

Programming

프로그래밍 언어의 개념과 종류

■ 프로그래밍 언어의 개념

- 프로그래밍 언어 : 컴퓨터가 이해하는 말로 컴퓨터에서 작동하는 소프트웨어(엑셀, 한글, 인터넷 익스플로러 등)를 만드는 도구
- 프로그래머 : 프로그래밍 언어를 사용해 소프트웨어나 앱을 만드는 사람



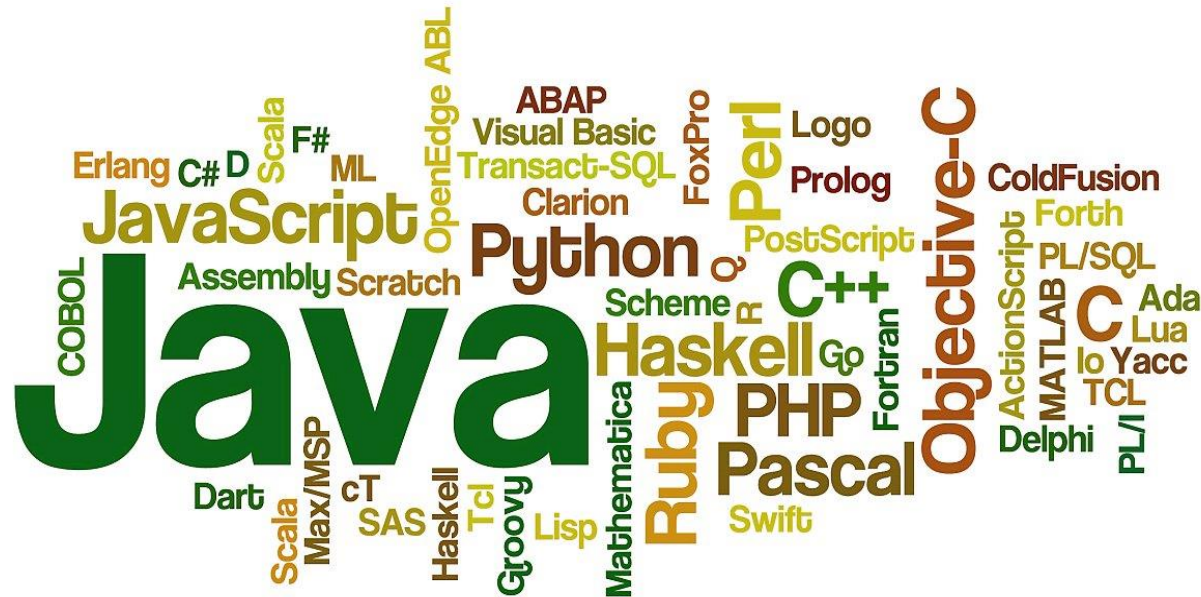
그림 1-1 프로그래머, 프로그래밍 언어, 소프트웨어

Programming Language

Programming Language

■ 프로그래밍 언어

▶ 컴퓨터가 이해할 수 있는 언어 (명령 전달을 위함)

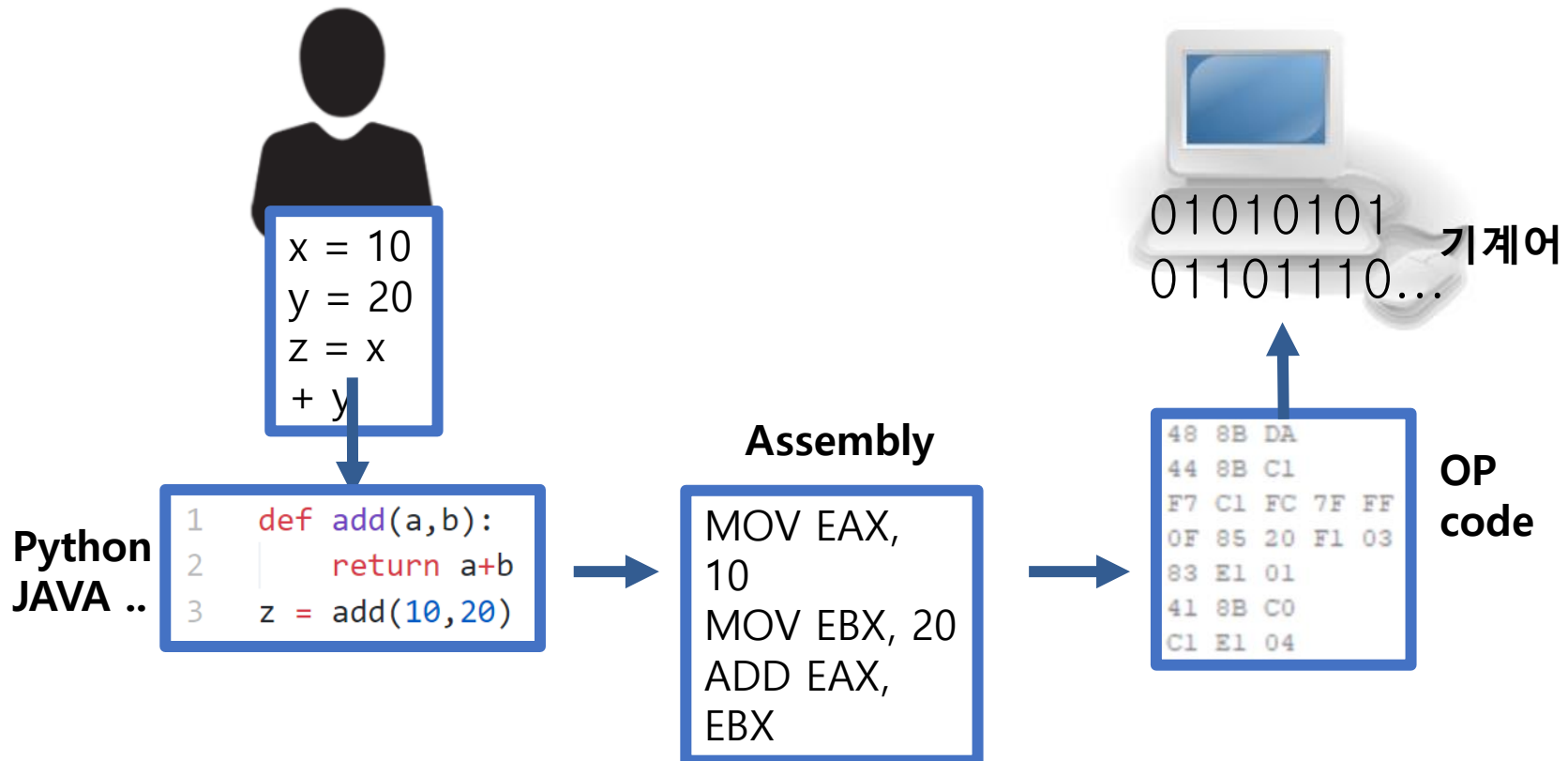


컴퓨터 프로그램을 만들기 위해서는 프로그래밍 언어를 배워야한다!

Programming Language

- 고급언어, 저급언어

.. ➡ **사람에게 친숙한 언어**일수록 고급언어



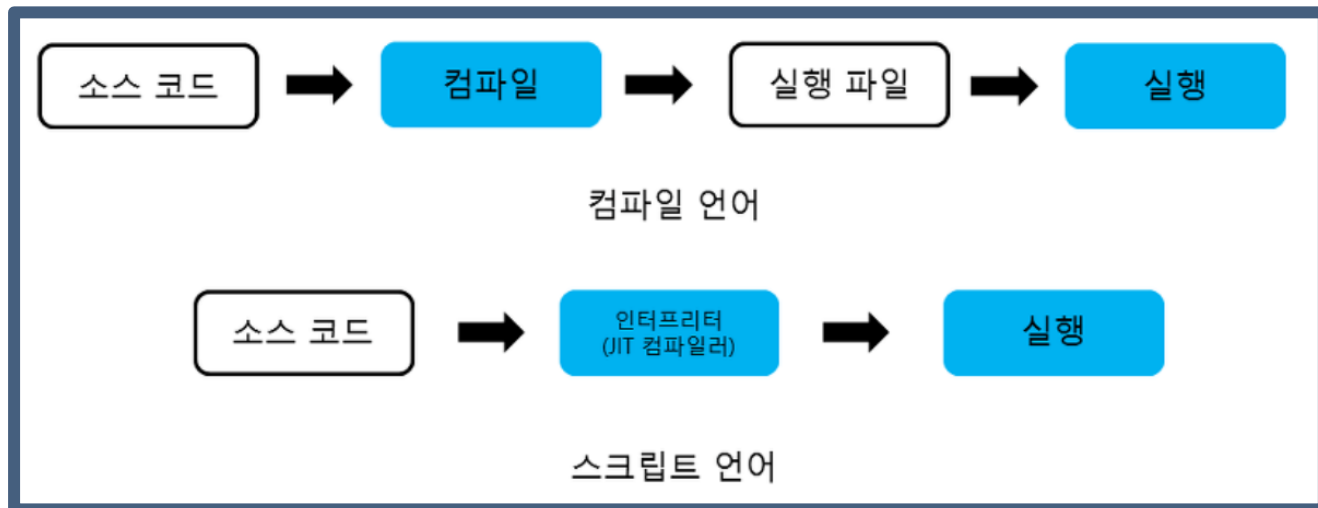
Programming Language

□ Compile 방식 C, C++

... ➡ 실행시간 이전에 코드 전체를 기계언어로 변환 후 실행

□ Interpret 방식 Python

... ➡ 실행시간에 줄 단위로 번역해서 실행함



Python

파이썬 특징

① 강력한 기능을 무료로 사용할 수 있다

- 파이썬은 오픈 소스이며, 비용을 지불하지 않고 무료로 사용 가능. 다양한 추가 라이브러리도 무료

② 읽기 쉽고 사용하기 쉽다

- 직관적인 코드를 사용해 C나 자바 같은 언어보다 읽기 쉬워 프로그램을 빨리 제작할 수 있어 비용 절감 효과 제공

③ 사물인터넷과 잘 연동된다

- 라즈베리파이 기반의 사물인터넷이 파이썬을 잘 지원하므로 사물인터넷 개발 및 운영에 적극 활용

④ 다양하고 강력한 외부 라이브러리들이 풍부하다

- 파이썬에서 제공하는 라이브러리뿐 아니라, 외부에서 제공하는 다양한 서드 파티(Third Party) 라이브러리까지 사용 가능

⑤ 강력한 웹 프레임워크를 사용할 수 있다

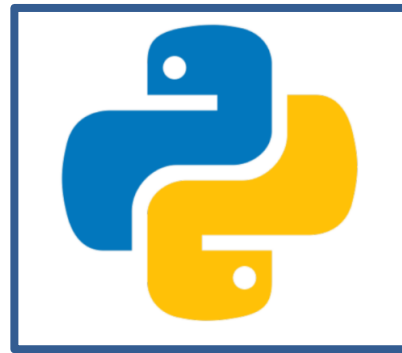
- 파이썬의 웹 프레임워크를 사용해 강력하고 빠른 웹 환경을 구축 가능

파이썬의 단점

- 느린 속도
 - 파이썬은 컴파일러 언어가 아닌 스크립트 언어이기 때문에 컴파일러 언어보다 느림
→ 이를 보완하려고 많은 파이썬 패키지를 최적화시키고 있음
- 모바일 컴퓨팅 분야에 지원이 약하고 하드웨어 제어 등과 관련된 부분 사용이 어려움

Python

- Python
 - 귀도 반 로섬이 1991년 발표한 고급 프로그래밍 언어
 - 영국 코미디프로그램 이름에서 유래



Python

□ 배우기 쉽다.

.. ➡ 문법이 간결하고, 이해하기 쉬움(직관적임)

C

```
1  #include<stdio.h>
2
3  int main(){
4      printf("Hello world");
5  }
```

JAVA

```
1  public class HelloWorld{
2      public static void main(String args[]){
3          System.out.println("Hello World");
4      }
5  }
```

Python

```
1  print("Hello world")
```

"Life is short (You need Python)"^[1]
인생은 짧으니, 당신은 파이썬이 필요하다.

- Bruce Eckel

Python

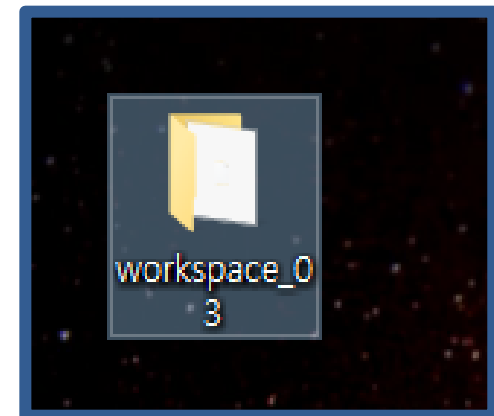
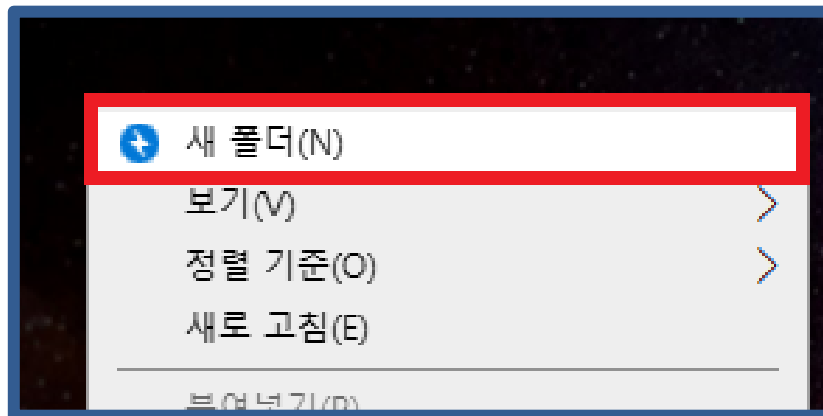
□ R vs Python



환경설치

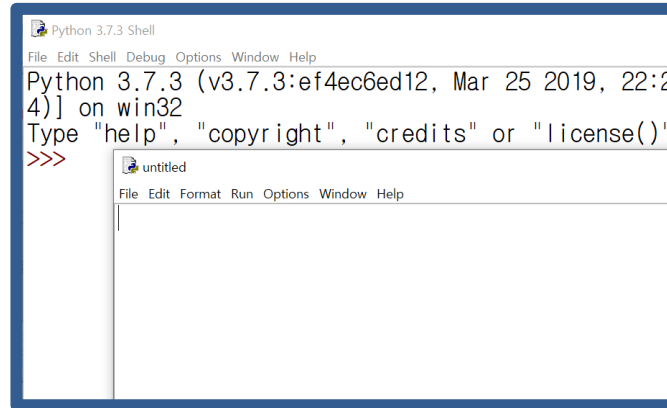
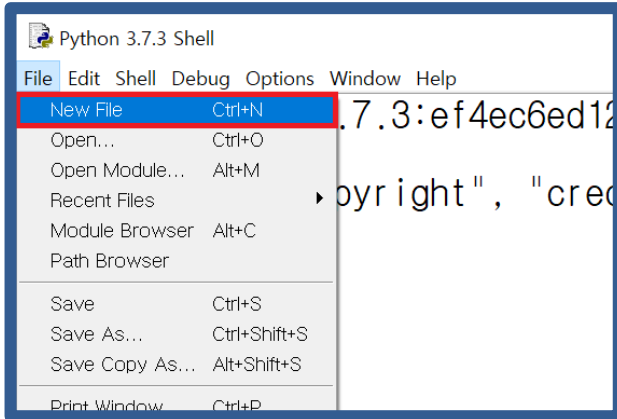
Python 설치

- Python 설치
 - python.org
- Workspace 폴더 생성
 - Sung

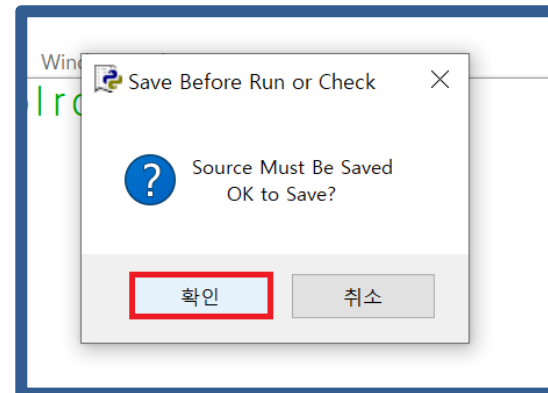
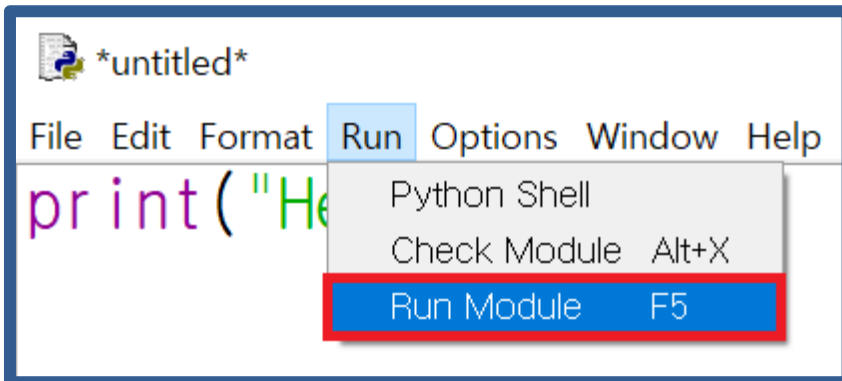


Python 설치

- Python 파일 생성
- [File] - [New File]



- Python 파일 실행
- [Run] - [Run Module]



Python 저장

■ 파이썬 파일 저장

- 스크립트 모드에서 [File]-[Save] 메뉴를 선택해 DWPsungW 폴더에 01-02 이름으로 저장(확장명 *.py가 자동으로 붙음)

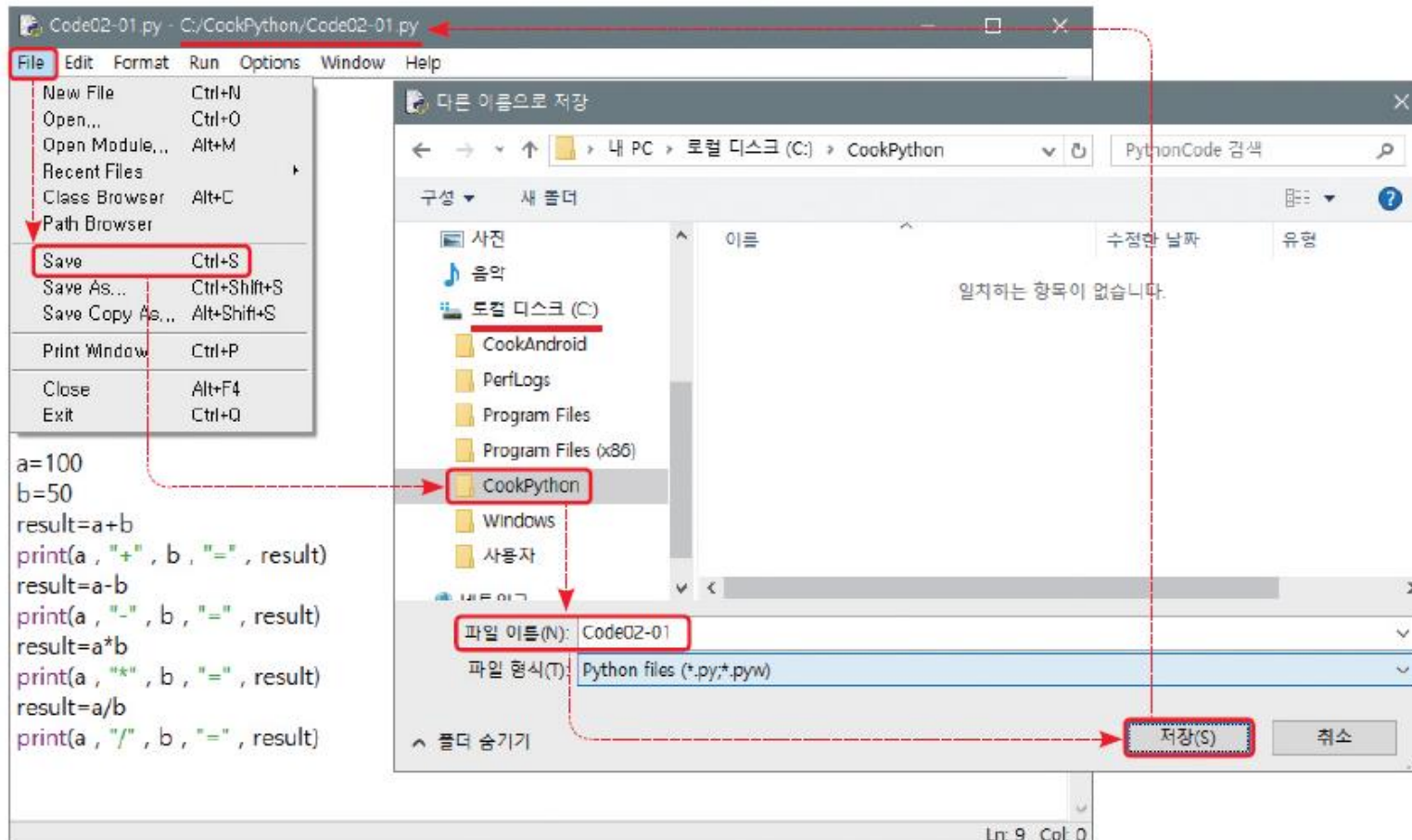


그림 2-12 파이썬 코드를 파일로 저장

Python 불러오기

□ Python 파일 실행 결과

```
===== RESTART: C:/Users/H4cker/Desktop/workspace_03/hello.py =====  
Hello wolrd!  
>>>
```

.. ➡ 파일 생성, 불러오기, 저장, 실행 과정이 번거로움

Python 확장자

□ 파일 확장자

- ➡ 해당 파일의 종류 및 역할에 따라 확장자가 정해져있음

[파일이름] . [확장자]

Python Shell

□ Python Shell

...➡ 대화형 모드라고 부름

...➡ 간단한 정보 확인 시 사용

Python 자료형

Python 에서 사용되는 자료의 형태

- 숫자
- 문자열
- Bool

주석

□ 한줄주석

... ➡ # 뒤의 문자들은 주석처리됨.

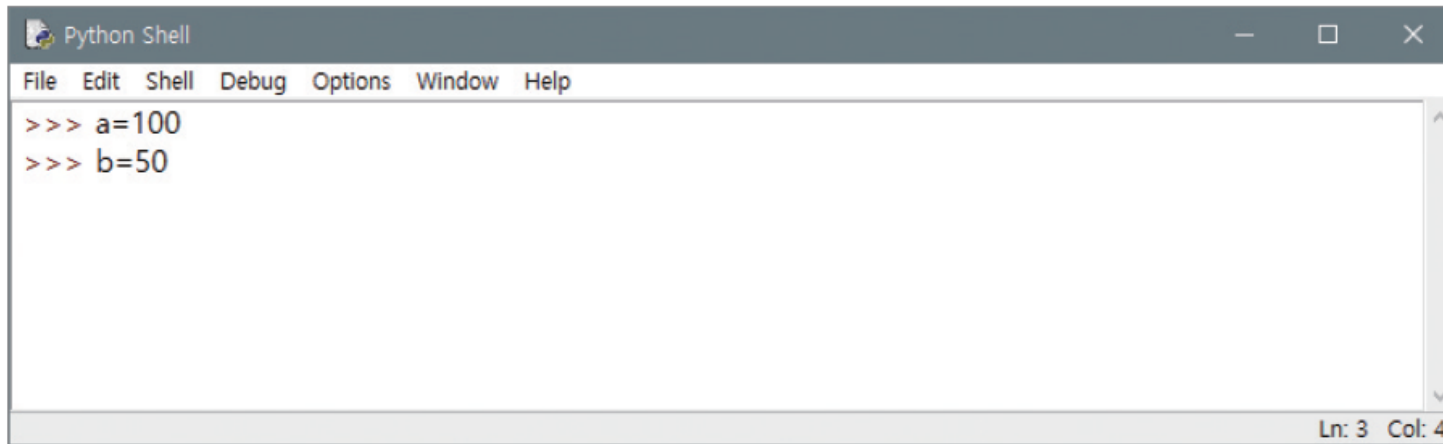
□ 여러줄 주석

... ➡ ''' ''', """ """ 사이의 문자들은 주석처리됨.

```
1  print('Hello world!', end=' ') # end 의 default 값은 \n
2  '''
3  | 하지만 end 값을 직접 설정해주면
4  | 개행을 하지 않습니다!
5  | '''
6  | """
7  | 주석은 코드가 실행되는데에
8  | 영향을 주지 않는다.
9  | print('a')
10 | """
```

필요한 변수 준비1

- = 기호는 같다는 의미가 아니라 '오른쪽의 것을 왼쪽으로 넣어라'는 의미의 대입 연산자
 - $a=100$ 은 $a \leftarrow 100$ 과 같은 개념
 - 내부적으로는 a 와 b 그릇이 생겨 a 그릇에는 100이, b 그릇에는 50이 담긴 상태
- 프로그래밍 언어에서 그릇과 같은 역할을 하는 것이 바로 변수



A screenshot of a Python Shell window. The title bar says 'Python Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows two lines of code: `>>> a=100` and `>>> b=50`. The status bar at the bottom right indicates 'Ln: 3 Col: 4'.

그림 2-1 변수 준비



그릇 이름 : a



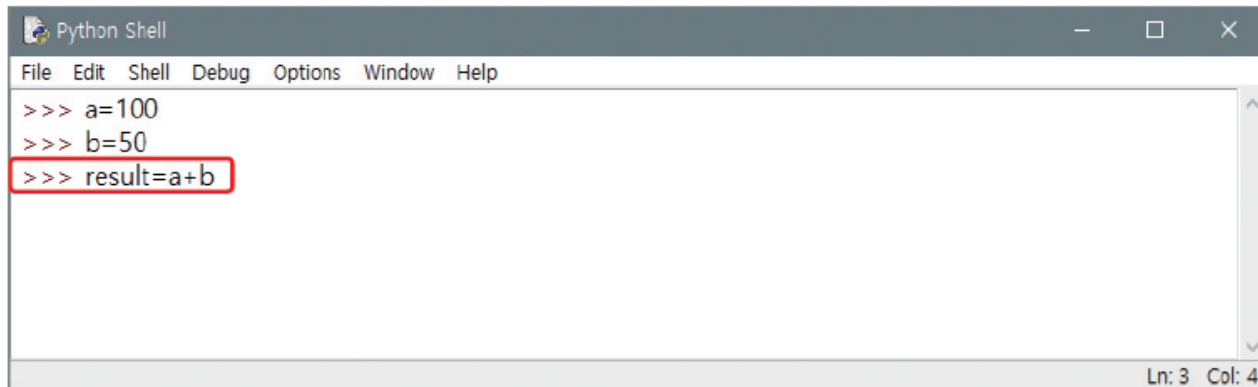
그릇 이름 : b

그림 2-2 그릇을 2개 준비

필요한 변수 준비2

■ 더하기 기능 구현

- a 그릇의 100과 b 그릇의 50을 합쳐 새로운 result 그릇에 들어간 상태가 됨
- 변수는 result에 값이 들어가더라도 a, b의 변수값이 그대로 남음



A screenshot of a Python Shell window. The title bar says 'Python Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains three lines of code: `>>> a=100`, `>>> b=50`, and `>>> result=a+b`. The third line is highlighted with a red rectangular box. The status bar at the bottom right shows 'Ln: 3 Col: 4'.

그림 2-3 더하기 구현



그림 2-4 더하는 작업

Python Shell

■ 파이썬 파일 저장

- 코드가 수십 줄인 경우는 **스크립트 모드** 사용(IDLE에서 [File]-[New File] 메뉴 선택)
- 메모장 같은 창인 스크립트 모드에서 코드를 여러 줄 입력 가능. 단, 실행은 되지 않음

■ 콘솔창

- 한줄에 **한문장**이 기본
- 이미 끝난 문장은 수정이 안됨
- 같은창에서 결과를 즉시볼수 있음

■ 스크립트창

- 한줄에 **여러문장**이 가능
- 이미 끝난 문장도 수정이 됨
- 스트립트 창에서 작성한 결과를 콘솔창에서 볼수 있음

Python 스크립트

■ 긴 프로그램을 코딩하는 순서

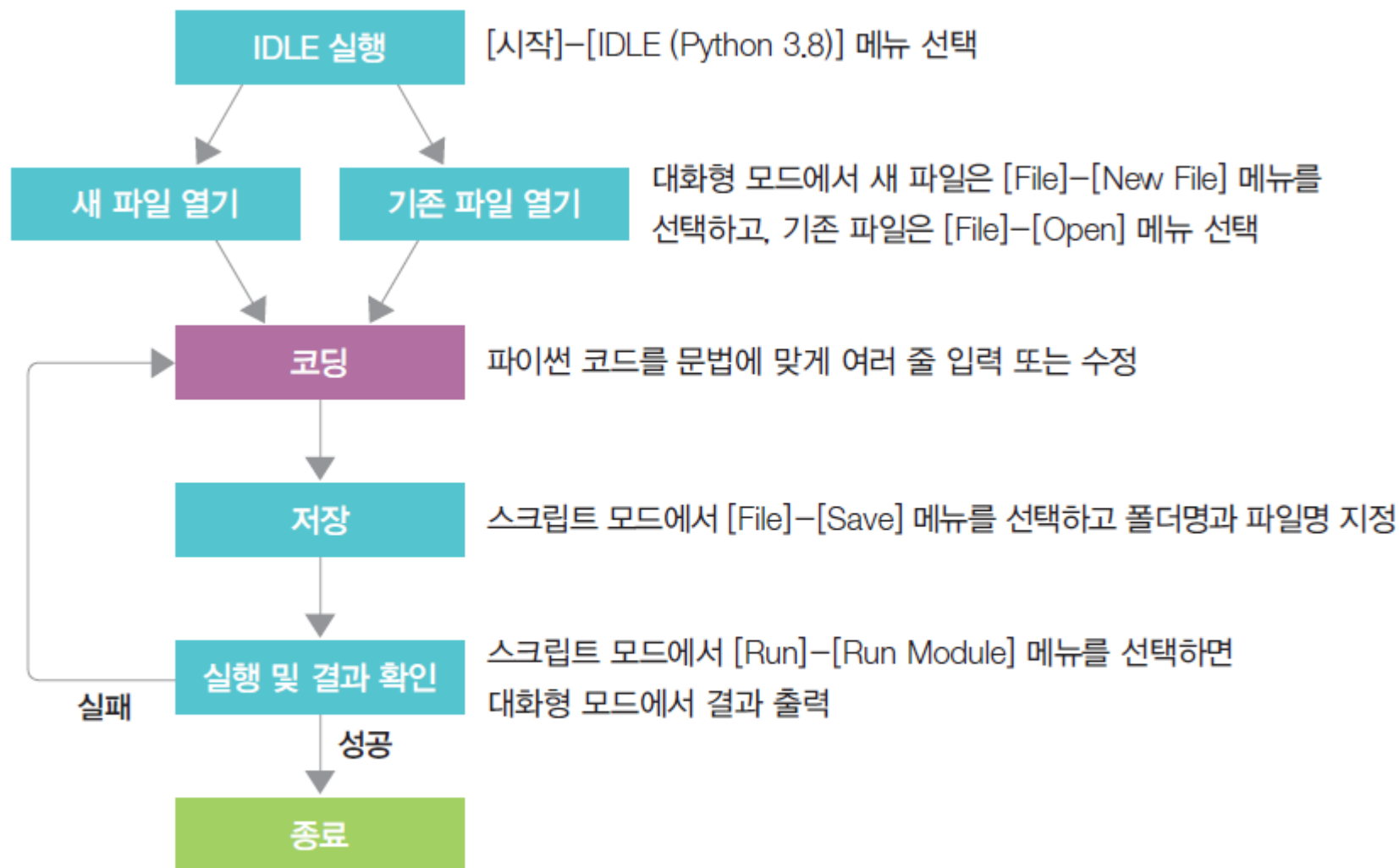


그림 2-18 긴 프로그램을 코딩하는 순서

Section04 데이터 표현 단위와 진수 변환

■ 비트

표 3-3 10진수, 2진수, 16진수 변환표

10진수(0~9)	2진수(0~1)	16진수(0~F)
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Section04 데이터 표현 단위와 진수 변환

■ 바이트

표 3-4 비트와 바이트 크기에 따른 숫자의 범위

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
1		$2^2=2$	0~1	0~1	0~1
2		$2^4=4$	0~11	0~3	0~3
4		$2^4=16$	0~1111	0~15	0~F
8	1	$2^8=256$	0~11111111	0~255	0~FF
16	2	$2^{16}=65536$	0~11111111 11111111	0~65535	0~FFFF
32	4	$2^{32}=\text{약 } 42\text{억}$	0~...	0~약 42억	0~FFFF FFFF
64	8	$2^{64}=\text{약 } 1800\text{경}$	0~... ..	0~약 1800경	0~... ..

Section04 데이터 표현 단위와 진수 변환

■ 비트

표 3-3 10진수, 2진수, 16진수 변환표

10진수(0~9)	2진수(0~1)	16진수(0~F)
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Section04 데이터 표현 단위와 진수 변환

■ 바이트

표 3-4 비트와 바이트 크기에 따른 숫자의 범위

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
1		$2^2=2$	0~1	0~1	0~1
2		$2^4=4$	0~11	0~3	0~3
4		$2^4=16$	0~1111	0~15	0~F
8	1	$2^8=256$	0~11111111	0~255	0~FF
16	2	$2^{16}=65536$	0~11111111 11111111	0~65535	0~FFFF
32	4	$2^{32}=\text{약 } 42\text{억}$	0~...	0~약 42억	0~FFFF FFFF
64	8	$2^{64}=\text{약 } 1800\text{경}$	0~... ..	0~약 1800경	0~... ..

조건문

기본 if 문

- 조건문(if 문)

- ~~ 라면, ~~가 아니라면

- if 조건식:

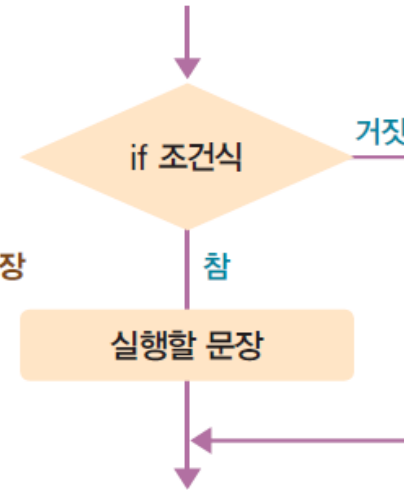
실행문장

```
a = 99
if a < 100 :
    print("100보다 작군요.")
```

출력 결과

100보다 작군요.

if 조건식 :
실행할 문장



- 문자열 자료형의 경우 "" 은 False, "" 이 아니면 True

- 비교연산자 , 논리연산자 > **Bool 값을 반환함**

- if **Bool** :

- if **관계연산** :

- if **논리연산** :

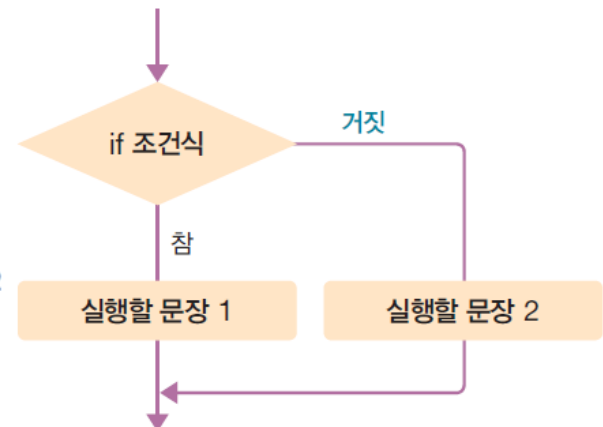
- if~else 문

```
1 a = 200
2
3 if a < 100 :
4     print("100보다 작군요.")
5 else :
6     print("100보다 크군요.")
```

출력 결과

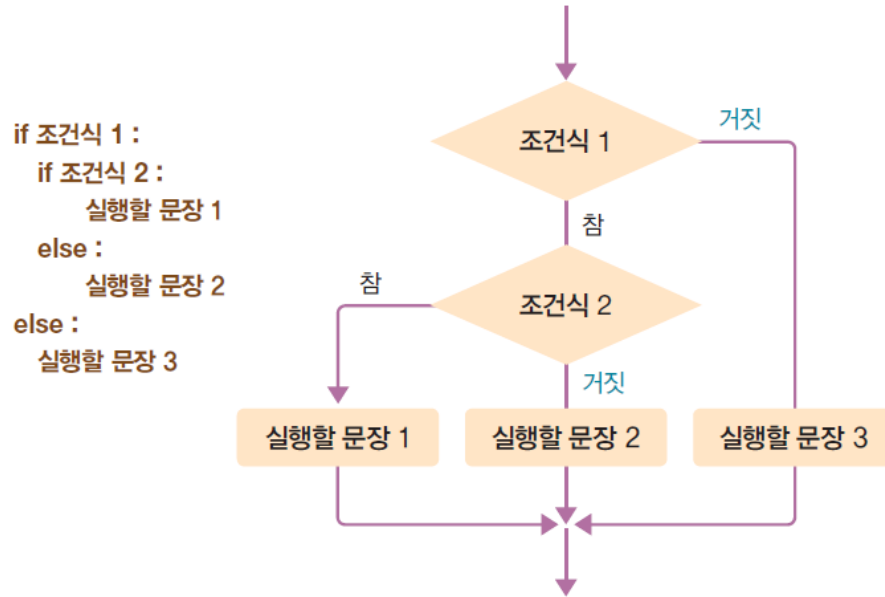
100보다 크군요.

if 조건식 :
실행할 문장 1
else :
실행할 문장 2



중첩 if 문

- if~else~if~else 문
 - if 문을 한 번 실행한 후 그 결과에서 if 문을 다시 실행하는 것



반복문

기본 for 문

- 반복문(for 문)

- for 변수 in range(시작값, 끝값+1, 증가값):
반복실행문장

```
1 for i in range(0, 3, 1) :  
2     print("안녕하세요? for 문을 공부 중입니다. ^^")
```

출력 결과

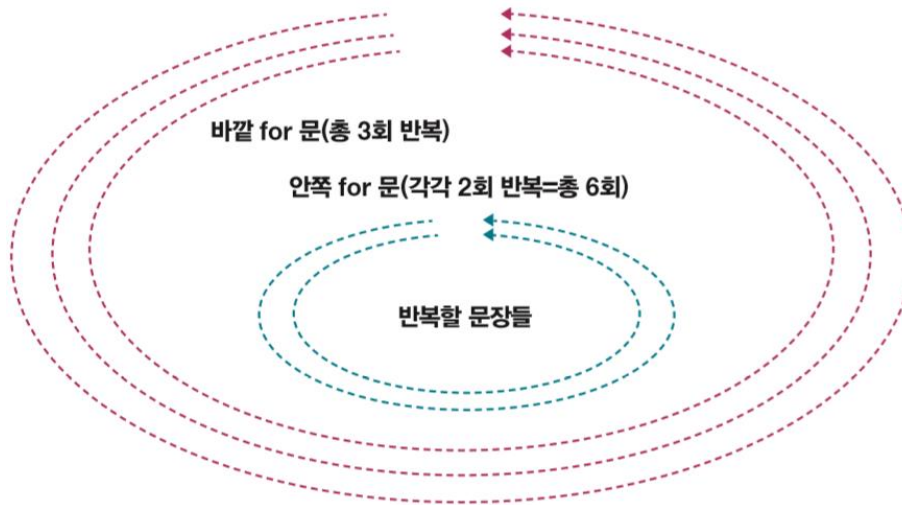
```
안녕하세요? for 문을 공부 중입니다. ^^  
안녕하세요? for 문을 공부 중입니다. ^^  
안녕하세요? for 문을 공부 중입니다. ^^
```

- for 문의 기본 형식

- range() 함수
 - 리스트(튜플) 형식

중첩 for 문

- 중첩 for 문
 - 반복문안에 반복문 (종속문장에 반복문이 존재)
- 처리 순서
 - 외부 변수인 i는 계속 0, 1, 2로 변경된 후 끝나지만, 내부 변수인 k는 0과 1을 계속 반복



```
for i in range(0, 3, 1):  
    for k in range(0, 2, 1):  
        print("파이썬 (i값 : %d k값: %d)" % (i,k))
```

실습

```
for i in range(0, 3, 1):  
    for k in range(0, 2, 1):  
        print("파이썬 (i값 : %d k값: %d)" % (i,k))  
#외부 변수인 i는 계속 0, 1, 2로 변경된 후 끝나지만, 내부 변수인 k는 0과 1을 계속 반복
```

while 문

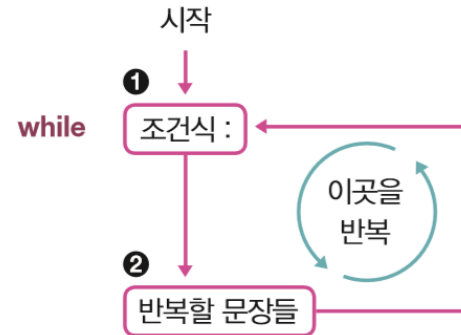
- for 문과 while 문 비교

- for 문의 형식

```
for 변수 in range(시작값, 끝값+1, 증가값):  
    반복실행문장
```

- while 문의 형식

```
변수 = 시작값  
while 변수 < 끝값:  
    반복실행문장  
    변수 = 변수 + 증가값
```



- 무한 루프를 하는 while 문
- 무한 루프 적용 : 'while 조건식 :'에 들어가는 조건식을 True로 지정
- 실행횟수가 정해져 있으면 for
- 실행횟수를 모를 경우 while

break 문 / continue 문

break 문 / continue 문

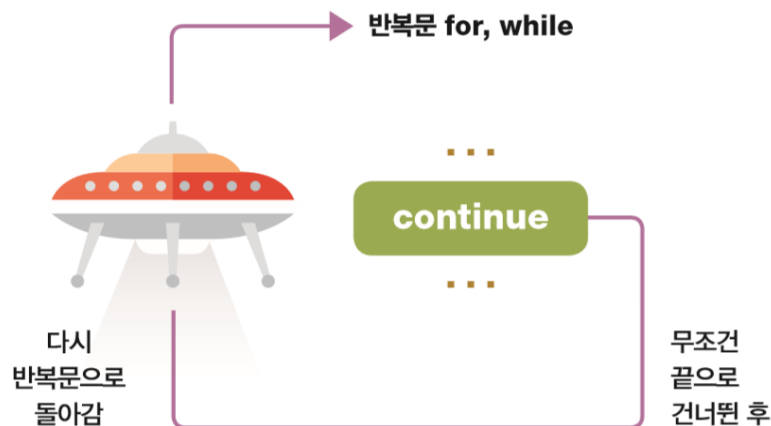
- 반복문을 탈출시키는 break 문
 - 계속되는 반복을 논리적으로 빠져나가는 방법

```
for i in range(1, 100) :  
    print("for 문을 %d번 실행했습니다." % i)  
    break
```

출력 결과

for 문을 1번 실행했습니다.

- 반복문으로 다시 돌아가게 하는 continue 문



random

random

randint(A, B)

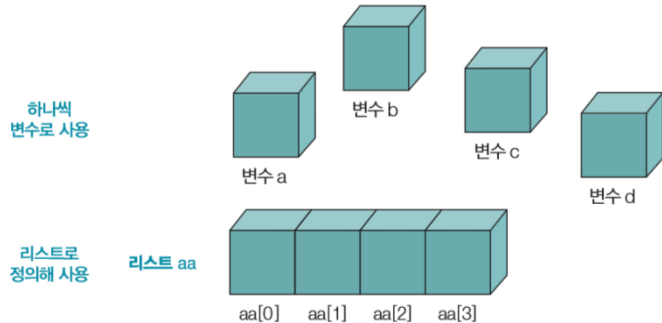
- A ~ B 까지 정수를 랜덤으로 반환 ($A < B$)
- 특정 이벤트를 발생시킬 때도 사용

리스트 / 튜플 / 딕셔너리

Section02 리스트의 기본

■ 리스트의 개념

- 여러 데이터를 하나의 []에 넣어서 하나의 변수 이름으로 사용하는 것



■ 리스트를 생성하는 방법

① 각 변수 사용

a, b, c, d = 10, 20, 30, 40

a 사용

b 사용

c 사용

d 사용

② 리스트 사용

aa = [10, 20, 30, 40]

aa[0] 사용

aa[1] 사용

aa[2] 사용

aa[3] 사용

■ 튜플의 생성

- 리스트는 대괄호 []로 생성, 튜플은 소괄호 ()로 생성
- 튜플은 값을 수정할 수 없으며, 읽기만 가능해 읽기 전용 자료를 저장할 때 사용
- 튜플은 소괄호 ()를 생략 가능, 항목이 하나인 튜플은 뒤에 쉼표(,) 붙임

■ 딕셔너리의 개념

- 쌍 2개가 하나로 묶인 자료구조
 - 예 : 'apple:사과'처럼 의미 있는 두 값을 연결해 구성

Tip • 다른 프로그래밍 언어에서는 해시(Hash), 연관 배열(Associative Array)이라 함

- 중괄호 {}로 묶어 구성, 키(Key)와 값(Value)의 쌍으로 구성

딕셔너리변수 = {키1:값1, 키2:값2, 키3:값3, ...}

Section02 리스트의 기본

- 리스트를 생성하는 방법

```
리스트명 = [값1, 값2, 값3, ...]
```

```
aa = [10, 20, 30, 40]
```

❶ 각 변수 사용

a, b, c, d = 10, 20, 30, 40

a 사용

b 사용

c 사용

d 사용

❷ 리스트 사용

aa = [10, 20, 30, 40]

aa[0] 사용

aa[1] 사용

aa[2] 사용

aa[3] 사용

Section02 리스트의 기본

■ 리스트의 생성과 초기화

■ 리스트 생성 코드

❶ aa = []
❷ bb = [10, 20, 30]
❸ cc = ['파이썬', '공부하는', '꿀잼']
❹ dd = [10, 20, '파이썬']

❶ 빈 리스트 생성
❷ 정수로만 구성된 리스트 생성
❸ 문자열로만 구성된 리스트 생성
❹ 다양한 데이터형을 섞어 리스트 생성

- 리스트 100개인 aa 0, 2, 4, 8, ...(2의 배수로 초기화 후 리스트 bb에 역순으로 넣는 과정

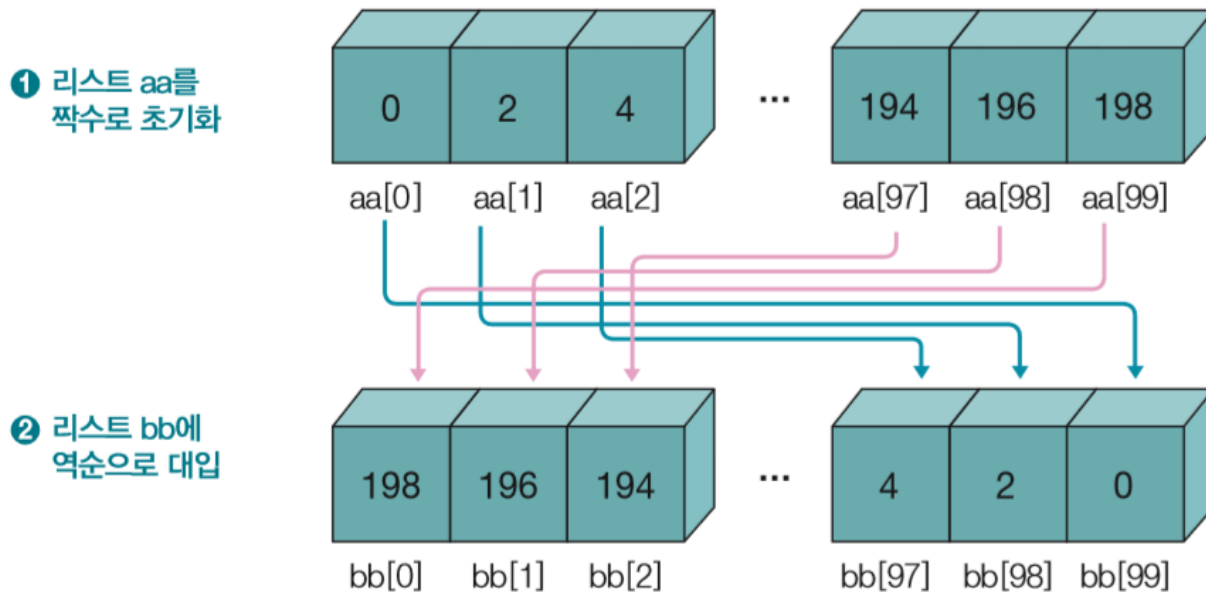


그림 7-3 리스트의 초기화 및 역순 대입

Section02 리스트의 기본

■ 리스트 조작 함수

표 7-1 리스트 조작 함수

함수	설명	사용법
append()	리스트 맨 뒤에 항목을 추가한다.	리스트명.append(값)
pop()	리스트 맨 뒤의 항목을 빼낸다(리스트에서 해당 항목이 삭제된다).	리스트명.pop()
sort()	리스트의 항목을 정렬한다.	리스트명.sort()
reverse()	리스트 항목의 순서를 역순으로 만든다.	리스트명.reverse()
index()	지정한 값을 찾아 해당 위치를 반환한다.	리스트명.index(찾을값)
insert()	지정된 위치에 값을 삽입한다.	리스트명.insert(위치, 값)
remove()	리스트에서 지정한 값을 삭제한다. 단 지정한 값이 여러 개면 첫 번째 값만 지운다.	리스트명.remove(지울값)
extend()	리스트 뒤에 리스트를 추가한다. 리스트의 더하기(+) 연산과 기능이 동일하다.	리스트명.extend(추가할리스트)
count()	리스트에서 해당 값의 개수를 센다.	리스트명.count(찾을값)
clear()	리스트의 내용을 모두 지운다.	리스트명.clear()
del()	리스트에서 해당 위치의 항목을 삭제한다.	del(리스트명[위치])
len()	리스트에 포함된 전체 항목의 개수를 센다.	len(리스트명)
copy()	리스트의 내용을 새로운 리스트에 복사한다.	새리스트=리스트명.copy()
sorted()	리스트의 항목을 정렬해서 새로운 리스트에 대입한다.	새리스트=sorted(리스트)

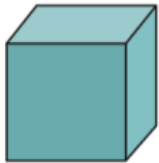
Section02 리스트의 기본

- 리스트의 첨자가 순서대로 변할 수 있도록 반복문과 함께 활용

for (4번 반복)

{

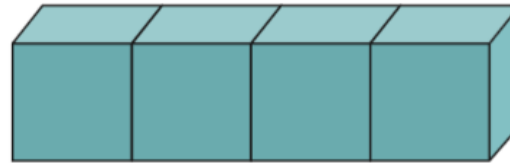
값 입력



aa[i]



aa[0]부터 aa[3]까지 4번 반복



aa[0]

aa[1]

aa[2]

aa[3]

}

i값이 0부터 3까지 변함

그림 7-2 for 문으로 리스트값 입력

Section03 2차원 리스트

■ 2차원 리스트의 개념

- 1차원 리스트를 여러 개 연결한 것, 첨자를 2개 사용

```
aa = [10, 20, 30]
```

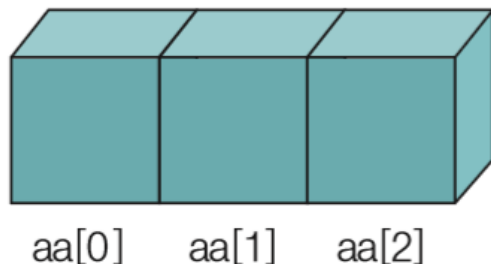
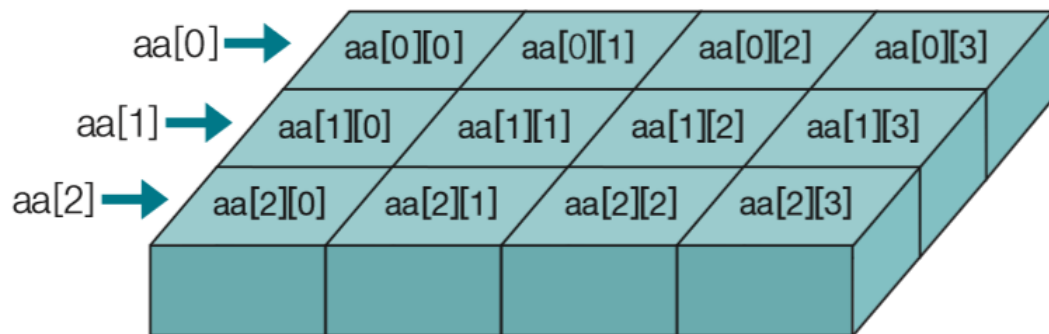


그림 7-4 1차원 리스트의 개념

```
aa = [[1, 2, 3, 4],  
      [5, 6, 7, 8],  
      [9, 10, 11, 12]]
```



전체 리스트명 : aa

그림 7-5 2차원 리스트의 개념

Section04 튜플

■ 튜플의 생성

- 리스트는 대괄호 []로 생성, 튜플은 소괄호 ()로 생성
- 튜플은 값을 수정할 수 없으며, 읽기만 가능해 읽기 전용 자료를 저장할 때 사용
- 튜플은 소괄호 ()를 생략 가능, 항목이 하나인 튜플은 뒤에 쉼표(,) 붙임

■ 딕셔너리의 개념

- 쌍 2개가 하나로 묶인 자료구조
 - 예 : 'apple:사과'처럼 의미 있는 두 값을 연결해 구성

Tip • 다른 프로그래밍 언어에서는 해시(Hash), 연관 배열(Associative Array)이라 함

- 중괄호 {}로 묶어 구성, 키(Key)와 값(Value)의 쌍으로 구성

문자열

Section02 문자열 기본

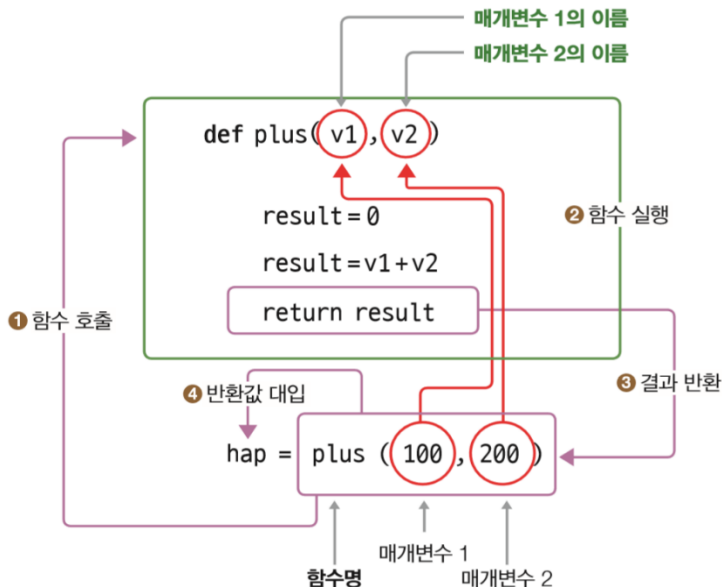
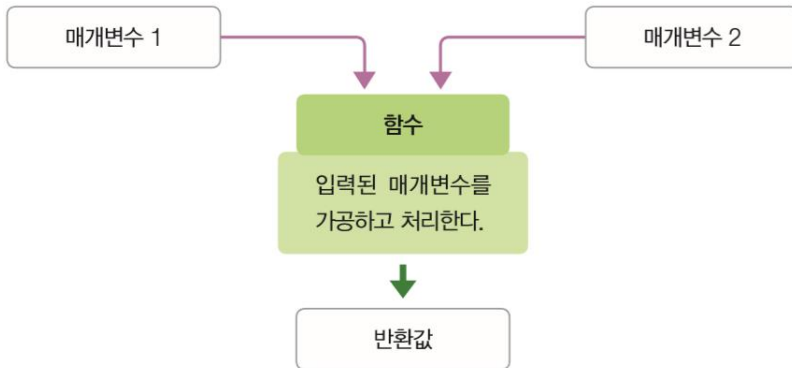
■ 문자열의 개념

- 리스트 코드와 비교 : 리스트는 대괄호 []로 묶고 문자열은 작은따옴표로 묶어 출력
- 더하기(+) 기호 사용해 연결. 또 곱하기(*) 기호 사용 문자열 반복
- len() 함수 : 리스트나 문자열의 개수를 셀 때 사용
- 대문자와 소문자 변환하기 : upper(), lower(), swapcase(), title()
- 문자열 찾기 : count(), find(), rfind(), index(), rindex(), startswith(), endswith()
- 문자열 공백 삭제·변경하기 : strip(),rstrip(), lstrip(), replace()
- 문자열 분리·결합하기 : split(), splitlines(), join()
- 함수명에 대입하기 : map()
- 문자열 정렬하기, 채우기 : center(), ljust(), rjust(), zfill()
- 문자열 구성 파악하기 : isdigit(), isalpha(), isalnum(), islower(), isupper(), isspace()

함수와 모듈

Section02 함수 기본

■ 함수의 형식과 활용



■ 지역 변수와 전역 변수의 이해

- 지역 변수 : 한정된 지역에서만 사용
- 전역 변수 : 프로그램 전체에서 사용

■ 함수의 반환값

함수의 반환값

Tip • 반환값은 return 문으로 반환되므로 리턴값이라고도 함. 매개변수는 파라미터라고도 함

반환값이 있는 함수

return 문을 사용

반환값이 없는 함수

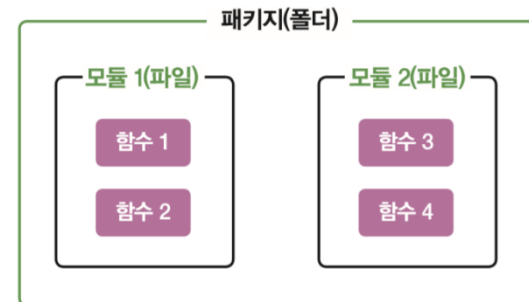
return 문이 없음

■ 모듈의 종류

- 표준 모듈, 사용자 정의 모듈, 서드 파티 모듈로 구분
- 표준 모듈 : 파이썬에서 제공하는 모듈
- 사용자 정의 모듈 : 직접 만들어서 사용하는 모듈
- 서드 파티(3rd Party) 모듈 : 파이썬이 아닌 외부 회사나 단체에서 제공하는 모듈
 - 파이썬 표준 모듈이 모든 기능을 제공 않음
 - 서드 파티 모듈 덕분에 파이썬에서도 고급 프로그래밍 가능
 - 게임 개발 기능이 있는 pyGame, 윈도우창을 제공 하는 PyGTK, 데이터베이스 기능의 SQLAlchemy 등

■ 패키지

- 모듈이 하나의 *.py 파일 안에 함수가 여러 개 들어 있는 것이라면, 패키지(Package)는 여러 모듈을 모아 놓은 것으로 폴더의 형태로 나타냄
- 모듈을 주제별로 분리할 때 주로 사용



객체지향프로그램

클래스/생성자/인스턴스변수/ 클래스변수/ 클래스상속

Section02 클래스

■ 클래스의 개념

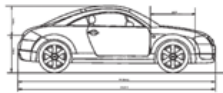
- 클래스의 모양과 생성
 - class 클래스명:
 - # 이 부분에 관련 코드 구현

현실 세계의 사물을 컴퓨터 안에서 구현하려고 고안된 개념

- 자동차를 클래스로 구현



자동차 설계도(클래스)



여러 번
찍어 내기

```
class 자동차:
    # 자동차의 속성
    색상
    속도
    # 자동차의 기능
    속도 올리개()
    속도 내리기()
```

자동차(인스턴스)



그림 12-2 클래스와 인스턴스의 개념



색상
속도



빨강
0km



색상
속도



파랑
0km



색상
속도



노랑
0km

그림 12-4 인스턴스의 필드에 값을 대입하는 개념

단계	작업	형식	예
1단계	클래스 선언	class 클래스명: # 필드 선언 # 메서드 선언	class Car : color = "" def upSpeed(self, value) : ...
2단계	인스턴스 생성	인스턴스 = 클래스명()	myCar1 = Car()
3단계	필드나 메서드 사용	인스턴스.필드명 = 값 인스턴스.메서드()	myCar1.color = "빨강" myCar1.upSpeed(30)

- 생성자의 개념 : 인스턴스를 생성하면서 필드값을 초기화시키는 함수

- 생성자의 기본

- 생성자의 기본 형태 : `__init__()` 라는 이름

Tip • `__init__()` 는 앞뒤에 언더바()가 2개씩, init 는 Initialize 의 약자로 초기화 의미

언더바가 2 개 붙은 것은 파이썬에서 예약해 놓은 것, 프로그램을 작성시 이 이름을 사

용해서 새로운 함수나 변수명을 만들지 말 것

Section04 인스턴스 변수와 클래스 변수

■ 인스턴스 변수의 개념

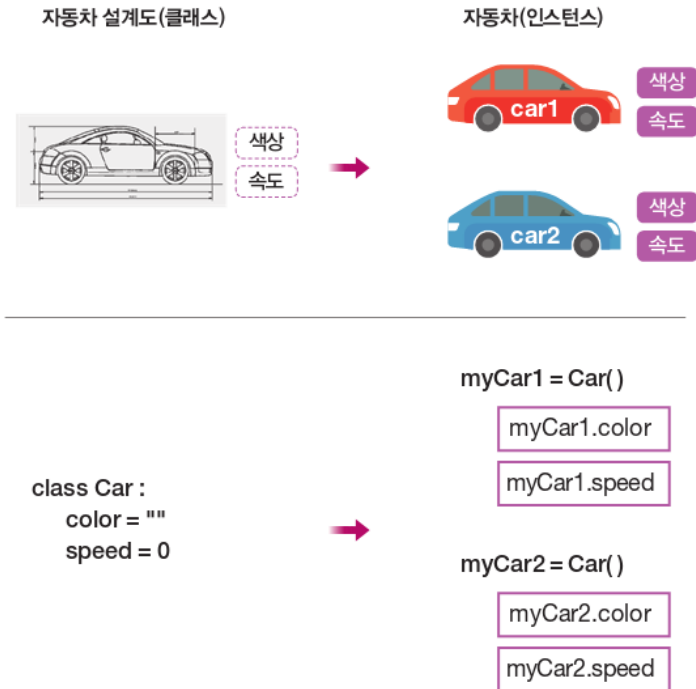


그림 12-5 인스턴스 변수의 개념

■ 클래스 변수

- 클래스 안에 공간이 할당된 변수, 여러 인스턴스가 클래스 변수 공간 함께 사용

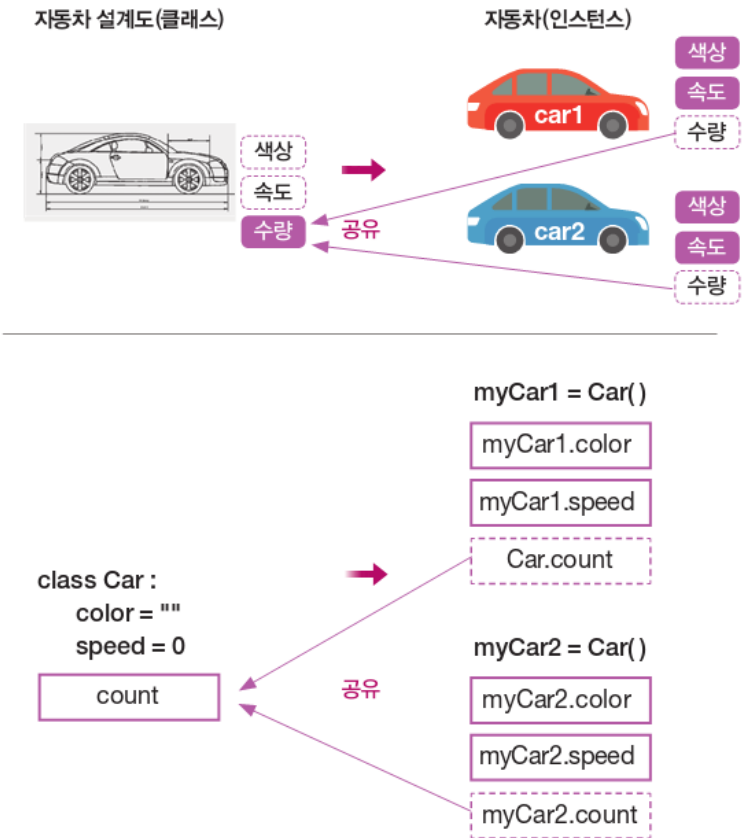


그림 12-6 클래스 변수의 개념

Section04 인스턴스 변수와 클래스 변수

■ 클래스 변수

- 클래스 안에 공간이 할당된 변수, 여러 인스턴스가 클래스 변수 공간 함께 사용

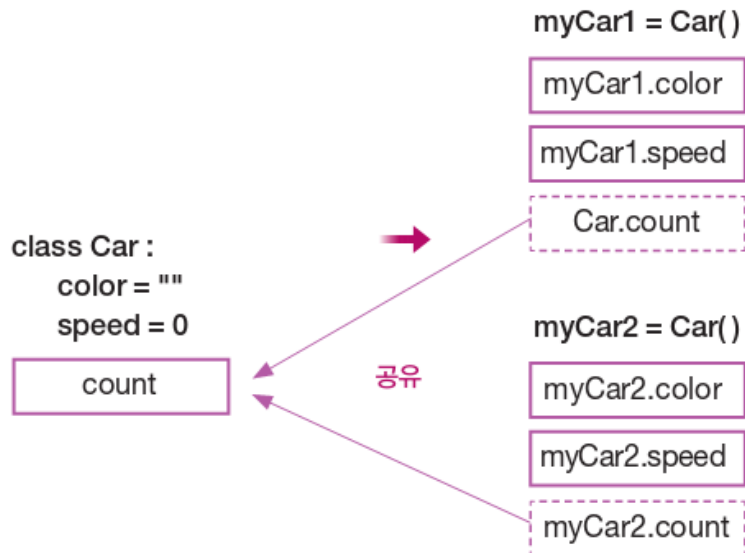
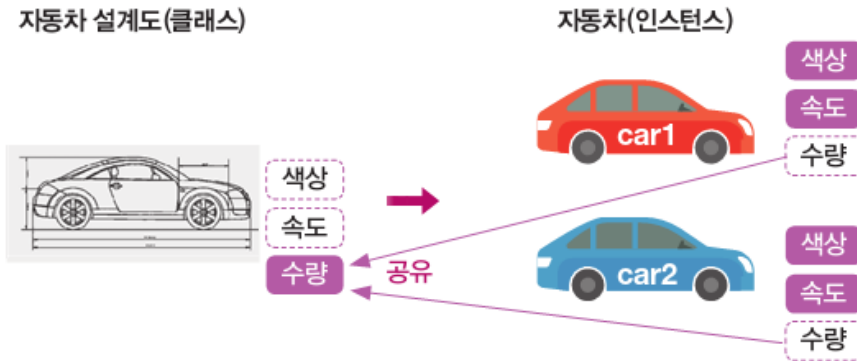


그림 12-6 클래스 변수의 개념

Section05 클래스의 상속

■ 상속의 개념

- 공통된 내용을 자동차 클래스에 두고 상속을 받음으로써 일관되고 효율적인 프로그래밍 가능
- 상위 클래스인 자동차 클래스를 슈퍼 클래스 또는 부모 클래스, 하위의 승용차와 트럭 클래스는 서브 클래스 또는 자식 클래스

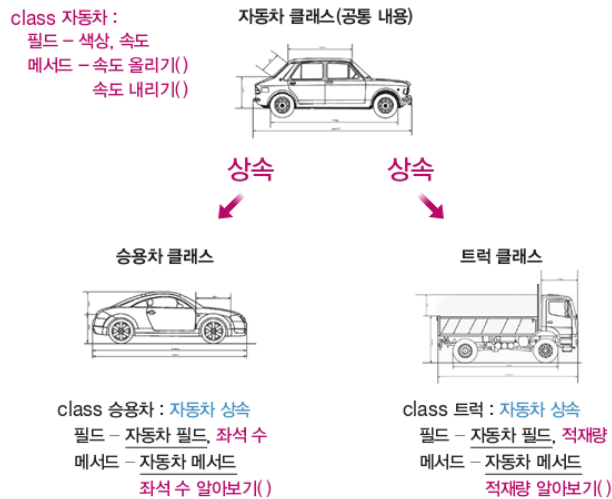


그림 12-8 상속의 개념

■ 메서드 오버라이딩

- 상위 클래스의 메서드를 서브 클래스에서 재정의

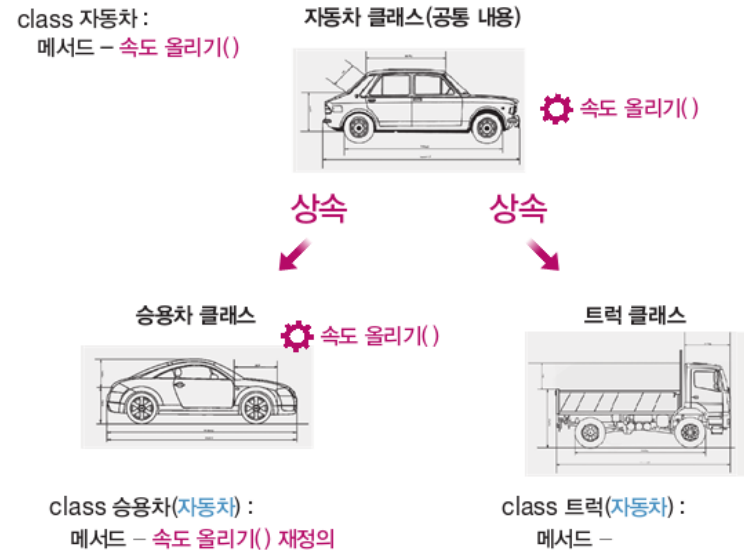


그림 12-9 메서드 오버라이딩의 개념(다른 필드나 메서드는 그림에서 생략)



Thank You
