

CHAPTER 10

윈도 프로그래밍

학습목표

- 윈도 프로그래밍의 개념을 익힌다.
- 윈도 프로그램에서 활용되는 다양한 위젯을 배운다.
- 메뉴의 구성을 이용한 고급 윈도 프로그래밍을 배운다.

SECTION 01 이 장에서 만들 프로그램

SECTION 02 기본 위젯 활용

SECTION 03 위젯의 배치와 크기 조절

SECTION 04 키보드와 마우스 이벤트 처리

SECTION 05 메뉴와 대화상자

요약

연습문제

응용예제

파이썬 for Beginner

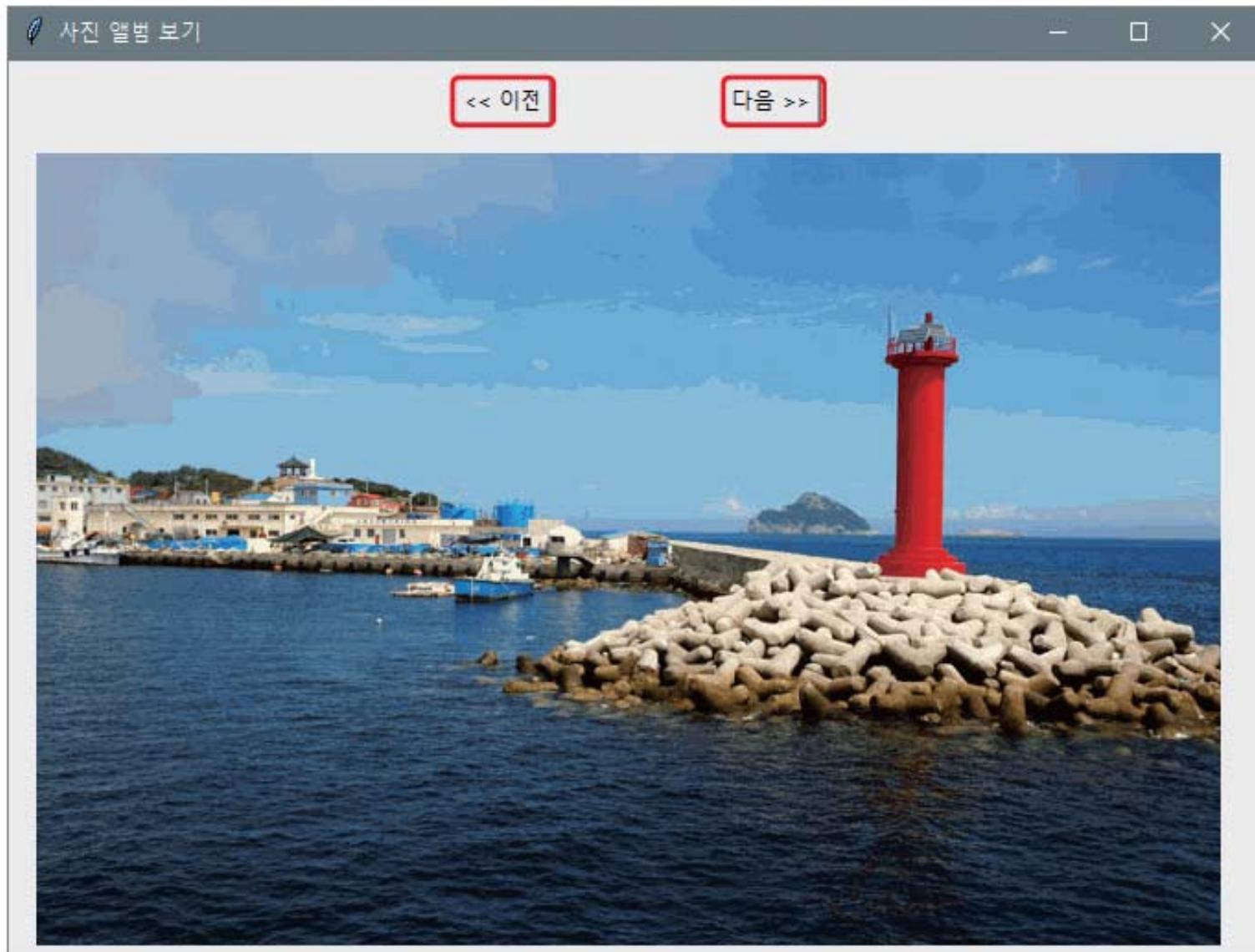
[강의교안 이용 안내]

- 본 강의교안의 저작권은 한빛아카데미(주)에 있습니다.
- 이 자료는 강의 보조자료로 제공되는 것으로 무단으로 전제하거나 배포하는 것을 금합니다.



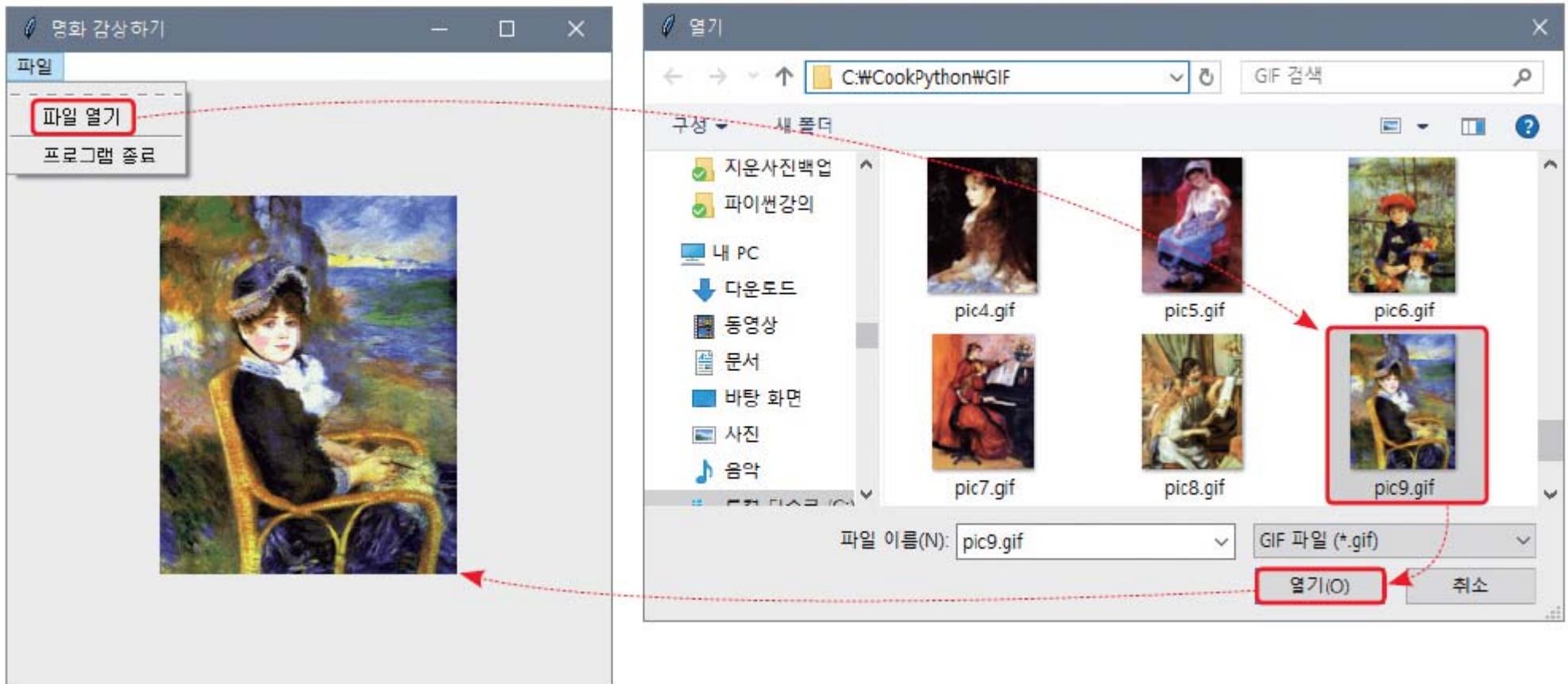
Section01 이 장에서 만들 프로그램

■ [프로그램 1] 사진 앤범



Section01 이 장에서 만들 프로그램

■ [프로그램 2] 영화 감상



■ Python GUI 프로그래밍

[출처] <http://pythonstudy.xyz/python/gui>

- 파이썬에서 여러 GUI Framework (혹은 Toolkit) 들을 사용할 수 있음

- 파이썬에서 기본적으로 제공되는 표준 GUI 라이브러리인 **Tkinter**
- Qt 프레임워크를 파이썬에서 사용하도록 한 **PyQt**와 **PySide**,
- GTK(gimp toolkit)을 파이썬에서 사용하게 한 **PyGTK** 등

- **PyQt**

- 동일한 파이썬코드를 사용하여 윈도우즈, Mac, 리눅스에서 모두 동작하는 GUI 프로그램을 작성
- 현재 PyQt5 버전으로 Qt v5를 파이썬에서 쓸 수 있도록 한 것임
- 하나의 언어에서 작성된 라이브러리나 서비스를 다른 언어에서 사용할 수 있도록 하는 것을 Language Binding 이라 한다. PyQt는 Qt의 Python Language Binding 중의 하나이다.
- Qt는 C++로 작성된 크로스 플랫폼 프레임워크로 The Qt Company에서 작성한 프로그램임

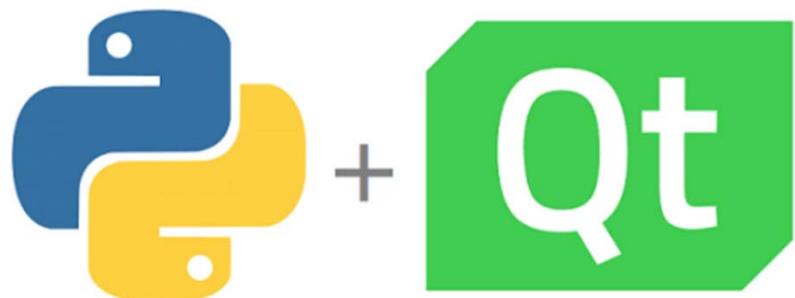
- **Tkinter**

- Tcl/Tk에 대한 파이썬 Wrapper로서 **Tcl/Tk**를 파이썬에 사용할 수 있도록 한 **Lightweight GUI 모듈**
- Tcl은 Tool Command Language의 약자로서 일종의 프로그래밍 언어이며, Tk는 크로스 플랫폼에 사용되는 일종의 GUI 툴킷임
- Tkinter는 타 GUI 프레임워크나 툴킷에 비해 지원되는 위젯들이 부족하고 UI도 그렇게 예쁘지 않다는 단점이 있지만, Python 설치시 기본적으로 내장되어 있는 파이썬 표준 라이브러리이기 때문에 쉽고 간단한 GUI 프로그램을 만들 때 활용됨

■ PyQt5

[출처] <http://codetorial.net/pyqt5/intro.html>

- PyQt5는 Qt5 어플리케이션 프레임워크에 대한 파이썬 버전임. Qt는 플랫폼에 관계없이 다양한 기능을 포함하는 C++ 라이브러리이자 개발툴임.
- PyQt5는 이러한 1,000여개의 클래스들을 포함하는 파이썬 모듈의 모음임.
- PyQt5는 윈도우, 리눅스, macOS, 안드로이드, iOS를 지원함.
- 공식 홈페이지에서 최신의, 그리고 안정적인 버전의 PyQt5와 최신 버전의 문서를 얻을 수 있음.
- PyQt5 개발자는 GPL과 상업용 라이선스 중 하나를 선택할 수 있음 (라이선스 관련)
- Riverbankcomputing에 의하면, PyQt는 Qt C++ 크로스-플랫폼 어플리케이션 프레임워크와 크로스-플랫폼 인터프리터 언어 파이썬의 장점을 결합하였다고 소개함.



Section02 기본 위젯 활용

■ 기본 윈도창의 구성

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

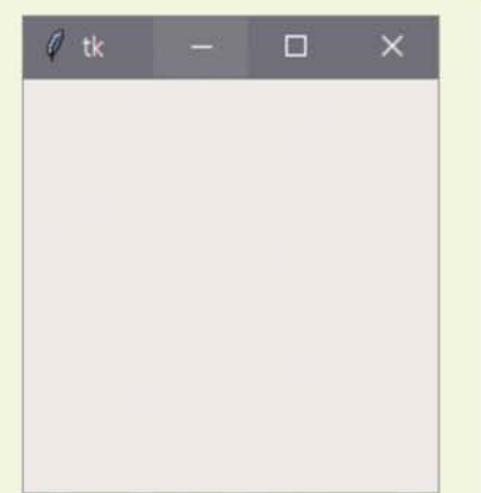
Code10-01.py

```
1 from tkinter import *      1행 : tkinter는 파이썬에서 GUI 관련 모듈  
2  
3 window = Tk()             2행 : Tk() 클래스 객체 생성 (기본이 되는  
4  
5 ## 이 부분에서 화면을 구성하고 처리 ##  
6  
7 window.mainloop()          3행 : mainloop() 메서드 실행
```

3행 : Tk() 클래스 객체 생성 (기본이 되는
윈도를 반환, 이를 루트 윈도우 또는 베이
스 윈도우라고 함). 3행 실행하면 윈도창
화면에 출력

↑
window 객체의 mainloop() 메서드 실행

mainloop()는 이벤트 메시지 루프로서 키보드나 마우스 혹은 화면 Redraw와 같은 다양
한 이벤트로부터 오는 메시지를 받고 전달하는 역할을 한다.



```
import tkinter  
window=tkinter.Tk()  
window.mainloop()
```

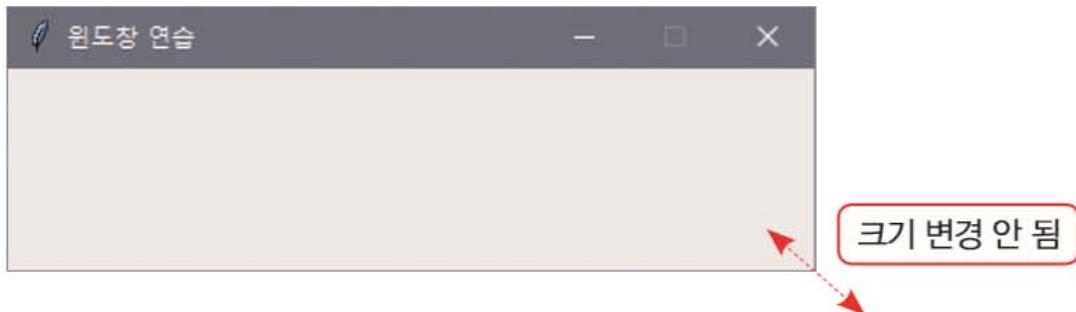
Section02 기본 위젯 활용

■ 윈도창 조절

Code10-02.py

```
1 from tkinter import *
2
3 window = Tk()
4 window.title("윈도창 연습")
5 window.geometry("400x100")
6 window.resizable(width = FALSE, height = FALSE)
7
8 window.mainloop()
```

4행 : 윈도창에 제목 표시
5행 : 윈도창의 초기 크기를 400×100으로 지정
6행 : 가로와 세로의 크기가 변경되지 않도록 지정



- Tk 객체명.geometry("너비x높이+x좌표+y좌표") : 윈도우 창의 너비와 높이, 초기 화면 위치의 x좌표와 y좌표를 설정
- Tk 객체명.resizeable(상하, 좌우) : 윈도우 창의 창 크기 조절 가능 여부를 설정함. True로 설정할 경우 윈도우 창의 크기를 조절함
- Tip : resizable()을 적용할 때, True=1, False=0을 의미하여 상수를 입력해도 적용이 가능

Section02 기본 위젯 활용

■ Tkinter 위젯

- Tkinter는 제한된 핵심 위젯들만을 제공

위젯	설명
Button	단순한 버튼
Label	텍스트 혹은 이미지 표시
CheckButton	체크박스
Entry	단순한 한 라인 텍스트 박스
ListBox	리스트 박스
RadioButton	옵션 버튼
Message	Label과 비슷하게 텍스트 표시. Label과 달리 자동 래핑 기능이 있다.
Scale	슬라이스 바
Scrollbar	스크롤 바
Text	멀티 라인 텍스트 박스로서 일부 Rich Text 기능 제공
Menu	메뉴 Pane
Menubutton	메뉴 버튼
Toplevel	새 윈도우를 생성할 때 사용. Tk()는 윈도우를 자동으로 생성하지만 추가로 새 윈도우 혹은 디얼로그를 만들 경우 Toplevel을 사용한다
Frame	컨테이너 위젯. 다른 위젯들을 그룹화할 때 사용
Canvas	그래프와 점들로 그림을 그릴 수 있으며, 커스텀 위젯을 만드는데 사용될 수도 있다

[출처] <http://pythonstudy.xyz/python/article/121-Tkinter-위젯>

- 위젯은 객체를 생성하여 필요한 속성들을 지정하여 사용한다.
- 위젯은 부모 컨테이너와 연관하여 어떤 상대적 위치에 놓이게 되는데, Geometry Manager를 사용하여 각 위젯의 위치를 정하게 된다.

Section02 기본 위젯 활용

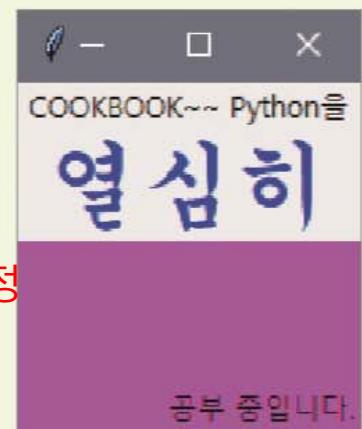
■ Label

- 텍스트 문자열이나 그림을 표시하는 위젯

Code10-03.py

```
1 from tkinter import *
2 window = Tk()
3
4 label1 = Label(window, text = "COOKBOOK~~ Python을")
5 label2 = Label(window, text = "열심히", font = ("궁서체", 30), fg = "blue")
6 label3 = Label(window, text = "공부 중입니다.", bg = "magenta", width = 20, height = 5,
7               anchor = SE)
8 label1.pack()
9 label2.pack()
10 label3.pack()
11
12 window.mainloop()
```

- 4행 : Label() 함수가 Label을 만들어 줌
- 8행 : pack() 함수로 해당 Label을 화면에 표시
- font : 글꼴과 크기 지정
- fg : foreground의 약어로 글자색을 지정
- bg : background의 약어로 배경색을 지정
- width/height : 위젯의 폭과 높이를 지정
- Anchor : 위젯이 어느 위치에 자리 잡을지 결정



- tkinter.Label(윈도우 창, 파라미터1, 파라미터2, 파라미터3, ...)을 사용하여 해당 윈도우 창에 표시 할 라벨의 속성을 설정
- pack() : 위젯 위치를 적당한 곳에 위치시킴 (auto)

Section02 기본 위젯 활용

▶ Label 파라미터

[출처] <https://076923.github.io/posts/Python-tkinter-2/>

라벨 문자열 설정

이름	의미	기본값	속성
text	라벨에 표시할 문자열	-	-
textvariable	라벨에 표시할 문자열을 가져올 변수	-	-
anchor	라벨안의 문자열 또는 이미지의 위치	center	n, ne, e, se, s, sw, w, nw, center
justify	라벨의 문자열이 여러 줄 일 경우 정렬 방법	center	center, left, right
wraplength	자동 줄내림 설정 너비	0	상수

라벨 형태 설정

이름	의미	기본값	속성
width	라벨의 너비	0	상수
height	라벨의 높이	0	상수
relief	라벨의 테두리 모양	flat	flat, groove, raised, ridge, solid, sunken
borderwidth=bd	라벨의 테두리 두께	2	상수
background=bg	라벨의 배경 색상	SystemButtonFace	color
foreground=fg	라벨의 문자열 색상	SystemButtonFace	color
padx	라벨의 테두리와 내용의 가로 여백	1	상수
pady	라벨의 테두리와 내용의 세로 여백	1	상수

라벨 형식 설정

이름	의미	기본값	속성
bitmap	라벨에 포함할 기본 이미지	-	info, warning, error, question, questhead, hourglass, gray12, gray25, gray50, gray75
image	라벨에 포함할 임의 이미지	-	-
compound	라벨에 문자열과 이미지를 동시에 표시할 때 이미지의 위치	none	bottom, center, left, none, right, top
font	라벨의 문자열 글꼴 설정	TkDefaultFont	font
cursor	라벨의 마우스 커서 모양	-	커서 속성

라벨 상태 설정

이름	의미	기본값	속성
state	상태 설정	normal	normal, active, disabled
activebackground	active 상태일 때 라벨의 배경 색상	SystemButtonFace	color
activeforeground	active 상태일 때 라벨의 문자열 색상	SystemButtonText	color
disabledforeground	disabled 상태일 때 라벨의 문자열 색상	SystemDisabledText	color

라벨 하이라이트 설정

이름	의미	기본값	속성
highlightcolor	라벨이 선택되었을 때 색상	SystemWindowFrame	color
highlightbackground	라벨이 선택되지 않았을 때 색상	SystemButtonFace	color
highlightthickness	라벨이 선택되었을 때 두께 (두께 설정)	0	상수

Section02 기본 위젯 활용

[출처] <https://076923.github.io/posts/Python-tkinter-10/>

▶ pack 함수

- 위젯이름.pack(파라미터1, 파라미터2, 파라미터3, ...) : 해당 윈도우 창에 표시할 위젯의 배치 속성을
- 배치되는 우선 순위는 가장 처음 선언한 pack부터 배치됨
- pack의 파라미터로 인하여 위젯의 크기가 변경됨

pack Parameter

이름	의미	기본값	속성
side	특정 위치로 공간 할당	top	top, bottom, left, right
anchor	할당된 공간 내에서 위치 지정	center	center, n, e, s, w, ne, nw, se, sw
fill	할당된 공간에 대한 크기 맞춤	none	none, x, y, both
expand	미사용 공간 확보	False	Boolean
ipadx	위젯에 대한 x 방향 내부 패딩	0	상수
ipady	위젯에 대한 y 방향 내부 패딩	0	상수
padx	위젯에 대한 x 방향 외부 패딩	0	상수
pady	위젯에 대한 y 방향 외부 패딩	0	상수

- side : 해당 구역으로 위젯을 이동
- anchor : 현재 배치된 구역 안에서 특정 위치로 이동
- fill : 할당된 공간에 맞게 크기가 변경
- expand : 할당되지 않은 미사용 공간을 모두 현재 위젯의 할당된 공간으로 변경

Section02 기본 위젯 활용

[출처] <https://076923.github.io/posts/Python-tkinter-10/>

▶ pack 함수

```
b1=tkinter.Button(window, text="top")
b1_1=tkinter.Button(window, text="top-1")

b2=tkinter.Button(window, text="bottom")
b2_1=tkinter.Button(window, text="bottom-1")

b3=tkinter.Button(window, text="left")
b3_1=tkinter.Button(window, text="left-1")

b4=tkinter.Button(window, text="right")
b4_1=tkinter.Button(window, text="right-1")

b5=tkinter.Button(window, text="center", bg="red")

b1.pack(side="top")
b1_1.pack(side="top", fill="x")

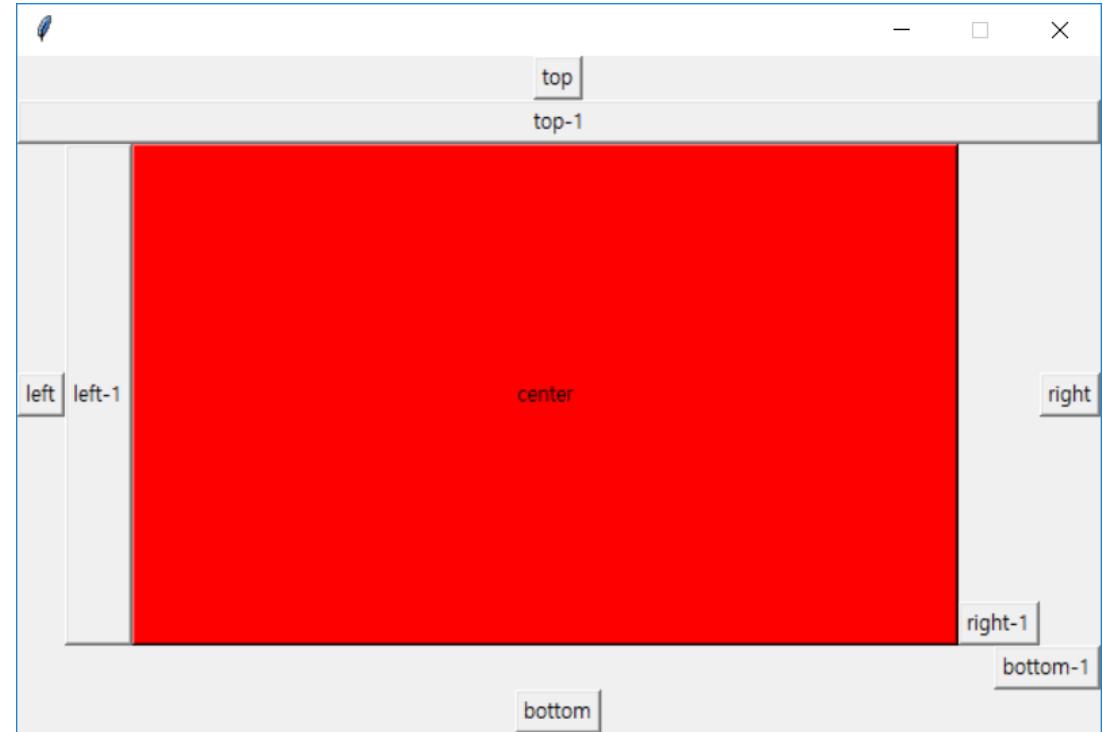
b2.pack(side="bottom")
b2_1.pack(side="bottom", anchor="e")

b3.pack(side="left")
b3_1.pack(side="left", fill="y")

b4.pack(side="right")
b4_1.pack(side="right", anchor="s")

b5.pack(expand=True, fill="both")

window.mainloop()
```

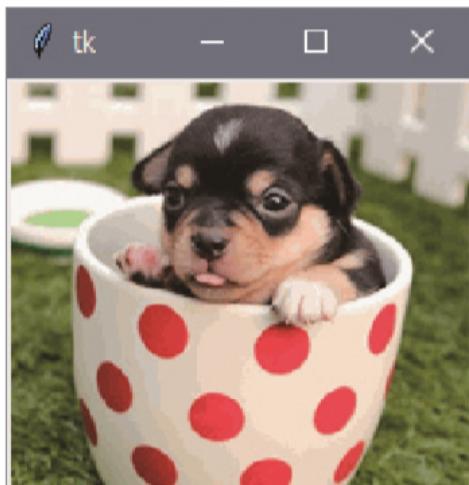


Section02 기본 위젯 활용

- Label에 글자 대신 이미지 넣기
 - PhotoImage()는 GIF 파일만 지원, JPEG나 BMP 등은 지원하지 않음

Code10-04.py

```
1 from tkinter import *
2 window = Tk()
3
4 photo = PhotoImage(file = "gif/dog.gif") 4행 : PhotoImage()에서 이미지 파일을 준비
5 label1 = Label(window, image = photo)      5행 : Label()의 옵션 image에서 속성을 지정
6
7 label1.pack()
8
9 window.mainloop()
```



Section02 기본 위젯 활용

SELF STUDY 10-1

Code10-04.py를 수정해서 이미지를 2개 출력해 보자.

힌트 위젯을 가로로 나타내려면 pack(side=LEFT)를 사용한다.



```
from tkinter import *
window = Tk()
# title은 생략
photo1 = PhotoImage(file = "gif/cat.gif")
photo2 = PhotoImage(file = "gif/cat2.gif")
label1 = Label(window, image = photo1)
label2 = Label(window, image = photo2)
label1.pack(side=LEFT);
label2.pack(side=RIGHT);
window.mainloop()
```

Section02 기본 위젯 활용

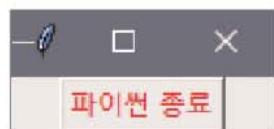
■ Button

- 마우스로 클릭하면 눌리는 효과와 함께 지정한 작업 실행
- 예: 버튼을 누르면 파이썬 IDLE이 종료되는 코드

Code10-05.py

```
1 from tkinter import *
2 window = Tk()
3
4 button1 = Button(window, text = "파이썬 종료", fg = "red", command = quit)
5
6 button1.pack()
7
8 window.mainloop()
```

4행 : command 옵션에 quit 명령



- command : 버튼이 active 상태일 때 실행하는 메소드(함수)

- tkinter.Button(윈도우 창, 파라미터1, 파라미터2, 파라미터3, ...) : 해당 윈도우 창에 표시할 버튼의 속성을 설정
- 파라미터 중 command를 이용하여 사용자 정의 함수를 실행시킬 수 있음

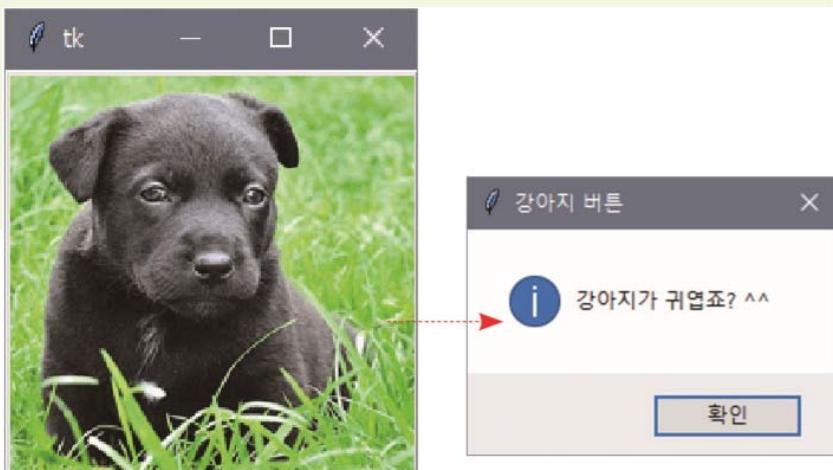
Section02 기본 위젯 활용

- 예: 이미지 버튼을 누르면 간단한 메시지창이 나오는 코드

Code10-06.py

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 ## 함수 선언 부분 ##
5 def myFunc() :
6     messagebox.showinfo("강아지 버튼", "강아지가 귀엽죠? ^^")
7
8 ## 메인 코드 부분 ##
9 window = Tk()
10
11 photo = PhotoImage(file = "gif/dog2.gif")
12 button1 = Button(window, image = photo, command = myFunc )
13
14 button1.pack()
15
16 window.mainloop()
```

2행 : messagebox 모듈 사용 임포트
5~6행 : myFunc() 함수 정의, 이 함수는 messagebox.showinfo(제목, 내용)
형식으로 화면에 메시지 상자 출력
11~12행 : 이미지 준비하고 버튼에 글자 대신 이미지 표현



Section02 기본 위젯 활용

▶ Button 파라미터

[출처] <https://076923.github.io/posts/Python-tkinter-2/>

Button Method

이름	의미
invoke()	버튼 실행
flash()	깜빡임

- Tip : `invoke()` : 버튼을 클릭했을 때와 동일한 실행
- Tip : `flash()` : `normal` 상태 배경 색상과 `active` 상태 배경 색상 사이에서 깜빡임

버튼 형태 설정

이름	의미	기본값	속성
width	버튼의 너비	0	상수
height	버튼의 높이	0	상수
relief	버튼의 테두리 모양	flat	flat, groove, raised, ridge, solid, sunken
overrelief	버튼에 마우스를 올렸을 때 버튼의 테두리 모양	raised	flat, groove, raised, ridge, solid, sunken
borderwidth=bd	버튼의 테두리 두께	2	상수
background=bg	버튼의 배경 색상	SystemButtonFace	color
foreground=fg	버튼의 문자열 색상	SystemButtonFace	color
padx	버튼의 테두리와 내용의 가로 여백	1	상수
pady	버튼의 테두리와 내용의 세로 여백	1	상수

버튼 문자열 설정

이름	의미	기본값	속성
text	버튼에 표시할 문자열	-	-
textvariable	버튼에 표시할 문자열을 가져올 변수	-	-
anchor	버튼안의 문자열 또는 이미지의 위치	center	n, ne, e, se, s, sw, w, nw, center
justify	버튼의 문자열이 여러 줄 일 경우 정렬 방법	center	center, left, right
wraplength	자동 줄내림 설정 너비	0	상수

버튼 상태 설정

이름	의미	기본값	속성
state	상태 설정	normal	normal, active, disabled
activebackground	active 상태일 때 버튼의 배경 색상	SystemButtonFace	color
activeforeground	active 상태일 때 버튼의 문자열 색상	SystemButtonText	color
disabledforeground	disabled 상태일 때 버튼의 문자열 색상	SystemDisabledText	color

Section02 기본 위젯 활용

➤ Button 파라미터

[출처] <https://076923.github.io/posts/Python-tkinter-2/>

버튼 동작 설정

이름	의미	기본값	속성
takefocus	Tab 키를 이용하여 위젯 이동 허용 여부	True	Boolean
command	버튼이 active 상태일 때 실행하는 메소드(함수)	-	메소드, 함수
repeatdelay	버튼이 눌러진 상태에서 command 실행까지의 대기 시간	0	상수(ms)
repeatinterval	버튼이 눌러진 상태에서 command 실행의 반복 시간	0	상수(ms)

버튼 형식 설정

이름	의미	기본값	속성
bitmap	버튼에 포함할 기본 이미지	-	info, warning, error, question, questhead, hourglass, gray12, gray25, gray50, gray75
image	버튼에 포함할 임의 이미지	-	-
compound	버튼에 문자열과 이미지를 동시에 표시할 때 이미지의 위치	none	bottom, center, left, none, right, top
font	버튼의 문자열 글꼴 설정	TkDefaultFont	font
cursor	버튼의 마우스 커서 모양	-	커서 속성

버튼 하이라이트 설정

이름	의미	기본값	속성
highlightcolor	버튼이 선택되었을 때 색상	SystemWindowFrame	color
highlightbackground	버튼이 선택되지 않았을 때 색상	SystemButtonFace	color
highlightthickness	버튼이 선택되었을 때 두께 (두께 설정)	0	상수

Section02 기본 위젯 활용

■ CheckButton

- 켜고 끄는 데 사용하는 위젯
- 예 : 체크버튼을 켜거나 끄면 메시지창이 열리게 하느니

Code10-07.py

```
1 from tkinter import *
2 from tkinter import messagebox
3 window = Tk()
4
5 ## 함수 선언 부분 ##
6 def myFunc() :
7     if chk.get() == 0 :
8         messagebox.showinfo("", "체크버튼이 꺼졌어요.")
9     else :
10        messagebox.showinfo("", "체크버튼이 켜졌어요.")
11
12 ## 메인 코드 부분 ##
13 chk = IntVar()
14 cb1 = Checkbutton(window, text = "클릭하세요", variable = chk, command = myFunc)
15
16 cb1.pack()
17
18 window.mainloop()
```

6~10행 : myFunc() 함수는 chk.get() 함수, 체크버튼에 설정된 값을 가져와서 메시지 출력

13행 : chk 변수를 준비, IntVar() 함수 사용
14행 : Checkbutton()의 옵션 중 variable에 chk 변수 사용

- IntVar() : 정수형 타입의 변수 생성
- tkinter.IntVar()에 저장된 value 값은 변수 이름 .get()을 통하여 불러올 수 있음



Section02 기본 위젯 활용

➤ CheckButton 파라미터

[출처] <https://076923.github.io/posts/Python-tkinter-6/>

체크버튼 동작 설정

이름	의미	기본값	속성
takefocus	Tab 키를 이용하여 위젯 이동 허용 여부	True	Boolean
command	체크버튼이 active 상태일 때 실행하는 메소드(함수)	-	메소드, 함수
variable	체크버튼의 상태를 저장할 제어 변수	-	tkinter.IntVar()
onvalue	체크버튼이 체크 상태일 때 연결된 제어 변수의 값	True	Boolean
offvalue	체크버튼이 해제 상태일 때 연결된 제어 변수의 값	False	Boolean
indicatoron	체크버튼의 위젯 일치화 여부	True	Boolean

- 체크버튼 문자열 설정, 형태 설정, 형식 설정, 상태 설정, 하이라이트 설정 등은 출처 참조

Section02 기본 위젯 활용

■ RadioButton

- 여러 개 중에서 하나를 선택하는 데 사용하는 위젯

Code10-08.py

```
1 from tkinter import *
2 window = Tk()
3
4 ## 함수 선언 부분 ##
5 def myFunc() :
6     if var.get() == 1 :
7         label1.configure(text = "파이썬")
8     elif var.get() == 2 :
9         label1.configure(text = "C++")
10    else :
11        label1.configure(text = "Java")
12
13 ## 메인 코드 부분 ##
14 var = IntVar()
15 rb1 = Radiobutton(window, text = "파이썬", variable = var, value = 1, command = myFunc)
16 rb2 = Radiobutton(window, text = "C++", variable = var, value = 2, command = myFunc)
17 rb3 = Radiobutton(window, text = "Java", variable = var, value = 3, command = myFunc)
```

5~11행 : myFunc() 함수는 var 변수값으로 맨 아래쪽
레이블의 텍스트를 변경
14행 : 정수형 변수 var를 준비
15~17행 : 라디오버튼을 3개 준비

Section02 기본 위젯 활용

```
18  
19  label1 = Label(window, text = "선택한 언어 : ", fg = "red")  
20  
21  rb1.pack()  
22  rb2.pack()  
23  rb3.pack()  
24  label1.pack()  
25  
26  window.mainloop()
```



- 파라미터 중 **command**를 이용하여 사용자 정의 함수 : myFunc() 함수를 실행시킴
- 파라미터 중 **value**를 이용하여 라디오버튼의 값을 설정할 수 있음. value의 값이 겹치는 경우 같이 선택됨.
- 파라미터 중 **variable**를 이용하여 tkinter.IntVar()의 그룹이 같을 경우 하나의 묶음으로 간주하며 'value'의 값이 저장됩니다

Section02 기본 위젯 활용

➤ RadioButton 파라미터

[출처] <https://076923.github.io/posts/Python-tkinter-7/>

라디오버튼 동작 설정

이름	의미	기본값	속성
takefocus	Tab 키를 이용하여 위젯 이동 허용 여부	True	Boolean
command	라디오버튼이 active 상태일 때 실행하는 메소드(함수)	-	메소드, 함수
variable	라디오버튼의 상태를 저장할 제어 변수	-	<code>tkinter.IntVar()</code> , <code>tkinter.StringVar()</code>
value	라디오버튼이 가지고 있는 값	-	<code>variable</code> 의 속성
indicatoron	라디오버튼의 위젯 일치화 여부	True	Boolean

- 라디오버튼 문자열 설정, 형태 설정, 형식 설정, 상태 설정, 하이라이트 설정 등은 출처 참조

Section03 위젯의 배치와 크기 조절

■ 수평 정렬

- 수평으로 정렬하는 방법 : pack() 함수의 옵션 중 side=LEFT, RIGHT

Code10-09.py

```
1 from tkinter import *
2 window = Tk()
3
4 button1 = Button(window, text = "버튼1")
5 button2 = Button(window, text = "버튼2")
6 button3 = Button(window, text = "버튼3")
7
8 button1.pack( side = LEFT )      8~10행 : side=LEFT 옵션은 왼쪽부터 채우라는 의미
9 button2.pack( side = LEFT )
10 button3.pack( side = LEFT )
11
12 window.mainloop()
```



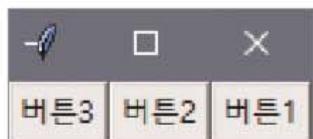
Section03 위젯의 배치와 크기 조절

- 리스트와 for 문 활용

Code10-10.py

```
1 from tkinter import *
2 window = Tk()
3
4 btnList = [None] * 3
5
6 for i in range(0, 3) :
7     btnList[i] = Button(window, text = "버튼" + str(i + 1))
8
9 for btn in btnList :
10    btn.pack( side = RIGHT )
11
12 window.mainloop()
```

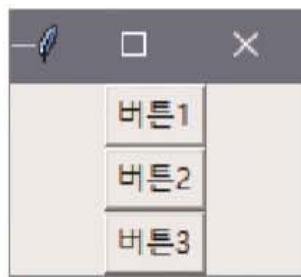
4행 : 비어 있는 리스트를 3개 준비
6~7행 : 3회 반복하면서 버튼 생성
9~10행 : 준비한 버튼 리스트를 화면에 출력



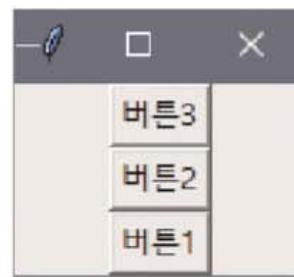
Section03 위젯의 배치와 크기 조절

■ 수직 정렬

- pack() 함수의 옵션 중 수직으로 정렬하는 방법 : side=TOP, BOTTOM
 - Code10-10.py 의 10행을 다음과 같이 변경하고 실행



btn.pack(side = TOP)



btn.pack(side = BOTTOM)

Section03 위젯의 배치와 크기 조절

■ 폭 조정

- pack() 함수의 옵션 중에서 윈도창 폭에 맞추는 방법 : `fill=X`이다.
 - Code10-10.py의 10행을 다음과 같이 변경하고 실행

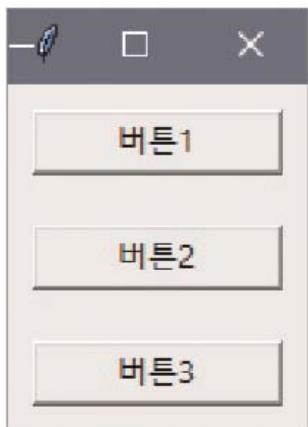


```
btn.pack(side = TOP, fill = X)
```

Section03 위젯의 배치와 크기 조절

■ 위젯 사이의 여백 조절

- pack() 함수의 옵션 중에서 위젯 사이에 여백 주는 방법 : padx=픽셀값 또는 pady=픽셀값
 - Code10-10.py의 10행을 다음과 같이 변경하고 실행

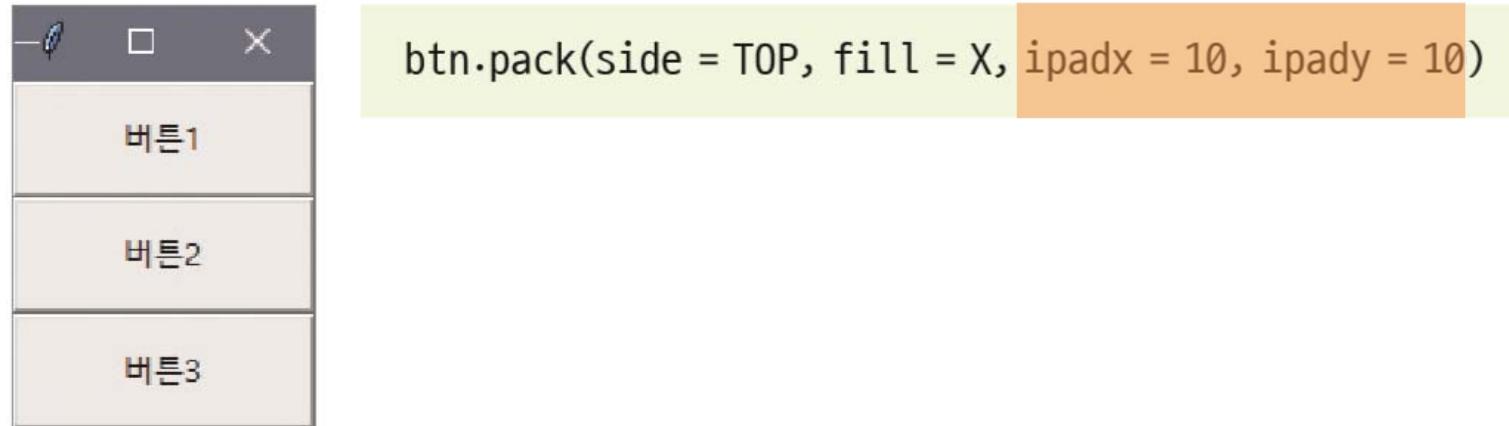


```
btn.pack(side = TOP, fill = X, padx = 10, pady = 10)
```

Section03 위젯의 배치와 크기 조절

■ 위젯 내부의 여백 조절

- pack() 함수의 옵션 중에서 위젯 내부에 여백 주는 방법 : ipadx=픽셀값 또는 ipady=픽셀값
 - Code10-10.py의 10행을 다음과 같이 변경하고 실행



Section03 위젯의 배치와 크기 조절

■ 위젯 내부의 여백 조절

- 위젯 내부와 외부에 모두 여백



```
btn.pack(side = TOP, fill = X, ipadx = 10, ipady = 10, padx = 10, pady = 10)
```

Section03 위젯의 배치와 크기 조절

■ 고정 위치에 배치

- 위젯을 고정 위치에 배치하려면 pack() 대신 **place()** 함수 사용
- 그림 9개를 2차원으로 배치하는 코드

Code10-11.py

```
1 from tkinter import *
2
3 ## 전역 변수 선언 부분 ##
4 btnList = [None] * 9
5 fnameList = ["froyo.gif", "gingerbread.gif", "honeycomb.gif", "icecream.gif",
6               "jellybean.gif", "kitkat.gif", "lollipop.gif", "marshmallow.gif", "nougat.gif"]
7 photoList = [None] * 9
8 i, k = 0, 0
9 xPos, yPos = 0, 0
10 num = 0
11 ## 메인 코드 부분 ##
12 window = Tk()
13 window.geometry("210x210")
14
```

- 4행 : 버튼을 저장할 9개짜리 리스트 준비
- 5행 : 이미지 파일명 9개를 리스트로 준비
- 6행 : PhotoImage()로 생성할 9개짜리 리스트
- 8행 : xPos, yPos는 그림을 출력할 X, Y 좌표로 사용
- 9행 : num 은 그림의 순차 번호로 사용

Section03 위젯의 배치와 크기 조절

```
15  for i in range(0, 9) :  
16      photoList[i] = PhotoImage(file = "gif/" + fnameList[i])  
17      btnList[i] = Button(window, image = photoList[i])  
18  
19  for i in range(0, 3) :          19~25행 : 3×3=9번 반복  
20      for k in range(0, 3) :  
21          btnList[num].place(x = xPos, y = yPos)  
22          num += 1  
23          xPos += 70  
24          xPos = 0  
25          yPos += 70  
26  
27 window.mainloop()
```



Section03 위젯의 배치와 크기 조절

SELF STUDY 10-2

Code10-11.py를 실행할 때마다 그림을 임의로 뒤섞어서 나타내게 하자.

힌트 random 모듈을 임포트하고 shuffle(리스트) 함수를 사용하면 리스트를 임의로 뒤섞어 준다.



```
from random import *
fnameList = ["honeycomb.gif", "icecream.gif", "jellybean.gif", "kitkat.gif", "lollipop.gif",
"marshmallow.gif", "nougat.gif", "oreo.gif", "pie.gif"]
shuffle(fnameList)
```

Section03 위젯의 배치와 크기 조절

■ [프로그램 1]의 완성

- <이전> 버튼이나 <다음> 버튼을 누르면 사진들을 표시하는 사진 앨범 프로그램

Code10-12.py

```
1 from tkinter import *
2 from time import *
3
4 ## 전역 변수 선언 부분 ##
5 fnameList = ["jeju1.gif", "jeju2.gif", "jeju3.gif", "jeju4.gif", "jeju5.gif",
               "jeju6.gif", "jeju7.gif", "jeju8.gif", "jeju9.gif"]
6 photoList = [None] * 9
7 num = 0
8
9 ## 함수 선언 부분 ##
10 def clickNext() :
11     global num
12     num += 1
```

- 5행 : fnameList 변수에 사진 9장의 파일명을 저장
- 6행 : photoList에는 PhotoImage() 함수로 생성할 변수 9개 준비
- 7행 : num은 현재 사진의 번호
- 10~17행 : <다음> 버튼을 누르면 실행되는 함수
- 11행 : global num은 num 전역 변수를 함수 안에서 사용 의미
- 12행 : 사진 번호를 하나 증가

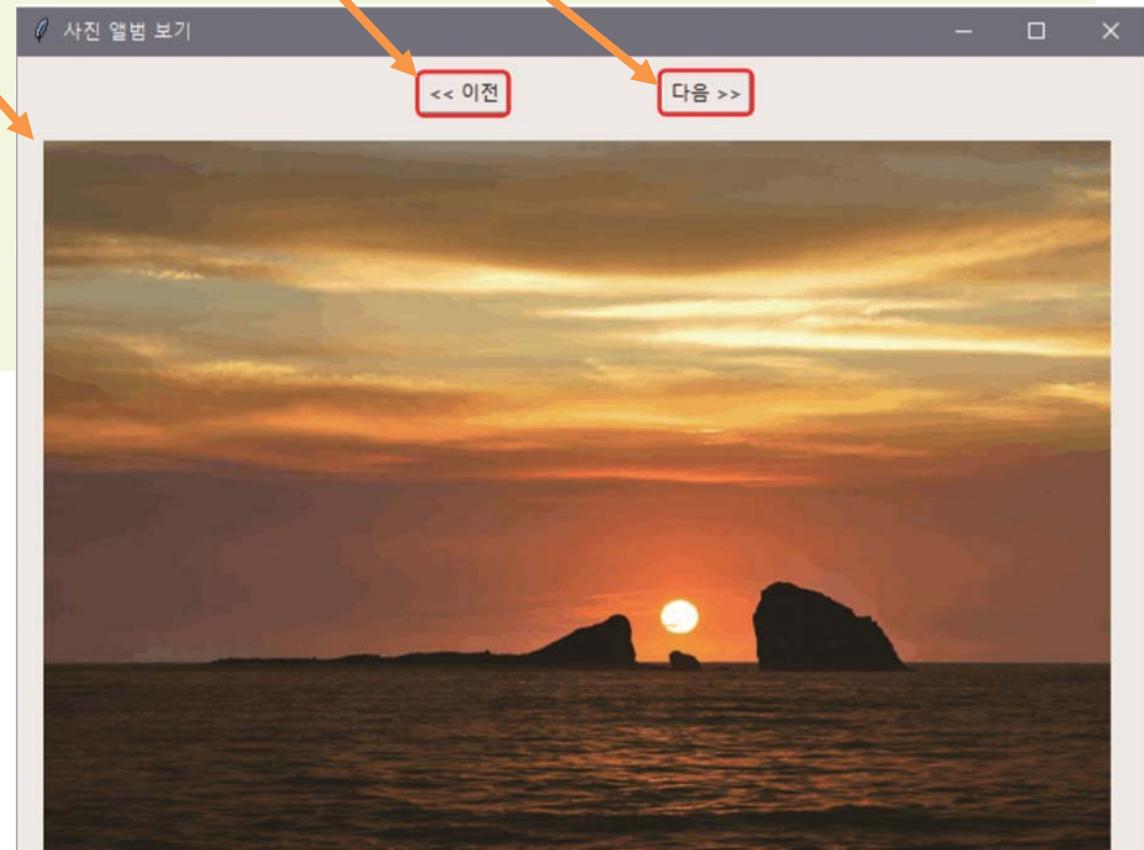
Section03 위젯의 배치와 크기 조절

```
13     if num > 8 :          • 13~14행 : 사진 번호가 최대 8, 8 넘으면 0번 사진 표시
14         num = 0
15
16         photo = PhotoImage(file = "gif/" + fnameList[num])
17         pLabel.configure(image = photo)
18         pLabel.image = photo
19
20 def clickPrev() : • 19~26행 : <이전> 버튼을 누르면 처리되는 함수
21     global num
22     num -= 1
23     if num < 0 :          • 변경된 사진 번호에 해당하는 이미지 파일로 pLabel 변경
24         num = 8            • pLabel은 main 코드에서 정의
25
26         photo = PhotoImage(file = "gif/" + fnameList[num])
27         pLabel.configure(image = photo)
28         pLabel.image = photo
29
30 ## 메인 코드 부분 ##
31 window = Tk()          • tkinter.PhotoImage : image objects (for images in PGM, PPM,
32 window.geometry("700x500")      GIF and PNG formats) 반환 -> photo로 image object 받음
33 window.title("사진 앨범 보기")    • pLabel에 있는 image를 photo로 변경
```

Section03 위젯의 배치와 크기 조절

```
32  
33 btnPrev = Button(window, text = "<< 이전", command = clickPrev) • 버튼 누르면 이 함수와 연결  
34 btnNext = Button(window, text = "다음 >>", command = clickNext)  
35  
36 photo = PhotoImage(file = "gif/" + fnameList[0]) • 프로그램 실행 후 첫 번째 사진 표시  
37 pLabel = Label(window, image = photo)  
38  
39 btnPrev.place(x = 250, y = 10)  
40 btnNext.place(x = 400, y = 10)  
41 pLabel.place(x = 15, y = 50)  
42  
43 window.mainloop()
```

39~41행 : 버튼 및 이미지 위치
place(x=좌표, y=좌표)로 지정



Section03 위젯의 배치와 크기 조절

SELF STUDY 10-3

Code10-12.py를 수정해서 버튼 사이에 파일명을 표시해 보자.



main 코드 부분

```
nameLabel = Label(window, text=fnameList[0])  
nameLabel.place(x=330, y=10)
```

clickNext, clickPrev 함수 내

```
nameLabel.configure(text=fnameList[num])
```

참고 Tkinter PhotoImage

The Tkinter PhotoImage Class

[출처] <http://effbot.org/tkinterbook/photoimage.htm>

- The PhotoImage class is used to display images (either grayscale or true color images) in **labels, buttons, canvases, and text widgets**.
- The PhotoImage class can read **GIF** and **PGM/PPM** images from files:

```
photo = PhotoImage(file="image.gif")
photo = PhotoImage(file="lenna.pgm")
```

- If you need to work with other file formats (대표적으로 jpg 파일), the **Python Imaging Library (PIL)** contains classes that lets you load images in over 30 formats, and convert them to Tkinter-compatible image objects:

```
from PIL import Image, ImageTk
image = Image.open("lenna.jpg")
photo = ImageTk.PhotoImage(image)
```

- You can use a PhotoImage instance everywhere Tkinter accepts an image object. An example:

```
label = Label(image=photo)
label.image = photo # keep a reference!
label.pack()
```

- ❖ You must keep a reference to the image object in your Python program, either by storing it in a global variable, or by attaching it to another object.

Section04 키보드와 마우스 이벤트 처리

■ 마우스 이벤트 기본 처리

표 10-1 마우스 이벤트

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

Section04 키보드와 마우스 이벤트 처리

- 마우스 이벤트가 처리 형식

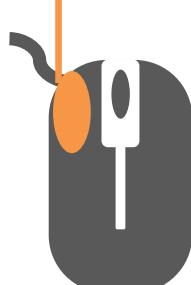
```
def 이벤트처리함수(event) :  
    # 이 부분에 마우스 이벤트가 발생할 때 작동할 내용 작성
```

```
위젯.bind("마우스이벤트", 이벤트처리함수)
```

“<이벤트코드>”

window.bind("<Button-1>", clickLeft)

왼쪽
버튼
클릭시



이벤트 코드
<Button-1>

이벤트 처리함수 실행
def clickLeft(event):

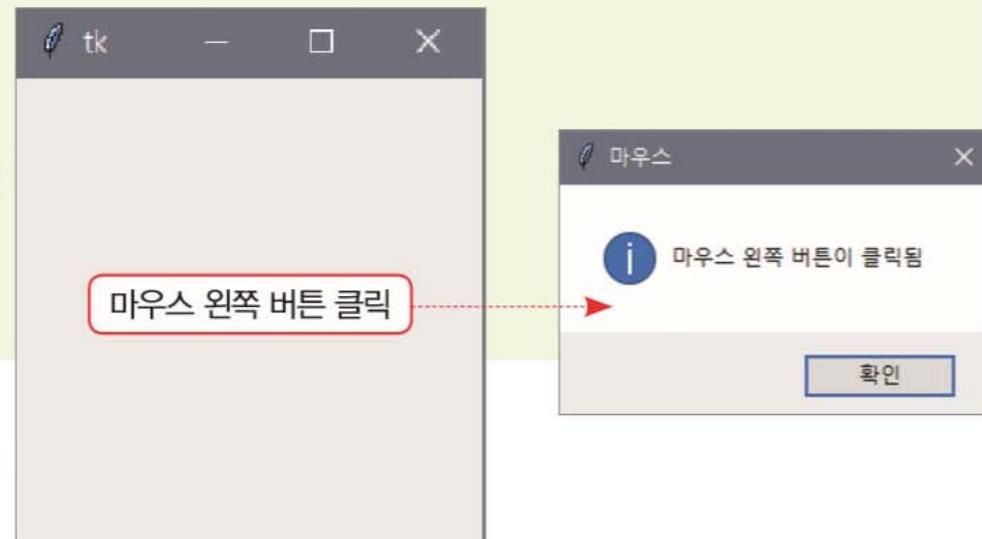
Section04 키보드와 마우스 이벤트 처리

- 마우스 왼쪽 버튼을 클릭했을 때 처리하는 방법

Code10-13.py

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 ## 함수 선언 부분 ##
5 def clickLeft(event) :
6     messagebox.showinfo("마우스", "마우스 왼쪽 버튼이 클릭됨")
7
8 ## 메인 코드 부분 ##
9 window = Tk()
10
11 window.bind("<Button-1>", clickLeft)
12
13 window.mainloop()
```

- 5~6행 : 마우스 이벤트가 발생할 때 작동할 함수 정의
- 11행 : window.bind() 함수에는 마우스 왼쪽 버튼 클릭할 때 발생하는 이벤트인 <Button-1> 설정하고 5행의 clickLeft 함수명을 지정



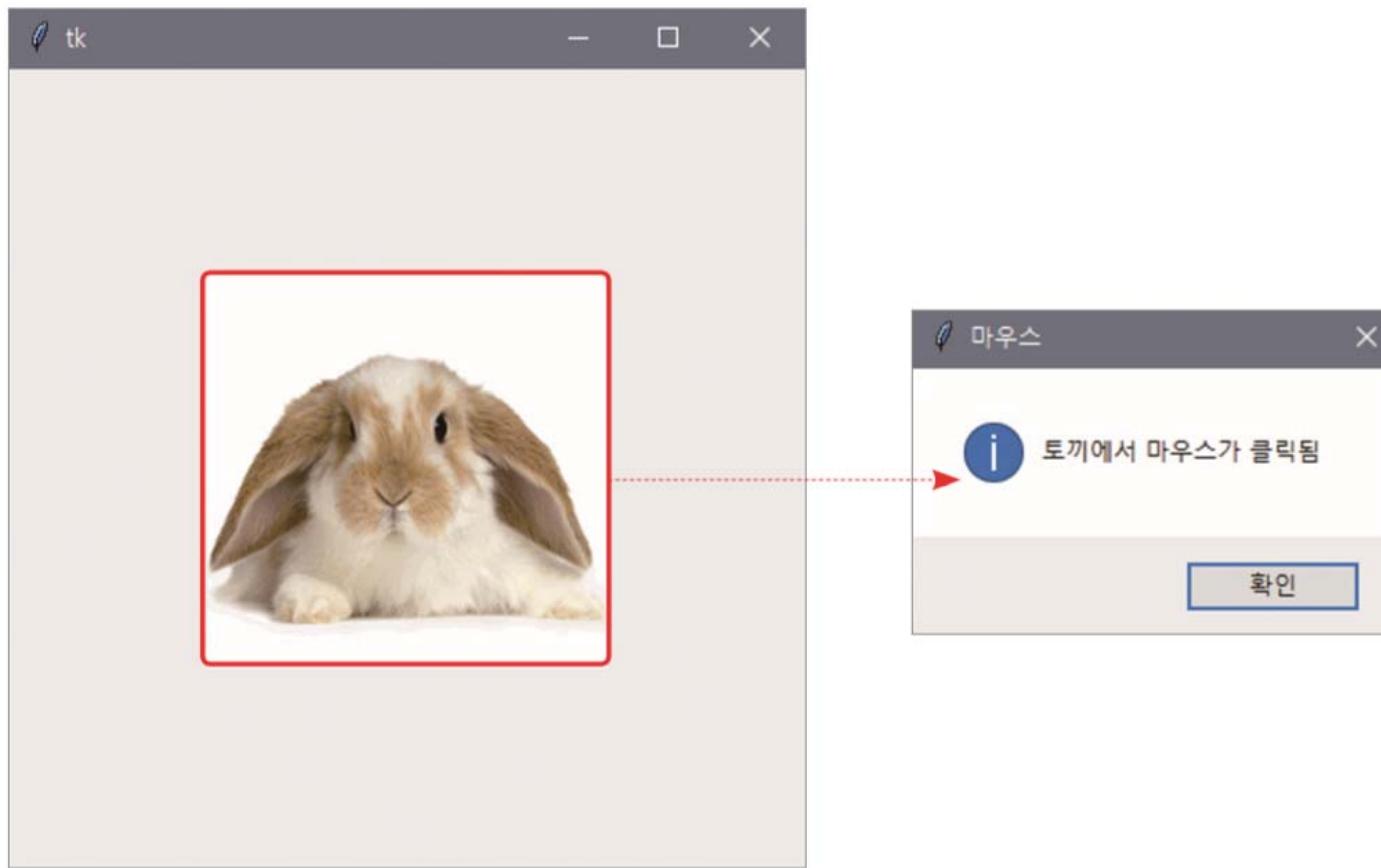
Section04 키보드와 마우스 이벤트 처리

- 지정된 위젯을 클릭했을 때 다른 함수 호출

Code10-14.py

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 ## 함수 선언 부분 ##
5 def clickImage(event) :
6     messagebox.showinfo("마우스", "토끼에서 마우스가 클릭됨")
7
8 ## 메인 코드 부분 ##
9 window = Tk()
10 window.geometry("400x400")
11
12 photo = PhotoImage(file = "gif/rabbit.gif")
13 label1 = Label(window, image = photo)
14
15 label1.bind("<Button>", clickImage)
16
17 label1.pack(expand = 1, anchor = CENTER)
18 window.mainloop()
```

Section04 키보드와 마우스 이벤트 처리



Code10-14.py는 이미지 클릭할 때만 이벤트 처리(15행에서 `window.bind()`가 아닌 `label1.bind()`를 사용했기 때문)
또 <Button> 이벤트를 사용했기 때문에 어떤 마우스 버튼 눌러도 메시지창 표시

Section04 키보드와 마우스 이벤트 처리

■ event 매개변수를 활용한 마우스 이벤트 처리

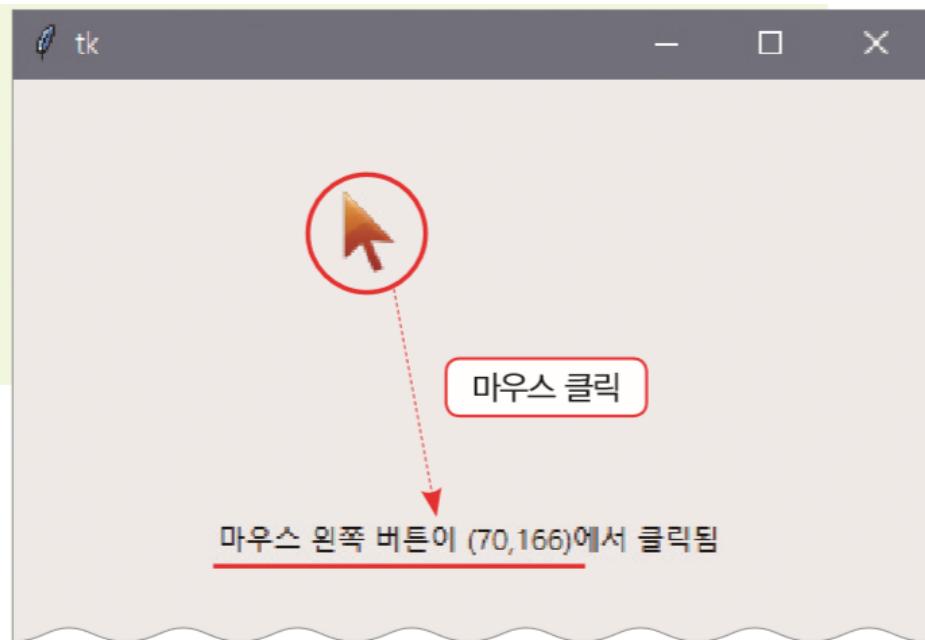
- 마우스를 클릭할 때마다 어떤 마우스가 클릭되었는지 보여 주고 클릭한 좌표 출력

Code10-15.py

```
1 from tkinter import *
2
3 ## 함수 선언 부분 ## • 마우스 클릭할 때 실행될 이벤트 함수 선언
4 def clickMouse( event ) :
5     txt = ""
6     if event.num == 1 :          • 6~9행 : event.num값은 마우스 왼쪽 버튼 클릭
7         txt += "마우스 왼쪽 버튼이 ("    했을 때 1값 가지고, 마우스 오른쪽 버튼 클릭했
8     elif event.num == 3 :          을 때 2값 가짐
9         txt += "마우스 오른쪽 버튼이 (" 11행 : event.x와 event.y는 클릭한 위치의 좌표를 가짐
10
11     txt += str( event.y ) + "," + str( event.x ) + ")에서 클릭됨"
12     label1.configure(text = txt)      • 12행 : 18행에서 화면에 표시한 레이블의 글자 변경
13
14 ## 메인 코드 부분 ##
15 window = Tk()
16 window.geometry("400x400")
```

Section04 키보드와 마우스 이벤트 처리

```
18 label1 = Label(window, text = "이곳이 바뀜")
19
20 window.bind("<Button>", clickMouse)
21
22 label1.pack(expand = 1, anchor = CENTER)
23 window.mainloop()
```



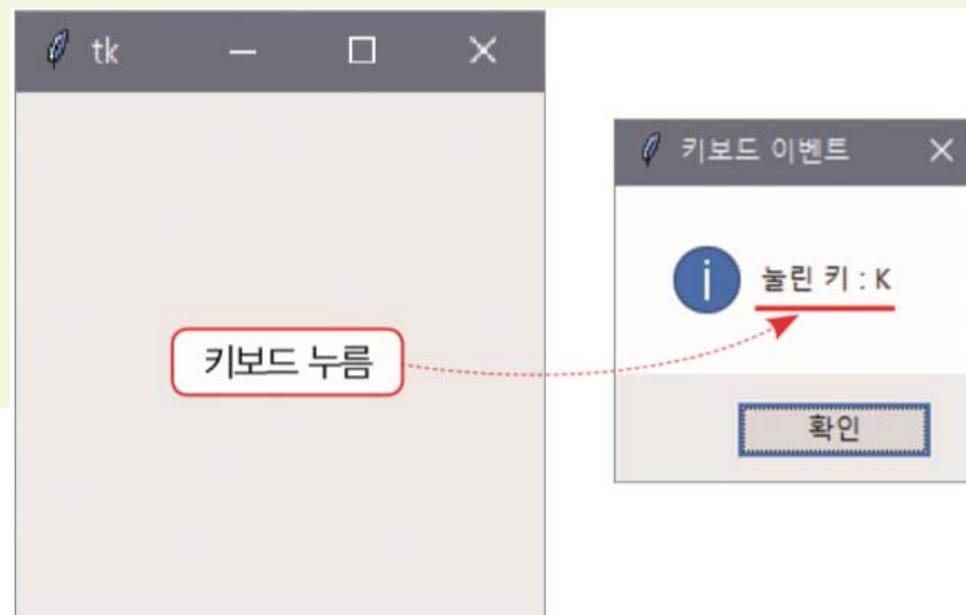
Section04 키보드와 마우스 이벤트 처리

■ 키보드 이벤트 기본 처리

- 키보드 이벤트는 위젯에서 키보드가 눌리면 발생

Code10-16.py

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 ## 함수 선언 부분 ##
5 def keyEvent(event) :
6     messagebox.showinfo("키보드 이벤트", "눌린 키 : " + chr( event.keycode))
7
8 ## 메인 코드 부분 ##
9 window = Tk()
10
11 window.bind( "<Key>" , keyEvent)
12
13 window.mainloop()
```



Section04 키보드와 마우스 이벤트 처리

■ 키보드 이벤트

표 10-2 키보드 이벤트

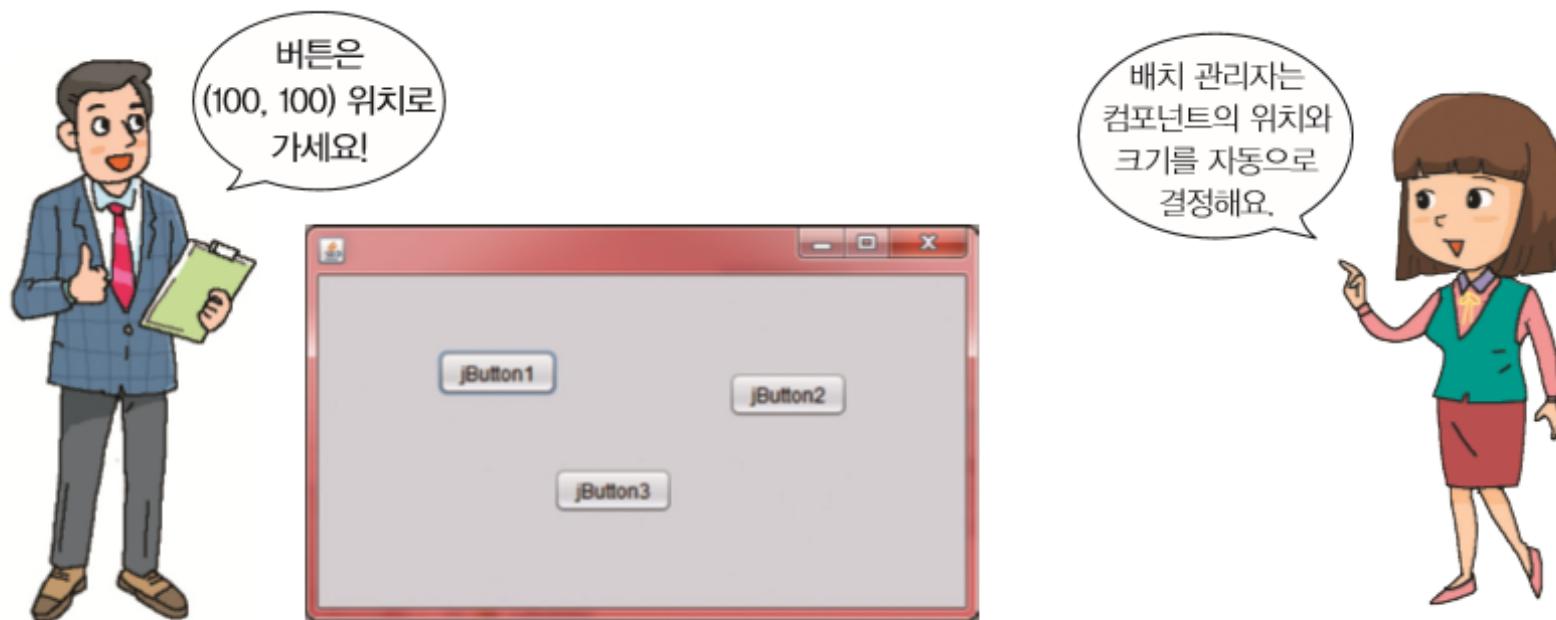
키보드 작동	이벤트 코드
모든 키를 누를 때	<code><Key></code>
특수 키를 누를 때	<code><Return></code> , <code><BackSpace></code> , <code><Tab></code> , <code><Shift_L></code> , <code><Control_L></code> , <code><Alt_L></code> , <code><Pause></code> , <code><Caps_Lock></code> , <code><Escape></code> , <code><End></code> , <code>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

배치 관리자

참조자료 두근두근 파이썬 10장 10장 tkinter로 GUI
만들기 강의자료

- 압축(pack) 배치 관리자
- 격자(grid) 배치 관리자
- 절대(place) 배치 관리자



격자 배치 관리자

참조자료: 두근두근 파이썬 10장 10장 tkinter로 GUI 만들기 강의자료

```
from tkinter import *

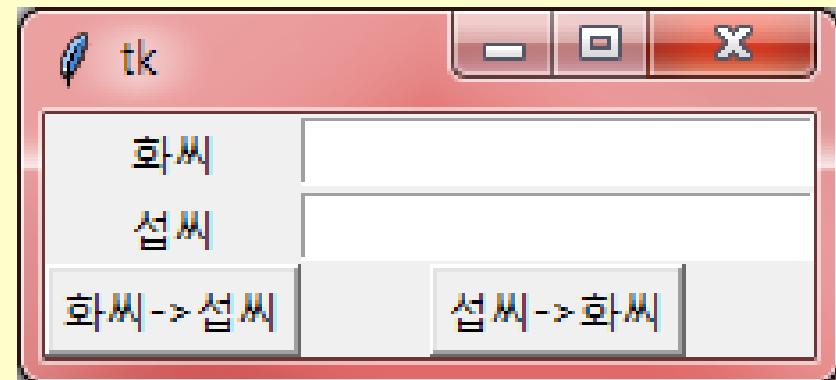
window = Tk()

l1 = Label(window , text="화씨")
l2 = Label(window, text="섭씨")
l1.grid(row=0, column=0)
l2.grid(row=1, column=0)

e1 = Entry(window)
e2 = Entry(window)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

b1 = Button(window, text="화씨->섭씨")
b2 = Button(window, text="섭씨->화씨")
b1.grid(row=2, column=0)
b2.grid(row=2, column=1)

window.mainloop()
```



```

from tkinter import *

def process():
    temperature = float(e1.get())
    mytemp = (temperature-32)*5/9
    e2.insert(0, str(mytemp))

window = Tk()

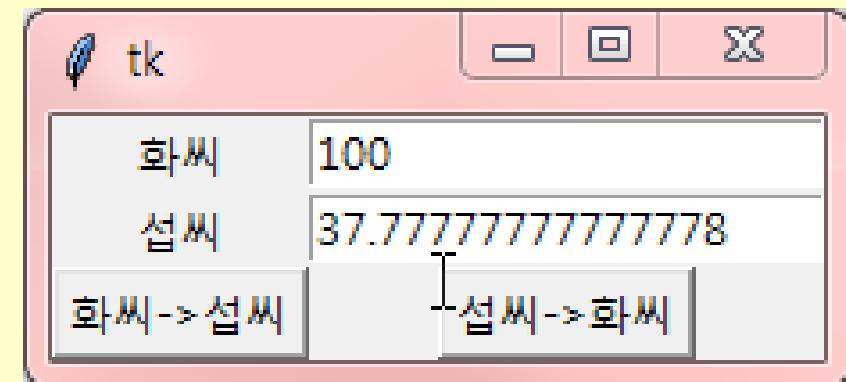
l1 = Label(window , text="화씨")
l2 = Label(window, text="섭씨")
l1.grid(row=0, column=0)
l2.grid(row=1, column=0)

e1 = Entry(window)
e2 = Entry(window)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)

b1 = Button(window, text="화씨->섭씨", command=process)
b2 = Button(window, text="섭씨->화씨")
b1.grid(row=2, column=0)
b2.grid(row=2, column=1)

window.mainloop()

```



```
from tkinter import *  
  
def process():  
    temperature = float(e1.get())  
    mytemp = (temperature-32)*5/9  
    e2.insert(0, str(mytemp))
```

```
window = Tk()
```

```
I1 = Label(window , text="화씨", font='helvetica 16 italic')  
I2 = Label(window, text="섭씨", font='helvetica 16 italic')
```

```
I1.grid(row=0, column=0)  
I2.grid(row=1, column=0)
```

```
e1 = Entry(window, bg="yellow", fg="white")  
e2 = Entry(window, bg="yellow", fg="white")  
e1.grid(row=0, column=1)  
e2.grid(row=1, column=1)
```

```
b1 = Button(window, text="화씨->섭씨", command=process)  
b2 = Button(window, text="섭씨->화씨")  
b1.grid(row=2, column=0)  
b2.grid(row=2, column=1)
```

```
window.mainloop()
```



절대 위치 배치 관리자

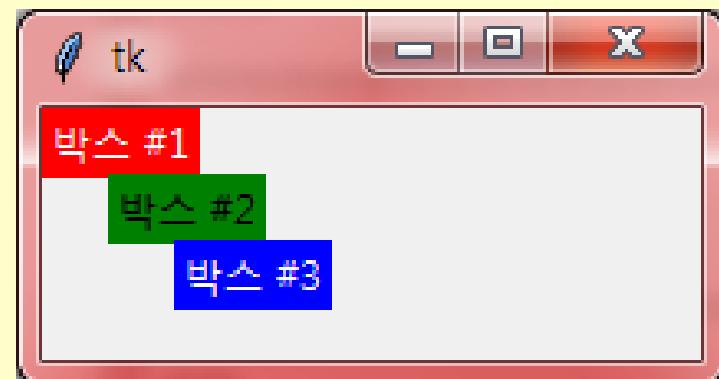
참조자료 : 두근두근 퍼이썬 10장 10장 tkinter로 GUI 만들기 강의자료

```
from tkinter import *

window = Tk()

w = Label(window, text="박스 #1", bg="red", fg="white")
w.place(x=0, y=0)
w = Label(window, text="박스 #2", bg="green", fg="black")
w.place(x=20, y=20)
w = Label(window, text="박스 #3", bg="blue", fg="white")
w.place(x=40, y=40)

window.mainloop()
```



Section05 메뉴와 대화상자

■ 메뉴의 생성

- 메뉴의 구성 개념과 형식

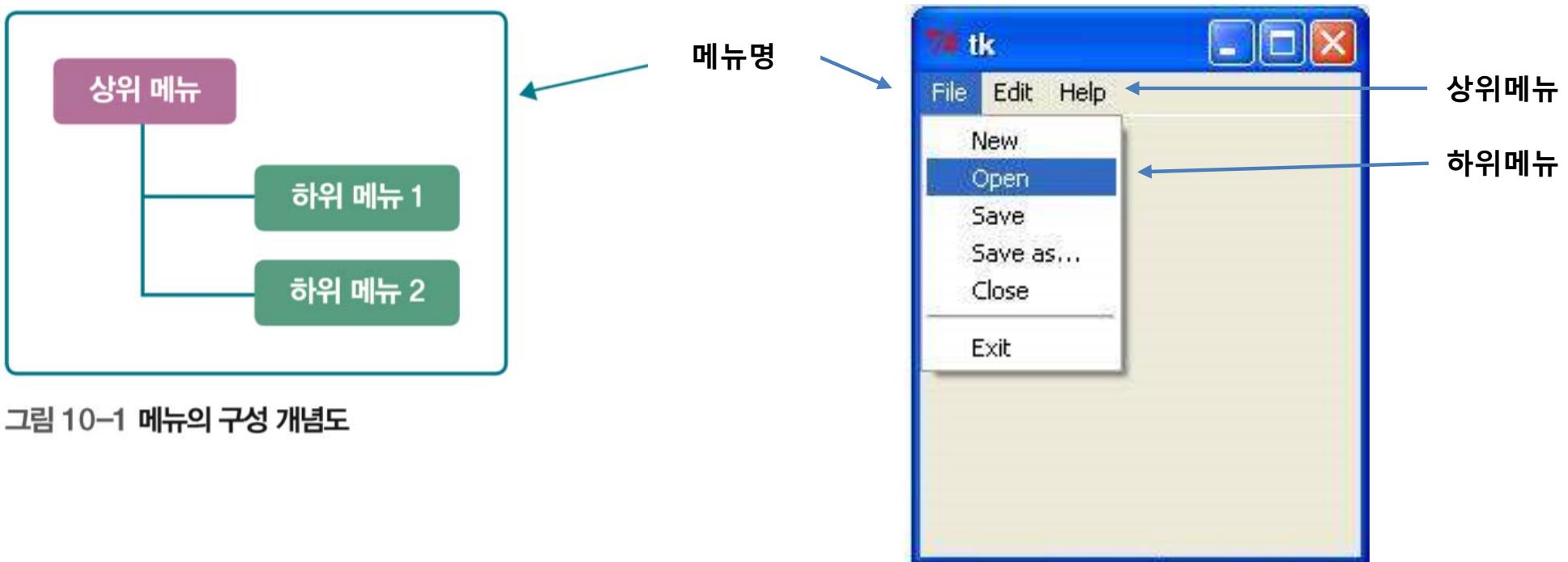


그림 10-1 메뉴의 구성 개념도

Section05 메뉴와 대화상자

메뉴명=tkinter.Menu(윈도우창)

윈도우창.config(menu=메뉴명)

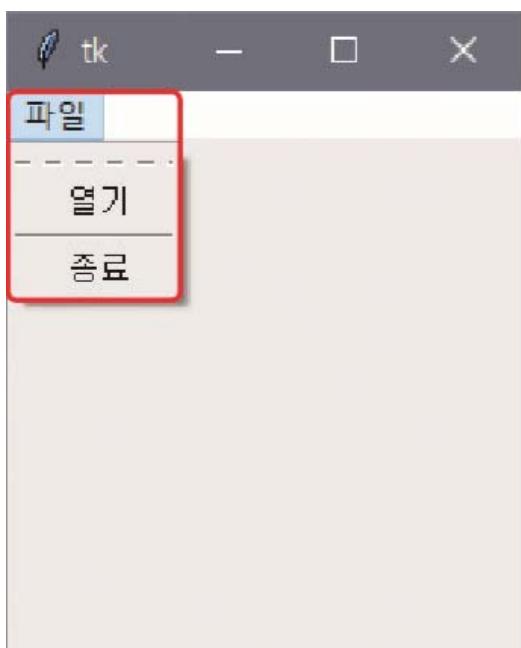
상위메뉴명=tkinter.Menu(메뉴명,파라미터1,파라미터2,...)

메뉴명.add_cascade(label='상위메뉴', menu=상위메뉴명)

상위메뉴명.add_command(label='하위메뉴1', command=함수1)

상위메뉴명.add_command(label='하위메뉴2', command=함수2)

- 윈도창(window)에 대한 메뉴 선언 (mainMenu)
- 윈도창(window)에 메뉴(mainMenu) 등록
- 메뉴명(mainMenu)에 표시할 상위메뉴 속성 설정하여, 이를 상위메뉴명 (fileMenu)으로 받음
- 메뉴명(mainMenu)에 상위메뉴 (fileMenu) 연결
- 상위메뉴명(fileMenu)에 하위메뉴1,2 생성



```
5  mainWindow = Menu(window)
6  window.config(menu = mainWindow)
7
8  fileMenu = Menu(mainWindow)
9  mainWindow.add_cascade(label = "파일", menu = fileMenu)
10 fileMenu.add_command(label = "열기")
11 fileMenu.add_separator()
12 fileMenu.add_command(label = "종료")
```

Section05 메뉴와 대화상자

■ 메뉴의 구성 개념과 형식

자료출처 : <https://076923.github.io/posts/Python-tkinter-8/>

Menu Method

이름	의미
add_command(파라미터)	기본 메뉴 항목 생성
add_radiobutton(파라미터)	라디오버튼 메뉴 항목 생성
add_checkbutton(파라미터)	체크버튼 메뉴 항목 생성
add_cascade(파라미터)	상위 메뉴와 하위 메뉴 연결
add_separator()	구분선 생성
add(유형, 파라미터)	특정 유형 의 메뉴 항목 생성
delete(start_index, end_index)	start_index 부터 end_index 까지의 항목 삭제
entryconfig(index, 파라미터)	index 위치의 메뉴 항목 수정
index(item)	item 메뉴 항목의 index 위치 반환
insert_separator (index)	index 위치에 구분선 생성
invoke(index)	index 위치의 항목 실행
type(속성)	선택 유형 반환 (command, radiobutton, checkbutton, cascade, separator, tearoff)

- Tip : 파라미터 중, : `label=이름` 을 이용하여 메뉴의 이름을 설정할 수 있습니다.
- Tip : `메뉴 이름.add_cascade(label="상위 메뉴 이름", menu=연결할 상위 메뉴)` 를 이용하여 메뉴를 부착할 수 있습니다.

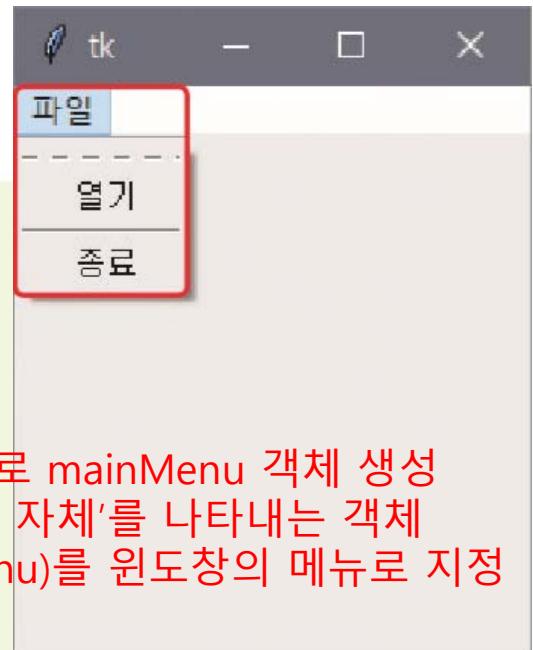
Section05 메뉴와 대화상자

- [파일] 메뉴 아래에 [열기]와 [종료] 하위 메뉴가 있는 코드

Code10-17.py

```
1 from tkinter import *
2
3 window = Tk()
4
5 mainMenu = Menu(window)
6 window.config(menu = mainMenu)
7
8 fileMenu = Menu(mainMenu)
9 mainMenu.add_cascade(label = "파일", menu = fileMenu)
10 fileMenu.add_command(label = "열기")
11 fileMenu.add_separator()
12 fileMenu.add_command(label = "종료")
13
14 window.mainloop()
```

- 5행 : Menu(window(부모 윈도))로 mainMenu 객체 생성
- mainMenu : [그림 10-1]의 '메뉴 자체'를 나타내는 객체
- 6행 : 생성한 메뉴 자체(mainMenu)를 윈도창의 메뉴로 지정
- 8~9행 : 상위 메뉴인 [파일] 생성, 메뉴 자체에 부착
[파일] 메뉴는 선택하고 끝나는 것이 아니라, 그
아래에 다른 메뉴가 확장되어야 하므로
add_cascade() 함수 사용
- 10행 : [파일] 메뉴의 하위에 [열기] 메뉴 준비
- [열기] 메뉴 : 선택할 때 어떤 작동을 해야 하므로,
add_command() 함수 사용
- 11행 : 메뉴 사이에 구분선 넣음
- 12행 : 같은 방식으로 하위 메뉴 생성



Section05 메뉴와 대화상자

- 메뉴를 선택하면 작동할 수 있도록 코드 추가

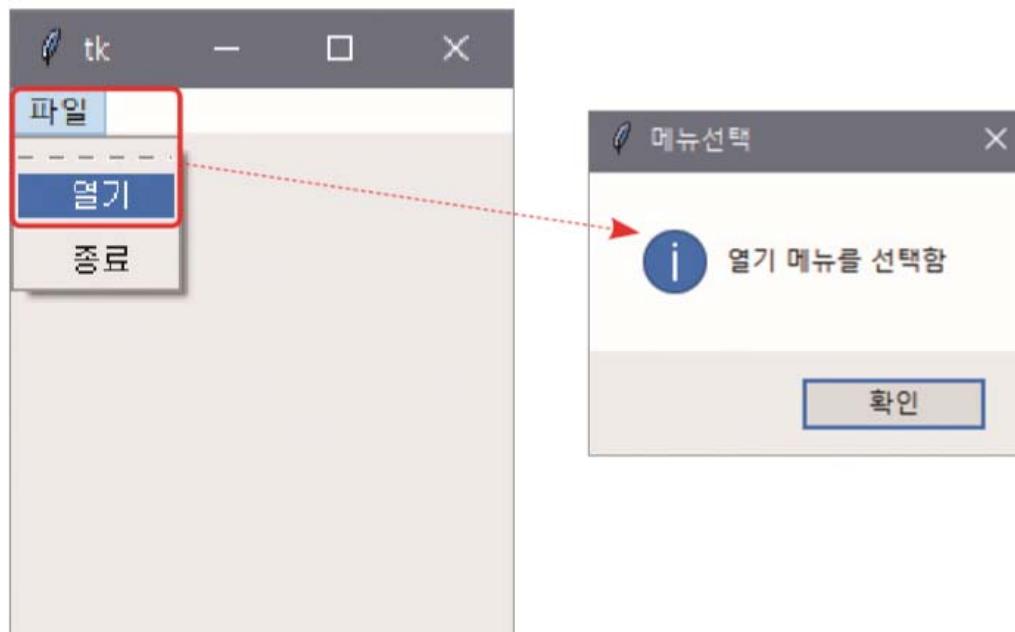
Code10-18.py

```
1 from tkinter import *
2 from tkinter import messagebox
3
4 ## 함수 선언 부분 ##
5 def func_open() :
6     messagebox.showinfo("메뉴선택", "열기 메뉴를 선택함")
7
8 def func_exit() :          5~6행 : [열기] 메뉴 선택하면 간단한 메시지창 열림
9     window.quit()           8~10행 : [종료] 메뉴를 선택하면 프로그램이 종료
10    window.destroy()
11
12 ## 메인 코드 부분 ##
13 window = Tk()
14
15 mainMenu = Menu(window)
16 window.config(menu = mainMenu)
17
```

Section05 메뉴와 대화상자

```
18 fileMenu = Menu(mainMenu)
19 mainMenu.add_cascade(label = "파일", menu = fileMenu)
20 fileMenu.add_command(label = "열기", command = func_open)
21 fileMenu.add_separator()
22 fileMenu.add_command(label = "종료", command = func_exit)
23
24 window.mainloop()
```

20행 : [열기] 메뉴 선택하면 무언가 작동해야 하므로
add_command() 함수 사용, 선택할 때 실행될 함수명을
command값으로 사용, 즉 [파일] 메뉴 선택하면 하위 메뉴가 확장,
[열기] 메뉴를 선택하면 5행의 func_open() 함수가 실행
22행 : 20행과 같은 방식으로 하위 메뉴를 생성한다.



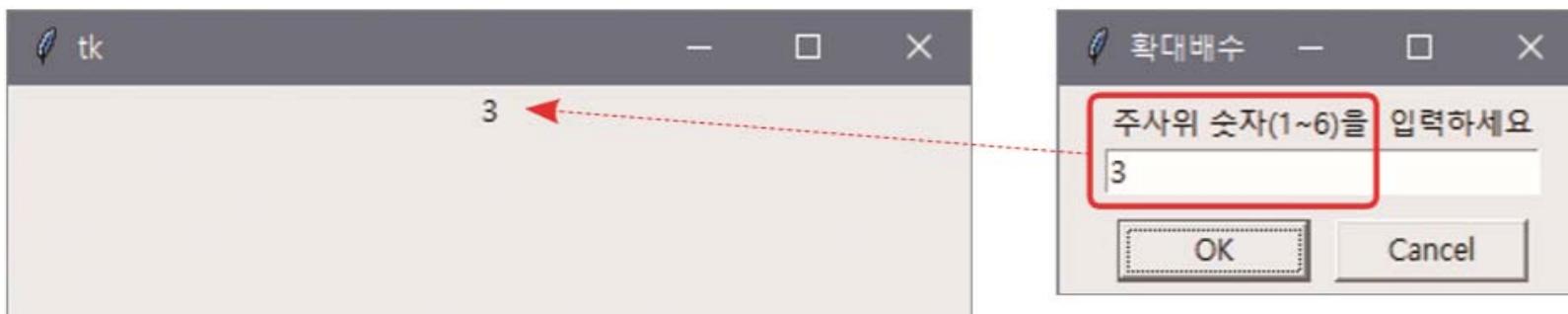
Section05 메뉴와 대화상자

■ 대화상자의 생성과 사용

- tkinter.simpledialog 모듈을 import한 후 askinteger() 및 askstring() 등을 사용

Code10-19.py

```
1 from tkinter import *
2 from tkinter.simpledialog import *          • 2행 : tkinter.simpledialog 모듈 임포트
3
4 ## 함수 선언 부분 ##
5 window = Tk()
6 window.geometry("400x100")
7
8 label1 = Label(window, text = "입력된 값")  • 8~9행 : 라벨 준비
9 label1.pack()
10
11 value = askinteger("확대배수", "주사위 숫자(1~6)을 입력하세요", minValue = 1, maxValue = 6)
12
13 label1.configure(text = str(value))        • 11행 : askinteger("제목", "내용", 옵션) 함수로 정수 입력
14 window.mainloop()                         • 13행 : 입력받은 숫자를 문자열로 변경해서 라벨에 씀
```



Section05 메뉴와 대화상자

■ 그림 파일인 GIF 파일을 선택하는 코드

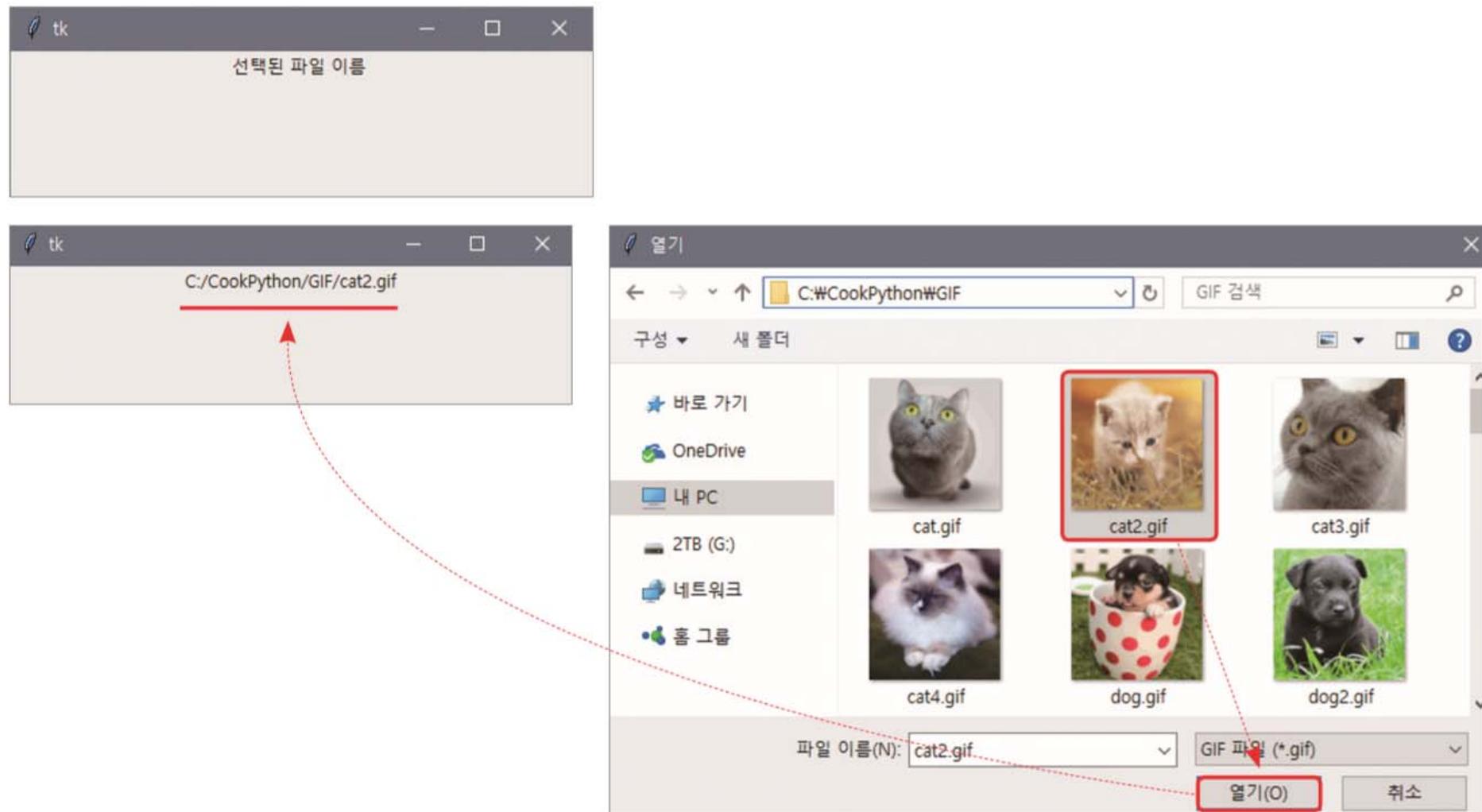
- Code10-20.py는 Code10-19.py의 2행과 11행만 변경

Code10-20.py

```
1 from tkinter import *
2 from tkinter.filedialog import *
3
4 ## 함수 선언 부분 ##
5 window = Tk()
6 window.geometry("400x100")
7
8 label1 = Label(window, text = "선택된 파일 이름")
9 label1.pack()
10
11 filename = askopenfilename(parent = window, filetypes = (("GIF 파일", "*.gif"),
12 ("모든 파일", "*.*")))
13
14 label1.configure(text = str(filename))
15 window.mainloop()
```

2행 : tkinter. filedialog 모듈 임포트
11행 : askopenfilename() 함수 사용
13행 : filename을 출력

Section05 메뉴와 대화상자



Section05 메뉴와 대화상자

- Code10-20.py의 11~13행 변경하고 실행

Code10-21.py

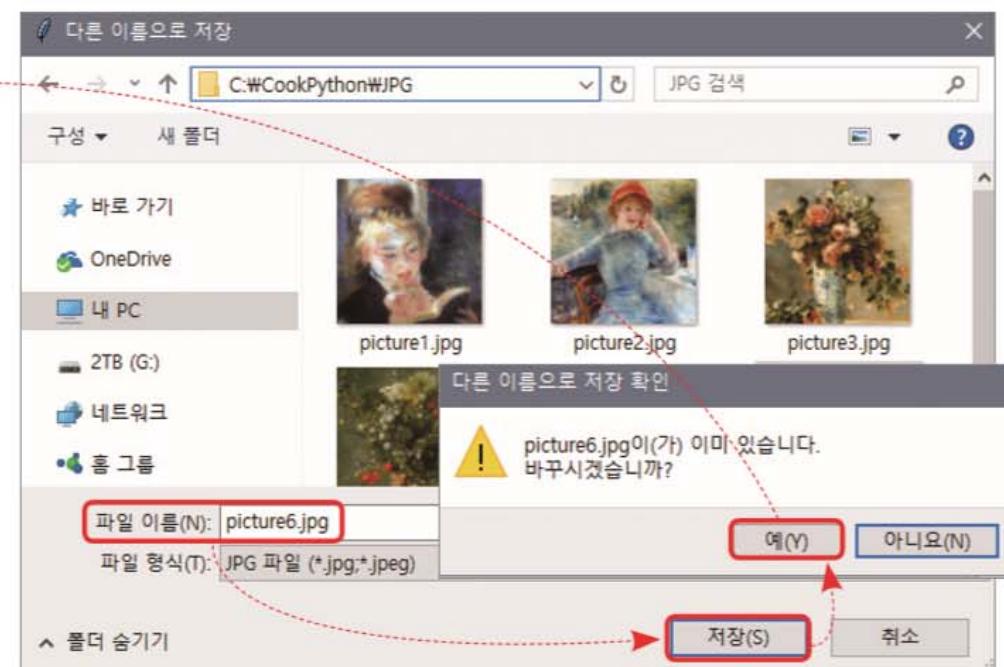
```
11 saveFp = asksaveasfile(parent = window, mode = "w", defaultextension = ".jpg",
                           filetypes = (("JPG 파일", "*.jpg;*.jpeg"), ("모든 파일", "*.*")))
12 label1.configure(text = saveFp)
13 saveFp.close()
```



11행 : asksaveasfile() 함수의 매개변수 중 mode="w"는 쓰기 모드라는 의미, defaultextension=".jpg"는 특별히 확장명을 지정하지 않으면 확장명을 jpg로 불인다는 의미임

12행 : saveFp는 자체를 text에 대입시켜서 출력

13행 : 파일 닫음



Section05 메뉴와 대화상자

■ [프로그램 2]의 완성

- 메뉴 처리와 파일 처리가 핵심

Code10-22.py

```
1 from tkinter import *
2 from tkinter.filedialog import *
3
4 ## 함수 선언 부분 ##
5 def func_open() :
6     filename = askopenfilename(parent = window, filetypes = (("GIF 파일", "*.gif"),
7                                         ("모든 파일", "*.*")))
8     photo = PhotoImage(file = filename)
9     pLabel.configure(image = photo)
10    pLabel.image = photo
11
12    def func_exit() :
13        window.quit()
14        window.destroy()
15
16    ## 메인 코드 부분 ##
17    window = Tk()
18    window.geometry("400x400")
```



5~9행 : [파일 열기] 메뉴를 선택하면 실행되는 함수
askopenfilename() 함수로 파일 선택 후 선택한
파일명을 7행에서 PhotoImage() 함수로 처리
8~9행 : 라벨에 이미지 표시

Section05 메뉴와 대화상자

```
18 window.title("명화 감상하기")
19
20 photo = PhotoImage()
21 pLabel = Label(window, image = photo)
22 pLabel.pack(expand = 1, anchor = CENTER)
23
24 mainMenu = Menu(window)
25 window.config(menu = mainMenu)
26 fileMenu = Menu(mainMenu)
27 mainMenu.add_cascade(label = "파일", menu = fileMenu)
28 fileMenu.add_command(label = "파일 열기", command = func_open)
29 fileMenu.add_separator()
30 fileMenu.add_command(label = "프로그램 종료", command = func_exit)
31
32 window.mainloop()
```

20~22행 : 선택한 GIF 윈도창에 출력하려고 라벨 준비
20행 : PhotoImage()에 별도의 매개변수 없이 빈 그림만
준비

24~30행 : 준비하는 메뉴는 앞에서 실습한 내용과 동일

Section05 메뉴와 대화상자

