

파이썬 프로그래밍

13~14주차 강의자료

나재호

CHAPTER
13

GUI 프로그래밍 기초

- 01 GUI 프로그래밍이란?
- 02 루트 윈도우의 기본 구성
- 03 기본 위젯
- 04 위젯의 배치
- 05 객체지향적으로 구성하기

이 단원을 마치고 나면

- 이벤트 기반의 GUI 프로그래밍의 동작 방식을 이해할 수 있다.
- tkinter 모듈을 이용해 윈도우 프로그램을 작성할 수 있다.
- 기본 위젯을 이용해 사용자 입력을 받을 수 있다.
- 위젯을 원하는 위치 상에 배치하여 GUI 프로그램을 구성할 수 있다.
- 객체지향적으로 GUI 프로그램을 작성할 수 있다.



01 | GUI 프로그래밍이란

인터페이스(interface)

- ◆ 서로 다른 두 구성요소 간에 정보를 교환하는 공통의 경계



그림 13.1 컴퓨터와 외부장치 간의 인터페이스

사용자 인터페이스(user interface, UI)

- ◆ 인간이 시스템과 상호작용하기 위해 제공되는 각종 매개체

- 하드웨어 인터페이스
 - 키보드나 마우스 그리고 모니터 같은 입출력장치
 - 물리적 매개체
- 소프트웨어 인터페이스
 - 특정 프로그램 내에서 제공되는 메뉴나 버튼 그리고 각종 대화상자 같은 화면 구성요소들
 - 가상적 매개체



그림 13.2 컴퓨터의 사용자 인터페이스

사용자 인터페이스의 구분

◆ CUI(Character User Interface)

- A.k.a. CLI (Command-Line Interface)
- 문자열을 기반으로 정보를 입력 받거나 출력하는 방식
 - 사용자로부터 정보를 입력 받기 위해 보통 키보드를 사용
 - 문자들의 나열을 통해 수행 결과를 모니터에 나타냄
- 고전적이고 기본적인 방식으로 사용자와 상호작용
 - 사용자는 어떤 명령들을, 어떤 순서로 입력해야 하는지를 알고 있어야 함
 - 초기 컴퓨터 시스템에서 사용
 - 오늘날에도 서버용 응용 프로그램에서 단순하고 효율적인 실행을 위해 사용

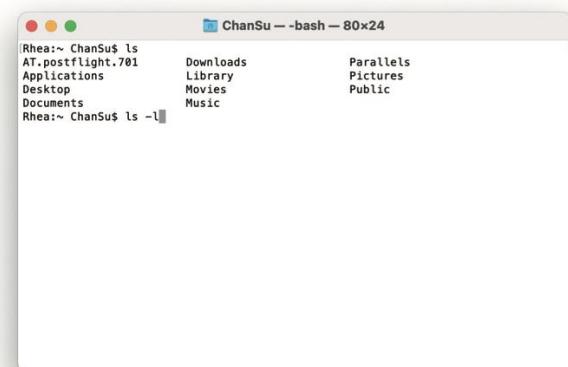


그림 13.3 CUI 방식으로 명령을 처리하는 macOS의 터미널 프로그램

사용자 인터페이스의 구분

◆ GUI(Graphical User Interface)

- 시각적인 요소들로 프로그램을 구성
 - 윈도우나 아이콘, 메뉴와 버튼과 같은 요소들로 화면을 구성
 - 키보드, 마우스, 트랙패드 등을 이용해 화면 구성요소를 선택해가면서 처리할 명령이나 데이터를 입력
 - 수행 결과를 문자열뿐만 아니라 그래픽적인 요소들을 이용해 다양한 방식으로 표시

- 사용자가 좀 더 직관적으로 프로그램을 사용할 수 있게 제공 함
 - 누구나 쉽게 사용 가능

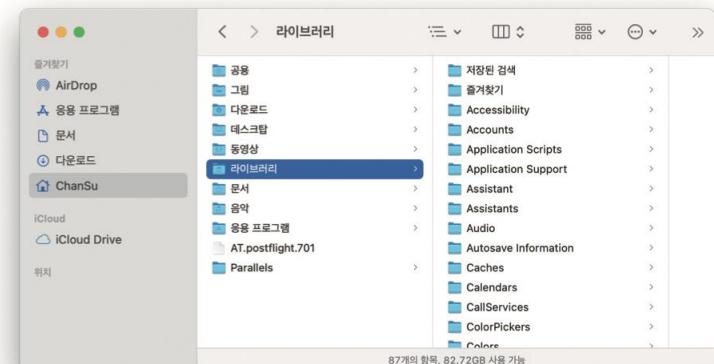


그림 13.4 GUI 방식으로 명령을 처리하는 macOS의 파일더 프로그램

사용자 인터페이스의 구분

◆ NUI(Natural User Interface)

- 사용자 인터페이스가 더 이상 화면에 제한되지 않음
- 인간에게 좀 더 자연스러운 방식 제공
 - 인간의 언어나 제스처와 같은 동작 등
- 조심스럽게 사용돼야 함
 - 부정확한 인식으로 인한 사용자가 원하지 않는 동작이 수행
 - 민감한 각 개인의 생체 정보에 대한 유지 관리에 대한 사회적 합의가 필요
 - 해킹이나 오남용으로 인해 개인의 프라이버시가 침해되지 않을 수 있도록 적절한 법적 제도적 장치가 필요

GUI 프로그래밍의 목적

- ◆ GUI 프로그램을 작성하는 과정
 - '윈도우 프로그래밍'이라 하기도
- ◆ 그래픽을 이용해 사용자가 이해하기 쉽고 사용하기 쉬운 프로그램 인터페이스를 제공



그림 13.6 손쉬운 GUI 프로그램의 사용

GUI 운영체제의 특징

- ◆ 멀티태스킹(multi-tasking)을 지원
 - 다수의 프로그램을 짧은 시간 간격마다 번갈아 수행함으로써 동시에 수행되는 것과 같은 효과
 - 사용자가 화면의 특정 부분을 마우스로 클릭했을 때
 - 운영체제는 이 동작을 현재 실행 중인 어떤 프로그램에게 전달해야 할 것인지를 결정해야 함
 - GUI 기반의 운영체제는 프로그램 내/외부의 각종 변화 발생을 메시지를 통해 해당 프로그램에게 알려줌



GUI OS에서 이벤트와 메시지 큐

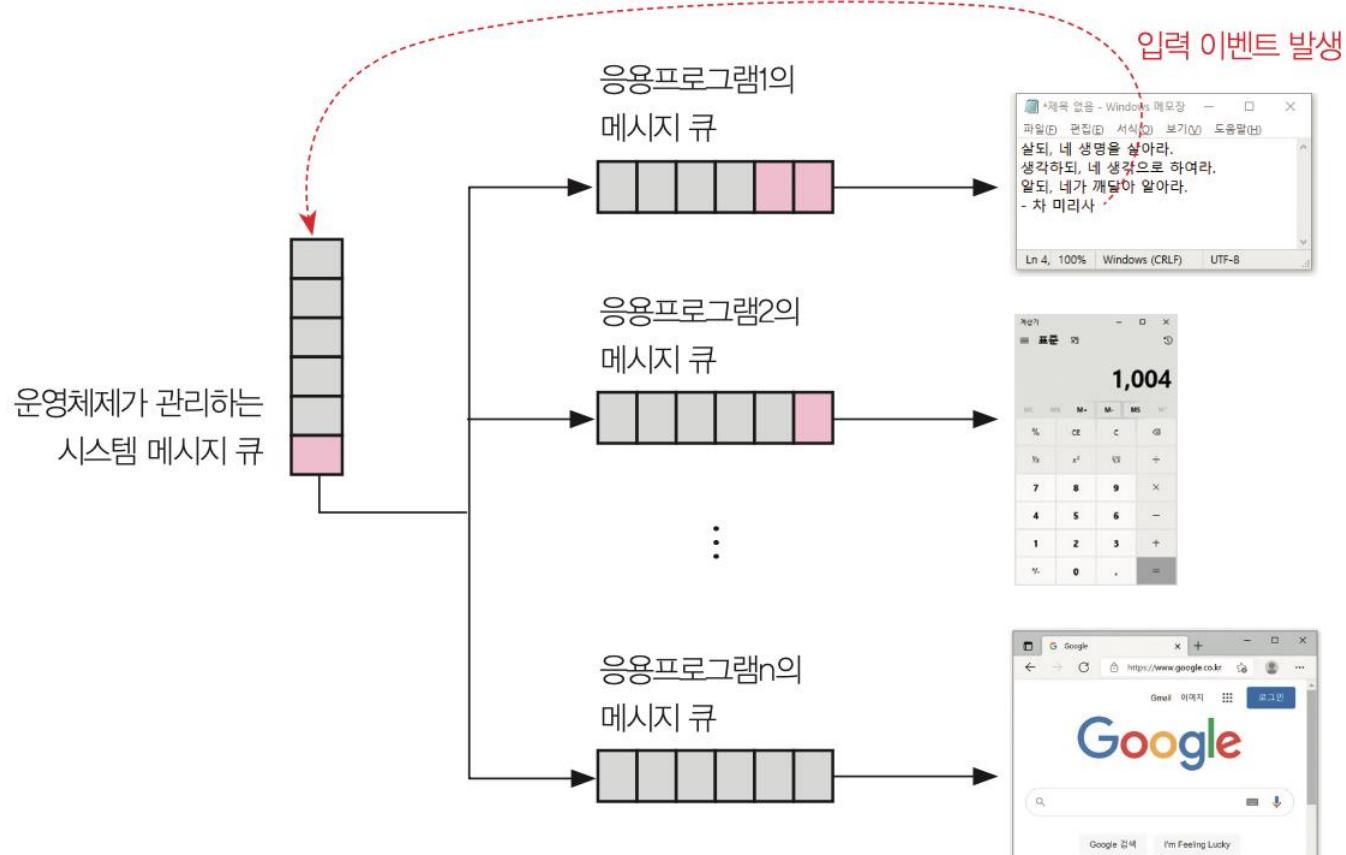


그림 13.7 이벤트와 메시지 큐

GUI 프로그래밍의 두 과정

- ◆ 사용자가 발생시키는 비동기적 이벤트에 대한 메시지 처리 필요
 - 메시지 기반(message driven) 프로그래밍
 - 이벤트 기반(event driven) 프로그래밍
- ◆ 응용 프로그램을 위한
 1. 화면 구성
 - 시각적인 요소로 화면을 구성하는 과정
 2. 이벤트 처리 과정
 - 구성된 화면을 통해 사용자가 발생시키는 혹은 프로그램 내부적으로 발생되는 각종 이벤트에 대한 메시지를 처리



02 | 루트 윈도우의 기본 구성

tkinter 모듈

- ◆ 파이썬에서 GUI 프로그램을 개발하기 위해 사용할 수 있는 모듈 중 하나
 - 파이썬 설치 시 기본적으로 내장되는 표준 모듈
 - 간단하게 GUI 프로그래밍 가능
 - 최근 버전에서는 개선된 UI를 제공
 - 다른 GUI 모듈에 비해 상대적으로 화면 구성을 위한 지원이 다소 부족하고 또 투박해 보인다는 기존의 단점 보완
 - 참고: TK interface의 약자
 - Tcl/Tk(Tool Command Language/Tool Kit)를 파이썬에서 사용할 수 있도록 구성된 모듈
 - Tcl/Tk는 유닉스 계열에서 간단하게 GUI 프로그램을 작성하기 위해 사용되던 스크립트 방식의 프로그래밍 언어

루트 윈도우 만들기

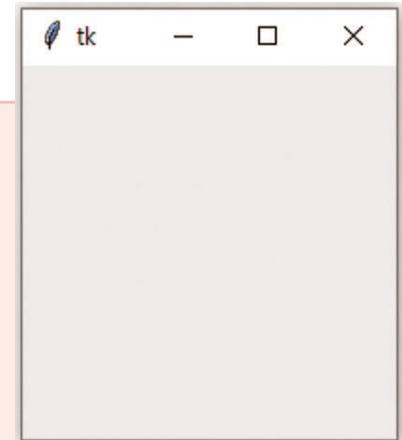
◆ 단 몇 줄만으로도 윈도우 생성 가능

#

코드 13-1

기본적인 루트 윈도우의 생성

```
01 import tkinter as tk  
02  
03 # 주 프로그램부  
04 root_win = tk.Tk()      # 기본 윈도우 객체 반환  
05  
06 # 화면 구성 및 이벤트 처리  
07 # ...  
08  
09 root_win.mainloop()    # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



지속적으로 사용자 이벤트를 전달받아
주어진 일을 처리할 수 있게

제목 표시줄 설정

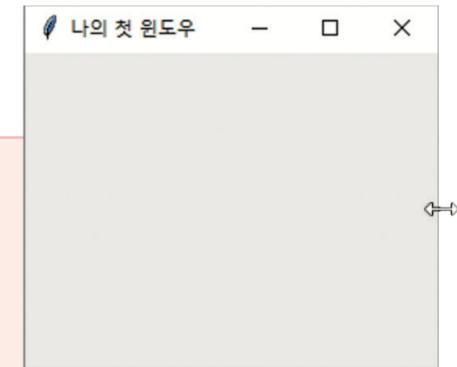
◆ 윈도우 객체에 대해 title() 메서드를 호출

#

코드 13-2

제목을 갖는 기본 윈도우

```
01 import tkinter as tk  
02  
03 # 주 프로그램부  
04 root_win = tk.Tk()          # 기본 윈도우 객체 반환  
05 root_win.title('나의 첫 윈도우') # 제목표시줄에 윈도우 제목 표시  
06  
07 # 화면 구성 및 이벤트 처리  
08 # ...  
09  
10 root_win.mainloop() # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



윈도우 크기와 위치 설정

루트 윈도우객체.geometry('창의너비/x창의높이/±가로위치값±세로위치값')

- 인수는 반드시 하나의 문자열로 지정
 - 문자열 내의 모든 내용은 공백 없이 작성
- 크기를 설정하기 위해선 숫자 사이에 알파벳 x를
- 위치를 설정하기 위해서는 +나 - 기호 사용

	+위치값	-위치값
가로 방향 위치	화면 왼쪽에서부터의 거리	화면 오른쪽에서부터의 거리
세로 방향 위치	화면 위쪽에서부터의 거리	화면 아래쪽에서부터의 거리

윈도우의 크기와 위치 설정 예

코드 13-3 일정한 크기로 설정된 기본 윈도우

```
01 import tkinter as tk  
02  
03 # 주 프로그램부  
04 root_win = tk.Tk()                      # 기본 윈도우 객체 반환  
05 root_win.title('나의 첫 윈도우')          # 제목표시줄에 윈도우 제목 표시  
06 root_win.geometry('500x200-0+50')        # 윈도우의 크기 및 위치 설정  
07  
08 # 화면 구성 및 이벤트 처리  
09 # ...  
10  
11 root_win.mainloop() # 윈도우에서 다양한 이벤트 처리 시작을 지시
```

윈도우 크기 고정

- ◆ 윈도우 객체에 대해 `resizable()` 메서드를 이용
 - 키워드 인수 `width`와 `height`를 가짐
 - 각각 고정 활성화/비활성화를 위해 `True/False` 전달

코드 13-4

윈도우의 크기 고정

```
01 import tkinter as tk
02
03 # 주 프로그램부
04 root_win = tk.Tk()                      # 기본 윈도우 객체 반환
05 root_win.title('나의 첫 윈도우')          # 제목표시줄에 윈도우 제목 표시
06 root_win.geometry('500x200-0+50')        # 윈도우의 크기 및 위치 설정
07 root_win.resizable(width=False, height=False)
08
09 # 화면 구성 및 이벤트 처리
10 # ...
11
12 root_win.mainloop() # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



03 | 기본 위젯

tkinter 위젯

- ◆ 각종 GUI 요소인 버튼, 레이블, 텍스트 입력 필드, 캔버스 등의 기본 위젯을 제공
 - 위젯은 GUI 프로그램의 화면을 구성하는 작은 부품과 같은 구성요소
 - 특정 사용자 인터페이스에 대한 독립된 책임을 가짐
 - https://tutorialspoint.com/python/python_gui_programming.htm
 - 각 위젯은
 - 객체 생성 후 적절한 속성 지정
 - 루트 윈도우에 부착되어 배치해야 사용 가능
 - (필요시) 사용자 입력에 따른 이벤트 처리 필요



그림 13.9 기본 위젯

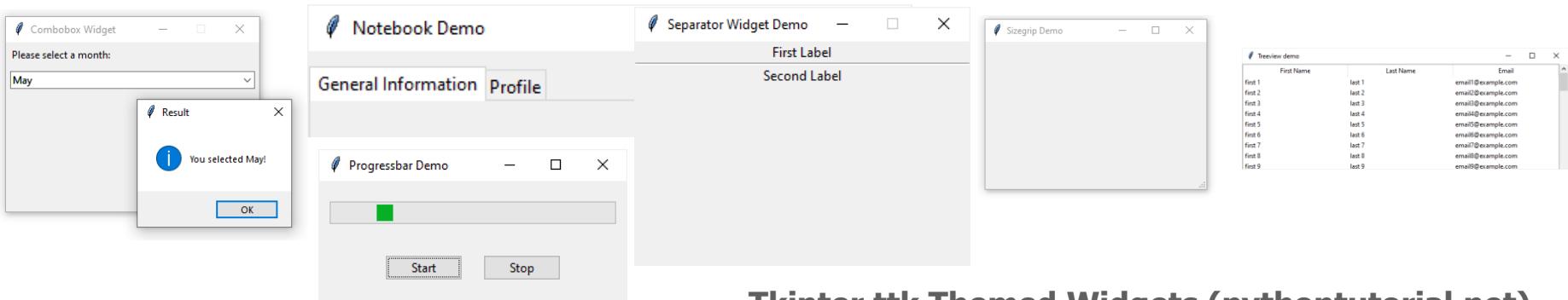
tkinter의 기본 위젯

위젯	설명
Label	텍스트 또는 이미지 표시
Button	푸쉬 버튼
CheckButton	체크박스 버튼
RadioButton	라디오 버튼
Entry	한 줄로 된 입력상자
Text	다행 입력상자, 일부 Rich Text 기능 제공
ListBox	목록 나열 및 선택상자
Message	Label과 비슷하게 텍스트 표시. 단, 자동 래핑 기능이 있음
Scale	슬라이스 바
Scrollbar	스크롤 바
Menu	메뉴
Menubutton	메뉴 버튼
Toplevel	대화상자 같은 추가의 새 윈도우 생성 시 사용
Frame	컨테이너 위젯. 다른 위젯들을 그룹화할 때 사용
Canvas	그림을 그릴 수 있음. 커스텀 위젯 생성 시 사용

ttk 위젯

- ◆ 개선된 룩앤플을 제공하는 추가 확장 모듈
 - ttk는 themed tk의 약자
 - 18개 중
 - 12개는 기존의 tk 위젯을 대체
 - 6개는 추가

tk 대체	Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale, Scrollbar, Spinbox
추가	Combobox, Notebook, Progressbar, Separator, Sizegrip, Treeview



Tkinter ttk Themed Widgets (pythontutorial.net)

tkinter 위젯과 ttk 위젯의 임포트

```
import tkinter as tk
from tkinter import ttk

# tkinter 위젯은 'tk.위젯이름'으로 사용
# ttk 위젯은 'ttk.위젯이름'으로 사용
```

- tkinter 모듈의 기본 위젯은 **tk.식별자**의 형태로 사용
- ttk 모듈의 확장 위젯은 **ttk.식별자**의 형태로 사용

ttk.Widget 클래스

- ◆ ttk 위젯들의 공통된 특성을 갖는 클래스
 - ttk 위젯들은 이 클래스에서 파생됨
 - 각 위젯이 지원하는 표준 옵션과 메서드를 정의함
 - 이 클래스로 직접 객체를 생성하지는 않음

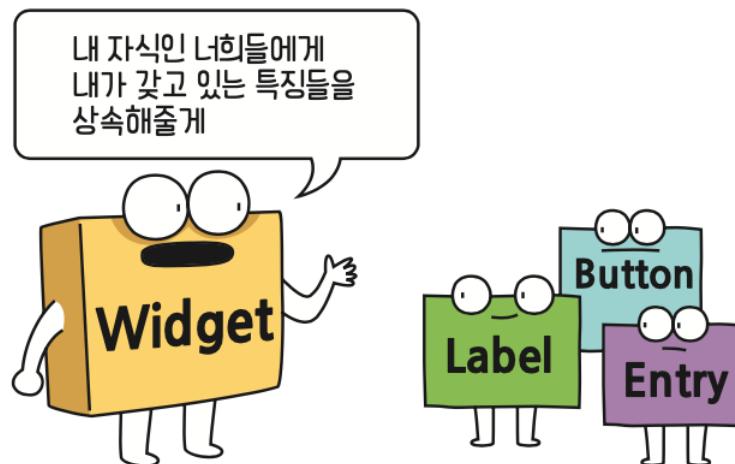


그림 13.10 기반 클래스인 Widget

위젯의 주요 옵션

- ◆ 주요 ttk 위젯이 갖는 기본적인 주요 공통 옵션
 - 위젯 객체 생성시 인수로 지정

주요 옵션	설명
style	사용자 정의 위젯 스타일을 지정
text	위젯 안에 표시할 텍스트 문자열을 지정
image	표시할 이미지를 지정
width	텍스트 레이블에 할당할 공간의 크기를 지정 0보다 크면 문자 너비 단위로, 0보다 작으면 최소 너비를 지정 0 혹은 미지정 시 텍스트 레이블의 자연 너비가 사용됨

- <https://docs.python.org/ko/3/library/tkinter.ttk.html#widget>

ttk 위젯의 옵션값 확인

◆ 위젯의 모든 옵션을 딕셔너리로 반환

위젯객체이름.configure()

- (참고) 반환값 형식: 각 옵션을 키로 가짐

{옵션이름 : (name, dbName, dbClass, default, current)}

옵션 세부 항목	설명
<i>name</i>	옵션 이름
<i>dbName</i>	옵션의 데이터베이스 이름
<i>dbClass</i>	옵션의 데이터베이스 클래스
<i>default</i>	옵션의 기본값
<i>current</i>	옵션의 현재 설정값

ttk.Label에 설정된 옵션값 확인 예

- ◆ 세부 옵션 항목을 기억하기 보다 필요할 때 확인

#

코드 13-5

위젯에 설정된 옵션값 확인

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 def display_options(widget):
05     config = widget.configure()
06     for key, value in config.items():
07         print(f'{key:15s}\t{value}')
08
09 widget = ttk.Label(None)
10 display_options(widget)
```

ttk.Label에 설정된 옵션값 확인 예



실행 결과

```
background      ('background', 'frameColor', 'FrameColor', '', '')
foreground     ('foreground', 'textColor', 'TextColor', '', '')
font          ('font', 'font', 'Font', '', '')
borderwidth    ('borderwidth', 'borderWidth', 'BorderWidth', '', '')
relief         ('relief', 'relief', 'Relief', '', '')
anchor         ('anchor', 'anchor', 'Anchor', '', '')
justify        ('justify', 'justify', 'Justify', '', '')
wraplength     ('wraplength', 'wrapLength', 'WrapLength', '', '')
takefocus      ('takefocus', 'takeFocus', 'TakeFocus', '', '')
text           ('text', 'text', 'Text', '', '')
textvariable   ('textvariable', 'textVariable', 'Variable', '', '')
underline      ('underline', 'underline', 'Underline', -1, -1)
width          ('width', 'width', 'Width', '', '')
image          ('image', 'image', 'Image', '', '')
compound       ('compound', 'compound', 'Compound', <string object: 'none'>,
                <string object: 'none'>)
padding        ('padding', 'padding', 'Pad', '', '')
state          ('state', 'state', 'State', <string object: 'normal'>, <string
                object: 'normal'>)
cursor         ('cursor', 'cursor', 'Cursor', '', '')
style          ('style', 'style', 'Style', '', '')
class          ('class', '', '', '', '')
```

ttk 위젯의 특정 옵션값 변경

- ◆ 대상 위젯에 대해 인수를 갖는 형태로 다음 메서드 호출

```
위젯객체이름.configure(설정옵션명=설정값)
```

- 설정 예:

```
text_label2 = ttk.Label(win)
text_label2.configure(text='반가워요',
                      foreground='white',
                      background='red',
                      font=( '맑은 고딕', 20))
)
```

라벨 위젯: ttk.Label

- ◆ 단순 문자열이나 이미지를 표시할 때 사용
- ◆ ttk 모듈의 Label 클래스로 위젯 객체 생성

```
객체이름 = Label(부모윈도우, text='문자열')
```

- 첫 번째 인수: 부착될 부모 윈도우 지정
- 나머지 인수: 키워드 인수로 위젯에 설정될 옵션 지정

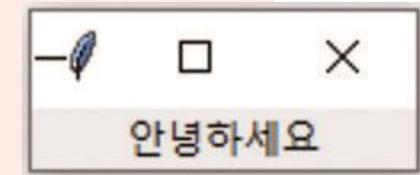
Label 위젯의 기본 사용 예

#

코드 13-6

Label 위젯의 기본 사용 예

```
01 import tkinter as tk          # tk라는 별명으로 tkinter 모듈 임포트
02 from tkinter import ttk      # tkinter에서 ttk 모듈 임포트
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     text_label1 = ttk.Label(win, text='안녕하세요')
07     text_label1.pack()
08
09 # 주 프로그램부
10 win = tk.Tk()                 # 기본 윈도우 객체 반환
11 win.title('라벨 위젯 예1')
12 buildGUI()                   # 화면 구성
13 win.mainloop()                # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



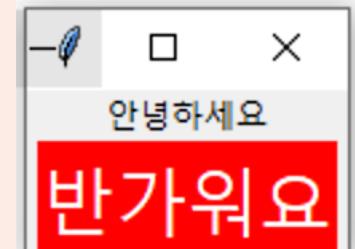
위젯 객체 생성 시 지정한
부모 윈도우 win에 나열식으로 배치

위젯 옵션 변경하기

코드 13-7

Label 위젯에 옵션 설정 예 [코드 13-6 수정]

```
01 import tkinter as tk          # tk라는 별명으로 tkinter 모듈 임포트
02 from tkinter import ttk      # tkinter에서 ttk 모듈 임포트
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     text_label1 = ttk.Label(win, text='안녕하세요')
07
08     text_label2 = ttk.Label(win)
09     text_label2.configure(text='반가워요',
10                           foreground='white',
11                           background='red',
12                           font=('맑은 고딕', 20))
13
14     text_label1.pack()
15     text_label2.pack()
16
17 ::
```



참고: ttk 스타일링

- ◆ 한 프로그램에서 사용되는 위젯들의 공통 양식 지정 필요

- 방법1:
 - 각 위젯에 대해 개별적으로 옵션 설정
- 방법2:
 - 좀 더 일관되고 효율적인 설정위해 style 옵션에 Style 객체를 이용해 지정

```
style = Style()
style.configure('스타일이름', 변경옵션이름1=값1 [, 변경옵션이름2=값2])
```

ttk 스타일링: 기본 스타일 이름 규칙

◆ 스타일 이름은 일정한 규칙을 가짐

표 13.1 위젯의 기본 스타일 이름

위젯 클래스	스타일 이름
Button	TButton
Checkbutton	TCheckbutton
Combobox	TCombobox
Entry	TEntry
Frame	TFrame
Label	TLabel
LabelFrame	TLabelFrame
Menubutton	TMenubutton
Notebook	TNotebook
PanedWindow	TPanedwindow (w가 소문자임에 주의)
Progressbar	orient 옵션에 따라 Horizontal.TProgressbar 또는 Vertical.TProgressbar
Radiobutton	TRadiobutton
Scale	orient 옵션에 따라 Horizontal.TScale 또는 Vertical.TScale
Scrollbar	orient 옵션에 따라 Horizontal.TScrollbar 또는 Vertical.TScrollbar
Separator	TSeparator
Sizegrip	TSizegrip
Treeview	Treeview (하나의 T로 시작함에 주의)

ttk 스타일링: 스타일 이름의 확인

- ◆ 위젯에 대해 `winfo_class()` 메서드를 호출

```
위젯객체이름.winfo_class()      # 위젯의 기본 스타일 이름 반환
```

#

코드 13-8

라벨 위젯의 기본 스타일 이름 확인하기

```
01 import tkinter as tk  
02 from tkinter import ttk  
03  
04 w = ttk.Label(None)  
05 print(w.winfo_class())      # Label 위젯의 기본 스타일 이름 출력
```



실행 결과

TLabel

ttk 스타일링: 새로운 스타일 이름 작성 규칙

◆ 새 스타일 정의

- 기존 스타일의 일부를 수정하는 형태로 정의
 - 스타일 객체에 대해 `configure()` 메서드 호출
 - 첫 번째 인수: 스타일의 이름을 '새스타일이름.기존스타일이름'의 형식으로 지정
 - 나머지 인수: 변경하려는 옵션명과 지정하는 키워드 인수 나열
 - 지정되지 않은 옵션은 기존 스타일을 물려 받음

◆ 위젯에 style 옵션 적용

- 방법1: 위젯 객체 생성시
- 방법2: 기존 위젯에 `configure()` 메서드 이용

ttk 스타일링: 다수 위젯에 스타일 설정 예

코드 13-9

Label 위젯의 스타일 설정 예 (buildGUI()만 나타냄)

```
05 def buildGUI():
06     s = ttk.Style()                      # 스타일 객체 생성
07     s.configure('WR.TLabel',             # 새 스타일 이름
08                 foreground='white',      # 전경색을 흰색으로 설정
09                 background='red',        # 배경색을 빨강색으로 설정
10                 font=('맑은 고딕', 20))   # 글꼴을 설정
11 )                                     # WR.TLabel 스타일 정의 끝
12
13 text_label1 = ttk.Label(win, text='안녕하세요', style='WR.TLabel')
14
15 text_label2 = ttk.Label(win)
16 text_label2.configure(text='반가워요', style='WR.TLabel')
17
18 text_label1.pack()
19 text_label2.pack()
```

푸시 버튼 위젯: ttk.Button

- ◆ 사용자로부터 확인 입력을 받고자 할 때 사용
 - 사용자는 버튼을 눌러주는 행위를 통해 어떤 작업이 일어나기를 프로그램에게 명령
- ◆ ttk 모듈의 Button 클래스로 위젯 객체 생성

```
객체이름 = Button(부모윈도우, text='문자열', command=핸들러함수이름)
```

- text 옵션을 통해 버튼에 표시될 문자열을 지정
- command 옵션을 통해 버튼이 눌렸을 때 처리될 동작을 기술한 함수 이름 지정
 - 함수 이름 다음에 ()가 붙지 않음에 유의

버튼 위젯의 사용 예

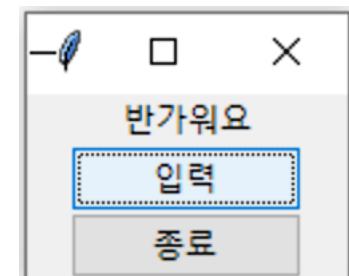
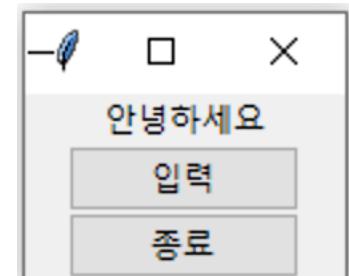
#

코드 13-10

버튼 위젯의 사용 예

```
01 import tkinter as tk          # tk라는 별명으로 tkinter 모듈 임포트
02 from tkinter import ttk      # tkinter에서 ttk 모듈 임포트
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     global text_label          # 이벤트 핸들러에서 접근 위해
07     text_label = ttk.Label(win, text='안녕하세요')
08
09     input_btn = ttk.Button(win, text='입력',
10                           command=input_btn_handler)
11     quit_btn = ttk.Button(win, text='종료',
12                           command=win.destroy)
13
14     text_label.pack()
15     input_btn.pack()
16     quit_btn.pack()
17
18 def input_btn_handler():      # 버튼 클릭에 대한 이벤트 핸들러 함수
19     text_label.configure(text='반가워요')
```

함수 이름 뒤에 ()가
붙지 않음에 유의



:

한 줄 입력 위젯: ttk.Entry

- ◆ 사용자로부터 문자열을 입력 받을 때 사용
 - 보통 한 줄로 입력 받는 경우
- ◆ ttk 모듈의 Entry 클래스로 위젯 객체 생성

```
tk변수객체이름 = StringVar()  
객체이름 = Entry(부모윈도우, textvariable=tk변수객체이름)
```

- textvariable 키워드 인수로 사용자로부터 입력받을 값을 저장할 tk변수 객체 지정

tk변수 객체

- ◆ tkinter를 이용한 GUI 프로그래밍에서 각 위젯과 연관된 정보를 손쉽게 이용하기 위해 사용

클래스 이름	의미
StringVar	문자열을 저장할 tk변수 객체
IntVar	정수를 저장할 tk변수 객체
DoubleVar	실수를 저장할 tk변수 객체

- get() 메서드를 이용해 입력 값 반환
- set() 메서드를 이용해 출력 값 설정

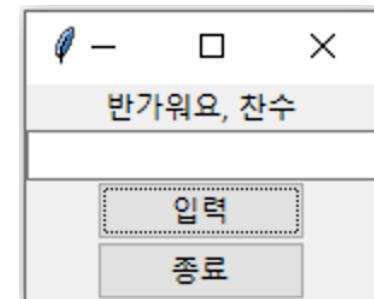
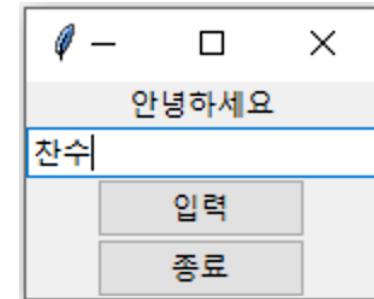
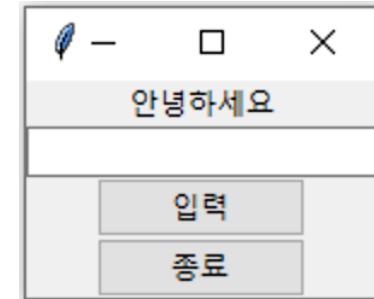
엔트리 위젯의 사용 예

#

코드 13-11

엔트리 위젯의 사용 예

```
...  
05 def buildGUI():  
06     global text_label  
07     text_label = ttk.Label(win, text='안녕하세요')  
08  
09     global name          # 이벤트 핸들러에서 접근 위해  
10     name = tk.StringVar() # 엔트리 위젯에서 문자열 저장 위해  
11     input_entry = ttk.Entry(win, textvariable=name)  
12  
13     input_btn = ttk.Button(win, text='입력',  
14                           command=input_btn_handler)  
15     quit_btn = ttk.Button(win, text='종료',  
16                           command=win.destroy)  
17  
18     text_label.pack()  
19     input_entry.pack()  
20     input_btn.pack()  
21     quit_btn.pack()  
...  
23     def input_btn_handler(): # 버튼 클릭에 대한 이벤트 핸들러 함수  
24         text_label.configure(text='반가워요, ' + name.get())  
25         name.set('')        # 작은따옴표 두 개로 빈문자열을 인수로 지정  
...
```



텍스트 위젯: tk.Text

- ◆ 여러 줄로 된 문자열 입력이 필요한 경우
- ◆ tkinter 모듈의 Text 클래스로 위젯 객체 생성
 - ttk 모듈에서는 따로 제공하지 않음

```
객체이름 = Text(부모윈도우, width=폭, height=높이,  
                    wrap=줄바꿈처리방식)
```

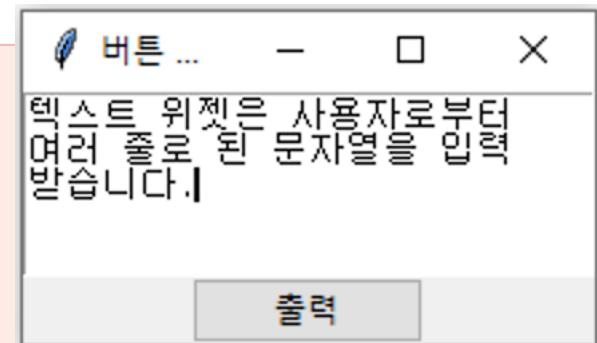
- width, height로 텍스트 위젯의 크기 지정
- wrap으로 한 행의 끝에서의 자동 줄바꿈 처리 지정
 - tk.CHAR : 문자 단위로
 - tk.WORD : 단어 단위로

텍스트 위젯의 사용 예

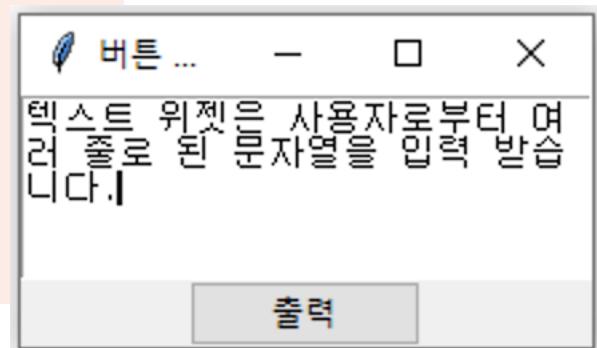
코드 13-12

텍스트 위젯의 사용 예

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     global text_area
07     text_area = tk.Text(win,
08                         width=30, height=5, wrap=tk.WORD)
09
10     input_btn = ttk.Button(win, text='출력',
11                           command=input_btn_handler)
12
13     text_area.pack()
14     input_btn.pack()
15
16 def input_btn_handler(): # 이벤트 핸들러 함수
17     print(text_area.get(0.0, tk.END))
18     text_area.delete(0.0, tk.END)
19
20
```



wrap=tk.WORD



wrap=tk.CHAR

스크롤 가능한 텍스트 위젯: ScrolledText

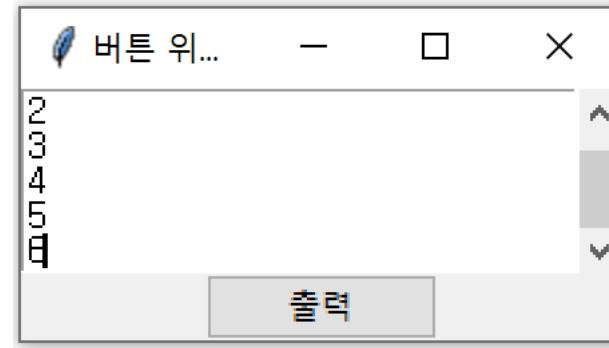
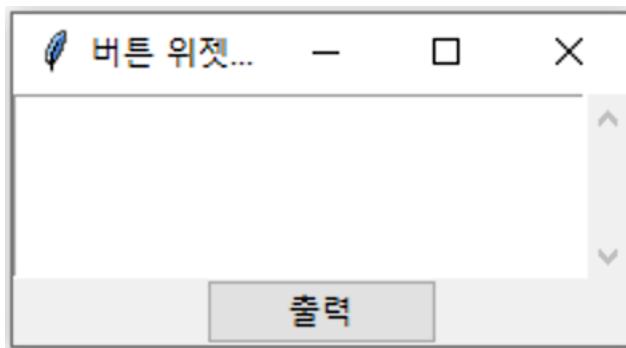
- ◆ 텍스트 위젯과 스크롤바를 하나로 묶어 제공하는 위젯
- ◆ `tkinter.scrolledtext` 모듈의 `ScrolledText` 클래스로 위젯 객체 생성
 - 사용법은 기본적으로 `Text` 위젯과 같음
 - 입력된 행의 수가 `height`로 지정한 값보다 많아지면 자동으로 세로 스크롤바가 나타남

스크롤 텍스트 위젯의 사용 예

코드 13-13

스크롤 텍스트 위젯의 사용 예

```
01 import tkinter as tk
02 from tkinter import ttk
03 from tkinter import scrolledtext
04 # 사용자 정의 함수부
05 def buildGUI():
06     global text_area
07     text_area = scrolledtext.ScrolledText(win,
08                                         width=30, height=5, wrap=tk.WORD)
09 // 이하 생략
```



체크 버튼 위젯: ttk.Checkbutton

- ◆ 다중 선택 입력을 받을 때 사용되는 위젯
 - 상태를 갖는 버튼
- ◆ ttk 모듈의 Checkbutton 클래스로 위젯 생성

```
tk변수객체이름 = IntVar()  
객체이름 = Checkbutton(부모윈도우, text='문자열',  
                         variable=tk변수객체이름,  
                         command=핸들러함수이름)
```

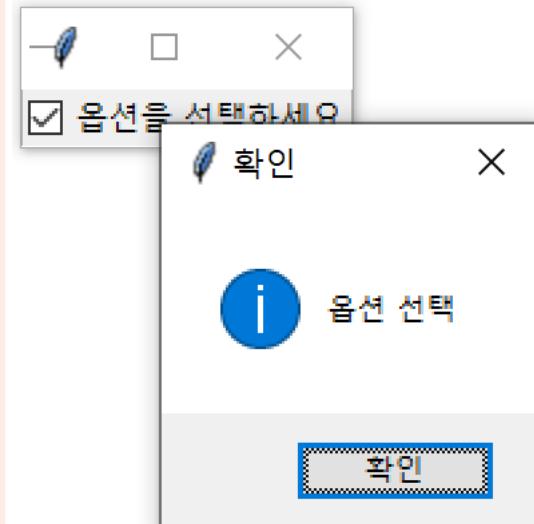
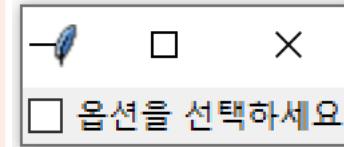
- 키워드 인수 variable에 정수형의 tk변수 객체 지정
 - 1 : 체크가 선택된 상태
 - 0 : 체크가 해제된 상태

체크 버튼 위젯의 사용 예

코드 13-14

체크버튼 위젯의 사용 예

```
01 import tkinter as tk
02 from tkinter import ttk
03 from tkinter import messagebox
04
05 # 사용자 정의 함수부
06 def buildGUI():
07     global check          # 이벤트 핸들러에서 접근 위해
08     check = tk.IntVar()    # 체크 상태 저장 위해
09     check_btn = ttk.Checkbutton(win, text='옵션을 선택하세요',
10                                variable=check,
11                                command=open_dialog_box)
12
13     check_btn.pack()
14
15 def open_dialog_box():
16     if check.get() == 1:
17         messagebox.showinfo('확인', '옵션 선택')
18     else:
19         messagebox.showinfo('확인', '옵션 해제')
20 :
```



라디오 버튼 위젯: ttk.Radiobutton

- ◆ 배타적인 선택 입력을 받을 때 사용되는 위젯
- ◆ ttk 모듈의 Radiobutton 클래스로 위젯 생성

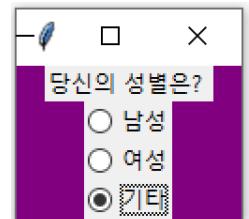
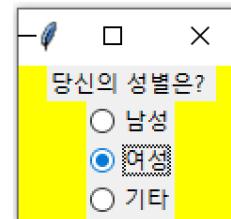
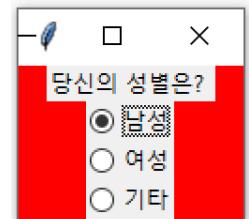
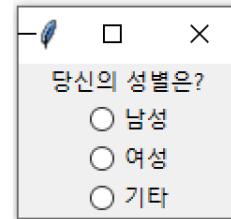
```
tk변수객체이름 = IntVar()  
객체이름 = Radiobutton(부모윈도우, text='문자열',  
                         variable=tk변수객체이름,  
                         value=각버튼의식별아이디값,  
                         command=핸들러함수이름)
```

- 키워드 인수 value에 각 버튼 식별값(아이디) 지정
- 키워드 인수 variable에 정수형의 tk변수 객체 지정
 - 한 묶음의 라디오 버튼은 공통의 tk변수 지정
 - 선택한 버튼의 value값이 저장됨

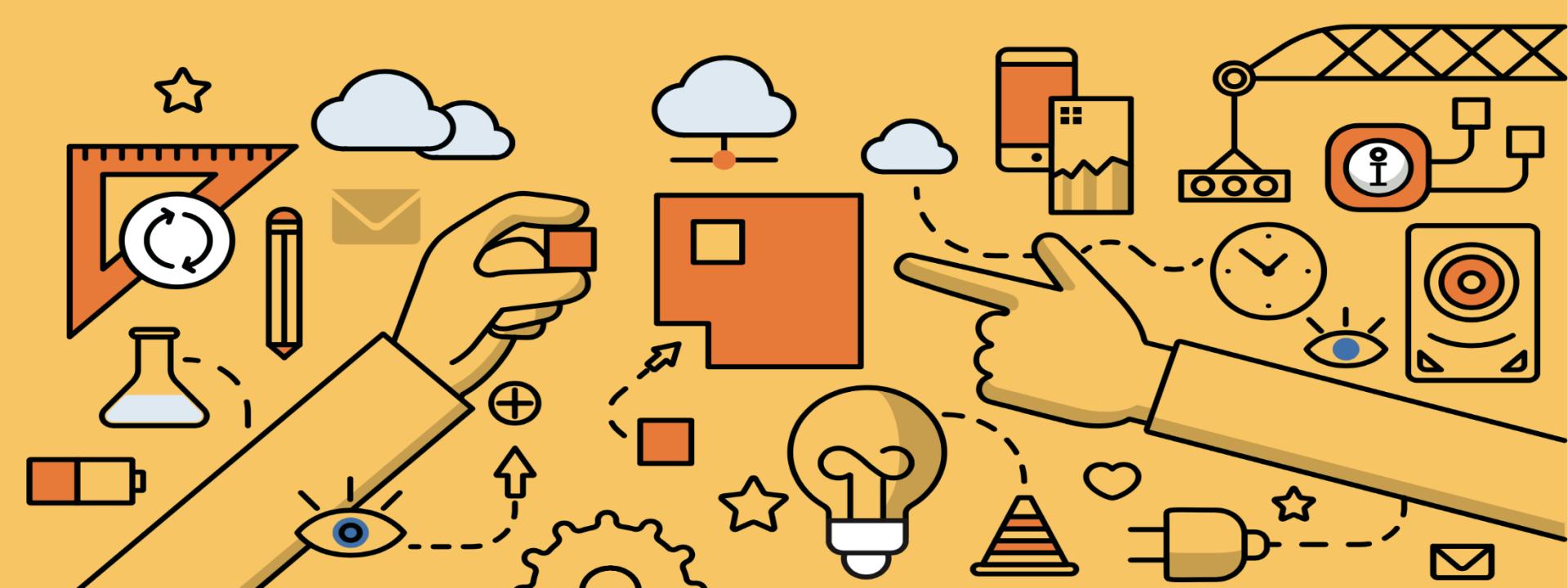
라디오버튼 위젯의 사용 예

코드 13-15 라디오버튼 위젯의 사용 예

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 GENDERS = [ '남성', '여성', '기타' ]
05 COLORS = [ 'red', 'yellow', 'purple' ]
06
07 # 사용자 정의 함수부
08 def buildGUI():
09     text_label = ttk.Label(win, text='당신의 성별은? ')
10     text_label.pack()
11
12     global gender          # 이벤트 핸들러에서 접근 위해
13     gender = tk.IntVar()    # 체크 상태 저장 위해
14     for i in range(3):
15         radio_btn = ttk.Radiobutton(win,
16                                     text=GENDERS[i],
17                                     value=i,
18                                     variable=gender,
19                                     command=radio_btn_hadler
20     )
21     radio_btn.pack()
22
23     gender.set(-1)        # 라디오 버튼 선택 지우기
```



```
24
25 def radio_btn_hadler():
26     choice = gender.get()
27     win.configure(background=COLORS[choice])
28
29 # 주 프로그램부
30 win = tk.Tk()          # 기본 윈도우 객체 반환
31 win.title('버튼 위젯 예')
32 buildGUI()            # 화면 구성
33 win.mainloop()         # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



04 | 위젯의 배치

배치 방법

- ◆ pack() 메서드 이용

- 현재 위젯을 블럭으로 구성하여 상위 컨테이너에 부착

- ◆ place() 메서드 이용

- 현재 위젯을 상위 컨테이너의 절대 위치에 부착

- ◆ grid() 메서드 이용

- 현재 위젯을 표 구조를 갖는 상위 컨테이너에 부착

pack() 메서드를 이용한 배치

◆ 각 위젯을 하나의 블록으로 구성하여 배치

- 위젯 객체 생성 시 첫 번째 인수로 지정된 윈도우에 pack()를 호출한 순서대로 나열해 배치
- side 키워드 인수로 나열 방향 지정
 - 기본값: tk.TOP

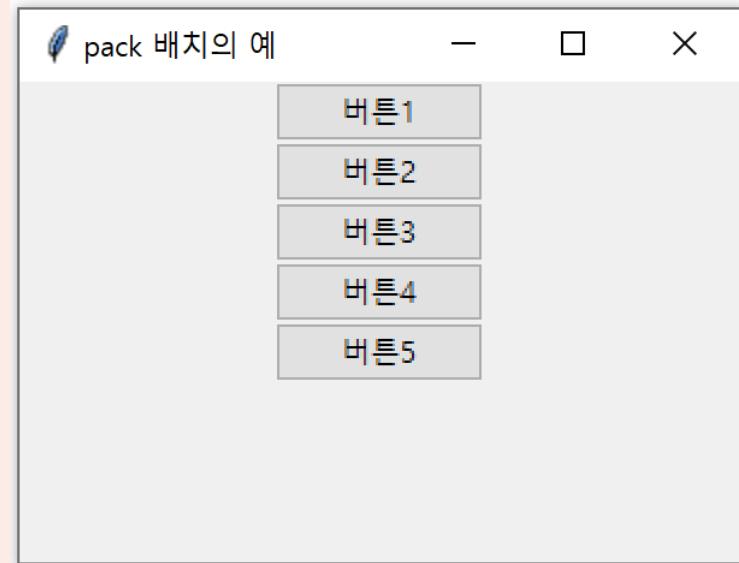


그림 13.11 식당에서의 자리 배치

pack()을 이용한 위젯 배치 예

코드 13-16 pack()을 이용한 위젯 배치

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     btn1 = ttk.Button(win, text='버튼1')
07     btn2 = ttk.Button(win, text='버튼2')
08     btn3 = ttk.Button(win, text='버튼3')
09     btn4 = ttk.Button(win, text='버튼4')
10     btn5 = ttk.Button(win, text='버튼5')
11
12     btn1.pack()      # btn1.pack(side=tk.TOP)과 동일
13     btn2.pack()
14     btn3.pack()
15     btn4.pack()
16     btn5.pack()
17
18 # 주 프로그램부
19 win = tk.Tk()          # 기본 윈도우 객체 반환
20 win.title('pack() 배치의 예')
21 win.geometry('300x200')
22 buildGUI()            # 화면 구성
23 win.mainloop()         # 윈도우에서 다양한 이벤트 처리 시작을 지시
```



위젯의 배치 방향 지정

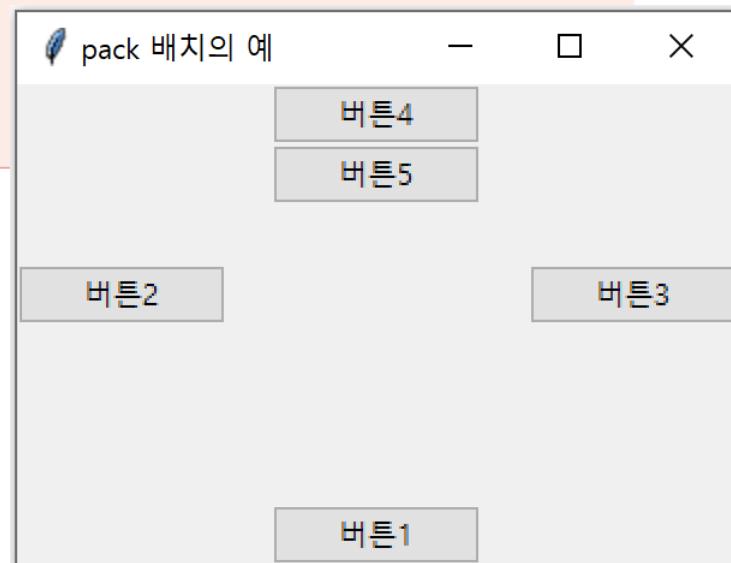
◆ 키워드 인수 side로 지정

#

코드 13-17

pack()에 side 키워드 인수를 지정한 예 [코드 13-16 수정]

```
12  btn1.pack(side=tk.BOTTOM)
13  btn2.pack(side=tk.LEFT)
14  btn3.pack(side=tk.RIGHT)
15  btn4.pack()
16  btn5.pack(side=tk.TOP)
```



위젯의 여백의 종류

◆ 안쪽 여백

- 위젯의 경계면부터 그 위젯의 내용까지의 여백
- 다음의 키워드 인수로 픽셀 단위로 지정
 - ipadx : 위젯의 왼쪽과 오른쪽의 안쪽 여백
 - ipady : 위쪽과 아래쪽의 안쪽 여백

◆ 바깥 여백

- 위젯의 경계면에서부터 컨테이너나 다른 위젯까지의 경계
- 다음의 키워드 인수로 픽셀 단위로 지정
 - padx : 위젯의 왼쪽과 오른쪽의 바깥 여백
 - pady : 위쪽과 아래쪽의 바깥 여백

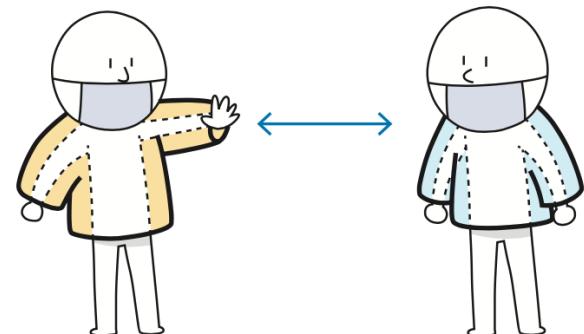


그림 13.12 안쪽여백과 바깥여백

위젯의 여백 지정 예

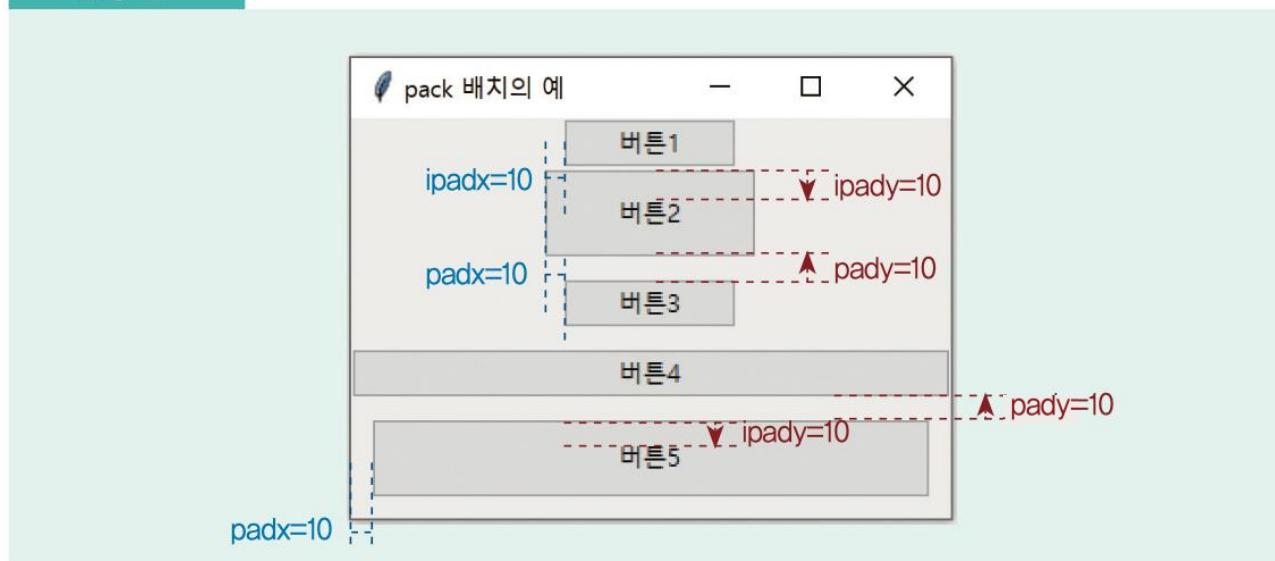
코드 13-18 pack()로 위젯의 여백 지정하기

```
12  btn1.pack()  
13  btn2.pack(ipadx=10, ipady=10)  
14  btn3.pack(padx=10, pady=10)  
15  btn4.pack(fill=tk.X)  
16  btn5.pack(fill=tk.X, padx=10, pady=10, ipadx=10, ipady=10)
```

컨테이너의 너비에 위젯의 크기를 맞춤



실행 결과



grid() 메서드를 이용한 배치

- ◆ 위젯 배치를 위해 화면 영역을 격자(셀)로 나누어 배치하는 방법
 - row와 column 키워드 인수를 통해 위젯을 어느 셀에 배치할지를 지정
 - 기본값 0

```
위젯이름.grid(row=배치할_셀의_로우값, column=배치할_셀의_컬럼값)
```

		키워드 인수 column의 값				
		0	1	2	...	c
키워드 인수 row의 값	0	(0, 0)	(0, 1)	(0, 2)	...	(0, c)
	1	(1, 0)	(1, 1)	(1, 2)	...	(1, c)
	2	(2, 0)	(2, 1)	(2, 2)	...	(2, c)

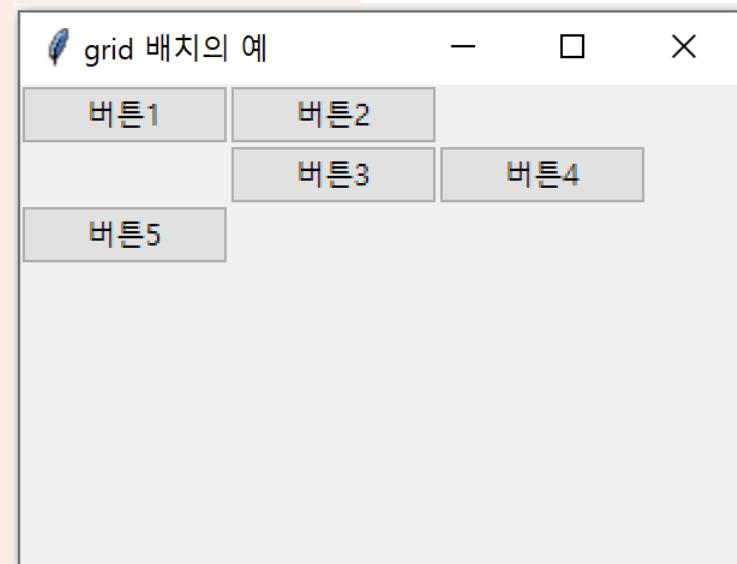
	r	(r, 0)	(r, 1)	(r, 2)	...	(r, c)

grid()를 이용한 위젯 배치 예

코드 13-19

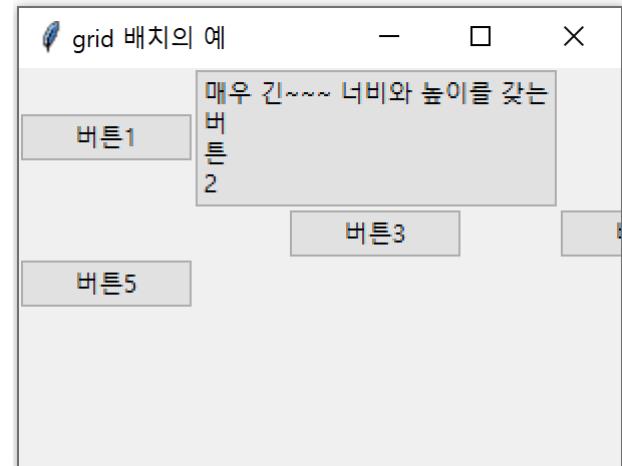
grid()을 이용한 위젯 배치

```
01 import tkinter as tk  
02 from tkinter import ttk  
03  
04 # 사용자 정의 함수부  
05 def buildGUI():  
06     btn1 = ttk.Button(win, text='버튼1')  
07     btn2 = ttk.Button(win, text='버튼2')  
08     btn3 = ttk.Button(win, text='버튼3')  
09     btn4 = ttk.Button(win, text='버튼4')  
10     btn5 = ttk.Button(win, text='버튼5')  
11  
12     btn1.grid(row=0, column=0)    # btn1.grid()와 동일  
13     btn2.grid(row=0, column=1)  
14     btn3.grid(row=1, column=1)  
15     btn4.grid(row=1, column=2)  
16     btn5.grid(row=2, column=0)  
  
:
```



grid()를 이용한 배치의 한계

- ◆ 각 셀의 크기는 그곳에 배치되는 위젯의 크기에 따라 달라짐
 - 셀의 너비: 같은 column에 배치된 위젯 중 가장 넓은 위젯에 의해 결정
 - 셀의 높이: 같은 row에 배치된 위젯 중 가장 높이가 높은 위젯에 의해 결정



코드 13-20

grid()을 이용한 다양한 크기의 위젯 배치

07

```
btn2 = ttk.Button(win, text='매우 긴~~~ 너비와 높이를 갖는\n버튼2')
```

셀 내의 정렬 방식 지정

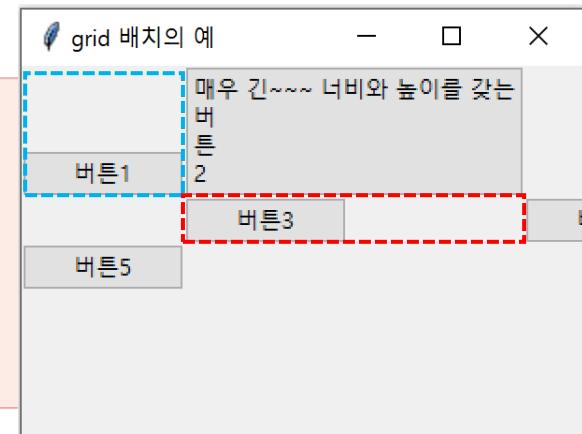
- ◆ 위젯이 할당된 셀에 여백이 생기는 경우 위젯은 기본적으로 해당 셀의 가운데에 배치됨
 - 셀 내에 여백이 존재할 경우 sticky 키워드 인수로 위젯을 특정 방향으로 치중해 배치할 수 있음

```
위젯명.grid(sticky=정렬방향)      # n, e, s, w, nw, ne, sw, se
```

코드 13-21

sticky 키워드 인수의 사용 예

```
12  btn1.grid(row=0, column=0, sticky='se')
13  btn2.grid(row=0, column=1)
14  btn3.grid(row=1, column=1, sticky='w')
15  btn4.grid(row=1, column=2)
16  btn5.grid(row=2, column=0)
```



셀 합치기

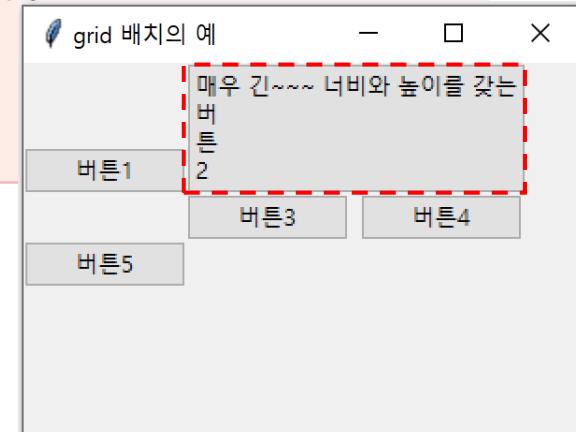
- ◆ 연속된 셀들을 합쳐 하나의 위젯을 배치
 - colspan: 가로 방향의 셀들을 합치는 효과
 - rowspan: 세로 방향의 셀들을 합치는 효과

#

코드 13-22

셀 합치기

```
12     btn1.grid(row=0, column=0, sticky='se')
13     btn2.grid(row=0, column=1, columnspan=2)
14     btn3.grid(row=1, column=1, sticky='w')
15     btn4.grid(row=1, column=2)
16     btn5.grid(row=2, column=0)
```

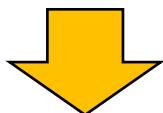


여전히 복잡한 배치

- ◆ 하나의 배치 메서드만으로 다양한 너비를 갖는 위젯들을 배치하는 것은 복잡하고 한계 존재
 - place() 메서드를 이용
 - 화면 좌측 상단 꼭지점 좌표값을 원점으로 하여 배치할 위젯의 절대 좌표값을 지정
 - 직관적으로 각 위젯을 원하는 곳에 단순하게 배치 가능
 - 위젯을 추가하거나 위젯의 위치를 수정/삭제 할 때마다 전체적으로 각 위젯의 배치 좌표를 수정해야 하는 번거로움 발생
 - 윈도우에 각 위젯을 좀 더 효율적으로 배치하기 위해 tkinter에서는 Frame이라는 컨테이너를 추가로 제공

Frame 위젯을 이용한 화면 구성

- ◆ 기본적으로 하나의 컨테이너 내에서는 하나의 배치 메서드만 사용 가능



- ◆ 복잡한 화면 구성을 위해 ttk 모듈의 Frame 클래스 이용
 - 비슷한 위치에 같은 방식으로 배치될 위젯은 프레임 위젯으로 묶자
 - 실생활의 쟁반과 비슷한 역할을 수행



그림 13.13 개인 쟁반을 이용한 상차림

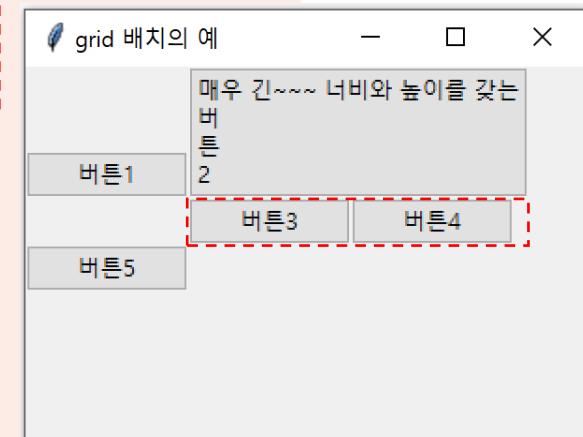
Frame을 이용한 배치 예

#

코드 13-23

Frame을 이용한 배치 [코드 13-19 수정]

```
05 def buildGUI():
06     btn1 = ttk.Button(win, text='버튼1')
07     btn2 = ttk.Button(win,
08                       text='매우 긴~~~ 너비와 높이를 갖는\n버튼2')
09     btn_group = ttk.Frame(win)
10     btn3 = ttk.Button(btn_group, text='버튼3')
11     btn4 = ttk.Button(btn_group, text='버튼4')
12     btn5 = ttk.Button(win, text='버튼5')
13
14     btn1.grid(row=0, column=0, sticky='se')
15     btn2.grid(row=0, column=1)
16     btn3.pack(side=tk.LEFT)
17     btn4.pack(side=tk.LEFT)
18     btn_group.grid(row=1, column=1, sticky='w')
19     btn5.grid(row=2, column=0)
```





05 | 객체지향적으로 구성하기

절차지향적으로 구성된 GUI

재인용

엔트리 위젯의 사용 예 (코드 13-11)

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 # 사용자 정의 함수부
05 def buildGUI():
06     global text_label
07     text_label = ttk.Label(win, text='안녕하세요')
08
09     global name           # 이벤트 핸들러에서 접근 위해
10     name = tk.StringVar() # 엔트리 위젯에서 문자열 저장 위해
11     # ... 생략 ...
12
13
14
15
16
17
18
19
20
21
22
23 def input_btn_handler(): # 버튼 클릭에 대한 이벤트 핸들러 함수
24     text_label.configure(text='반가워요, ' + name.get())
25     name.set('')          # 작은따옴표 두 개로 빈문자열을 인수로 지정
26
27 # 주 프로그램부
28 win = tk.Tk()          # 기본 윈도우 객체 반환
29 win.title('버튼 위젯 예')
30 buildGUI()              # 화면 구성
31 win.mainloop()          # 윈도우에서 다양한 이벤트 처리 시작을 지시
```

다른 함수에서 선언된 위젯을 참조위해
전역변수 사용

객체지향적인 윈도우 구성

- ◆ 나만의 사용자 윈도우를 객체로 구성
 - 객체 생성시 자동으로 윈도우 구성하기
 - 전역변수 대신 객체 속성 사용
 - 화면 구성 및 이벤트 핸들러 메서드 추가하기

```
import tkinter as tk

# 사용자정의 클래스부
class MyWindow:
    def __init__(self):
        self.__window = tk.Tk()      # 기본 윈도우 반환
        self.__window.title("프로그램 제목")

        self.buildGUI()            # 화면 구성 및 이벤트 처리

    def buildGUI(self) :
        # 화면 구성
        pass

    def start(self):
        self.__window.mainloop()   # 윈도우에서의 다양한 이벤트 처리

# 주 프로그램
mywin = MyWindow()
mywin.start()
```

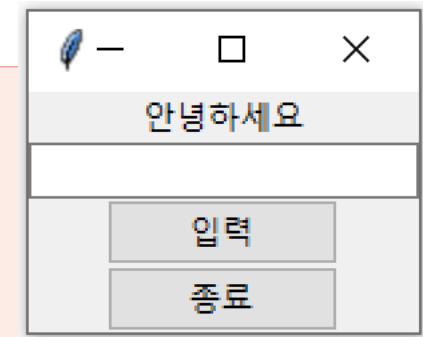
객체지향적으로 구성된 GUI(1/2)

#

코드 13-24

객체지향적으로 GUI 구성

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 # 클래스 정의부
05 class SayHelloWin:
06     def __init__(self):
07         self.win = tk.Tk()      # 기본 윈도우 객체 생성
08         self.win.title('버튼 위젯 예-OOP')
09         self.__buildGUI()      # 화면 구성
10
11     def __buildGUI(self):
12         self.text_label = ttk.Label(self.win,
13                                     text='안녕하세요')
14
15         self.name = tk.StringVar()
16         input_entry = ttk.Entry(self.win,
17                                textvariable=self.name)
18
19         input_btn = ttk.Button(self.win, text='입력',
20                               command=self.__input_btn_handler)
```

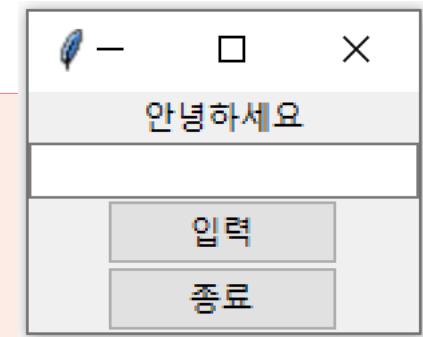


객체지향적으로 구성된 GUI(2/2)

코드 13-24

객체지향적으로 GUI 구성 (cont)

```
18     quit_btn = ttk.Button(self.win, text='종료',
                           command=self.win.destroy)
19
20     self.text_label.pack()
21     input_entry.pack()
22     input_btn.pack()
23     quit_btn.pack()
24
25     def __input_btn_handler(self):
26         self.text_label.configure(
27             text='반가워요, ' + self.name.get())
28         self.name.set('')
29
30     # 주 프로그램부
31     hello = SayHelloWin()
32     hello.win.mainloop()
```



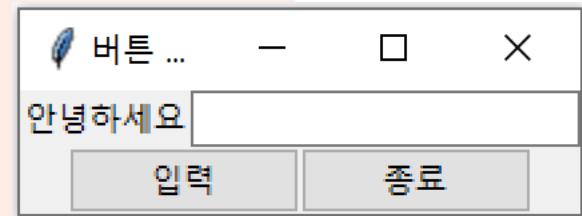
좀 더 복잡한 화면 구성을 갖는 경우의 구성

코드 13-25

객체지향적으로 GUI 구성

```
01 import tkinter as tk
02 from tkinter import ttk
03
04 # 클래스 정의부
05 class SayHelloWin :
06     def __init__(self):
07         self.win = tk.Tk()      # 기본 윈도우 객체 생성
08         self.win.title('버튼 위젯 예-OOP')
09         self.__buildGUI()      # 화면 구성
10
11     def __buildGUI(self):
12         self.__create_input_frame().pack()
13         self.__create_button_frame().pack()
14
15     def __create_input_frame(self):
16         frame = ttk.Frame(self.win)
17
18         self.text_label = ttk.Label(frame, text='안녕하세요')
19
20         self.name = tk.StringVar()
21         input_entry = ttk.Entry(frame, textvariable=self.name)
22
23         self.text_label.grid()
24         input_entry.grid(row=0, column=1)
25
26         return frame
```

코드 13-24를 수정



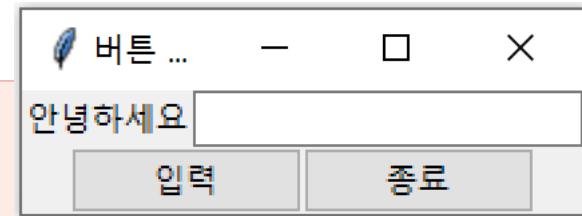
좀 더 복잡한 화면 구성을 갖는 경우의 구성

#

코드 13-25

객체지향적으로 GUI 구성 (cont)

```
27  
28     def __create_button_frame(self):  
29         frame = ttk.Frame(self.win)  
30  
31         input_btn = ttk.Button(frame, text='입력', command=self.__input_btn_handler)  
32         quit_btn = ttk.Button(frame, text='종료', command=self.win.destroy)  
33  
34         input_btn.pack(side=tk.LEFT)  
35         quit_btn.pack(side=tk.LEFT)  
36  
37         return frame  
38  
39     def __input_btn_handler(self):  
40         self.text_label.configure(  
41             text='반가워요, ' + self.name.get())  
42         self.name.set('')  
43  
44     # 주 프로그램부  
45     hello = SayHelloWin()  
46     hello.win.mainloop()
```



연습문제 13.1-13.3

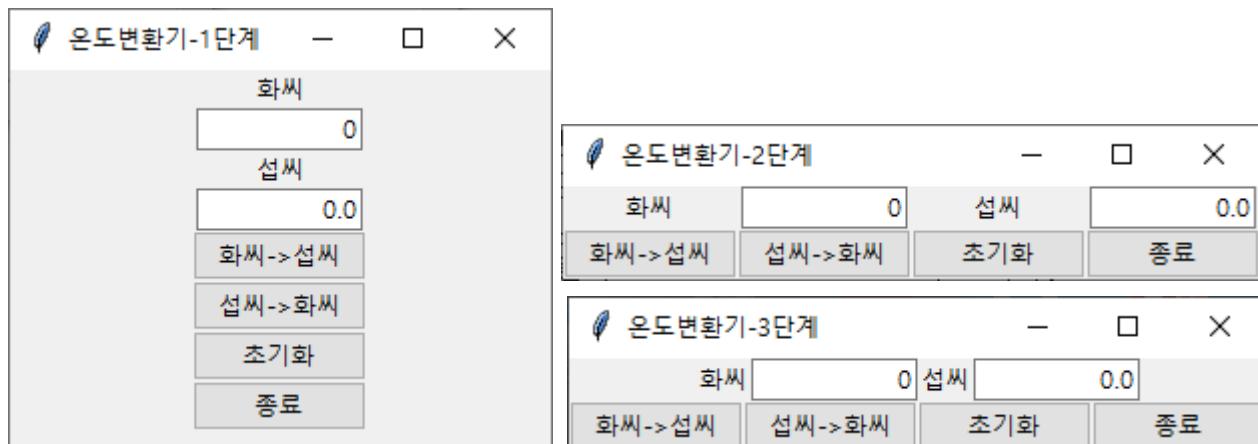
- ◆ 섭씨온도-화씨온도 변환을 처리하는 GUI 프로그램 작성
 - 화씨 필드에 값 입력 → 화씨->섭씨 버튼 클릭 → 섭씨 필드에 값 출력
 $(\text{화씨온도} - 32) \div 1.8 = \text{섭씨온도}$
 - 섭씨 필드에 값 입력 → 섭씨->화씨 버튼 클릭 → 화씨 필드에 값 출력
 $(\text{섭씨온도} \times 1.8) + 32 = \text{화씨온도}$
 - 초기화 버튼 클릭 → 섭씨/화씨 필드에 출력된 값 삭제
 - 종료 버튼 클릭 → 종료

◆ 2단계

- `grid()` 메서드로 배치

◆ 3단계

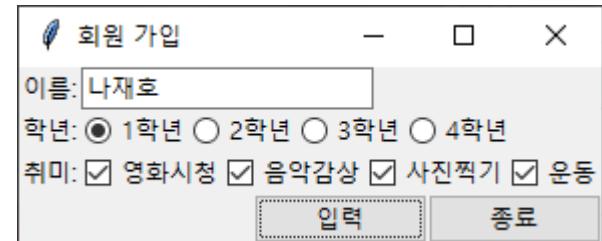
- 프레임 위젯 사용



연습문제 13.4

◆ 사용자 정보 입력을 처리하는 GUI 프로그램 작성

- 이름은 한 줄로 된 문자열로 입력
- 학년은 배타적으로 하나만 선택
- 취미는 다중 선택 가능
- 입력 버튼이 눌리면 콘솔창에 오른쪽과 같이 결과값 출력
- 종료 버튼이 눌리면 종료

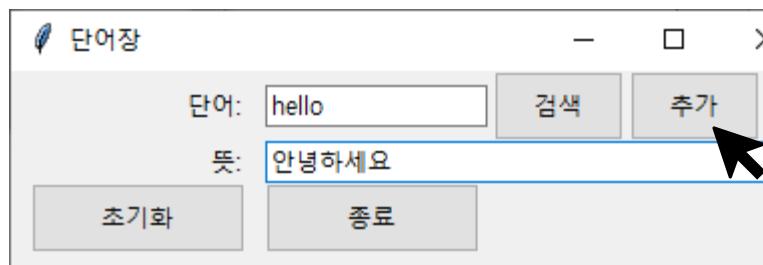


나재호
1
영화시청
음악감상
사진찍기
운동

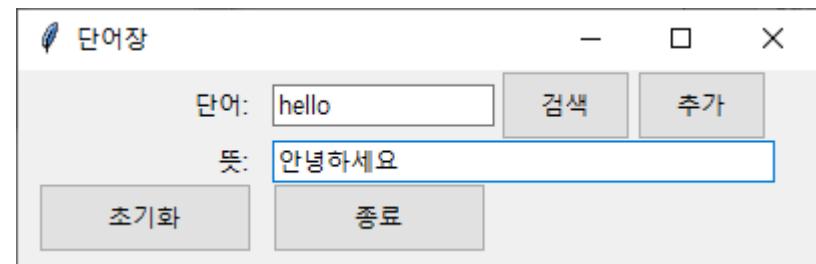
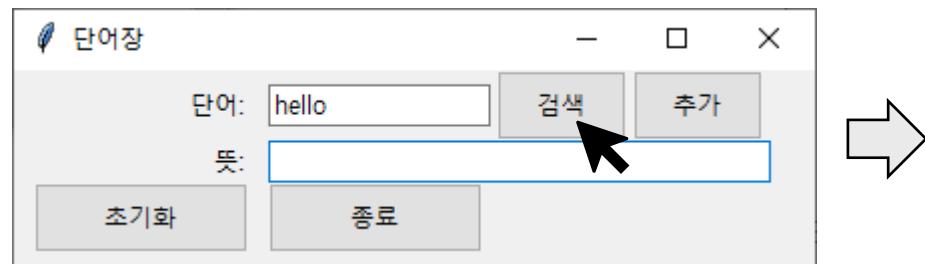
연습문제 13.5

◆ 단어장 GUI 프로그램 작성

- 단어, 뜻을 입력한 후 추가 버튼을 누르면 아래와 같이 창을 띄움



- 초기화 버튼을 누르면 단어, 뜻 필드의 내용 삭제
- 이후 단어를 입력 후 검색 버튼을 누르면, 이 단어의 뜻을 찾아 출력



연습문제 13.5

◆ 단어장 GUI 프로그램 작성

- 단어장에 해당 단어가 없으면 오류 메시지 출력



- 종료 버튼이 눌리면 종료
- 키, 값 쌍을 함께 저장하는 딕셔너리를 이용하여 구현

◆ 구현 옵션

- 프로그램 시작시 words.txt 파일이 존재하면, 그 내용을 불러들여 words에 유지. 없으면 빈 딕셔너리 준비
- 프로그램 종료시 딕셔너리의 내용을 words.txt에 기록

학기 마무리

- ◆ 후속 프로그래밍 과목들
 - C프로그래밍1, 2 – 절차적 프로그래밍
 - 객체지향프로그래밍,
고급객체지향프로그래밍 – OOP 개념
 - ◆ 파이썬을 사용할 가능성이 있는 과목들
 - 일부 수학 과목들 – numpy, matplotlib 등을 사용하여 수학 문제를 풀고 그래프를 그림
 - 자료구조 – 분반에 따라 파이썬 사용 가능
 - 인공지능 관련 과목들 – 최근 관련 연구들은 대부분 파이썬 사용
 - 설계 과목들 – 경우에 따라 파이썬으로 개인/팀 프로젝트 수행 가능
 - ◆ 시험 준비 잘 하시고, 부족한 부분은 방학 때 복습하기 바랍니다.

