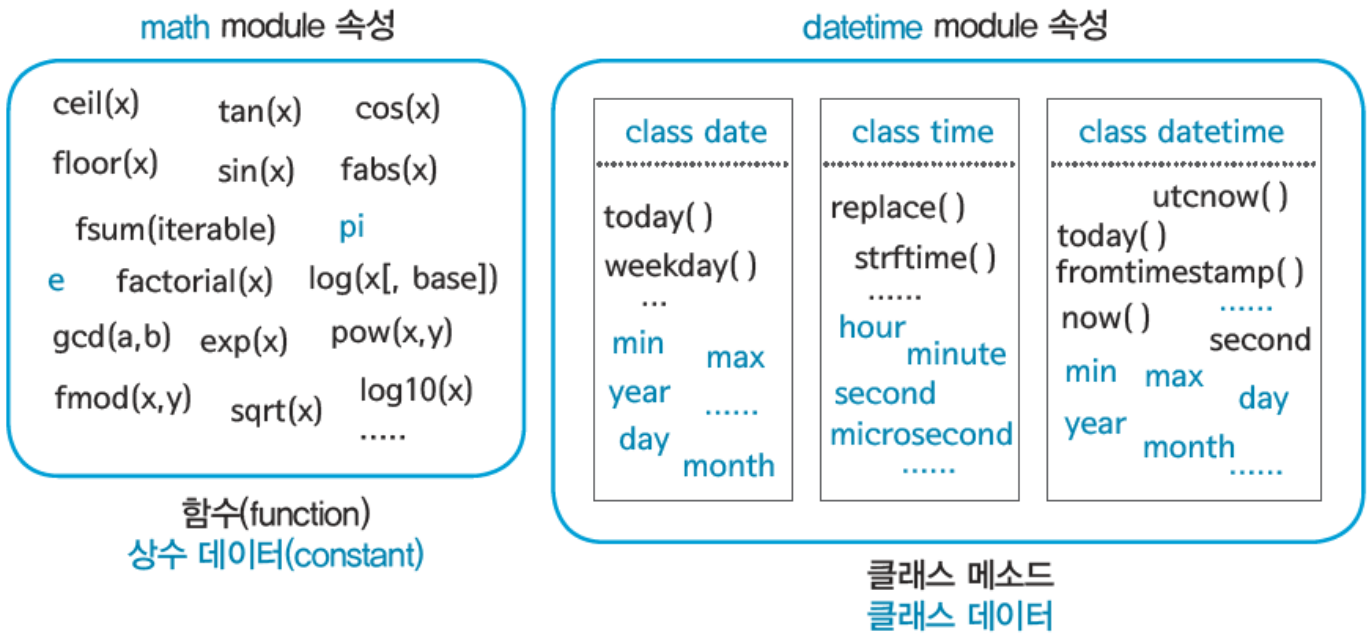
- 파이썬의 세 가지 모듈
  - 표준 모듈 : 파이썬 패키지에 기본적으로 포함된 모듈
  - 사용자 정의 모듈 : 사용자가 직접 만들어서 사용하는 모듈
  - 써드파티(third party) 모듈 : 제 3자가 만들어서 제공하는 모듈
- 표준 모듈인 math 모듈에 속한 함수 목록 보기

```
〉〉〉 import math          # math 모듈을 import 합니다.
〉〉〉 dir(math)            # math 모듈에 있는 함수와 데이터 목록을 알아봅니다.
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees',
'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp',
'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

1. 모듈은 무엇일까요

◆ 표준 모듈인 datetime 모듈에 속한 함수 목록 보기

```
〉〉〉 import datetime
〉〉〉 dir(datetime)
['MAXYEAR', 'MINYEAR', '__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__spec__', '_divide_and_round', 'date',
'datetime', 'datetime_CAPI', 'time', 'timedelta', 'timezone', 'tzinfo']
```



math 모듈과 datetime 모듈의 구성도

2. 모듈 사용하기

◆ import 구문으로 모듈 사용하기

- ‘import 모듈’을 한 후에, ‘모듈.함수()’, ‘모듈.상수’로 사용함.

```
>>> import math
>>> math.pi                    # 원주율 파이값을 알 수 있습니다.
3.141592653589793
>>> math.factorial(5)          # 5 팩토리얼 값을 계산해 줍니다.
120
>>> math.pow(2,5)              # 25을 계산해 줍니다.
32.0
```

2. 모듈 사용하기

◆ import 구문으로 모듈 사용하기

▪ 모듈에서 일부분만 가져다 사용하기


```
〉〉〉 from math import sqrt      # math 모듈에서 sqrt 함수만 사용하겠다는 의미예요.
〉〉〉 sqrt(9)                    # 그냥 함수이름만 사용합니다.
3.0
〉〉〉 math.sqrt(16)
... ...
NameError: name 'math' is not defined
```

2. 모듈 사용하기

- import 구문으로 모듈 사용하기
  - 모듈에서 일부분만 가져다 사용하기
    - import 하지 않은 함수는 사용할 수 없음.

```
>>> from math import pow
>>> pow(2,5)
32.0
>>> sqrt(9)          # math 모듈에 있지만 import하지 않았기 때문에 에러가 발생합니다.
... ...
NameError: name 'sqrt' is not defined
>>> math.sqrt(9)  # 앞에 모듈명을 붙여도 에러를 냅니다.
... ...
NameError: name 'math' is not defined
```

2. 모듈 사용하기

◆ import 구문으로 모듈 사용하기

▪ 모듈에서 일부분만 가져다 사용하기

- 모듈에 있는 함수 여러 개를 가져다 사용하려면 콤마를 이용해서 나열함.

```
>>> from math import sqrt, factorial, pow
>>> sqrt(100)
10.0
>>> factorial(3)
6
>>> pow(2,3)
8.0
```

2. 모듈 사용하기

◆ import 구문으로 모듈 사용하기

▪ 모듈에서 일부분만 가져다 사용하기

- 모듈에 있는 함수 모두 가져다 사용하려면 *를 이용함.

```
>>> from math import *          # math 모듈 멤버 모두 이름으로만 사용합니다.
>>> sqrt(100)                   # 100의 루트 값을 구합니다.
10.0
>>> gcd(20, 32)                 # 20과 32의 최대공약수를 구합니다.
4
```

2. 모듈 사용하기

◆ import 구문으로 모듈 사용하기

- 모듈명에 별명을 붙여서 사용하는 방식

```
import <모듈> as <별명>
```

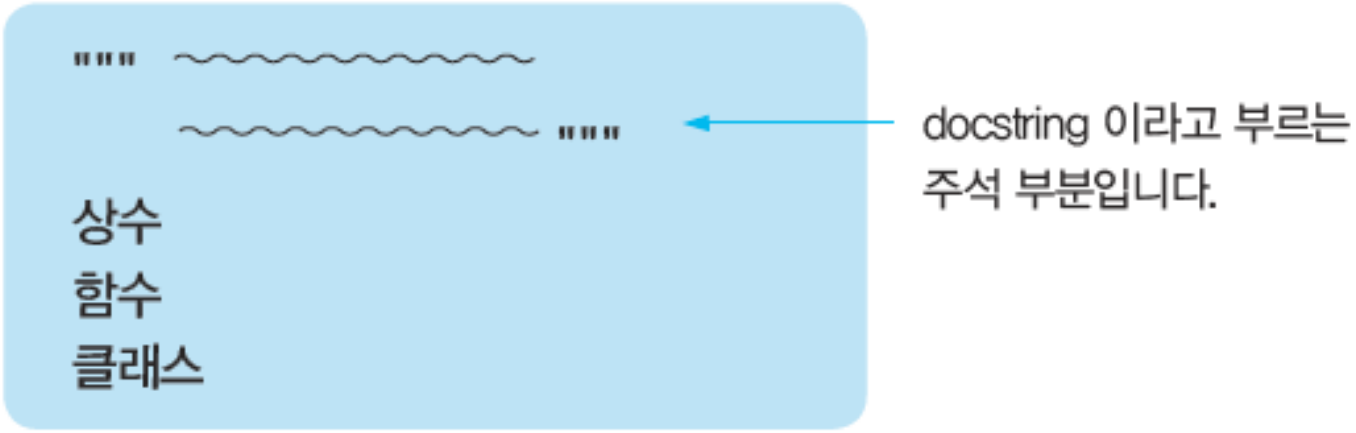
```
>>> import math as mt      # math 대신에 mt라고 쓰겠다는 뜻이에요.
>>> mt.sqrt(100)
10.0
>>> math.sqrt(100)         # 이제는 math라고 쓰면 에러가 발생합니다.
... ...
NameError: name 'math' is not defined
>>> mt.pow(3,5)
243.0
```

3. 모듈 만들기

◆ 사용자가 직접 모듈 만들기

- 모듈에는 상수, 함수, 클래스 등을 저장함.
- 모듈은 다음과 같이 파이썬 코드임.

mymodule.py



- 파일명에서 확장자 py를 떼면 모듈명이 됨.

3. 모듈 만들기

◆ 사용자가 만든 three_calculator 모듈

three_calculator.py (파일명)

```
""" 모듈명 : three_calculator
   add_three(a,b,c) : a,b,c 세 수의 합을 반환합니다.
   multiply_three(a,b,c) : a,b,c 세 수의 곱을 반환합니다.
   count : 상수
"""

count = 3                    # 데이터 상수 넣기

def add_three(a,b,c):        # 사용자 정의 함수 넣기
    total = a + b + c
    return total

def multiply_three(a,b,c):
    result = a * b * c
    return result
```

docstring : 주석

3. 모듈 만들기

◆ 사용자가 만든 three_calculator 모듈 이용하기 (1)

```
import three_calculator    # py를 뗀 이름이 모듈명임.

x = int(input('Enter integer 1 : '))
y = int(input('Enter integer 2 : '))
z = int(input('Enter integer 3 : '))

answer1 = three_calculator.add_three(x, y, z)
answer2 = three_calculator.multiply_three(x, y, z)
print('add : {}'.format(answer1))
print('multiply : {}'.format(answer2))
print('calculate for {} numbers'.format(three_calculator.count))
```

3. 모듈 만들기

◆ 사용자가 만든 three_calculator 모듈 이용하기 (2)

```
from three_calculator import add_three, multiply_three, count
x = int(input('Enter integer 1 : '))
y = int(input('Enter integer 2 : '))
z = int(input('Enter integer 3 : '))

answer1 = add_three(x, y, z)       # three_calculator.add_three라고 쓰지 못합니다.
answer2 = multiply_three(x, y, z)

print('add : {}'.format(answer1))
print('multiply : {}'.format(answer2))
print('calculate for {} numbers'.format(count))
```

4. random 모듈

◆ random 모듈 함수 목록

```
〉〉〉 import random
〉〉〉 dir(random)
[....., 'betavariate', 'choice', 'choices', 'expovariate', 'gammavariate',
'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariate',
'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed',
'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate',
'weibullvariate']
```

4. random 모듈

◆ random 모듈 함수 목록

함수		설명
임의의 정수 선택	randint(a,b)	a <= N <=b 사이의 임의의 정수 N 선택 (a,b: 정수)
	randrange(,,)	range( ) 함수의 결과 중에서 임의의 값 선택
임의의 실수 선택	random( )	0.0 <= F < 1.0 사이의 임의의 실수 F 선택
	uniform(a,b)	a <= F < b 사이의 임의의 실수 선택(a,b: 정수,실수)
컨테이너 자료형 X에서 선택	choice(X)	X에는 시퀀스 자료형인 문자열, 리스트, 튜플만 넣을 수 있음 (집합과 사전에는 사용할 수 없음). X에서 임의의 원소 한 개를 선택함.
	sample(X, k)	X에는 시퀀스 자료형과 집합만 넣을 수 있음(사전에 사용하려면 list( )로 변환해서 사용해야 함). X에서 k개의 원소를 임의로 중복 없이 선택함.
	shuffle(X)	X에는 리스트만 넣을 수 있고, X의 데이터들을 섞어서 리스트로 반환함.

참고 파이썬에서 범위를 나타낼 때, (a, b)처럼 표현하는 경우에 a는 포함되고 b는 포함되지 않는 경우가 대부분이에요. 예를 들어서, range(1,5)라고 하면 1, 2, 3, 4 이렇게 5가 포함되지 않죠. 그런데 random.randint(a,b) 함수는 b가 포함됩니다. 즉, a 이상 b 이하의 수에서 임의의 수를 선택합니다.

4. random 모듈

- random 모듈 함수 - randint(a,b)
  - randint(a,b)의 a,b에는 정수를 넣어야 하고, a 이상 b 이하의 정수 중에서 하나의 정수를 임의로 반환함.

```
>>> import random
>>> random.randint(0,1)      # 0이상 1이하
0
>>> random.randint(5,10)     # 5이상 10이하
5
>>> random.randint(100, 200)
152
>>> random.randint(50000, 100000)
93091
```

```
>>> import random
>>> for i in range(5):
random.randint(10, 20)

17
10
10
20
19
```

4. random 모듈

CODE 87 프로그램을 수행하면 프로그램이 1 ~ 5 사이의 정수를 하나 선택하도록 합니다. 그리고 나서 프로그램이 선택한 수를 맞추는 프로그램을 작성해 볼게요.

```
import random
data = random.randint(1, 5)          # random.randrange(1, 6)도 됩니다.
x = int(input('Choose one number between 1 and 5 : '))
if data == x:
    print('Bingo! You and computer both choose {}.'.format(data))
else:
    print('Wrong! Computer chooses {} and you choose {}.'.format(data, x))
```

[결과 1]

```
Choose one number between 1 and 5 : 2
Wrong! Computer chooses 4 and you choose 2.
```

[결과 2]

```
Choose one number between 1 and 5 : 3
Bingo! You and computer both choose 3.
```

[결과 3]

```
Choose one number between 1 and 5 : 4
Wrong! Computer chooses 3 and you choose 4.
```

4. random 모듈

◆ random 모듈 함수 - randrange(,,)

- randrange(,,) 함수의 인수는 range() 함수와 인수와 같음. range() 범위에 있는 수들 중에서 하나를 임의로 선택해서 반환함.

```
>>> random.randrange(5)          # range(5)에서 임의의 정수 선택
0
>>> random.randrange(3)          # range(3)에서 임의의 정수 선택
2
>>> random.randrange(5,10)       # range(5,10)에서 임의의 정수 선택
7
>>> random.randrange(7,10)       # range(7,10)에서 임의의 정수 선택
8
>>> random.randrange(1, 20, 4)   # range(1, 20, 4)에서 임의의 정수 선택
9
```

4. random 모듈

- random 모듈 함수 – random()
  - random() 함수는 인수가 없고, 0.0 이상 1.0 미만의 실수를 반환함.

>>> random.random() 0.9235512183687059	>>> random.random() 0.5817530846913979

4. random 모듈

- random 모듈 함수 - uniform(a,b)
  - uniform(a,b)의 a와 b에는 실수 또는 정수가 올 수 있고, 함수의 결과로 a 이상 b 미만의 실수를 반환함.

```
>>> random.uniform(3,4)
3.586912161723947
>>> random.uniform(10, 15)
10.164453875505
>>> random.uniform(11,12)
11.52993873767126
```

```
>>> random.uniform(2.2, 3.9)
2.242162960416483
>>> random.uniform(2, 4.9)
3.962491103531939 5
>>> random.uniform(-5.5, 5.5)
-3.396261703231439 3
```

4. random 모듈

◆ random 모듈 함수 - choice(X)

- choice(X) 함수의 인수 X에는 시퀀스 자료형만 넣을 수 있음. X에 있는 원소 중에서 임의로 하나를 선택해서 반환함.

```
>>> S = 'python'
>>> L = [4, 9, 13, 2, 10, 7, 30, 100, 8, 22]
>>> T = (5, 2, 3, 1)
>>> W = {3, 6, 8, 10, 11, 14}
>>> D = {'red':2, 'blue':5, 'green':7, 'white':1}
```

```
>>> import random
>>> random.choice(S)
'h'
>>> random.choice(L)
22
>>> random.choice(T)
2
```

```
>>> random.choice(W)    # 집합
......
TypeError: 'set' object does not support indexing
>>> random.choice(D)    # 사전
......
KeyError: ...
```

4. random 모듈

- random 모듈 함수 - choice(X)
  - choice(X) - List comprehension에서 random.choice(x) 함수를 이용하면 시퀀스 자료형에서 손쉽게 임의의 데이터를 모을 수 있음.

```
>>> L = [4, 100, 22, 89, 54, 67, 15, 8, 33]
>>> A = [random.choice(L) for i in range(5)]  # random.choice(L)을 5회 반복합니다.
>>> print(A)
[4, 33, 67, 8, 33]
>>> B = [random.choice(L) for i in range(3)]  # 중복해서 선택할 수 있어요.
>>> print(B)
[67, 4, 67]
```

4. random 모듈

- random 모듈 함수 목록 - sample(X, k)
  - 컨테이너 자료형 X에서 k를 임의로 중복없이 선택함.
  - k에는 정수가 와야 하고, 0 <= k <= len(X) 범위의 값을 넣어야 함.
  - X에는 문자열, 리스트, 튜플, 집합을 넣을 수 있음. (사전은 안됨)

자료형	예 제

리스트	>>> A = [4, 2, 8, 9, 13, 7, 20]
	>>> random.sample(A, 3) # 리스트 A에서 세 개를 중복 없이 선택합니다.
	[4, 13, 20]
	>>> L = [4, 2, 5, 8, 2] # k가 리스트 길이와 같으면 모두 선택합니다.
	>>> random.sample(L, 5)
	[2, 4, 2, 8, 5]
	>>> random.sample(L,0) # k가 0이면, 빈 리스트가 나옵니다.
	[]

튜플	>>> B = (7, 9, 13, 2, 9, 1)
	>>> random.sample(B, 3)
	[1, 7, 9]
	>>> random.sample(B, 5)
	[9, 7, 2, 13, 9]

4. random 모듈

◆ random 모듈 함수 목록 - sample(X, k)

자료형	예 제
문자열	⟩⟩⟩ C = 'abcdefgh' ⟩⟩⟩ random.sample(C, 3) # 문자열에서 문자들을 선택합니다. ['e', 'a', 'g']
집합	⟩⟩⟩ D = {4, 3, 9, 10, 5} ⟩⟩⟩ random.sample(D, 3) [4, 10, 9] ⟩⟩⟩ E = {4, 3, 9, 10, 5, 3} # 3이 중복되어 있어서 원소가 5개입니다. ⟩⟩⟩ random.sample(E, 6) ..... ValueError: Sample larger than population or is negative ⟩⟩⟩ random.sample(E, 5) [4, 9, 3, 5, 10]
사전	⟩⟩⟩ d = {1:'one', 2:'two', 3:'three', 4:'four'} ⟩⟩⟩ random.sample(d, 2) ..... TypeError: Population must be a sequence or set. For dicts, use list(d). ⟩⟩⟩ random.sample(list(d), 2) [1, 3]

4. random 모듈

- random 모듈 함수 목록 - shuffle(X)
  - X에는 시퀀스 자료형 중에서 mutable한 자료형에만 적용할 수 있음.
  - 리스트에만 적용 할 수 있음.

```
〉〉〉 A = [4, 6, 10, 5, 2, 1, 9]
〉〉〉 random.shuffle(A)
〉〉〉 print(A)
[10, 4, 9, 1, 2, 6, 5]
```

5. itertools 모듈

- itertools 모듈
  - iterable 객체에 대해서 어떤 반복적인 일을 처리하는 데 도움을 주는 함수들로 구성되어 있음.

```
〉〉〉 import itertools
〉〉〉 dir(itertools)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', '_grouper',
'_tee', '_tee_dataobject', 'accumulate', 'chain', 'combinations',
'combinations_with_ replacement', 'compress', 'count', 'cycle', 'dropwhile',
'filterfalse', 'groupby', 'islice', 'permutations', 'product', 'repeat',
'starmap', 'takewhile', 'tee', 'zip_longest']
```

5. itertools 모듈

◆ chain() 함수

- iterable 객체에서 원소를 하나씩 넘겨 줌.
- 두 리스트를 연속해서 출력하는 일을 chain() 함수를 이용하면 효율적으로 할 수 있음.

```
from itertools import chain
L1 = [1,3,5]
L2 = [2,4,6,8]
for x in chain(L1, L2):
    print(x, end=' ')
```

L1이 순서대로 출력되고 L2가 연이어서 출력됩니다.

[결과]

1 3 5 2 4 6 8

5. itertools 모듈

◆ chain() 함수

- itertools.chain() 함수를 사용하지 않더라도 리스트 두 개를 +로 연결하여 처리할 수도 있음.
- 그러나 + 로 연결하여 처리하면, 두 리스트를 합한 새로운 리스트가 중간에 생기게 됨. 따라서 itertools.chain()이 효율적임.

```
L1 = [1, 3, 5]
L2 = [2, 4, 6, 8]
for x in L1 + L2:          # L1 + L2 자리에 두 리스트를 합한 새로운 리스트가 생깁니다.
    print(x, end=' ')
```

5. itertools 모듈

- chain() 함수 예제

import itertools
 	
L = [1, 3, 5]
T = (10, 20)
S = 'python'
 	
for x in itertools.chain(L, T, S):
  print(x, end=' ')
	import itertools
	
	S = {'red', 'blue', 'yellow'}
	D = {100:'hundred', 1000:'thousand'}
	
	for x in itertools.chain(S, D, range(5)):
	  print(x, end=' ')

[결과] 1 3 5 10 20 p y t h o n	[결과] red yellow blue 100 1000 0 1 2 3 4

5. itertools 모듈

◆ count() 함수

- 인수로 넣는 정수로부터 수열을 무한히 만들어 나갈 수 있음.

count( )	0 1 2 3 4 5 6 7 8 9 10 ....
count(100)	100 101 102 103 104 105 106 107 108 109 110 111 112 ...
count(100, 3)	100 103 106 109 112 115 118 121 124 127 130 133 136 ...

```
from itertools import count
for x in count(100):
    print(x, end=' ')
```

100 101 102 103 104 105 ...
한없이 출력됩니다.

- zip() 내장 함수와 같이 사용하면 효율적임.

```
from itertools import count
A = ['one', 'two', 'three']
B = zip(count(1), A)
print(list(B))
```

[결과]

[(1, 'one'), (2, 'two'), (3, 'three')]

5. itertools 모듈

- dropwhile(f, X) 함수
  - f - 참 또는 거짓을 반환하는 함수, X - iterable 객체
  - X에 저장된 각 데이터에 대해서 함수 f를 적용하여 결과가 True인 데이터는 버리고 처음으로 False가 되는 데이터를 찾아서 그 데이터 이후의 모든 데이터를 반환함.

```
from itertools import dropwhile
def find_even(x):
    if x%2 == 0:
        return True
    return False
L = [4, 8, 10, 3, 6, 7, 1]
for x in dropwhile(find_even, L):
    print(x, end=' ')
```

[결과]
3 6 7 1

```
from itertools import dropwhile

L = [4, 3, 10, 2, 6, 7, 1]
for x in dropwhile(lambda x: x<5, L):
    print(x, end=' ')
```

[결과]
10 2 6 7 1

5. itertools 모듈

- takewhile(f, X) 함수
  - f - 참 또는 거짓을 반환하는 함수, X - iterable 객체
  - X에 저장된 각 데이터에 대해서 함수 f를 적용하여 결과가 True인 동안의 데이터를 모두 반환함.

```
from itertools import takewhile
def find_even(x):
    if x%2 == 0:
        return True
    return False
L = [4, 8, 10, 3, 6, 7, 1]
for x in takewhile(find_even, L):
    print(x, end=' ')
```

[결과]
4 8 10

```
from itertools import takewhile

L = [4, 3, 10, 2, 6, 7, 1]
for x in takewhile(lambda x: x<5, L):
    print(x, end=' ')
```

[결과]
4 3

5. itertools 모듈

- groupby(X, f) 함수
  - X - iterable 객체 (두 번째 인수는 없을 수도 있음, 있다면 f는 함수임)
  - f가 없다면, X에 있는 데이터 자체가 키가 되어 그룹으로 묶음.
  - f가 있다면, 함수 f를 기준으로 X를 그룹으로 묶음.

```
>>> from itertools import groupby
>>> L = [1, 1, 2, 2, 2, 2, 3, 3, 3]
>>> x = groupby(L)      # 두 번째 인수가 없어서 L에 있는 데이터 자체가 키가 됩니다.
>>> for k, v in x:
        print(k, ':', list(v))

1 : [1, 1]
2 : [2, 2, 2, 2]
3 : [3, 3, 3]
```

5. itertools 모듈

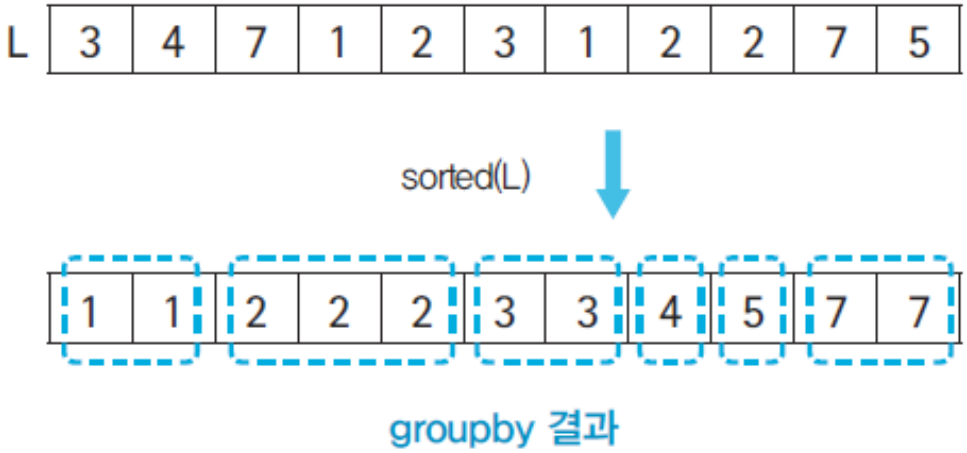
- groupby(X, f) 함수

```
from itertools import groupby

L = [3, 4, 7, 1, 2, 3, 1, 2, 2, 7, 5]
for k, v in groupby(sorted(L)):
    print(k, list(v))
```

[결과]

```
1 [1, 1]
2 [2, 2, 2]
3 [3, 3]
4 [4]
5 [5]
7 [7, 7]
```



5. itertools 모듈

- groupby(X, f) 함수 예제
  - 리스트 L은 정수들이 저장되어 있음. 각 정수가 몇 번 반복해서 나오는지 출력하는 코드임.

```
from itertools import groupby

L = [3, 4, 7, 1, 2, 3, 1, 2, 2, 7, 5]
for k, v in groupby(sorted(L)):
    print('{} : {}개'.format(k, len(list(v))))
```

[결과]

```
1 : 2개
2 : 3개
3 : 2개
4 : 1개
5 : 1개
7 : 2개
```

5. itertools 모듈

◆ groupby(X, f) 함수 예제

- 리스트 words에는 단어 여러 개가 저장되어 있음. 이 리스트에서 같은 단어만을 묶어서 출력하는 코드임.

```
from itertools import groupby

words = ['table', 'book', 'boy', 'red', 'table',
'boy', 'home', 'home', 'boy', 'rose','table',
'boy', 'top', 'pass', 'hat','rose', 'rose', 'dog',
'nose', 'table']

for k, group in groupby(sorted(words)):
    print(k, list(group))
```

[결과]

```
book ['book']
boy ['boy', 'boy', 'boy', 'boy']
dog ['dog']
hat ['hat']
home ['home', 'home']
nose ['nose']
pass ['pass']
red ['red']
rose ['rose', 'rose', 'rose']
table ['table', 'table', 'table', 'table']
top ['top']
```

5. itertools 모듈

◆ groupby(X, f) 함수 예제

- 리스트 words에서 같은 알파벳으로 시작하는 단어들을 묶어서 출력하는 코드 (groupby() 사용하지 않음).

```
words = ['table', 'book', 'pen', 'red', 'phone', 'eraser', 'house', 'home',
         'computer', 'chair', 'mirror', 'boy', 'top', 'pass', 'hat', 'cat',
         'air', 'rose', 'dog', 'nose', 'boat', 'apple']
```

```
d = {}
for word in sorted(words):
    if word[0] in d:
        d[word[0]].append(word)
    else:
        d[word[0]] = []
        d[word[0]].append(word)
for k in d.keys():
    d[k].sort()
for k, v in d.items():
    print(k, v)
```

[결과]

```
a ['air', 'apple']
b ['boat', 'book', 'boy']
c ['cat', 'chair', 'computer']
d ['dog']
e ['eraser']
h ['hat', 'home', 'house']
m ['mirror']
n ['nose']
p ['pass', 'pen', 'phone']
r ['red', 'rose']
t ['table', 'top']
```

5. itertools 모듈

◆ groupby(X, f) 함수 예제

CODE 88 앞의 코드를 groupby() 함수를 이용하여 수정하기.

```
from itertools import groupby

words = ['table', 'book', 'pen', 'red', 'phone', 'eraser', 'house', 'home',
'computer', 'chair', 'mirror', 'boy', 'top', 'pass', 'hat', 'cat',
'air', 'rose', 'dog', 'nose', 'boat', 'apple']

for k, group in groupby(sorted(words), lambda w: w[0]):
    print(k, list(group))
```

5. itertools 모듈

CODE 89 다음과 같이 colors 리스트에 데이터가 저장되어 있습니다. groupby 함수를 이용 하여 오른쪽과 같은 결과가 나오도록 코딩해 볼게요(이 문제는 그룹을 짓는 문제가 아니고, groupby( ) 함수의 결과가 사전의 형태임을 연습해 보는 문제예요).

```
from itertools import groupby
flowers = [('red', 'rose'), ('yellow', 'sunflower'), ('yellow', 'iris'),
           ('red', 'anemone'), ('red', 'peony'), ('purple', 'lavendar'),
           ('orange', 'begonia'), ('purple', 'windflower')]
F = {}
for key, group in groupby(sorted(flowers), lambda x: x[0]):
    F[key] = list(group)
import pprint
pprint.pprint(F, width='40')
```

[결과]

```
{ 'orange': [('orange', 'begonia')],
'purple' : [('purple', 'lavenda')
             ('purple', 'windflower')],
'red': [('red', 'anemone'),
        ('red', 'peony'),
        ('red', 'rose')],
'yellow': [('yellow', 'iris')
           ('yellow', 'sunflower')]}
```

6. keyword 모듈

- keyword 모듈 – 파이썬 키워드 정보를 알 수 있음.

```
>>> import keyword
>>> dir(keyword)
['__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__spec__', 'iskeyword', 'kwlist', 'main']
>>> keyword.kwlist                    # 모든 키워드 목록이 나옵니다.
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del',
'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',
'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>> len(keyword.kwlist)               # 키워드의 개수가 33개임을 알 수가 있습니다.
33
>>> keyword.iskeyword('global')       # 'global'은 파이썬 키워드입니다.
True
>>> keyword.iskeyword('delete')       # 'delete'는 파이썬 키워드가 아닙니다.
False
```

7. sys 모듈

◆ sys 모듈 - 파이썬 인터프리터에 대한 정보

- 파이썬 버전 정보 - sys.version, sys.version_info

```
>>> sys.version
'3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]'
>>> sys.version_info
sys.version_info(major=3, minor=6, micro=3, releaselevel='final', serial=0)
```

- 표준 모듈 목록 - sys.modules

```
for module_name in sys.modules.keys():    # sys.modules 결과는 사전입니다.
    print(module_name)
```

7. sys 모듈

- sys 모듈 - 파이썬 인터프리터에 대한 정보
  - 표준 모듈의 위치 - sys.path

```
>>> sys.path
['C:/Users/sogang-pc/AppData/Local/Programs/Python/Python36-32', ..... ,
'C:₩₩ Users₩₩sogang-pc₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩Python36-32',
'C:₩₩Users₩₩sogang-pc₩₩AppData₩₩Local₩₩Programs₩₩Python₩₩ Python36-
32₩₩lib₩₩site-packages']
```

  - 사용 중인 기본 문자열 인코딩 - sys.getdefaultencoding()

```
>>> sys.getdefaultencoding()
'utf-8'
```

7. sys 모듈

◆ sys 모듈 - sys.stdin()

- 표준 입력 - sys.stdin()을 사용하면 표준 입력을 좀 더 융통성있게 해결할 수 있음.

```
first_name = input('Enter first name : ')
last_name = input('Enter last name : ')
print('You are ' + first_name + ' ' + last_name + '.')
```

sys 모듈을 이용하면 이 두 줄을 한 줄로 해결할 수 있습니다.

[결과]

```
Enter first name : Alice
Enter last name : Lee
You are Alice Lee.
```

7. sys 모듈

◆ sys 모듈 - sys.stdin()

```
① import sys
② print('Enter your name : ', end='')
③ name = sys.stdin.readline( )
④ first_name, last_name = name.split( )
⑤ print('You are ' + first_name + ' ' + last_name + '.')
```

한 줄에 여러 개의 데이터를 입력받을 수 있음.
입력받은 데이터는 문자열임.

[결과]

```
Enter your name : Alice Lee
You are Alice Lee.
```

7. sys 모듈

◆ sys 모듈 - sys.stdin()

- 여러 개의 숫자를 입력받아 합을 구하는 코드

```
① import sys
② print('Enter integers : ', end='')
③ numbers = sys.stdin.readline( )
④ num = numbers.split( )
⑤ total = 0
⑥ for n in num:
⑦       total += int(n)
⑧ print('total :', total)
```

[결과]

Enter integers : 3 9 10 12 5 1
total : 40

7. sys 모듈

CODE 90 앞의 코드를 여러 개를 입력받는 방식과 itertools.groupby( ) 함수를 이용해서 다음과 같은 결과가 출력되는 코드를 작성해 보겠습니다.

[결과 1]

Enter two integers : 15 4
0 : [0, 1, 2, 3]
1 : [4, 5, 6, 7]
2 : [8, 9, 10, 11]
3 : [12, 13, 14, 15]

0부터 15까지의 수를 크기가 4인 그룹으로 묶기

[결과 2]

Enter two integers : 20 6
0 : [0, 1, 2, 3, 4, 5]
1 : [6, 7, 8, 9, 10, 11]
2 : [12, 13, 14, 15, 16, 17]
3 : [18, 19, 20]

[결과 3]

Enter two integers : 20 7
0 : [0, 1, 2, 3, 4, 5, 6]
1 : [7, 8, 9, 10, 11, 12, 13]
2 : [14, 15, 16, 17, 18, 19, 20]

[결과 4]

Enter two integers : 7 3
0 : [0, 1, 2]
1 : [3, 4, 5]
2 : [6, 7]

7. sys 모듈

CODE 90 (계속)

```
import sys
import itertools

print('Enter two integers : ', end='')
numbers = sys.stdin.readline( )
num1, num2 = numbers.split( )
num1, num2 = int(num1), int(num2)

for k, group in itertools.groupby(range(num1+1), lambda x : x // num2):
    print(k, ':', list(group))
```

8. copy 모듈

◆ copy 관련 메소드들

- mutable 자료형만 copy() 메소드를 갖고 있음 (리스트, 집합, 사전).
- copy 모듈의 copy() 함수와 deepcopy() 함수

얕은 복사	깊은 복사
▪ mutable 자료형의 copy() 메소드 ▪ copy 모듈의 copy() 함수	▪ copy 모듈의 deepcopy() 함수

8. copy 모듈

◆ 얕은 복사

- mutable 자료형의 copy() 메소드

```
>>> L = [3, 7, 20, 5]
>>> M = L        # L, M은 객체를 공유
>>> id(L), id(M)
(35467312, 35467312)
>>> M[2] = 100
>>> print(L, M)
[3, 7, 100, 5]  [3, 7, 100, 5]
```

```
>>> L = [3, 7, 20, 5]
>>> M = L.copy()    # L, M은 다른 객체
>>> id(L), id(M)
(35599424, 35599784)
>>> M[2] = 100
>>> print(L, M)
[3, 7, 20, 5]  [3, 7, 100, 5]
```

8. copy 모듈

◆ 얕은 복사

- mutable 자료형의 copy() 메소드, [:], [::]

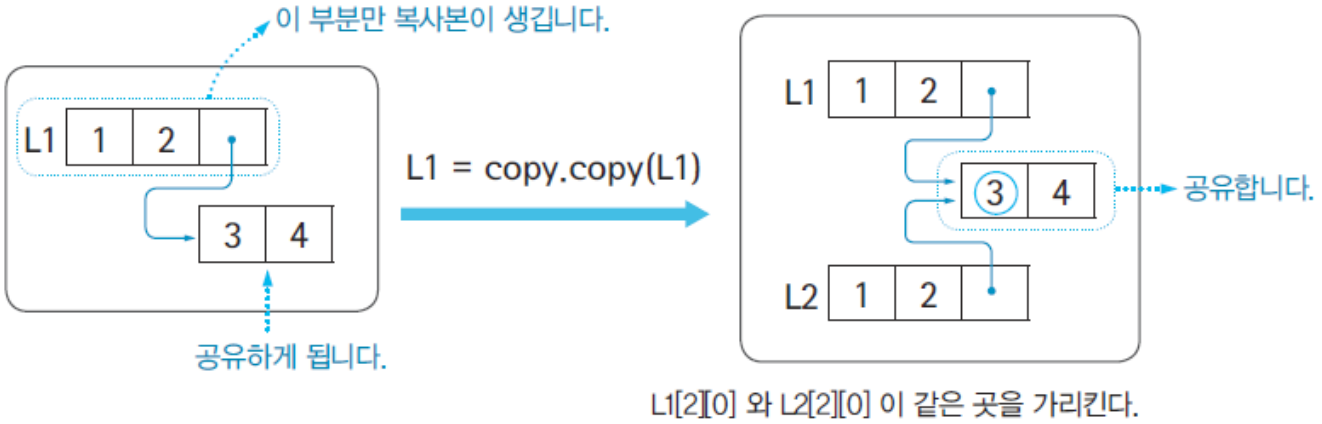
L1 = [1, 2, [3, 4]] L2 = L1.copy( ) L2[2] [0] = 100 print('L1 :', L1) print('L2 :', L2)	L1 = [1, 2, [3, 4]] L2 = L1 [:] L2[2] [0] = 100 print('L1 :', L1) print('L2 :', L2)	import copy L1 = [1, 2, [3, 4]] L2 = copy.copy(L1) L2[2] [0] = 100 print('L1 :', L1) print('L2 :', L2)
[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]	[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]	[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]

8. copy 모듈

◆ 얕은 복사

- mutable 자료형의 copy() 메소드, copy 모듈의 copy() 함수, [:], [::]

L1 = [1, 2, [3, 4]] L2 = L1.copy( ) L2 [2] [0] = 100 print('L1 :', L1) print('L2 :', L2)	L1 = [1, 2, [3, 4]] L2 = L1 [:] L2 [2] [0] = 100 print('L1 :', L1) print('L2 :', L2)	import copy L1 = [1, 2, [3, 4]] L2 = copy.copy(L1) L2 [2] [0] = 100 print('L1 :', L1) print('L2 :', L2)
[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]	[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]	[결과] L1 : [1, 2, [100, 4]] L2 : [1, 2, [100, 4]]



8. copy 모듈

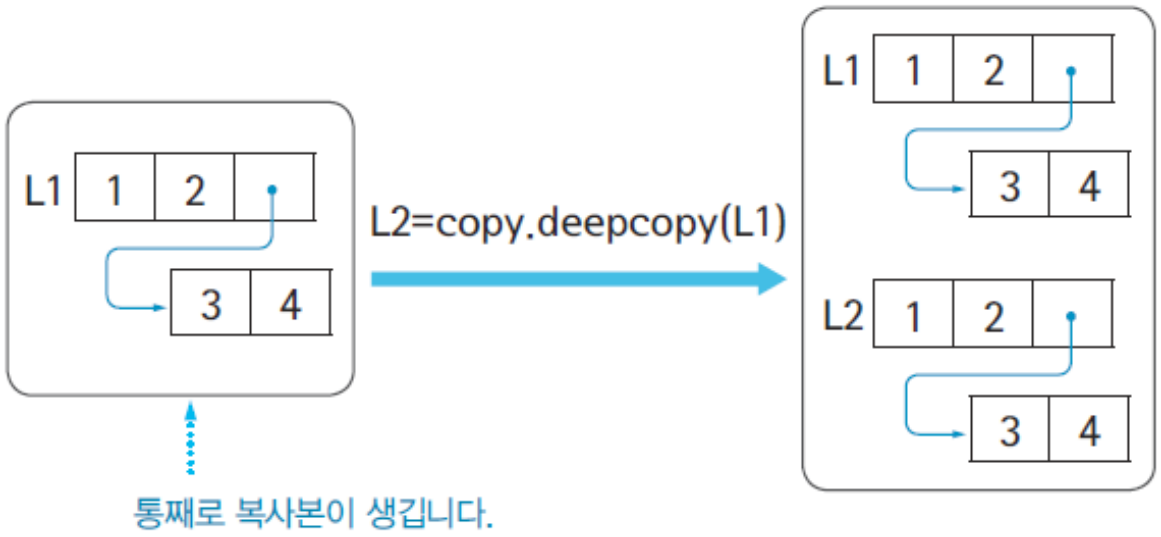
◆ 깊은 복사

▪ copy 모듈의 deepcopy() 함수

```
import copy
L1 = [1, 2, [3, 4]]
L2 = copy.deepcopy(L1)
L2[2][0] = 100
print('L1 :', L1)
print('L2 :', L2)
```

[결과]

```
L1 : [1, 2, [3, 4]]
L2 : [1, 2, [100, 4]]
```



9. 정리

- 파이썬 모듈에는 표준 모듈, 사용자 정의 모듈, 써드파티 모듈이 있음.
- 모듈에는 함수, 클래스, 상수 등이 포함됨.
- math 모듈, random 모듈, itertools 모듈, sys 모듈에는 유용한 함수들이 있음.
- 사용자가 필요할 때는 모듈을 만들어 사용할 수 있음.