

9장 tkinter를 이용한 GUI 프로그래밍

첫 번째 tkinter 프로그램

```
from tkinter import *  
  
window = Tk()  
  
label = Label(window, text="Hello World!")  
label.pack()  
  
window.mainloop()
```



버튼과 이벤트 처리

```
from tkinter import *
```

```
window = Tk()
```

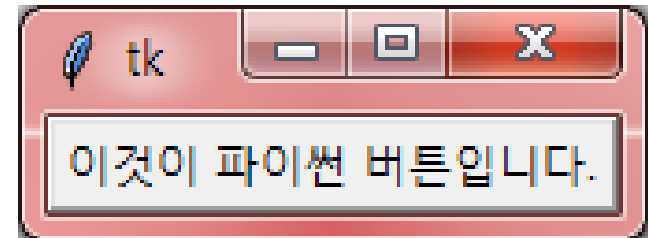
원도우 창을 하나 생성하고
(아직 보이지는 않음)

```
b1 = Button(window, text="이것이 파이썬 버튼입니다.")  
b1.pack()
```

그 위에서 동작할 요소들을
프로그래밍 한 후에

```
window.mainloop()
```

원도우 창을 실행시킴
(X버튼이 눌릴 때까지 위의
동작이 계속됨)



배치 관리자 소개

```
from tkinter import *
```

```
window = Tk()
```

```
b1 = Button(window, text="첫번째 버튼")
```

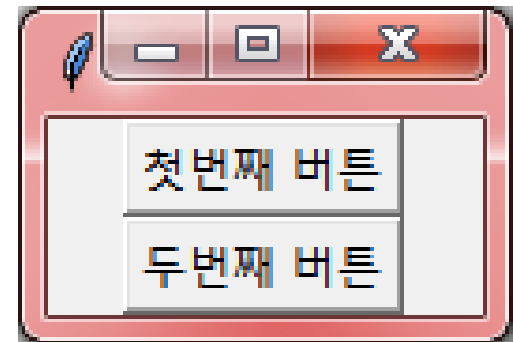
```
b2 = Button(window, text="두번째 버튼")
```

```
b1.pack()
```

```
b2.pack()
```

```
window.mainloop()
```

← Button 만들 때 연결해 놓았던 window
창에 배치하는 작업임.



배치 관리자 소개

```
from tkinter import *
```

```
window = Tk()
```

```
b1 = Button(window, text="첫번째 버튼")
```

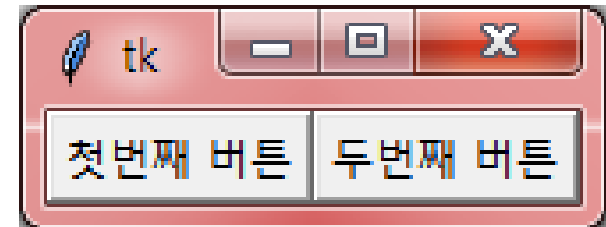
```
b2 = Button(window, text="두번째 버튼")
```

```
b1.pack(side=LEFT)
```

```
b2.pack(side=LEFT)
```

나란히 배치

```
window.mainloop()
```



배치 관리자 소개

```
from tkinter import *
```

```
window = Tk()
```

```
b1 = Button(window, text="첫번째 버튼")
```

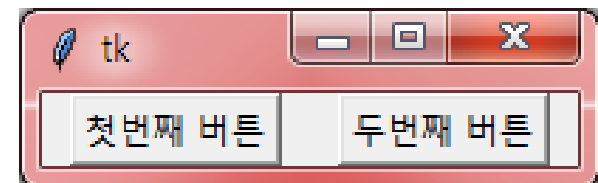
```
b2 = Button(window, text="두번째 버튼")
```

```
b1.pack(side=LEFT, padx=10)
```

```
b2.pack(side=LEFT, padx=10)
```

```
window.mainloop()
```

← 나란히 간격을 두고 배치



버튼의 텍스트 변경

```
from tkinter import *
```

```
window = Tk()
```

```
b1 = Button(window, text="첫번째 버튼")
```

```
b2 = Button(window, text="두번째 버튼")
```

```
b1.pack(side=LEFT, padx=10)
```

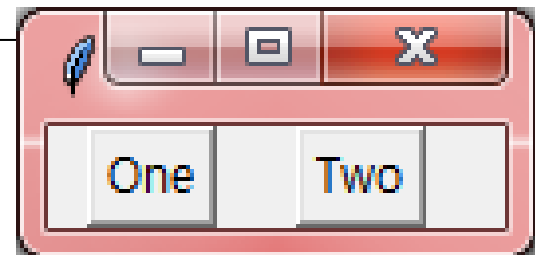
```
b2.pack(side=LEFT, padx=10)
```

```
b1["text"] = "One"
```

```
b2["text"] = "Two"
```

```
window.mainloop()
```

버튼의 글씨를
바꾸고 싶을 때



버튼의 이벤트를 처리하려면

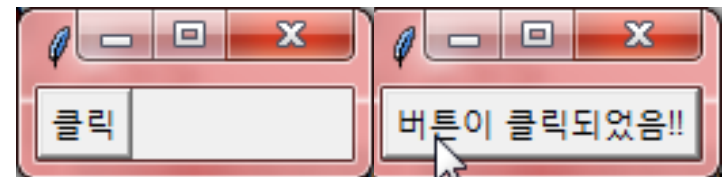
```
from tkinter import *
```

```
def my_func():  
    button["text"] = "버튼이 클릭되었음!"
```

버튼이 눌렸을 때 처리할
함수를 여기에 적음



```
window = Tk()  
button = Button(window, text="클릭", command=my_func)  
button.pack(side=LEFT)  
  
window.mainloop()
```

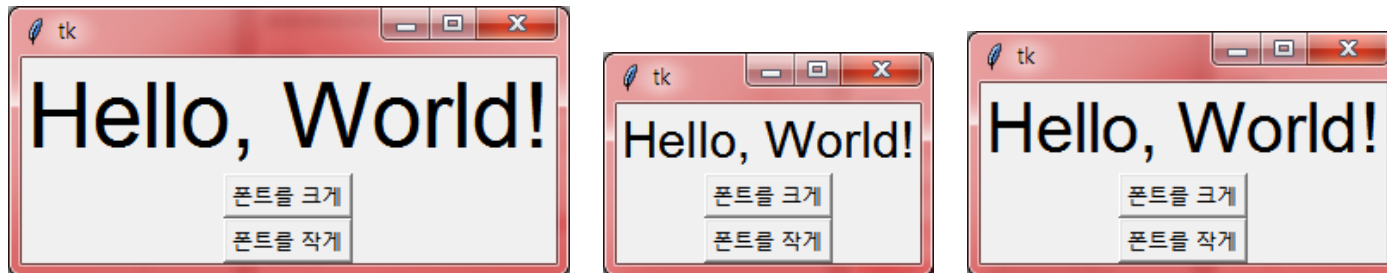


버튼의 색깔을 바꾸려면

```
from tkinter import *  
window = Tk()  
button = Button(window, text="버튼을 클릭하세요")  
button.pack()  
button["fg"] = "yellow" ← 버튼 글씨색  
button["bg"] = "green" ← 버튼 배경색
```



연습문제1. 글씨를 크게! 작게! (1/3)



버튼을 눌러서 글씨를 확대하거나 작게하는
프로그램을 만들어보자

연습문제1 . 글씨를 크게! 작게! (2/3)

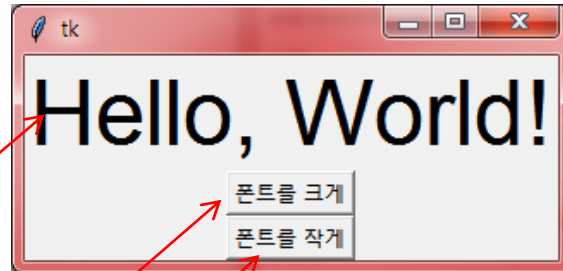
```
from tkinter import *
from tkinter.font import *

wn=Tk()
my_font = Font(family="Helvetica", size=12) ← 폰트 초기값 설정

def make_bigger():
    my_font['size'] += 2
    # size = my_font['size']
    # my_font.configure(size=size + 2) ← 버튼이 눌렀을 때 실행될 함수 정의

def make_smaller():
    my_font['size'] -=2
    # size = my_font['size']
    # my_font.configure(size=size - 2)
```

연습문제1. 글씨를 크게! 작게!



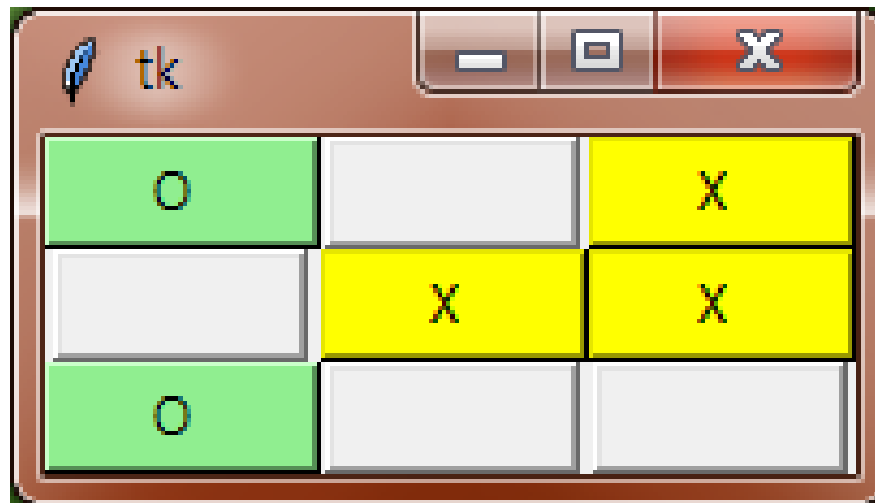
```
my_text = Label(wn, text="Hello, World!", font=my_font)
my_text.pack()

big_butt = Button(wn, text="폰트를 크게", command=make_bigger)
small_butt = Button(wn, text="폰트를 작게", command=make_smaller)
big_butt.pack()
small_butt.pack()

wn.mainloop()
```

연습문제 2: Tic-Tac-Toe

- Tic-Tac-Toe는 3×3칸을 가지는 게임판을 만들고, 경기자 2명이 동그라미 심볼(O)와 가위표 심볼(X)을 고른다. 경기자는 번갈아가며 게임판에 동그라미나 가위표를 놓는다. 가로, 세로, 대각선으로 동일한 심볼을 먼저 만들면 승리하게 된다.



연습문제 2: Tic-Tae-Toe

```
from tkinter import *
```

```
def checked(i):
```

버튼이 눌렸을 때 실행할 함수

```
    global player, color, blank, semi_blank  
    button = blist[i]
```

```
    if button["text"] != blank:  
        return
```

버튼의 상태를 바꿔 줌

```
    button["text"] = semi_blank + player + semi_blank  
    button["bg"] = color
```

```
    if player=="X":  
        player, color = "O", "lightgreen"  
    else :  
        player, color = "X", "yellow"
```

연습문제 2: Tic-Tae-Toe

```
wn = Tk()
```

```
blank = "          "
```


```
semi_blank = "      "
```

```
player = "X"
```

```
color = "yellow"
```

```
blist = []
```

이벤트 처리 함수에
인자 넘기는 방법 : 람다
함수 이용



```
for i in range(9):
```

```
    b = Button(wn, text=blank, command=lambda k=i: checked(k))
```

```
    b.grid(row=i//3, column=i%3)  버튼 배치
```

```
    blist.append(b)
```



```
wn.mainloop()
```

버튼을 순서대로 리스트에 들고 있음. 각
인덱스에 해당하는 버튼을 접근할 때
필요함.

엔트리 위젯

전체적인 구조



```
w = Entry(parent , option, ... )
```


예제

```
from tkinter import *
```

```
window = Tk()
```

```
Label(window , text="이름").grid(row=0)
```

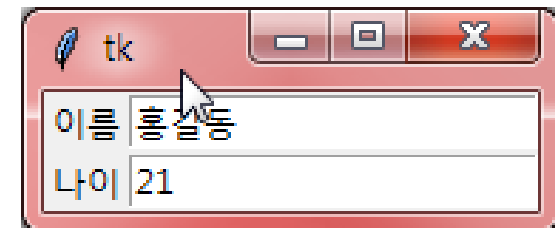
```
Label(window, text="나이").grid(row=1)
```

```
e1 = Entry(window) ← 사용자 데이터 입력할 수 있도록  
e2 = Entry(window) 엔트리를 생성함.
```

```
e1.grid(row=0, column=1)
```

```
e2.grid(row=1, column=1)
```

```
window.mainloop( )
```



예제

```
from tkinter import *
```

사용자가 입력한 데이터를 가져오는 함수

```
def show():
```

```
    print("이름: %s\n나이: %s" % (e1.get(), e2.get()))
```

```
parent = Tk()
```

```
Label(parent, text="이름").grid(row=0)
```

```
Label(parent, text="나이").grid(row=1)
```

```
e1 = Entry(parent)
```

```
e2 = Entry(parent)
```

```
e1.grid(row=0, column=1)
```

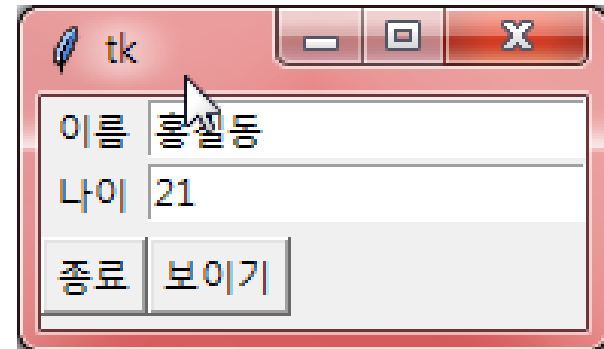
```
e2.grid(row=1, column=1)
```

```
Button(parent, text='보이기', command=show).grid(row=3,  
column=1, sticky=W, pady=4)
```

```
Button(parent, text='종료', command=parent.quit).grid(row=3,  
column=0, sticky=W, pady=4)
```

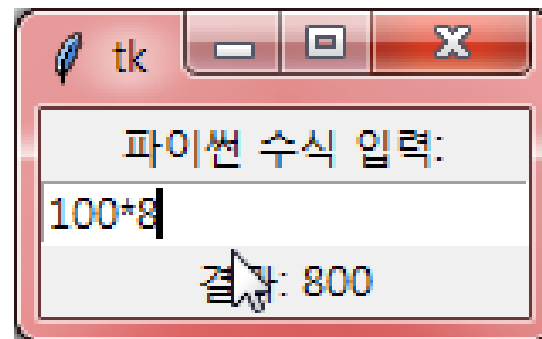
```
mainloop( )
```

버튼이 눌리면 호출되는 함수



Lab: 계산기

- 수식을 텍스트로 입력하면 이것을 평가하고 그 결과를 출력할 수 있는 간단한 계산기를 작성하여 본다. 수식의 형식은 파이썬과 동일하여야 한다. `eval()` 함수를 사용하여 사용자가 입력한 수식을 계산할 수 있다.



Solution

```
from tkinter import *
from math import *

def calculate(event):
    label.configure(text = "결과: " + str(eval(entry.get()))))

window = Tk()

Label(window, text="파이썬 수식 입력:").pack()
entry = Entry(window)
entry.bind("<Return>", calculate) ← 사용자가 return 키를
entry.pack()                      누르면 calculate 함수를
                                   부르도록 설정

label = Label(window, text = "결과:")
label.pack()

w.mainloop()
```

배치 관리자

■ Grid

- 격자 배치 관리자(**grid geometry manager**)는 테이블 형태의 배치

■ Pack

- 압축 배치 관리자(**pack geometry manager**)는 위젯들을 부모 위젯 안에 압축

■ Place

- 절대 배치 관리자(**place geometry manager**)는 주어진 위치에 위젯을 배치

다른 종류의 배치관리자는 혼합하여 사용할 수 없음.

혼합하여 사용하고 싶은 경우 **Frame** 위젯을 이용해야 함.

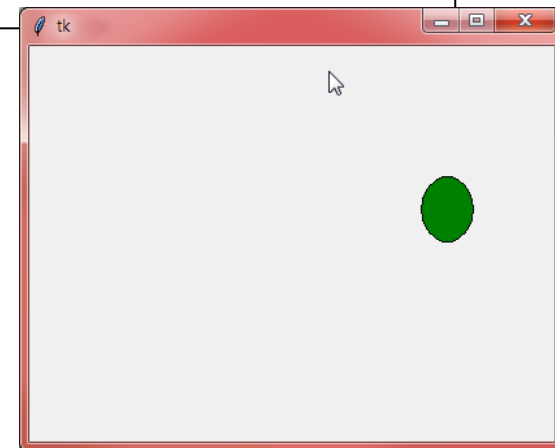
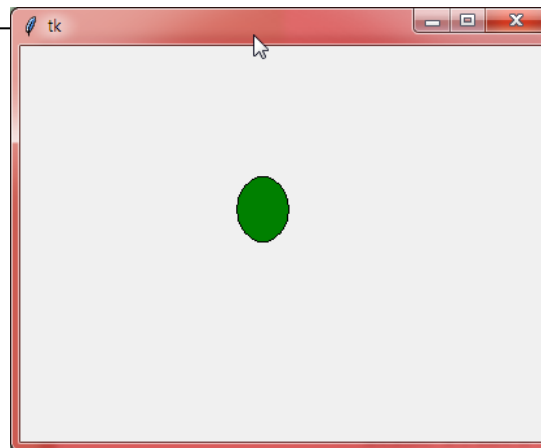
tkinter를 이용한 그래픽

- Canvas 위젯을 이용하여 그래프를 그린다거나 그래픽 에디터를 작성할 수도 있고 많은 종류의 커스텀 위젯을 작성할 수 있다.

애니메이션을 만들어 보자.

```
import time
from tkinter import *
window = Tk()
canvas = Canvas(window, width=400, height=300)
canvas.pack()
id=canvas.create_oval(10, 100, 50, 150, fill="green")

for i in range(100):
    canvas.move(id, 3, 0)
    window.update()
    time.sleep(0.05)
```



Q & A

