

함수와 모듈

Section02 함수 기본

1. 함수의 개념과 필요성

- 함수(Function) : '무엇'을 넣으면, '어떤 것'을 돌려주는 요술 상자
- 메서드(Method)와 차이점 : 함수는 외부에 별도로 존재, 메서드는 클래스 안에 존재
- 함수의 형식
 - 함수명()
- 함수는 1함수호출 -> 2함수실행 -> 3함수반환 -> 4함수대입으로 실행된다

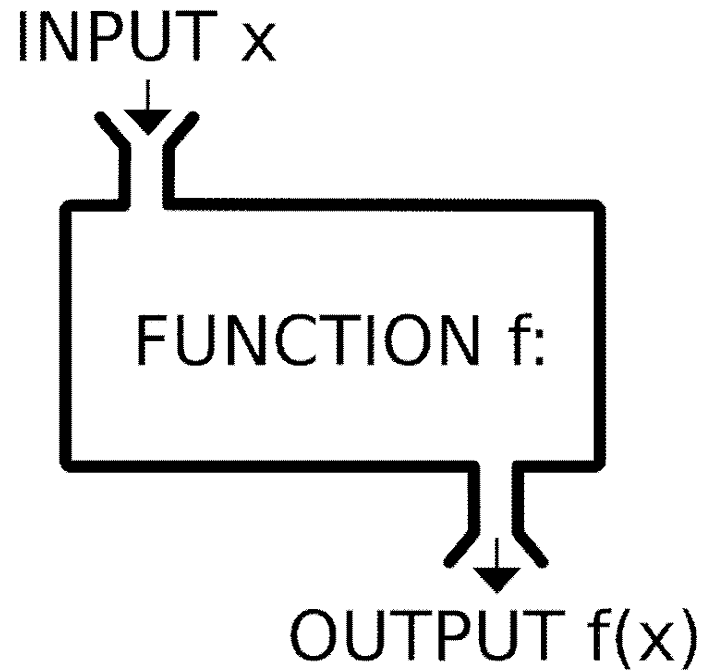
```
함수명()
```

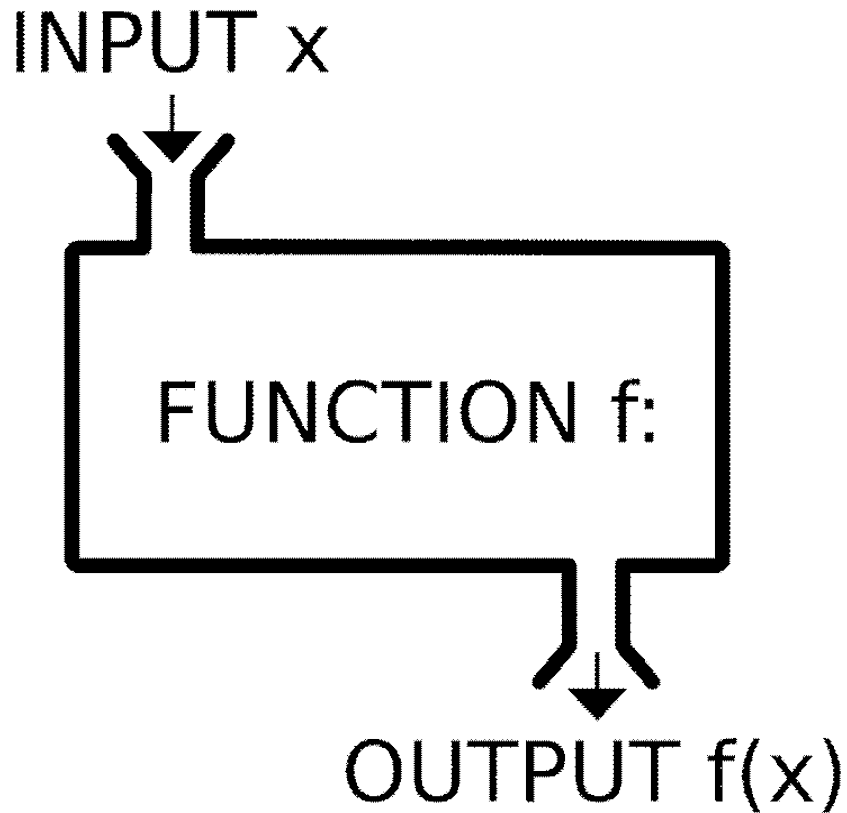
- print() 함수

```
print("CookBook-파이썬")
```

□ 함수

→ 특정 기능을 수행하는 코드 집합





□ 입력값 x

□ 기능구현

□ 반환값 y

input(자료)

Section02 함수 기본

■ 함수의 형식과 활용

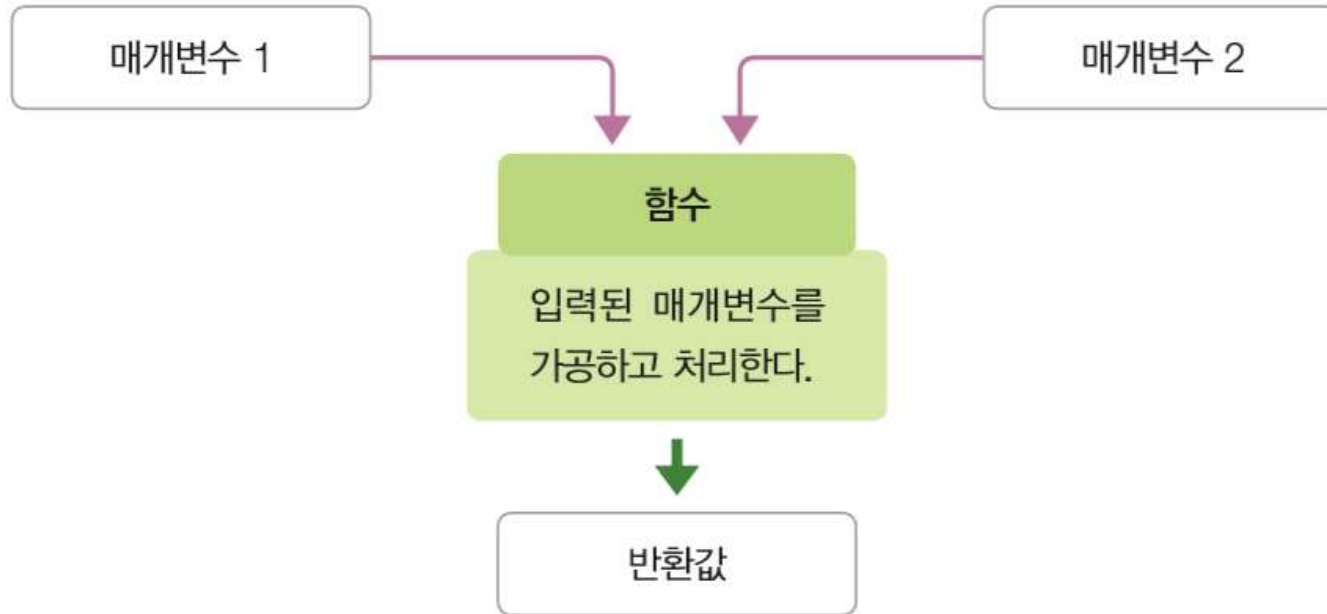
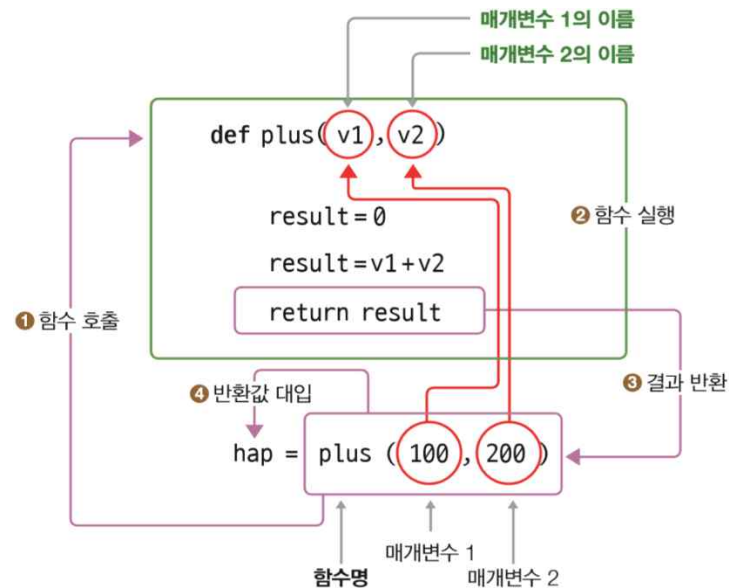
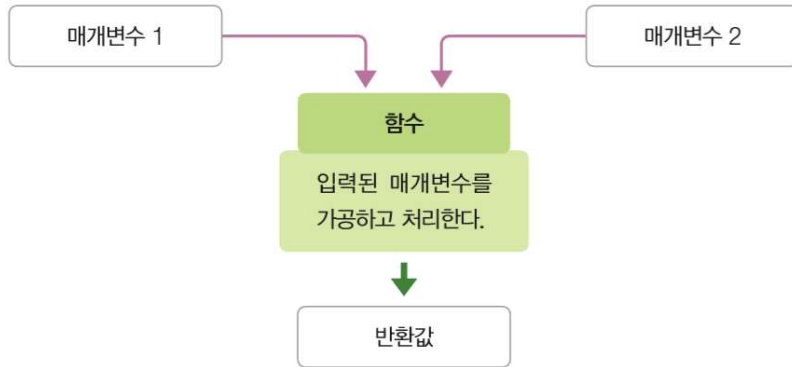


그림 9-3 함수의 기본 형식

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

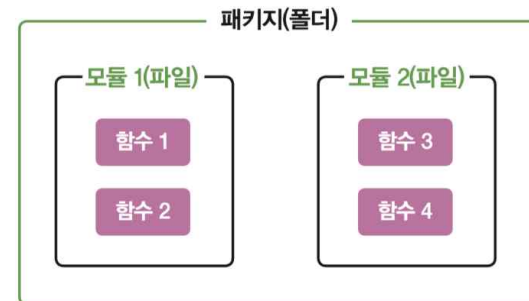
return 문이 없음

■ 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 제공하는 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



Section03 지역 변수, 전역 변수

■ 지역 변수와 전역 변수의 이해

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

① 지역 변수의 생존 범위

함수 1

a = 10

a가 뭔지 함수 1에서 안다.

함수 2

a가 뭔지 함수 2에서 모른다.

② 전역 변수의 생존 범위

함수 1

b가 뭔지 함수 1에서 안다.

함수 2

b가 뭔지 함수 2에서 안다.

b = 20

그림 9-6 지역 변수와 전역 변수의 생존 범위

Section04 함수의 반환값과 매개변수

- 반환값이 없는 함수

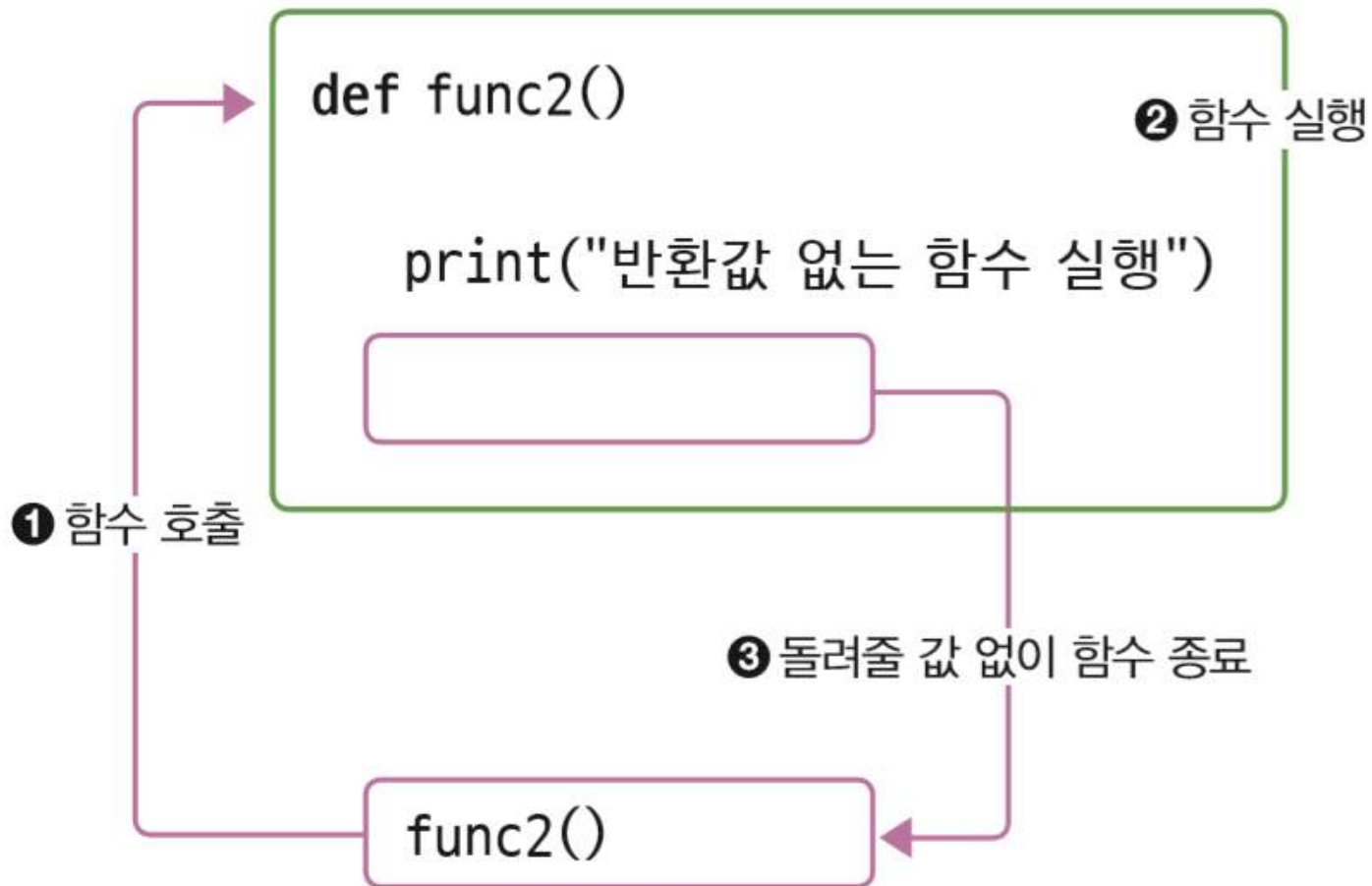


그림 9-9 반환값이 없는 함수의 작동

Section06 함수의 심화 내용

■ 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

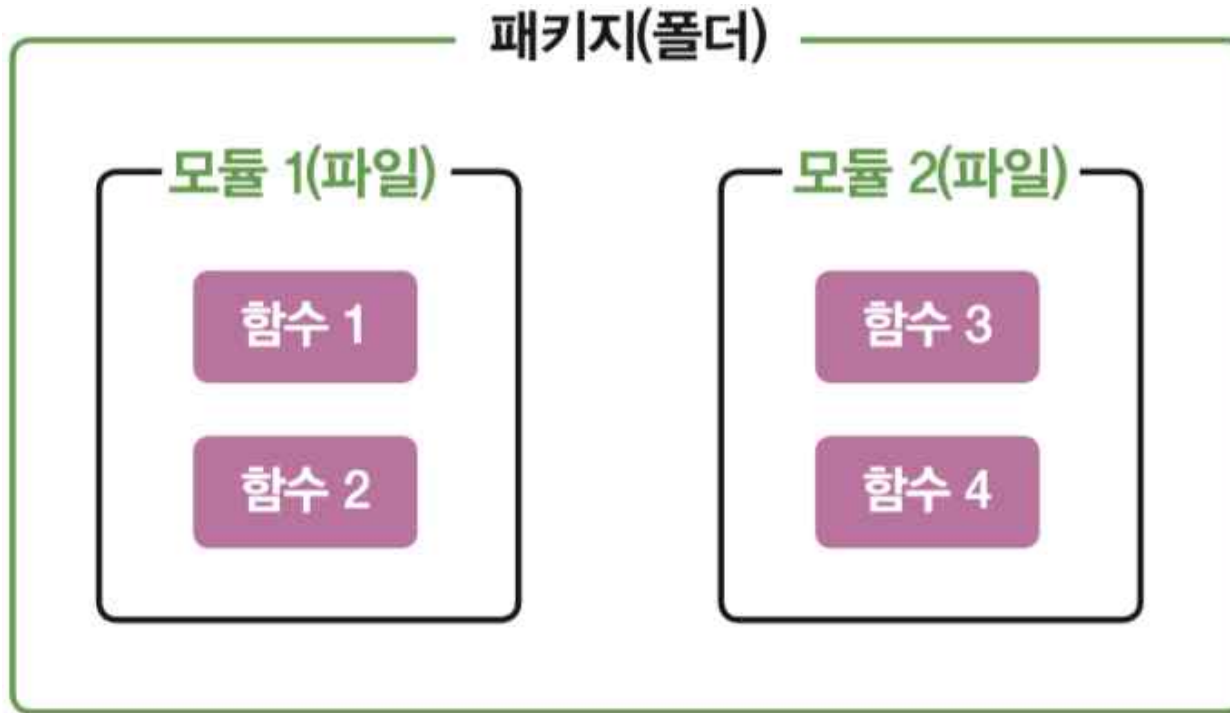


그림 9-11 패키지의 개념



Thank You

객체지향프로그램

클래스/생성자/인스턴스변수/ 클래스변수/ 클래스상속

Section02 클래스

■ 클래스의 개념

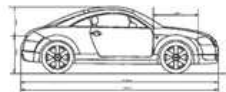
- 클래스의 모양과 생성
 - class 클래스명:
 - # 이 부분에 관련 코드 구현

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 내리개()
```

자동차(인스턴스)



그림 12-2 클래스와 인스턴스의 개념



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

Tlp • `__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

Section02 클래스

- 자동차 클래스의 개념을 실제 코드로 구현
 - 자동차의 속성은 지금까지 사용 한 변수처럼 생성(필드(Field))
 - 자동차의 기능은 지금까지 사용한 함수 형식으로 구현
 - 클래스 안에서 구현된 함수는 함수라고 하지 않고 메서드라고 함.

```
class Car :  
    # 자동차의 필드  
    색상 = ""  
    현재_속도 = 0  
  
    # 자동차의 메서드  
    def upSpeed(증가할_속도량) :  
        # 현재 속도에서 증가할_속도량만큼 속도를 올리는 코드  
  
    def downSpeed(감소할_속도량) :  
        # 현재 속도에서 감소할_속도량만큼 속도를 내리는 코드
```

Section02 클래스

- 인스턴스 구현 형식

자동차 설계도(클래스)

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

자동차(인스턴스)

```
자동차1 = 자동차()  
자동차2 = 자동차()  
자동차3 = 자동차()
```

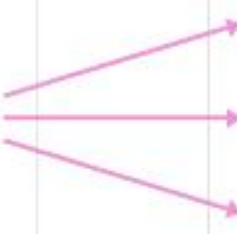


그림 12-3 클래스와 인스턴스 코드 구성

- 자동차 세 대의 인스턴스 생성 코드

```
myCar1 = Car()  
myCar2 = Car()  
myCar3 = Car()
```

Section02 클래스

■ 클래스 사용 순서

클래스 사용 순서
1단계(클래스 선언)
2단계(인스턴스 생성)
3단계(필드나 메서드 사용)

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

Section03 생성자

- 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 사용하여 새로운 함수나 변수명을 만들지 말 것

```
class 클래스명 :  
    def __init__(self) :  
        # 이 부분에 초기화할 코드 입력
```

- 기본 생성자
- 매개변수가 `self` 만 있는 생성자

Section03 생성자

- 기본 생성자

- 매개변수가 self 만 있는 생성자

Tip • Code12 - 03 . py 에서는 코드양 줄이려고 Code12 - 02 . py 와 달리 myCar1 과 myCar2 만 사용

- 매개변수가 있는 생성자

Section04 인스턴스 변수와 클래스 변수

■ 인스턴스 변수

- 예 : Car 클래스 2 개의 필드

```
class Car :  
    color = ""    # 필드 : 인스턴스 변수  
    speed = 0     # 필드 : 인스턴스 변수
```

- 클래스를 이용해 메인 코드에서 인스턴스 만들기

```
myCar1 = Car()  
myCar2 = Car()
```

Section04 인스턴스 변수와 클래스 변수

■ 인스턴스 변수의 개념

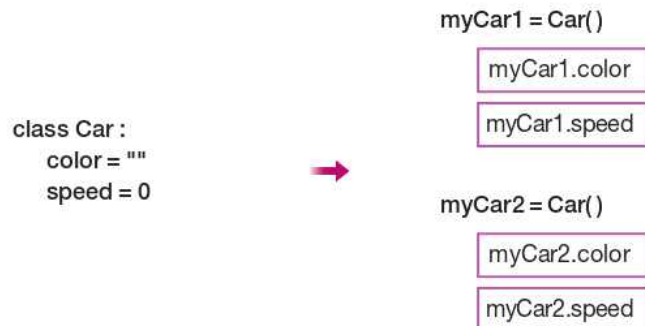
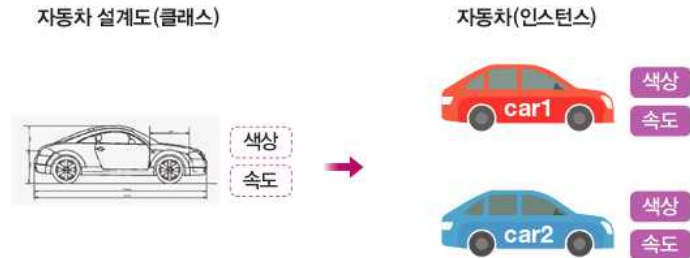


그림 12-5 인스턴스 변수의 개념

Section04 인스턴스 변수와 클래스 변수

■ 클래스 변수

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

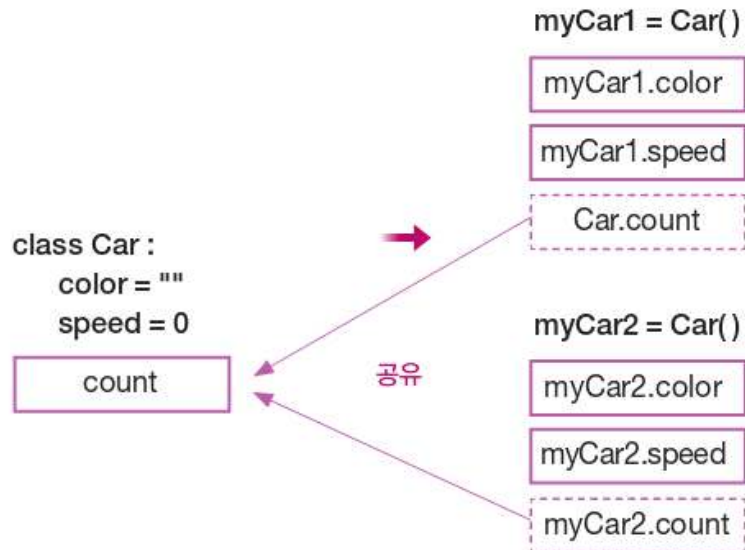
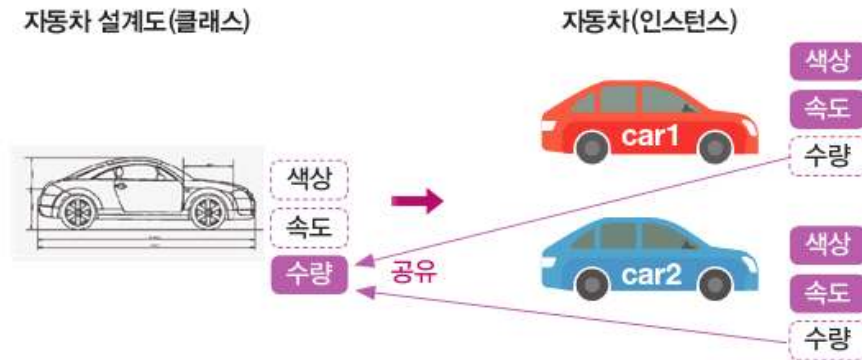


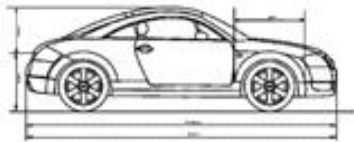
그림 12-6 클래스 변수의 개념

Section05 클래스의 상속

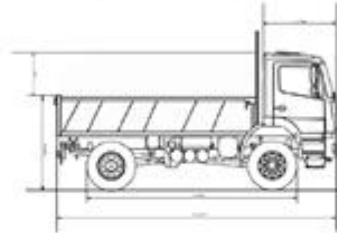
■ 상속의 개념

- 클래스의 상속(Inheritance) : 기존 클래스에 있는 필드와 메서드를 그대로 물려받는 새로운 클래스를 만드는 것

승용차 클래스



트럭 클래스



class 승용차 :

필드 - 색상, 속도, **좌석 수**

메서드 - 속도 올리기()

속도 내리기()

좌석 수 알아보기()

class 트럭 :

필드 - 색상, 속도, **적재량**

메서드 - 속도 올리기()

속도 내리기()

적재량 알아보기()

그림 12-7 승용차와 트럭 클래스의 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

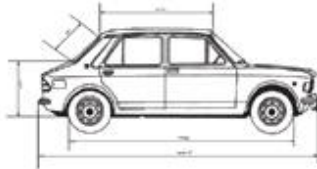
class 자동차 :

필드 - 색상, 속도

메서드 - 속도 올리기()

속도 내리기()

자동차 클래스(공통 내용)



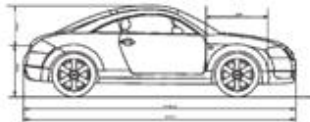
상속



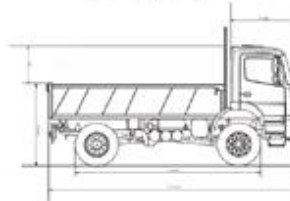
상속



승용차 클래스



트럭 클래스



class 승용차 : 자동차 상속

필드 - 자동차 필드, 좌석 수

메서드 - 자동차 메서드

좌석 수 알아보기()

class 트럭 : 자동차 상속

필드 - 자동차 필드, 적재량

메서드 - 자동차 메서드

적재량 알아보기()

그림 12-8 상속의 개념

Section05 클래스의 상속

■ 메서드 오버라이딩

- 상위 클래스의 메서드를 서브 클래스에서 재정의

class 자동차 :
 메서드 - 속도 올리기()

자동차 클래스(공통 내용)



⚙️ 속도 올리기()

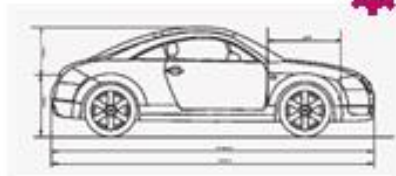
상속



상속

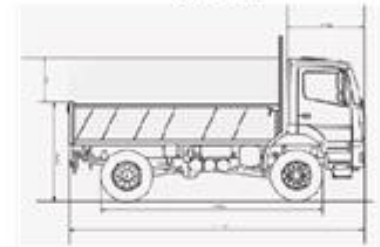


승용차 클래스



⚙️ 속도 올리기()

트럭 클래스



class 승용차(자동차) :
 메서드 - 속도 올리기() 재정의

class 트럭(자동차) :
 메서드 -

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

Section06 객체지향 프로그래밍의 심화 내용

■ 클래스의 특별한 메서드

- `__del__()` 메서드
 - 소멸자(Destructor), 생성자와 반대로 인스턴스 삭제할 때 자동 호출.
- `__repr__()` 메서드
 - 인스턴스를 `print()` 문으로 출력할 때 실행
- `__add__()` 메서드
 - 인스턴스 사이에 덧셈 작업이 일어날 때 실행되는 메서드, 인스턴스 사이의 덧셈 작업 가능
- 비교 메서드 : `__lt__()`, `__le__()`, `__gt__()`, `__ge__()`, `__eq__()`, `__ne__()`
 - 인스턴스 사이의 비교 연산자(<, <=, >, >=, ==, != 등) 사용할 때 호출



Thank You

파일입출력

Section02 파일 입출력의 기본

■ 파일 입출력의 개념

- 표준 입출력(키보드입력, 모니터 출력)과 파일 입출력(파일입력, 파일출력)함수

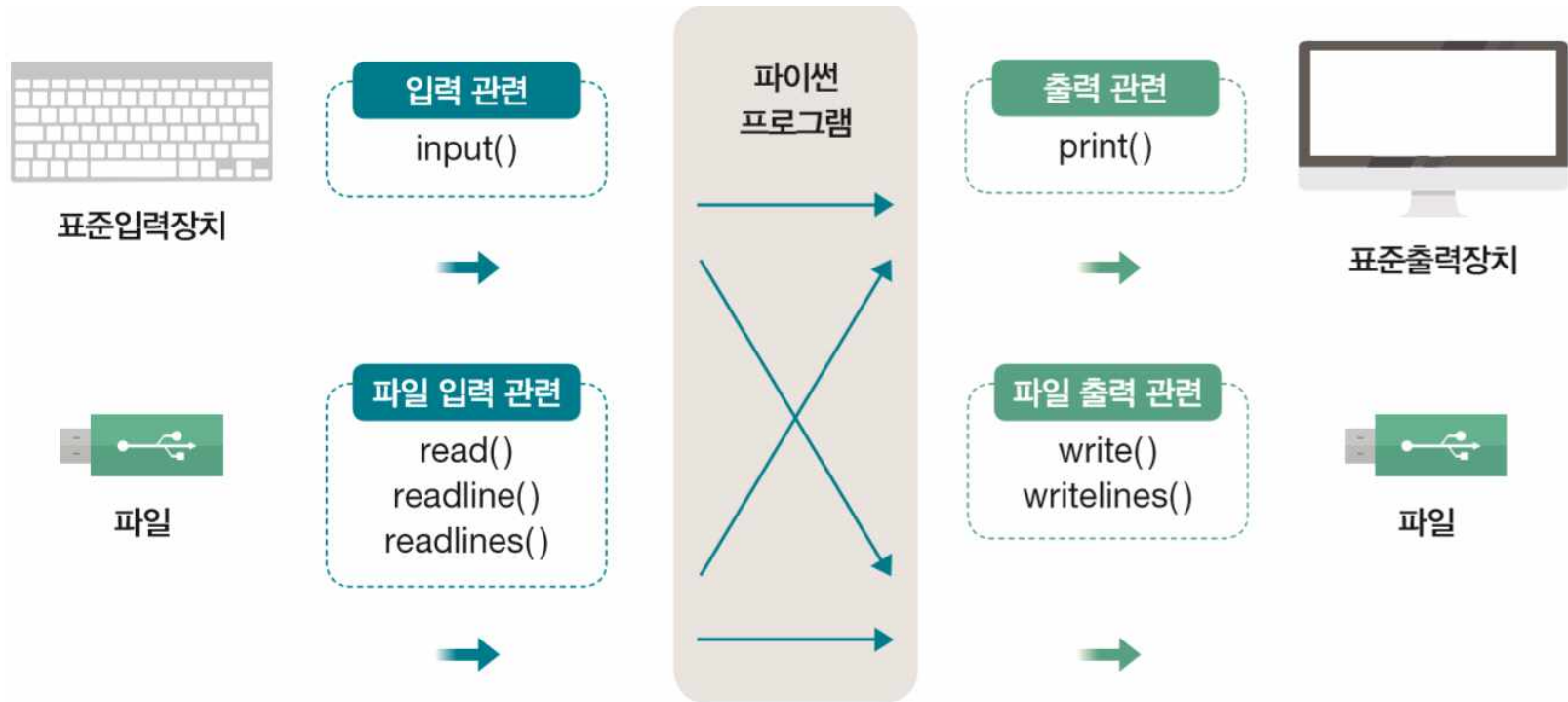


그림 11-1 표준 입출력과 파일 입출력 함수

Section02 파일 입출력의 기본

■ 파일 입출력의 기본 과정



그림 11-2 파일 처리의 3단계

■ 1단계 : 파일 열기

읽기용 : 변수명 = open("파일명", "r")

쓰기용 : 변수명 = open("파일명", "w")

Section02 파일 입출력의 기본

- 모드(Mode) : open() 함수의 마지막 매개변수
- r :읽기모드
w :쓰기 모드

표 11-1 파일의 열기 모드

종류	설명
생략	r과 동일하다.
r	읽기 모드, 기본값이다.
w	쓰기 모드, 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드, 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드, 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드, 이진 파일을 처리한다.

Section02 파일 입출력의 기본

- 모드(Mode) : open() 함수의 마지막 매개변수

표 11-1 파일의 열기 모드

종류	설명
생략	r과 동일하다.
r	읽기 모드, 기본값이다.
w	쓰기 모드, 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드, 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드, 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드, 이진 파일을 처리한다.

- 2단계 : 파일 처리
- 3단계 : 파일 닫기
 - 1단계에서 open() 함수로 연 변수명

```
변수명.close();
```

Section03 텍스트 파일 입출력

■ 파일을 이용한 입력



그림 11-3 파일 입력과 표준 출력

Section03 텍스트 파일 입출력

■ 한 행씩 읽어 들이기

- readline() 함수 사용
- 파일로 데이터 입력 후 이를 화면에 출력하는 프로그램

CookBook 파이썬을 공부합니다.

완전 재미있어요. ^^

파이썬을 공부하기 잘했네요~~

Section03 텍스트 파일 입출력

■ 파일을 이용한 출력

- 출력 결과를 파일에 저장하는 방식(파일에 내용 쓸 때 `write()`나 `writelines()` 함수 사용)



그림 11-5 표준 입력과 파일 출력

Section04 이진 파일 입출력

■ 이진 파일의 개념

- 이진(Binary : 바이너리) 파일 : 글자가 아닌 비트(Bit) 단위로 의미가 있는 파일
 - 텍스트 파일을 제외한 나머지 파일
 - 그림 파일, 음악 파일, 동영상 파일, 엑셀 파일, 실행 EXE 파일 등이 모두 이진 파일
- 텍스트 파일과 이진 파일을 구분하는 간단한 방법(파일을 메모장에서 열기)
 - 열었을 때 글자처럼 보이면 텍스트 파일, 이상하게 보이면 이진 파일

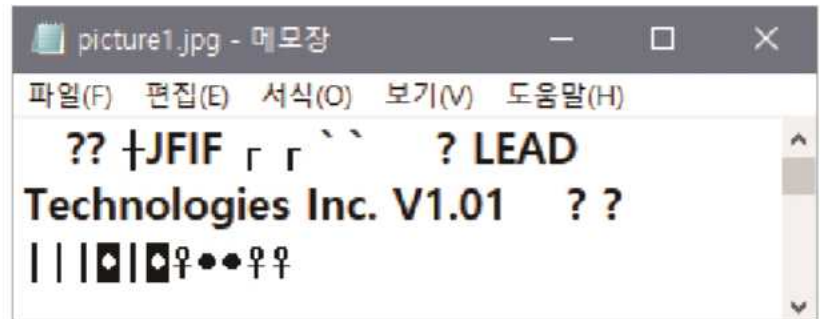
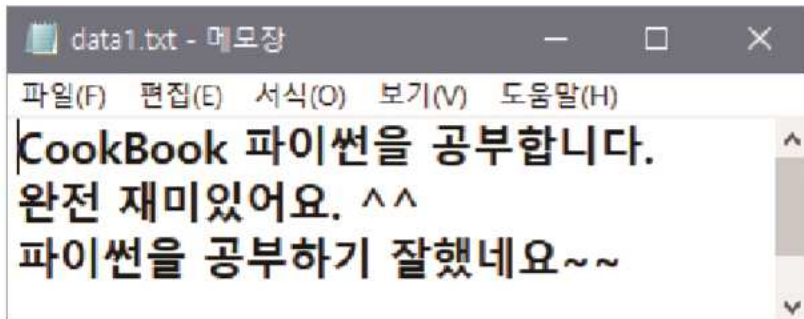


그림 11-9 텍스트 파일과 이진 파일을 메모장에서 열 때

Section04 이진 파일 입출력

■ [프로그램 2]의 완성

- 사진 파일 읽어서 출력
- RAW 사진 파일의 구성 이해
 - *.raw 파일 : 256×256픽셀 크기의 흑백 이미지

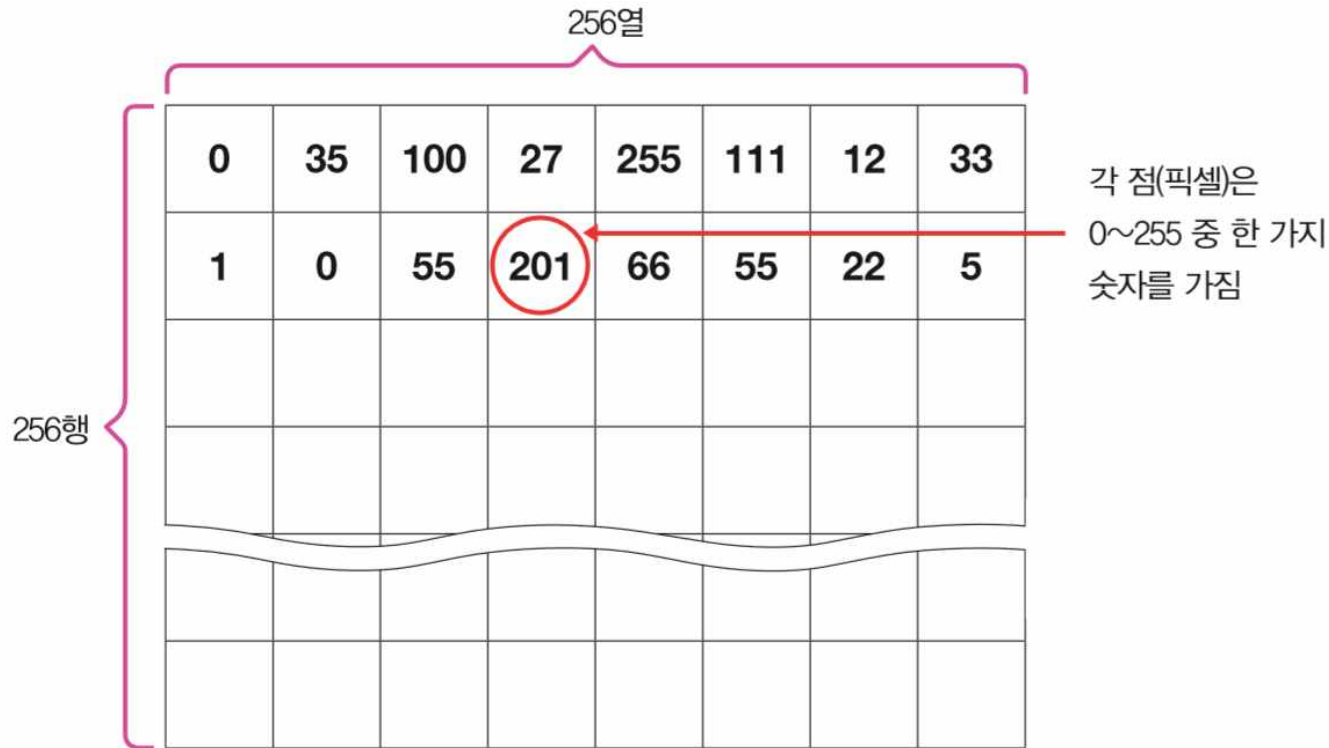


그림 11-10 RAW 사진 파일의 구조

Section04 이진 파일 입출력

- RAW 이미지 출력
 - Code11-11.py의 11행에 다음 두 행을 추가로 코딩

```
paper = PhotoImage(width = XSIZE, height = YSIZE)
canvas.create_image((XSIZE / 2, YSIZE / 2), image = paper, state = "normal")
```

- paper 변수는 캔버스 위에 흰색 종이를 한 장 붙인 것

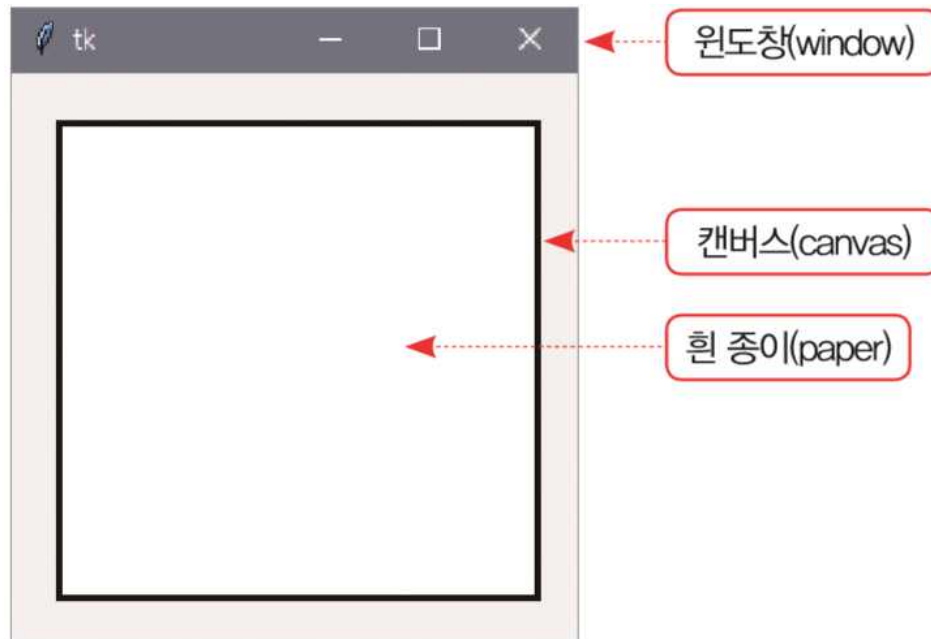


그림 11-11 흰 종이를 붙인 개념의 윈도창

Section05 파일 입출력의 심화 내용

- 대표적인 예외종류

표 11-2 대표적인 예외 종류

종류	설명
ImportError	import 문에서 오류가 발생할 때
IndexError	리스트 등 첨자의 범위를 벗어날 때
KeyError	딕셔너리에서 키가 없을 때
KeyboardInterrupt	프로그램 실행 중 Ctrl + C 를 누를 때
NameError	변수명이 없는 것에 접근할 때
RecursionError	재귀 호출의 횟수가 시스템에서 설정한 것보다 넘칠 때(1000번)
RuntimeError	실행이 발생할 때
SyntaxError	exec()나 eval()에서 문법상 오류가 발생할 때
TypeError	변수형의 오류가 발생할 때. 예 '문자열-문자열' 연산
ValueError	함수의 매개변수에 잘못된 값을 넘길 때. 예 int('파이썬')
ZeroDivisionError	0으로 나눌 때
IOError	파일 처리 등 오류일 때



Thank You

윈도우프로그래밍

Tkinter란?

- Tkinter는 Python에서 사용하는 GUI(graphical user interface) widget set
 - Window
 - 일반적으로 결과가 나타나는 스크린의 직사각형 구역을 뜻한다
 - top-level window
 - 스크린 위에 독립적으로 존재하는 윈도우
 - 표준 프레임이며
 - 시스템의 데스크 탑 매니저를 컨트롤한다

Tkinter 구성요소

- **위젯(widget)**
 - GUI에서 어플리케이션을 만드는 블록을 구성하는 포괄적인 용어이다
 - 위젯의 예
 - buttons, radio buttons, text fields, frames, and text labels
- **프레임(frame)**
 - 프레임 위젯은 복잡한 레이아웃을 조직하는 기본단위
 - 프레임은 다른 위젯을 포함하는 직사각형 영역이다
- **child, parent**
 - 어떤 위젯이 만들어질 때, parent-child 관계가 만들어진다
 - 예를 들어,
 - text label을 프레임 안에 위치시킬 때,
 - 프레임은 label의 parent이다

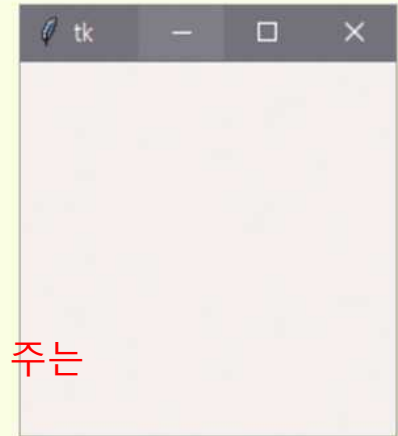
Section02 기본 위젯 활용

■ 기본 윈도우창의 구성

- 위젯(Widget) : 윈도우창에 나올 수 있는 문자, 버튼, 체크박스, 라디오버튼 등을 의미

Code10-01.py

```
1 from tkinter import *
2
3 window = Tk()
4
5 ## 이 부분에서 화면을 구성하고 처리 ##
6
7 window.mainloop()
```



1행 : tkinter는 파이썬에서 GUI 관련 모듈 제공해 주는
표준 윈도우 라이브러리

3행 : Tk()는 기본이 되는 윈도우를 반환, 이를 루트 윈도우
또는 베이스 윈도우라고 함. 3행 실행하면 윈도우창
화면에 출력

3행에서 베이스 윈도우를 window 변수에 넣고 7행에서
window.mainloop() 함수 실행

Tip • tkinter 는 TK Interface의 약어. Tk는 Tcl/Tk라는 전통적인 GUI 인터페이스 윈도우, 리눅스,
맥 등에서 모두 동일한 코드로 사용 가능

입력을 위한 Tkinter widgets

widget	Description
Entry	가장 기본적인 text box이다 보통 User가 한 줄의 text 를 입력하도록 한다 다양한 서식 설정을 허용하지 않는다
Text	User에게 여러 줄의 text 를 입력하도록 한다 입력 된 그 text를 저장한다 서식 설정 옵션을 제공한다(style, attributes)
Button	User가 GUI에 명령을 수행하도록 하는 기본방법, e.g. "OK" or "Cancel" in a dialog
Radiobutton	User에게 목록으로부터 하나의 옵션을 선택 하도록 한다
Checkbutton	User에게 목록으로부터 여러 개의 옵션을 선택 하도록 한다

출력을 위한 Tkinter widgets

widget	Description
Label	Text나 images을 보여주는 가장 기본적인 방법
PhotoImage/BitmapImage	실제 widget보다 class 객체(objects)에 가깝다 Label이나 다른 display widgets 과 혼합하여 사용된다 Images와 bitmaps을 각각 보여준다
Listbox	User가 강조할 수 있는 Text items의 목록을 보여준다
Menu	User에게 menu를 만들 수 있는 기능을 제공한다

그래픽 요소들을 처리하는 명령어

Geometry Manager	Description
pack	Widgets과 함께 공간을 채우는 데 사용된다 이전에 기술한 그래픽 요소들이 화면에 나타나게 한다
grid	Widget을 parent grid위에 둔다 각각의 widget이 몇몇 옵션과 함께 다중의 cells을 덮도록 만들어 질 수 있다 하지만 자신의 box나 cell을 grid내에서 표시된다
place	window의 크기와 위치를 정확히 지정한다 일반적인 layout보다 custom layout을 실행하는데 사용된다

Tkinter widget option

option	description
text	글자를 화면에 출력
image	그림을 화면에 출력
width	이미지나 글자의 넓이 지정, 지정하지 않으면 원래 데이터의 크기로 보인다
height	이미지나 글자의 높이 지정, 지정하지 않으면 원래 데이터의 크기로 보인다
relief	보더 스타일(flat(default), Groove, Raised, Ridge, Sunken..)
borderwidth	보더의 넓이, 지정하지 않으면 '0'로 선이 나타나지 않는다
background	배경 색상
foreground	전경 색상
font	폰트이름 지정

버튼의 종류

- **Button**

- 눌러서 뭔가 다른 기능을 시행하게 하는 역할
- 원하는 문자가 들어가는 버튼을 만들 수 있다

- **Radiobutton**

- 선택 가능한 리스트를 나열하고
- 그 중에서 한 개만 선택하게 한다

- **Checkbox**

- 선택 가능한 리스트를 나열하고
- 그 중에서 원하는 경우 여러 개를 선택하게 한다

Section04 키보드와 마우스 이벤트 처리

■ 마우스 이벤트 기본 처리

표 10-1 마우스 이벤트

마우스 작동	마우스 버튼	이벤트 코드
클릭할 때	모든 버튼 공통	〈Button〉
	왼쪽 버튼	〈Button-1〉
	가운데 버튼	〈Button-2〉
	오른쪽 버튼	〈Button-3〉
떼었을 때	모든 버튼 공통	〈ButtonRelease〉
	왼쪽 버튼	〈ButtonRelease-1〉
	가운데 버튼	〈ButtonRelease-2〉
	오른쪽 버튼	〈ButtonRelease-3〉
더블클릭할 때	모든 버튼 공통	〈Double-Button〉
	왼쪽 버튼	〈Double-Button-1〉
	가운데 버튼	〈Double-Button-2〉
	오른쪽 버튼	〈Double-Button-3〉
드래그할 때	왼쪽 버튼	〈B1-Motion〉
	가운데 버튼	〈B2-Motion〉
	오른쪽 버튼	〈B3-Motion〉
마우스 커서가 위젯 위로 올라왔을 때		〈Enter〉
마우스 커서가 위젯에서 떠났을 때		〈Leave〉

Section04 키보드와 마우스 이벤트 처리

- 마우스 이벤트가 처리 형식

```
def 이벤트처리함수(event) :  
    # 이 부분에 마우스 이벤트가 발생할 때 작동할 내용 작성  
  
위젯.bind("마우스이벤트", 이벤트처리함수)
```

Section04 키보드와 마우스 이벤트 처리

■ 키보드 이벤트

표 10-2 키보드 이벤트

키보드 작동	이벤트 코드
모든 키를 누를 때	<Key>
특수 키를 누를 때	<Return>, <BackSpace>, <Tab>, <Shift_L>, <Control_L>, <Alt_L>, <Pause>, <Caps_Lock>, <Escape>, <End>, <Home>, <Left>, <Right>, <Up>, <Down>, <Num_Lock>, <Delete>, <F1>~<F12> 등
일반 키를 누를 때	a~z, A~Z, 0~9, <space>, <less>
화살표 키와 조합	<Shift-Up>, <Shift-Down>, <Shift-Left>, <Shift-Right> 등

- Enter 를 처리하려면 Code10-16.py에서는 11행의 <Key> 대신 <Return>을 사용
- 대·소문자 등도 구분해서 처리 가능
- 소문자 r 은 11행의 <Key> 대신에 r을 사용해 처리
- 일반 키를 누를 때 주의할 점은 SpaceBar 는 <Space>로, < 는 <less>로 사용

Section05 메뉴와 대화상자

■ 메뉴의 생성

■ 메뉴의 구성 개념과 형식

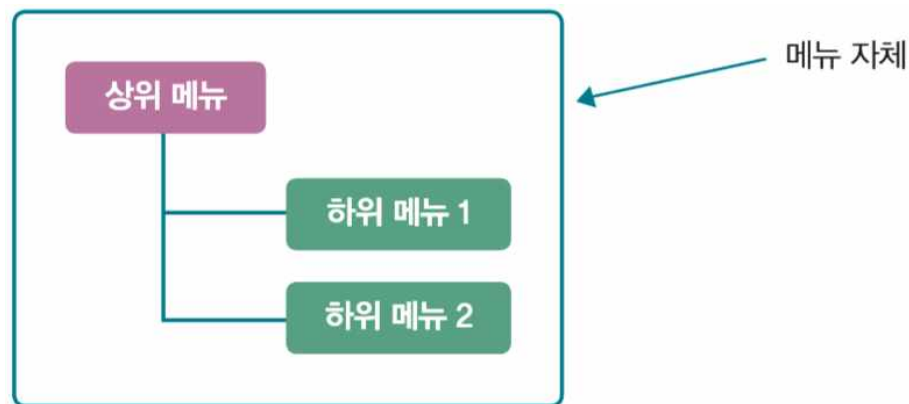


그림 10-1 메뉴의 구성 개념도

```
메뉴자체 = Menu(부모윈도)
```

```
부모윈도.config(menu = 메뉴자체)
```

```
상위메뉴 = Menu(메뉴자체)
```

```
메뉴자체.add_cascade(label = "상위메뉴텍스트", menu = 상위메뉴)
```

```
상위메뉴.add_command(label = "하위메뉴1", command = 함수1)
```

```
상위메뉴.add_command(label = "하위메뉴2", command = 함수2)
```

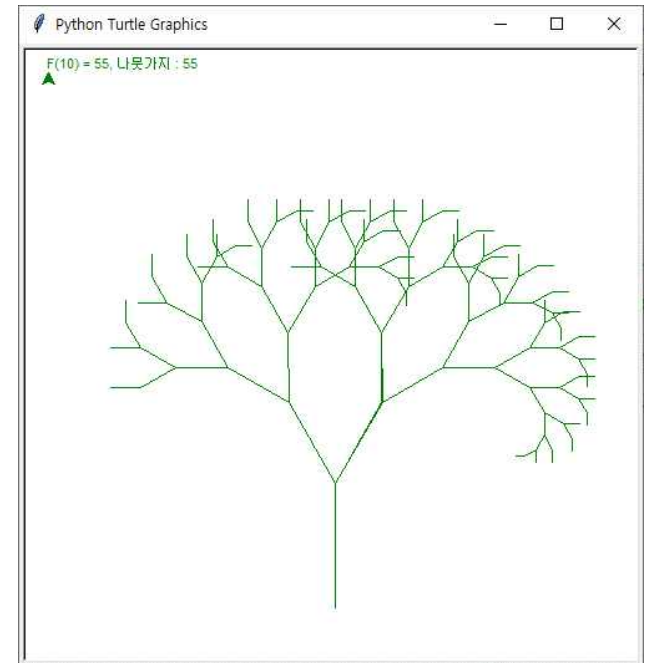


Thank You

Turtle

- **터틀 그래픽** (turtle graphic)

- 1966, **교육용** 프로그래밍 언어인 Logo에서 처음 소개
- 꼬리에 잉크가 묻은 거북이를 종이에 올려놓고 리모컨으로 조작하는 방식으로 동작
- 화면에서 거북이를 이용하여 지나간 흔적으로 만들어지는 그림
- 거북이가 펜을 가지고 있고 프로그래머가 명령을 이용하여 거북이를 움직이면 그림이 그려짐

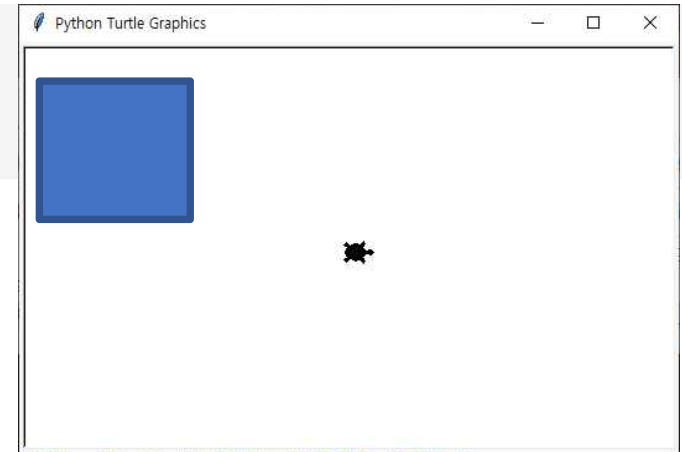








- 터틀 그래픽의 사용

- **import** 예약어로 turtle 모듈을 불러와 사용
- **turtle.shape("turtle")**에 의해 거북이가 캔버스에 나타남

```
>>> import turtle
>>> turtle.shape("turtle")
```

- [Python Turtle Graphics] 화면의 중앙(x:0, y:0)에 거북이가 나타남
- turtle.**shape**("turtle")에 의해 거북이 모양 변경

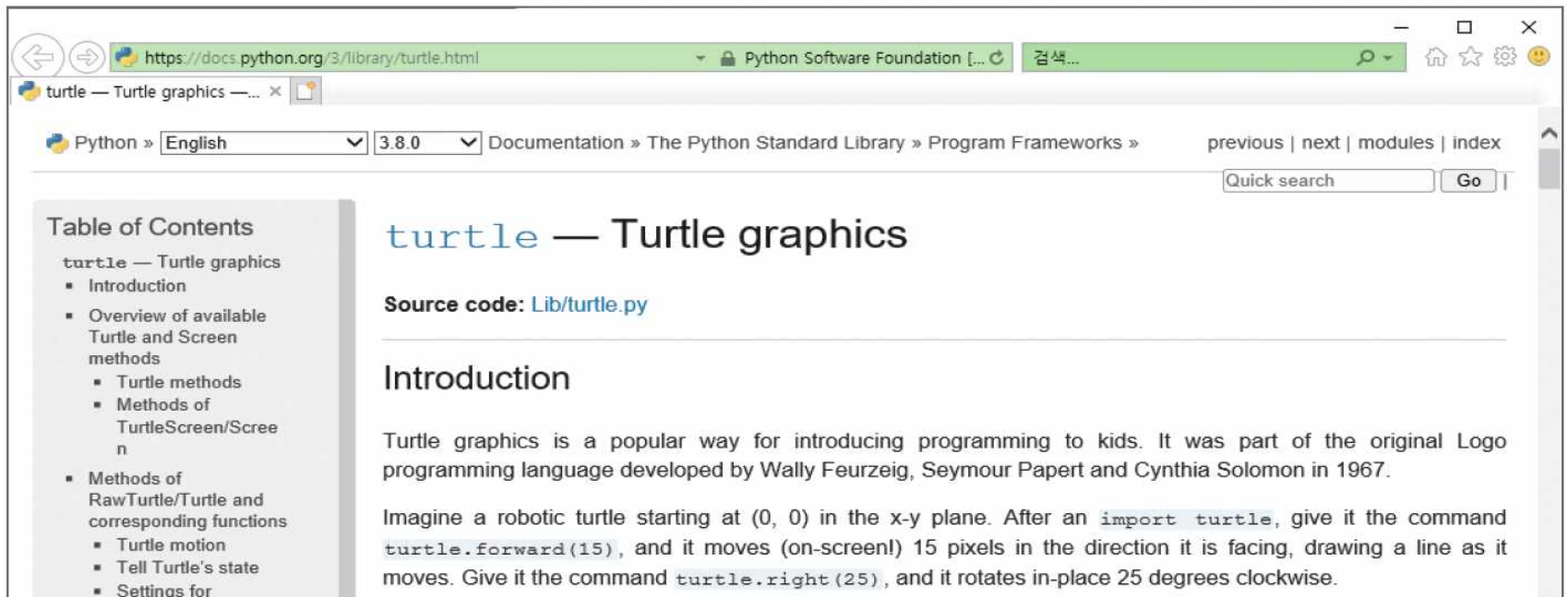


classic	arrow	turtle	circle	square	triangle
					

8. 거북이의 방향에 따라 앞(forward), 왼쪽(left), 오른쪽(right), 뒤(backward)가 결정된다.

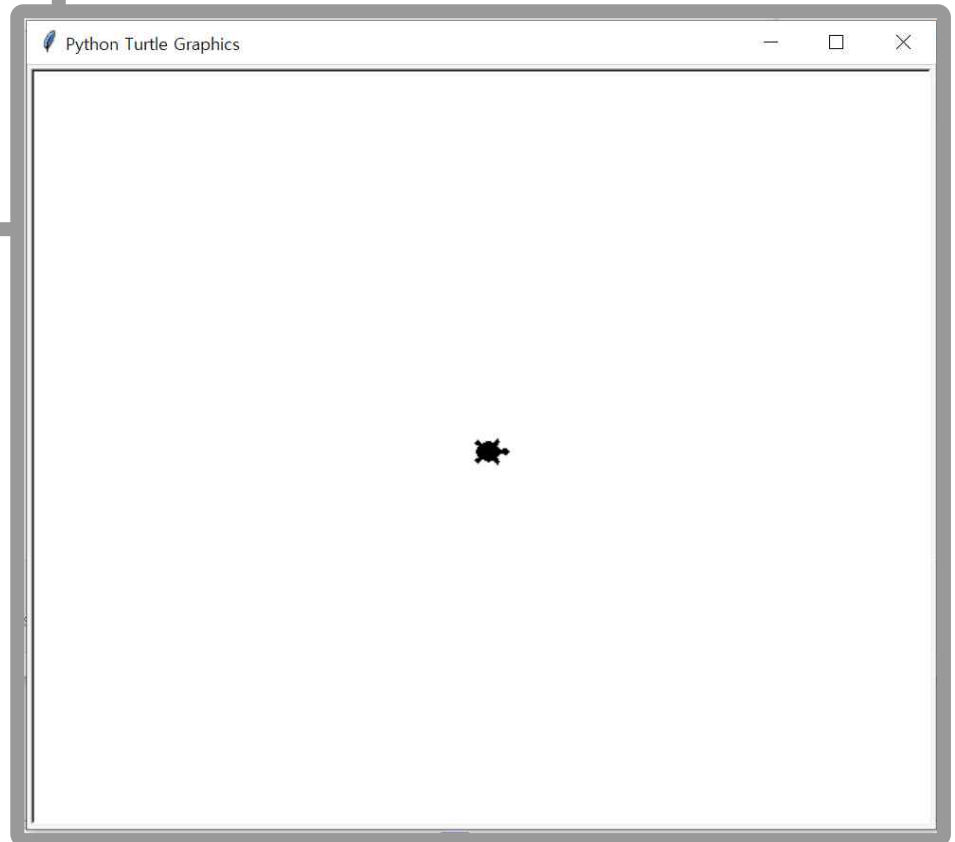


9. turtle 모듈의 공식 문서는 <https://docs.python.org/3/library/turtle.html> 홈페이지를 통해 확인할 수 있다.



turtle 로 배우는 python 기본문법 실습

```
1 import turtle as t  
2  
3 t.shape("turtle")  
4 t.mainloop()
```



□ `turtle.shape(" 설정할 모양 ")`

▣ ➡ 거북이 모양설정

□ `turtle.getshapes()`

▣ ➡ 설정할 수 있는 모양들 반환

□ `turtle.color("red")`

▣ ➡ 커서 색깔 설정

□ `turtle.penup()`

▮ ➡ 펜을 들겠다! (이동간, 선을 그리지 않음)

□ `turtle.pendown()`

▮ ➡ 펜을 내리겠다! (이동간, 선을 그림)

□ `turtle.hideturtle()`

▮ ➡ 커서를 숨긴다.

□ `turtle.showturtle()`

▮ ➡ 커서를 드러냄.

□ turtle.forward(X)

▣ ➡ X 만큼 전진!

□ turtle.left(X)

▣ ➡ 왼쪽으로 X 만큼 돌아라!

□ turtle.right(X)

▣ ➡ 오른쪽으로 X 만큼 돌아라!

□ turtle.speed(X)

▣ ➡ 이동 스피드 조정 (0~1)

프로그램c02-07

■ 필요한 변수 준비

- 거북이로 그릴 선의 두께(pSize)와 거북이의 크기(tSize) 변수
- 색상을 표현하는 빨강(r), 초록(g), 파랑(b) 변수

```
pSize, tSize = 10, 0  
r, g, b = 0.0, 0.0, 0.0
```

또는

```
r = 0.0  
g = 0.0  
b = 0.0
```

■ 기능1 구현

- 마우스 왼쪽 버튼을 누르면 거북이가 클릭한 지점까지 임의의 색상으로 선을 그리면서 따라오도록 하는 함수

```
def screenLeftClick(x, y) :  
    global r, g, b  
    turtle.pencolor((r, g, b))  
    turtle.pendown()  
    turtle.goto(x, y)
```

■ 기능2 구현

- 마우스 오른쪽 버튼을 누르면 거북이가 클릭한 지점까지 선을 그리지 않고 이동만 하도록 하는 함수

```
def screenRightClick(x, y) :  
    turtle.penup()  
    turtle.goto(x, y)
```

■ 기능3 구현

- 마우스 가운데 버튼을 누르면 거북이가 임의로 크기를 확대 또는 축소하는 함수
- 마우스 가운데 버튼을 누를 때는 선의 색상(0~0.99999)이 임의로 선택되도록 하고 거북이의 크기도 1부터 9까지 임의로 설정되도록 함

```
def screenMidClick(x, y) :  
    global r, g, b  
    tSize = random.randrange(1, 10)  
    turtle.shapesize(tSize)  
    r = random.random()  
    g = random.random()  
    b = random.random()
```



Thank You

Game

Random

□ Randint(A, B)

▣ ➡ A ~ B 까지 정수를 랜덤으로 반환 ($A < B$)

특정 이벤트를 발생시킬 때도 사용

random 을 배워볼까요?

왜냐면 while 이라는 구문이 반복횟수가 정해져 있지 않기 때문에

게임 코드에 자주 쓰입니다.

자, 우선 게임은 변수가 있어야 재밌죠.

import random # import [] random 을 이 파일에서 사용하겠다

 # 정도로만 이해하고 가시면 되요.

 # 나중에 class 와 import 파트가 따로 준비되어있는데 거기

서 다룰게요

random.randint(1, 10) # 다음과 같이 A(정수), B(정수) 를 넣어주면

 # A 에서 B 까지의 수가 랜덤으로 나옵니다.

 # 즉, 실행할 때마다 예측할 수 없는 수가

업다운 게임

Random

연산 게임

Section02 슈팅 게임 프로젝트

■ 외부 라이브러리 설치

- pygame 외부 라이브러리 사용

```
pip install pygame
```

TIP • 파이썬 최신 버전을 사용할 경우에 pip install pygame 명령이 실패할 수도 있다. 이런 경우에는 관련 파일을 직접 다운로드해서 설치해도 좋다. <https://www.lfd.uci.edu/~gohlke/pythonlibs/#pygame>이나 책의 사이트(<http://www.hanbit.co.kr/src/4466>)에서 Pygame 관련 파일을 다운로드한 후 적당한 폴더(예:C:\Temp)에 복사한다. 그런 뒤 명령 프롬프트를 열고 다음 명령으로 설치하면 된다.

```
CD C:\Temp  
pip install 다운로드_파일
```

이 책을 집필하는 시점에는 32비트 파이썬용 pygame-1.9.6-cp38-cp38-win32.whl 또는 64비트 파이썬용 pygame-1.9.6-cp38-cp38-win_amd64.whl 파일을 다운로드할 수 있으며, 이 책에서 사용한 Python 3.8.0 버전에 잘 설치된다. 참고로 파일 이름에서 cp38이 파이썬 3.8버전을 의미한다. 파이썬 3.9용의 경우 cp39로 표기될 것이다.

```
import pygame
```



Thank You
