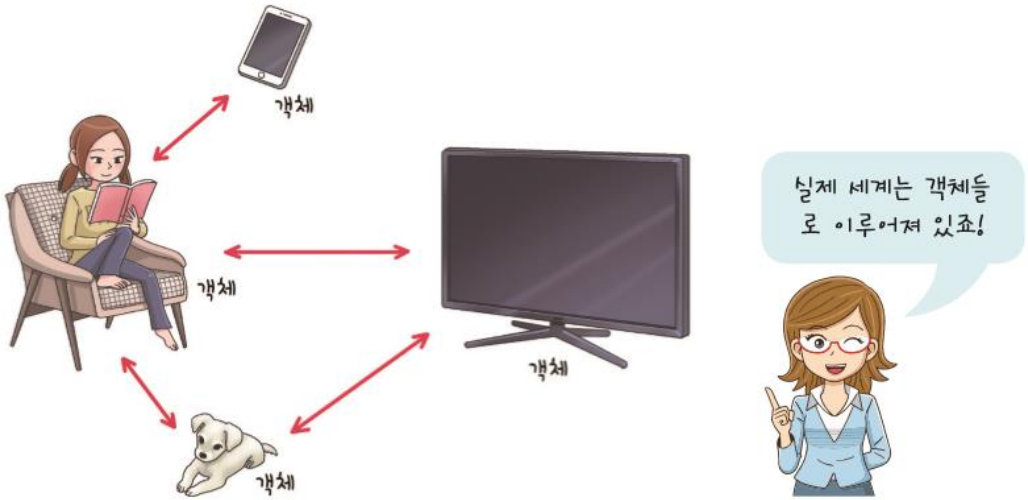
6장 클래스와 모듈

객체지향 프로그래밍

- 객체 지향 프로그래밍(OOP: object-oriented programming)은 우리가 살고 있는 실제 세계가 객체(object)들로 구성되어 있는 것과 비슷하게, 소프트웨어도 객체로 구성하는 방법이다.

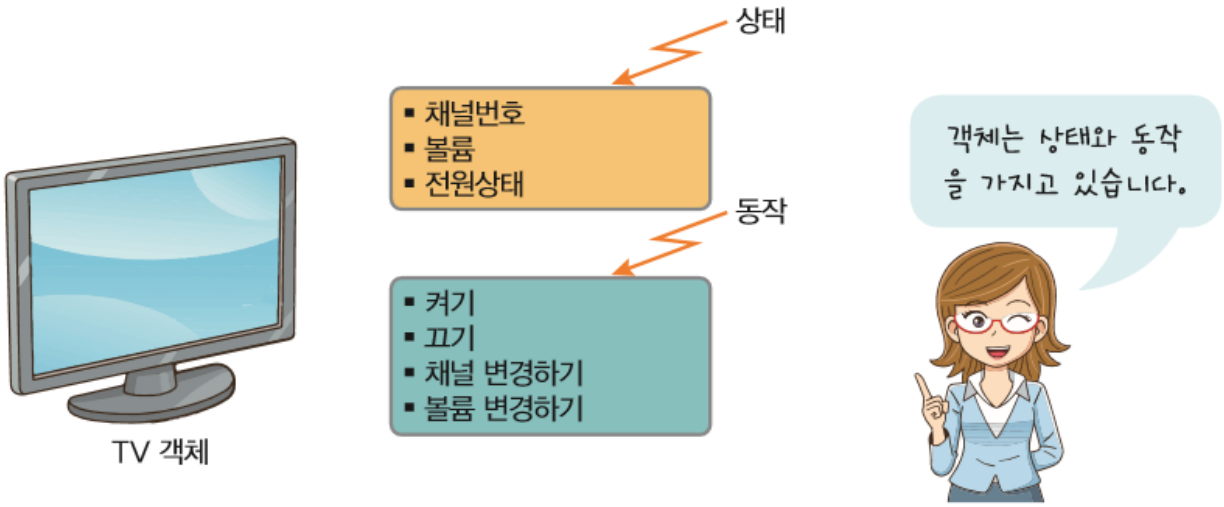


객체

객체는 상태와 동작을 가지고 있다.

객체의 상태(state)는 객체의 속성이다.

객체의 동작(behavior)은 객체가 취할 수 있는 동작(기능)이다.



인스턴스 변수와 메소드

객체=필드+메소드

텔레비전 객체

필드

- channel
- volume
- on

메소드

- turnOn()
- turnOff()
- changeChannel()
- changeVolume()

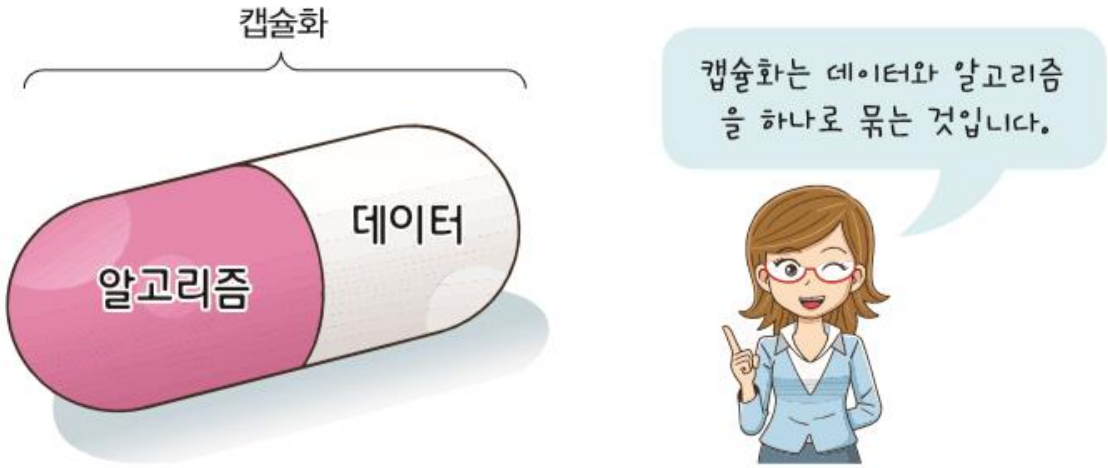
클래스란?

객체에 대한 설계도를 클래스(class)라고 한다.

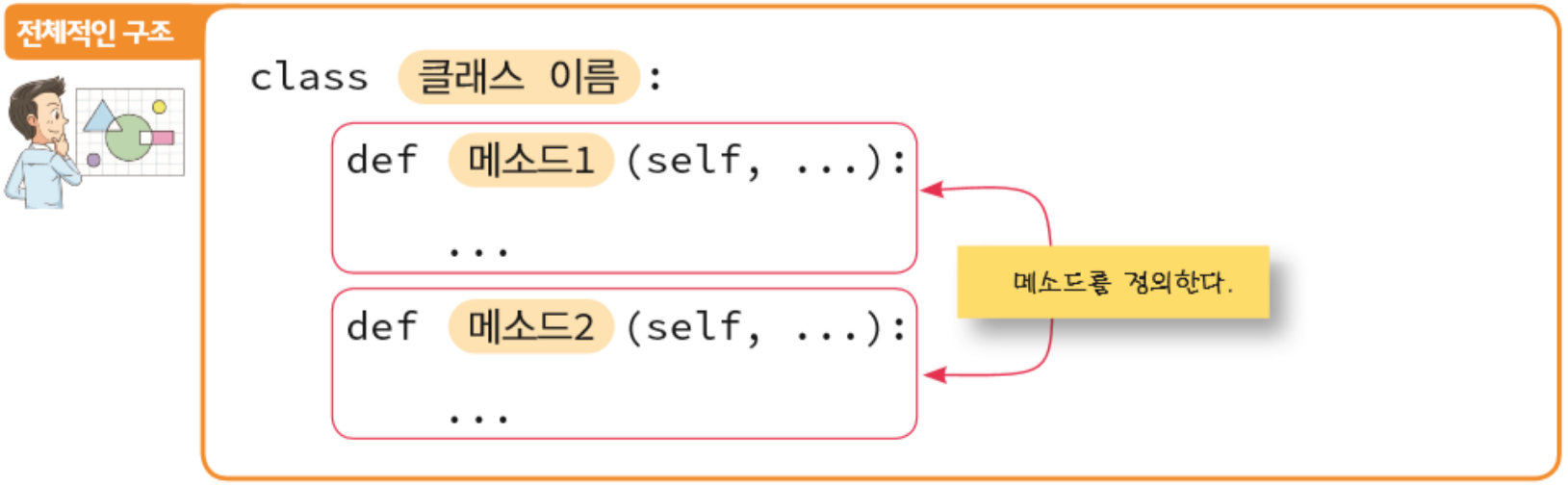
클래스로부터 만들어지는 각각의 객체를 그 클래스의 인스턴스(instance)라고 한다.

캡슐화

데이터와 알고리즘을 하나로 묶고 공용 인터페이스만 제공하고 구현 세부 사항을 감추는 것은 캡슐화(encapsulation)라고 한다.

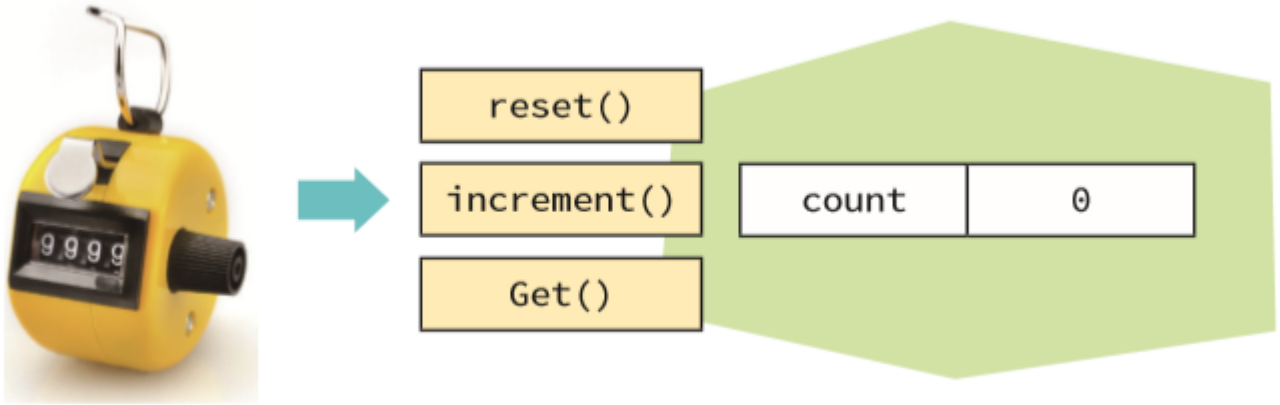


클래스 작성하기



클래스의 예

Counter 클래스를 작성하여 보자. Counter 클래스는 기계식 계수기를 나타내며 경기장이나 콘서트에 입장하는 관객 수를 세기 위하여 사용할 수 있다.



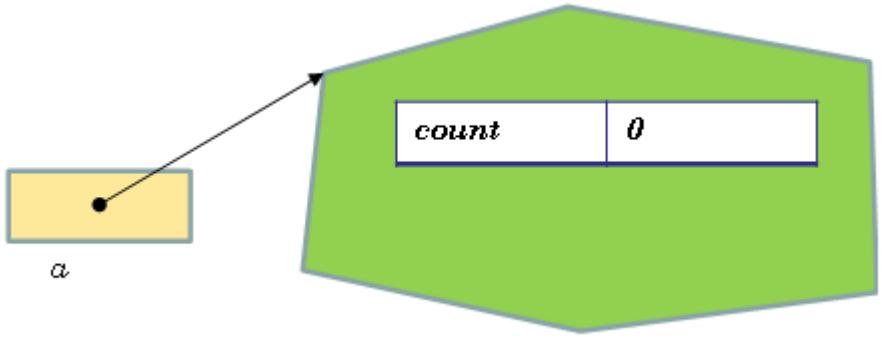
COUNTER 클래스

```
class Counter:
    def reset(self):
        self.count = 0
    def increment(self):
        self.count += 1
    def get(self):
        return self.count
```

객체 생성

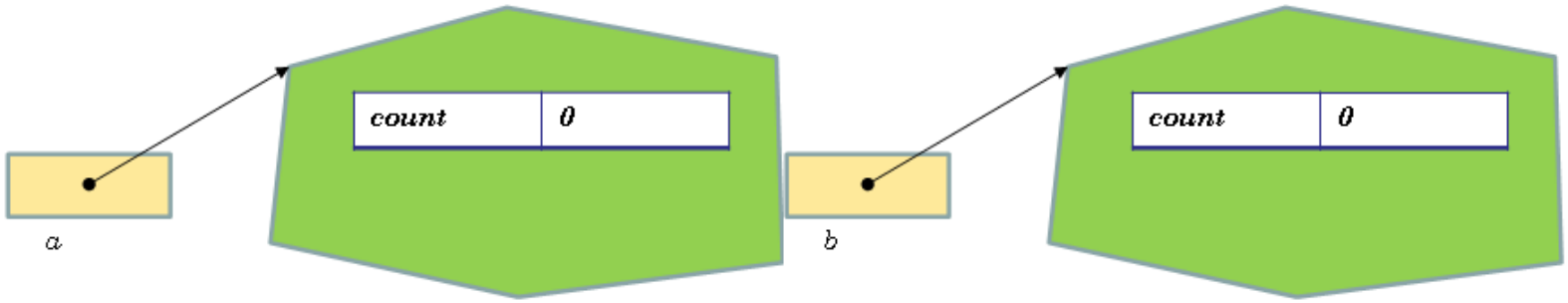
```
a = Counter()

a.reset()
a.increment()
print("카운터 a의 값은", a.get())
```

```
카운터 a의 값은 1
```



객체 2개 생성하기

```
a = Counter()
b = Counter()

a.reset()
b.reset()
```



생성자

- 생성자(constructor)는 객체가 생성될 때 객체를 기본값으로 초기화하는 특수한 메소드이다.

생성자의 예

전체적인 구조

```
class 클래스 이름:
    def __init__(self, ...):
        ...
```

__init__() 메소드가 생성자이다.
여기서 객체의 초기화를 담당한다.

```
class Counter:
    def __init__(self) :
        self.count = 0
    def reset(self) :
        self.count = 0
    def increment(self):
        self.count += 1
    def get(self):
        return self.count
```

메소드 정의

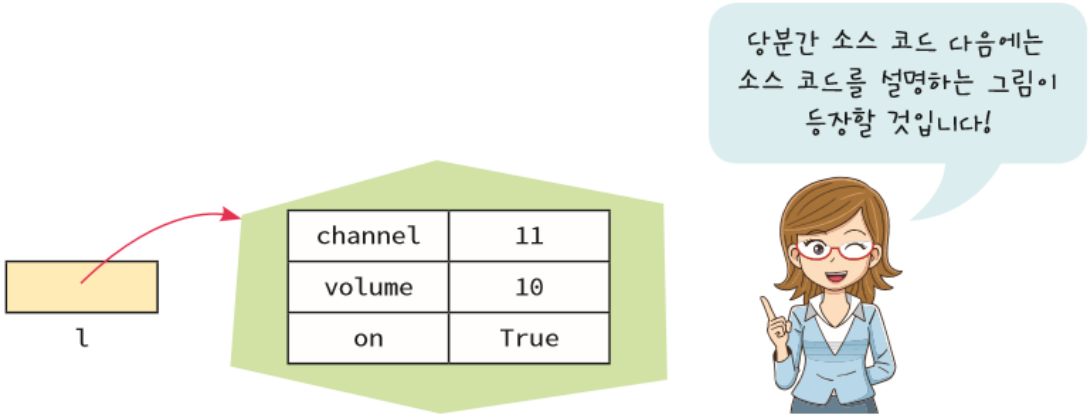
- **메소드는 클래스 안에 정의된 함수이므로 함수를 정의하는 것과 아주 유사하다. 하지만 첫 번째 매개변수는 항상 self이어야 한다.**

```
class Television:
        def __init__(self, channel, volume, on):
                self.channel = channel
                self.volume = volume
                self.on = on

        def show(self):
                print(self.channel, self.volume, self.on)

        def setChannel(self, channel):
                self.channel = channel

        def getChannel(self):
                return self.channel
```

메소드호출

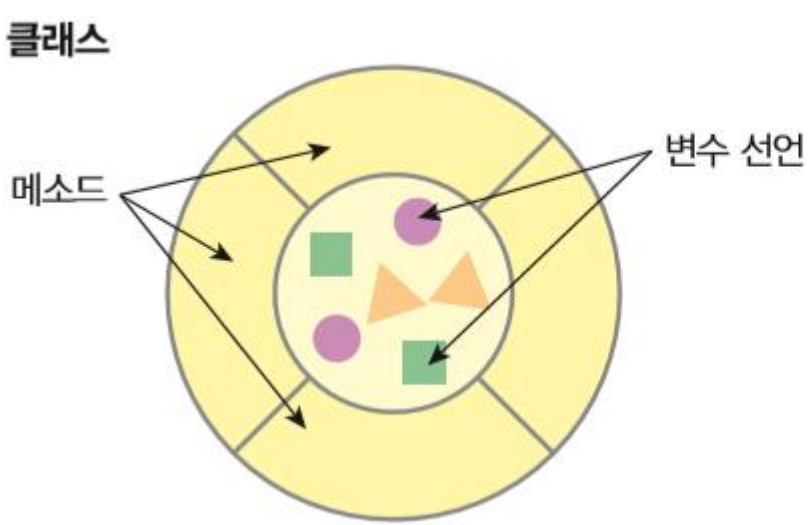
```
t = Television(9, 10, True)

t.show()
t.setChannel(11)
t.show()
```

```
9 10 True
11 10 True
```



정보 은닉

구현의 세부 사항을 클래스 안에 감추는 것



```
class Student:
        def __init__(self, name=None, age=0):
                self.__name = name
                self.__age = age

obj=Student()
print(obj.__age)
```

```
...
AttributeError: 'Student' object has no attribute '__age'
```

접근자와 설정자

- 하나는 인스턴스 변수값을 반환하는 접근자(getters)이고 또 하나는 인스턴스 변수값을 설정하는 설정자(setters)이다.

```
class Student:
        def __init__(self, name=None, age=0):
                self.__name = name
                self.__age = age

        def getAge(self):
                return self.__age

        def getName(self):
                return self.__name

        def setAge(self, age):
                self.__age=age

        def setName(self, name):
                self.__name=name

obj=Student("Hong", 20)
obj.getName()
```

객체를 함수로 전달할 때

우리가 작성한 객체가 전달되면 함수가 객체를 변경할 수 있다.

```
# 사각형을 클래스로 정의한다.
class Rectangle:
        def __init__(self, side=0):
                self.side = side

        def getArea(self):
                return self.side*self.side

# 사각형 객체와 반복횟수를 받아서 변을 증가시키면서 면적을 출력한다.
def printAreas(r, n):
        while n >= 1:
                print(r.side, "\t", r.getArea())
                r.side = r.side + 1
                n = n - 1
```

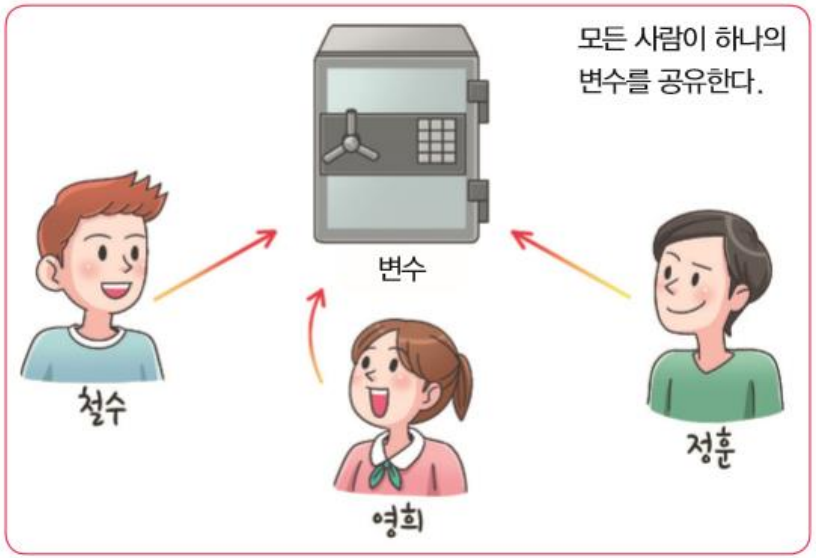
객체를 함수로 전달할 때

```
# printAreas()을 호출하여서 객체의 내용이 변경되는지를 확인한다.
myRect = Rectangle();
count = 5
printAreas(myRect, count)
print("사각형의 변=", myRect.side)
print("반복횟수=", count)
```

```
0         0
1         1
2         4
3         9
4         16
사각형의 변= 5
반복횟수= 5
```

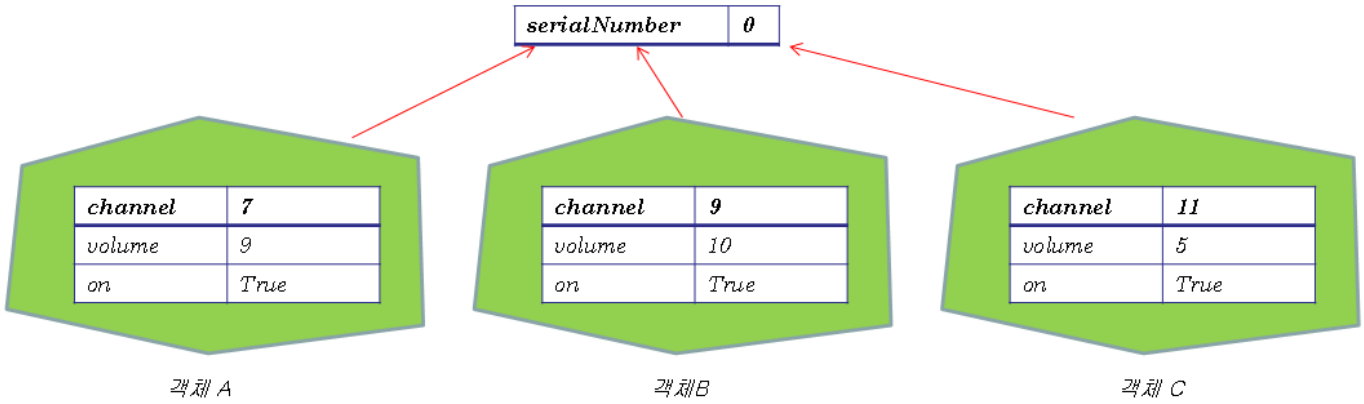
정적 변수

- 이들 변수는 모든 객체를 통틀어서 하나만 생성되고 모든 객체가 이것을 공유하게 된다. 이러한 변수를 정적 멤버 또는 클래스 멤버(class member)라고 한다.



정적 변수

```
class Television:
        serialNumber = 0          # 이것이 정적 변수이다.
        def __init__(self):
                Television.serialNumber += 1
        self.number = Television.serialNumber
                ...
```



특수 메소드

파이썬에는 연산자(+, -, *, /)에 관련된 특수 메소드(special method)가 있다.

```
class Circle:
        ...
        def __eq__(self, other):
                return self.radius == other.radius

c1 = Circle(10)
c2 = Circle(10)
if c1 == c2:
        print("원의 반지름은 동일합니다. ")
```

특수 메소드

연산자	메소드	설명
x + y	__add__(self, y)	덧셈
x - y	__sub__(self, y)	뺄셈
x * y	__mul__(self, y)	곱셈
x / y	__truediv__(self, y)	실수나눗셈
x // y	__floordiv__(self, y)	정수나눗셈
x % y	__mod__(self, y)	나머지
divmod(x, y)	__divmod__(self, y)	실수나눗셈과 나머지
x ** y	__pow__(self, y)	지수
x << y	__lshift__(self, y)	왼쪽 비트 이동
x >> y	__rshift__(self, y)	오른쪽 비트 이동
x <= y	__le__(self, y)	less than or equal(작거나 같다)
x < y	__lt__(self, y)	less than(작다)
x >= y	__ge__(self, y)	greater than or equal(크거나 같다)
x > y	__gt__(self, y)	greater than(크다)
x == y	__eq__(self, y)	같다
x != y	__neq__(self, y)	같지않다

파이썬에서의 변수의 종류

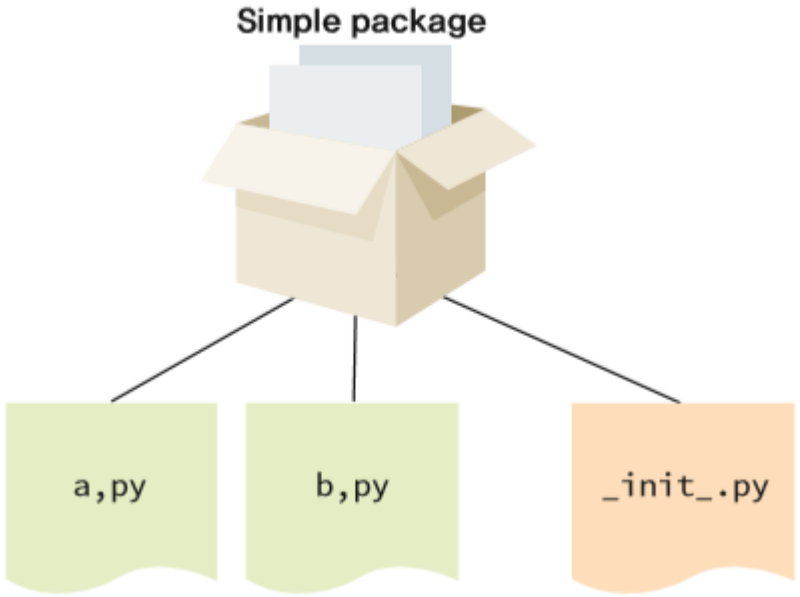
지역 변수 – 함수 안에서 선언되는 변수

전역 변수 – 함수 외부에서 선언되는 변수

인스턴스 변수 – 클래스 안에 선언된 변수, 앞에 self.가 붙는다.

모듈이란?

함수나 변수들을 모아 놓은 파일을 모듈(module)



모듈 작성

fibo.py

```
# 피보나치 수열 모듈

def fib(n):    # 피보나치 수열을 화면에 출력한다.
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()
```

모듈 사용

```
>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> fibo.__name__
'fibo’
```

```
>>> from fibo import *
>>> fib(500)
1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

모듈을 스크립트로 실행하기

만약 파이썬 모듈을 다음과 같이 명령어 프롬프트를 이용하여 실행한다면

```
C> python fibo.py <arguments>
```

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

예

```
C> python fibo.py 50
1 1 2 3 5 8 13 21 34
```

```
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

함수를 사용한 프로그램 설계

1. **문제를 한 번에 해결하려고 하지 말고 더 작은 크기의 문제들로 분해한다. 문제가 충분히 작아질 때까지 계속해서 분해한다.**
2. **문제가 충분히 작아졌으면 각각의 문제를 함수로 작성한다.**
3. **이들 함수들을 조립하면 최종 프로그램이 완성된다.**

예제

```
def readList():
        nlist = []
        flag = True;
        while flag :
                number = int(input("숫자를 입력하시오: "))
                if number < 0:
                        flag = False
                else :
                        nlist.append(number)
        return nlist

def processList(nlist):
        nlist.sort()
        return nlist


def printList(nlist):
        for i in nlist:
                print("성적=", i)
```

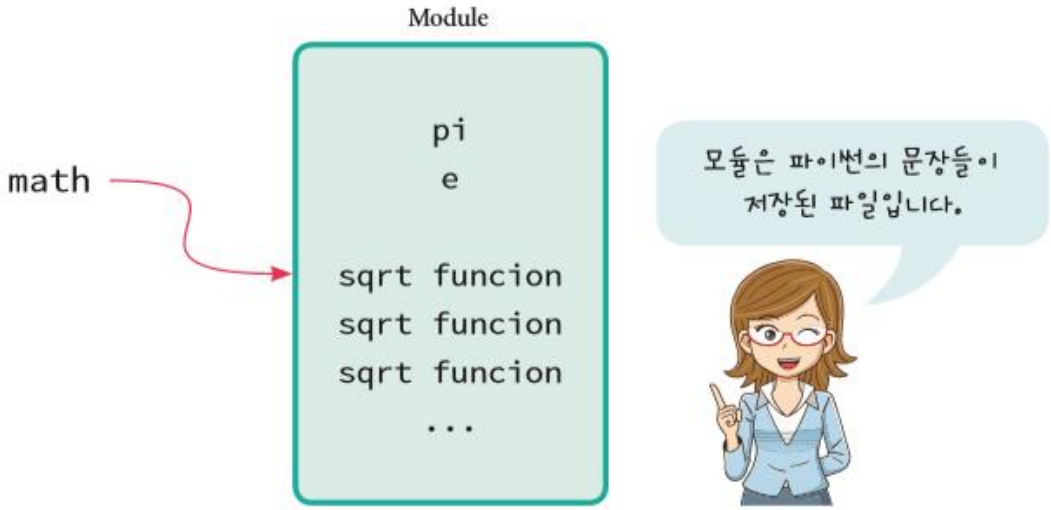
예제

```
def main():
        nlist = readList()
        processList(nlist)
        printList(nlist)

if __name__ == "__main__":
        main()
```

```
숫자를 입력하시오: 30
숫자를 입력하시오: 50
숫자를 입력하시오: 10
숫자를 입력하시오: 90
숫자를 입력하시오: 60
숫자를 입력하시오: -1
성적= 10
성적= 30
성적= 50
성적= 60
성적= 90
```

모듈

파이썬에서 모듈(module)이란 함수나 변수 또는 클래스 들을 모아 놓은 파일이다.



모듈 작성하기

fibo.py

```
# 피보나치 수열 모듈

def fib(n):             # 피보나치 수열 출력
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

def fib2(n):            # 피보나치 수열을 리스트로 반환
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)
        a, b = b, a+b
    return result
```

모듈 사용하기

```
>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> fibo.fib2(100)
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

>>> fibo.__name__
'fibo'
```

모듈 실행하기

```
C> python fibo.py <arguments>
```

```
...
if __name__ == "__main__":
    import sys
    fib(int(sys.argv[1]))
```

```
C> python fibo.py 50
1 1 2 3 5 8 13 21 34
```

모듈 탐색 경로

① 입력 스크립트가 있는 디렉토리(파일이 지정되지 않으면 현재 디렉토리)

② PYTHONPATH 환경 변수

③ 설치에 의존하는 디폴트값

유용한 모듈

프로그래밍에서 중요한 하나의 원칙은 이전에 개발된 코드를 적극적으로 재활용 하자는 것

COPY 모듈

```
import copy
colors = ["red", "blue", "green"]
clone = copy.deepcopy(colors)

clone[0] = "white"
print(colors)
print(clone)
```

```
['red', 'blue', 'green']
['white', 'blue', 'green']
```

KEYWORD 모듈

```
import keyword

name = input("변수 이름을 입력하시오: ")

if keyword.iskeyword(name):
    print (name, "은 예약어임.")
    print ("아래는 키워드의 전체 리스트임: ")
    print (keyword.kwlist)
else:
    print (name, "은 사용할 수 있는 변수이름임.")
```

변수 이름을 입력하시오: for
for 은 예약어임.
아래는 키워드의 전체 리스트임:
['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']

RANDOM 모듈

```
>>> import random
>>> print(random.randint(1, 6))
6
>>> print(random.randint(1, 6))
3

>>> import random
>>> print(random.random()*100)
81.1618515880431

>>> myList = [ "red", "green", "blue" ]
>>> random.choice(myList)
'blue'
```

RANDOM 모듈

```
>>> myList = [ [x] for x in range(10) ]
>>> random.shuffle(myList)
>>> myList
[[3], [2], [7], [9], [8], [1], [4], [6], [0], [5]]

>>> for i in range(3):
        print(random.randrange(0, 101, 3))

81
21
57
```

os 모듈

```
>>> import os
>>> os.system("calc")
```

```
>>> os.getcwd()
'D:\\'
>>> os.chdir("D:\\tmp")
>>> os.getcwd()
'D:\\tmp’

>>> os.listdir(".")
[ 'chap01', 'chap02', 'chap03', 'chap04', 'chap05', 'chap06', 'chap07',
'chap08', 'chap09', 'chap10', 'chap11', 'chap12', 'chap13', 'chap14', 'chap15',
'chap16', 'chap17', 'chap18', 'chap19', 'chap20' ]
```

SYS 모듈

```
>>> import sys
>>> sys.prefix # 파이썬이 설치된 경로
'C:\\Users\\chun\\AppData\\Local\\Programs\\Python\\Python35-32’

>>> sys.executable
'C:\\Users\\chun\\AppData\\Local\\Programs\\Python\\Python35-
32\\pythonw.exe’
```

TIME 모듈

```
>>> import time
>>> time.time()
1461203464.6591916
```

예제

```
import time
def fib(n):    # 피보나치 수열 출력
    a, b = 0, 1
    while b < n:
        print(b, end=' ')
        a, b = b, a+b
    print()

start = time.time()
fib(1000)
end = time.time()
print(end-start)
```

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
0.0350019931793212 9

CALENDAR 모듈

```
import calendar

cal = calendar.month(2016, 8)
print(cal)
```

```
    August 2016
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```