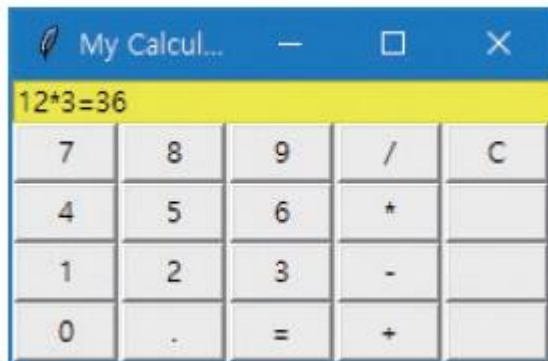


9장 G U I 프로그래밍

GUI 앱 vs 텍스트 앱

- 그래픽 사용자 인터페이스(GUI: graphical user interface)



첫 번째 수를 입력하시오:45

두 번째 수를 입력하시오:6

연산의 종류를 입력하시오(+, -, *, /):-
연산의 결과는 39 입니다

tkinter란?

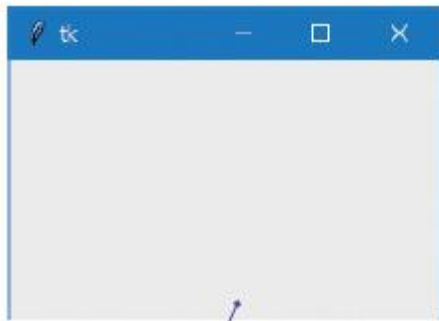
- tkinter는 파이썬에서 **그래픽 사용자 인터페이스(GUI: graphical user interface)**를 개발할 때 필요한 모듈



tkinter 시작하기

Widget: 작은 장치

- 윈도우(window)를 생성하고 여기에 필요한 위젯들을 추가한다.



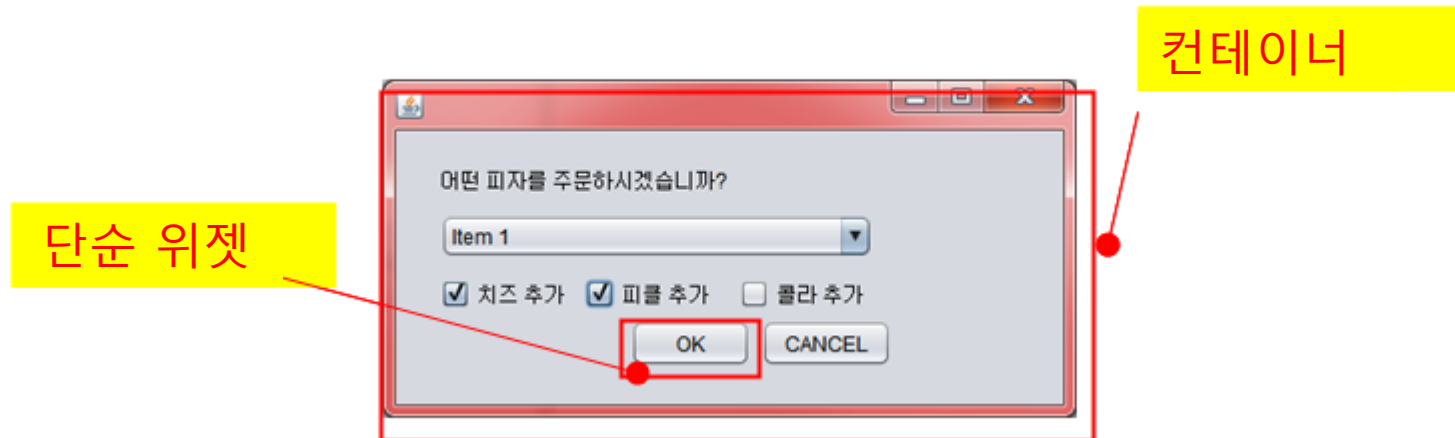
(1) 비어 있는 윈도우를 생성한다.



(2) 윈도우에 필요한 위젯을 추가한다.

단순 위젯과 컨테이너 위젯

- 단순 위젯: Button, Canvas, Checkbutton, Entry, Label, Message 등이 여기에 속한다.
- 컨테이너 컴포넌트: 다른 컴포넌트를 안에 포함할 수 있는 컴포넌트로서 Frame, Toplevel, LabelFrame, PanedWindow 등이 여기에 속한다.



레이블이 있는 윈도우를 생성해보자

- 하나의 레이블이 있는 윈도우를 생성해보자
- Label 위젯은 입력 기능이 없고, 텍스트나 그림을 보여 줌

```
from tkinter import *
```

```
window = Tk()
```

```
label = Label(window, text="Hello tkinter")
```

```
label.pack()
```

```
window.mainloop()
```

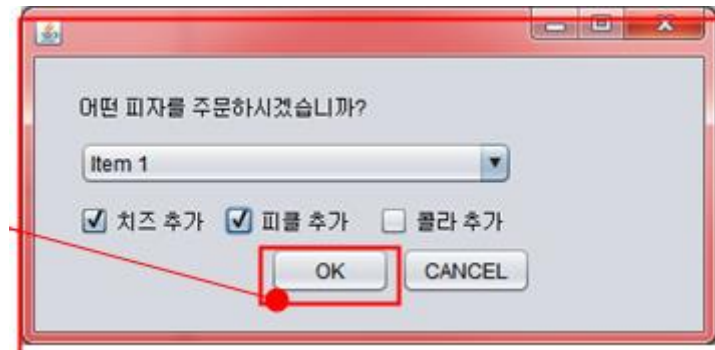
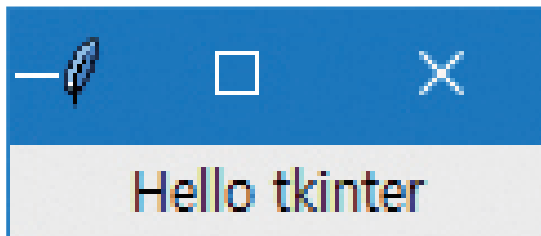
```
# tkinter 모듈을 포함
```

```
# 루트 윈도우를 생성
```

```
# 레이블 위젯을 생성
```

```
# 레이블 위젯을 윈도우에 배치
```

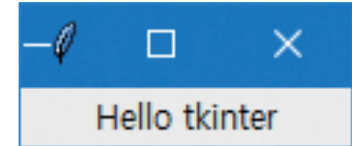
```
# 윈도우가 사용자 동작을 대기
```



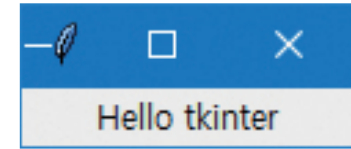
분석

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

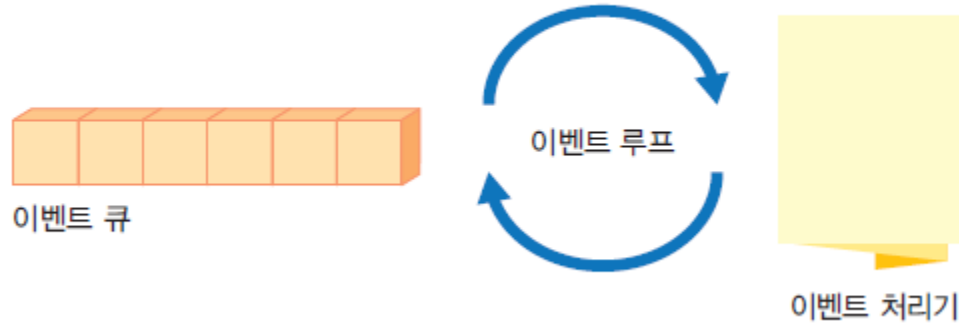
- from tkinter import *
 - ▣ tkinter 모듈을 포함시키는 것이다.
- window = Tk()
 - ▣ Tk()을 호출하면 Tk 클래스의 객체가 생성되면서 화면에 하나의 윈도우가 생성된다.
- label = Label(window, text="Hello tkinter")
 - ▣ 레이블 위젯을 생성한다.
- label.pack()
 - ▣ pack()은 압축 배치 관리자를 이용하여서 레이블을 컨테이너에 위치시킨다.



분석

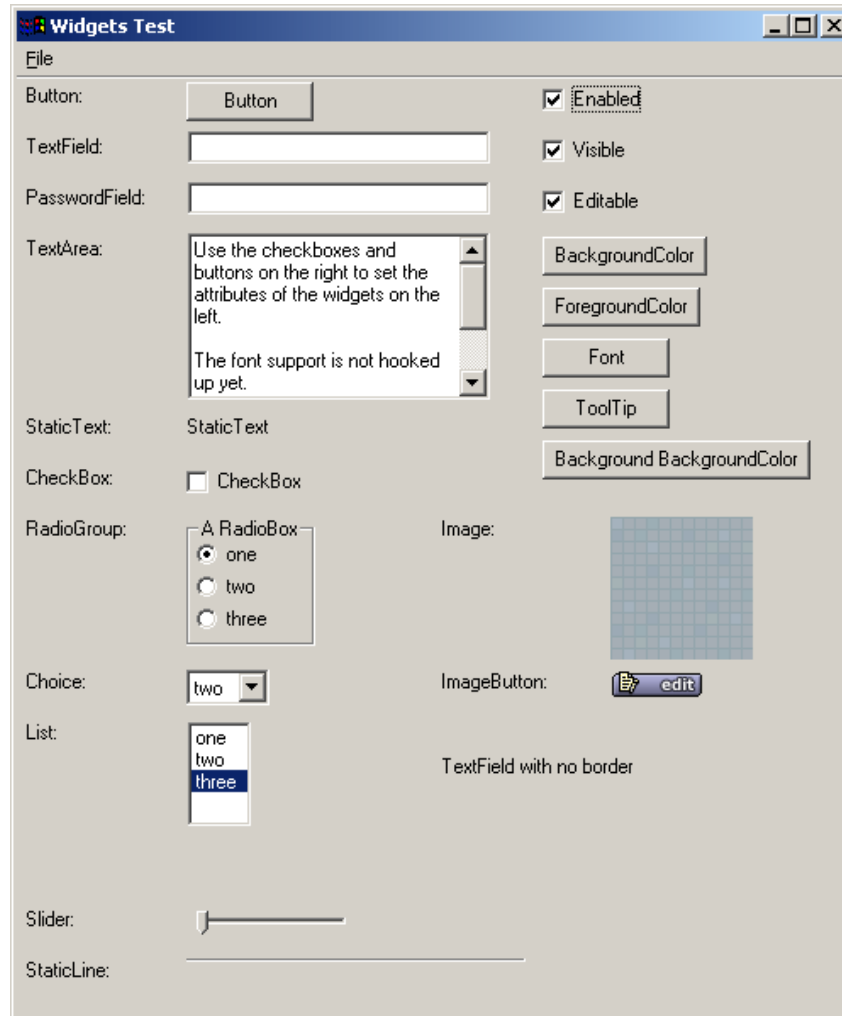


- `window.mainloop()`
 - ▣ 루트 윈도우의 `mainloop()`는 이벤트 처리 루프로서 사용자로부터 오는 마우스나 키보드 이벤트를 처리한다



tkinter의 위젯들

Widget: 작은 장치

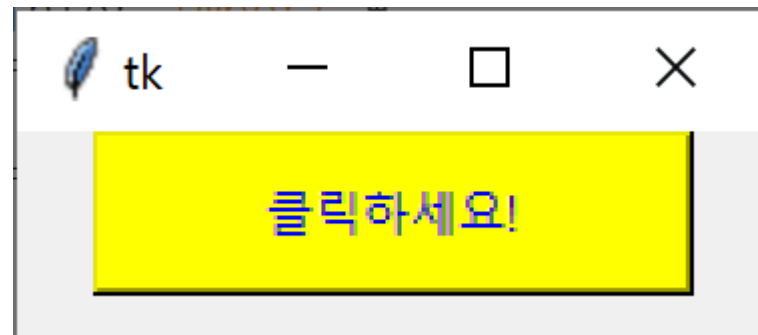


| 위젯 | 설명 |
|-------------|---|
| Button | 간단한 버튼으로 명령을 수행할 때 사용된다. |
| Canvas | 화면에 무언가를 그릴 때 사용한다. |
| Checkbutton | 2가지의 구별되는 값을 가지는 변수를 표현한다. |
| Entry | 한 줄의 텍스트를 입력받는 필드이다. |
| Frame | 컨테이너 클래스이다. 프레임은 경계선과 배경을 가지고 있다. 다른 위젯들을 그룹핑하는데 사용된다. |
| Label | 텍스트나 이미지를 표시한다. |
| Listbox | 선택 사항을 표시한다. |
| Menu | 메뉴를 표시한다. 풀다운 메뉴나 팝업 메뉴가 가능하다. |
| Menubutton | 메뉴 버튼이다. 풀다운 메뉴가 가능하다. |
| Message | 텍스트를 표시한다. 레이블 위젯과 비슷하다. 하지만 자동적으로 주어진 크기로 텍스트를 축소할 수 있다. |
| Radiobutton | 여러 값을 가질 수 있는 변수를 표시한다. |
| Scale | 슬라이더를 끌어서 수치를 입력하는데 사용된다. |
| Scrollbar | 캔버스, 엔트리, 리스트 박스, 텍스트 위젯을 위한 스크롤 바를 제공한다. |
| Text | 형식을 가지는 텍스트를 표시한다. 여러 가지 스타일과 속성으로 텍스트를 표시할 수 있다. |
| Toplevel | 최상위 윈도우로 표시되는 독립적인 컨테이너 위젯이다. |
| LabelFrame | 경계선과 제목을 가지는 프레임 위젯의 변형이다. |
| PanedWindow | 자식 위젯들을 크기조절이 가능한 패널로 관리하는 컨테이너 위젯이다. |
| Spinbox | 특정한 범위에서 값을 선택하는 엔트리 위젯의 변형 |

버튼 위젯

픽셀 단위
텍스트 단위
행 단위

```
from tkinter import *  
window = Tk()  
  
button = Button(window, text="클릭하세요!",  
                 bg="yellow", fg="blue",      # 전경색과 배경색 설정  
                 width=20, height=2         # 크기 설정  
)  
  
button.pack()  
window.mainloop()
```



엔트리 위젯

```
from tkinter import *  
window = Tk()  
  
entry = Entry(window, fg="black", bg="yellow", width=80)  
  
entry.pack()  
window.mainloop()
```



배치 관리자

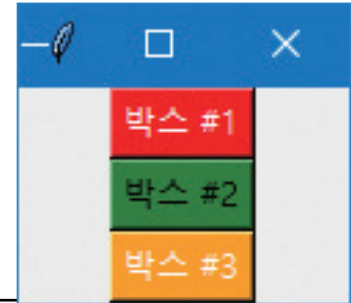
- 압축(pack) 배치 관리자
- 격자(grid) 배치 관리자
- 절대(place) 배치 관리자



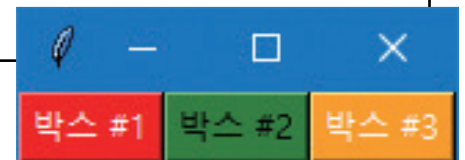
압축 배치 관리자 (pack geometry manager)

- 위젯을 최대한 압축하여서 배치한다. pack() 사용.

```
from tkinter import *  
window = Tk()  
  
label = Label(window, text="박스 #1", bg="red", fg="white")  
label.pack()  
Label(window, text="박스 #2", bg="green", fg="black").pack()  
Label(window, text="박스 #3", bg="blue", fg="white").pack()  
  
window.mainloop()
```



```
from tkinter import *  
window = Tk()                                     # TOP, BOTTOM, LEFT, RIGHT  
  
Button(window, text="박스 #1", bg="red", fg="white").pack(side=LEFT)  
Button(window, text="박스 #2", bg="green", fg="black").pack(side=LEFT)  
Button(window, text="박스 #3", bg="orange", fg="white").pack(side=LEFT)  
  
window.mainloop()
```



격자 배치 관리자 (grid geometry manager)

- 위젯을 테이블 형태로 배치한다. grid() 사용.

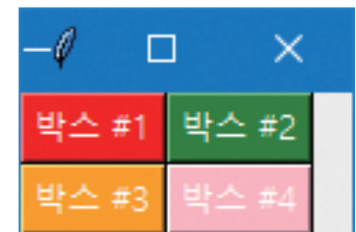
```
from tkinter import *  
window = Tk()
```

```
b1 = Button(window, text="박스 #1", bg="red", fg="white")  
b2 = Button(window, text="박스 #2", bg="green", fg="white")  
b3 = Button(window, text="박스 #3", bg="orange", fg="white")  
b4 = Button(window, text="박스 #4", bg="pink", fg="white")
```

```
b1.grid(row=0, column=0)    # 0행 0열  
b2.grid(row=0, column=1)    # 0행 1열  
b3.grid(row=1, column=0)    # 1행 0열  
b4.grid(row=1, column=1)    # 1행 1열
```

```
window.mainloop()
```

| | Column 0 | Column 1 | Column 2 |
|-------|----------|----------|----------|
| Row 0 | | | |
| Row 1 | | | |
| Row 2 | | | |
| Row 3 | | | |



절대 위치 배치 관리자(place geometry manager)

- 절대 위치를 사용하여 위젯을 배치한다. `place()` 사용.
- `x,y`는 픽셀 단위. 왼쪽위의 좌표가 (0,0)

```
from tkinter import *  
  
window = Tk()  
  
b1 = Button(window, text="박스 #1", bg="red", fg="white")  
b1.place(x=0, y=0)  
b2 = Button(window, text="박스 #2", bg="green", fg="black")  
b2.place(x=20, y=30)  
b3 = Button(window, text="박스 #3", bg="orange", fg="white")  
b3.place(x=40, y=60)  
  
window.mainloop()
```



여러 배치 관리자 혼용하기

```
from tkinter import *

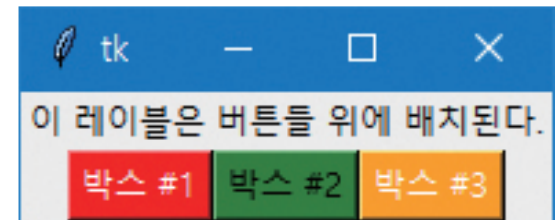
window = Tk()
f = Frame(window)

b1 = Button(f, text="박스 #1", bg="red", fg="white")
b2 = Button(f, text="박스 #2", bg="green", fg="black")
b3 = Button(f, text="박스 #3", bg="orange", fg="white")
b1.pack(side=LEFT)
b2.pack(side=LEFT)
b3.pack(side=LEFT)

l = Label(window, text="이 레이블은 버튼들 위에 배치된다.")

l.pack()
f.pack()

window.mainloop()
```



윈도우의 크기와 위젯의 크기 설정하기

픽셀 단위
텍스트 단위
행 단위

```
from tkinter import *  
  
window = Tk()  
window.geometry("600x100")           # Width x Height  
  
Button(window, text="박스 #1", width=10, height=1).pack()  
Button(window, text="박스 #2", width=10, height=1).pack()  
Button(window, text="박스 #3", width=10, height=1).pack()  
  
window.mainloop()
```



Lab: 온도 변환기

```
from tkinter import *
```

```
window = Tk()
```

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

```
Label(window, text="섭씨").grid(row=1, column=0)
```

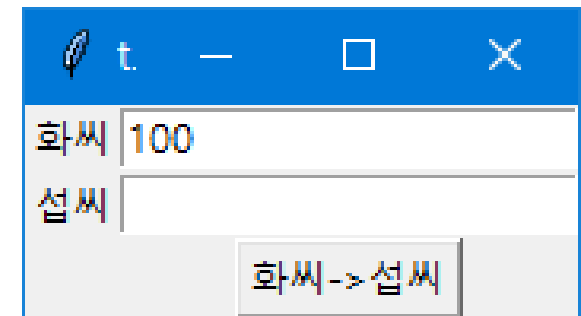
```
e1 = Entry(window).grid(row=0, column=1)
```

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

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

```
window.mainloop()
```

| | Column 0 | Column 1 | Column 2 |
|-------|----------|----------|----------|
| Row 0 | | | |
| Row 1 | | | |
| Row 2 | | | |
| Row 3 | | | |



Lab: 격자 배치 관리자 실습

```
from tkinter import *
```

```
window = Tk()
```

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

```
Label(window, text="높이").grid(row=1, column=0)
```

```
e1 = Entry(window)
```

```
e2 = Entry(window)
```

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

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

```
photo = PhotoImage(file="d://fig3.png")
```

```
label = Label(window, image=photo)
```

```
label.grid(row=0, column=2, columnspan=2, rowspan=2)
```

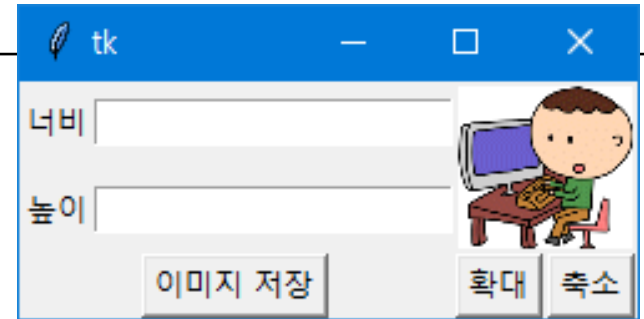
```
Button(window, text='이미지 저장').grid(row=2, column=0, columnspan=2)
```

```
# 두 번째 버튼과 세 번째 버튼은 2열과 3열에 표시한다.
```

```
Button(window, text='확대').grid(row=2, column=2)
```

```
Button(window, text='축소').grid(row=2, column=3)
```

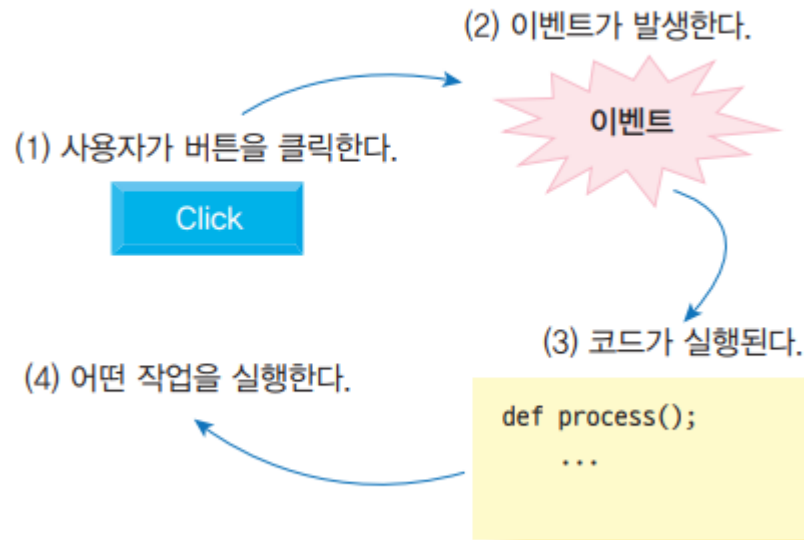
```
window.mainloop()
```



| | 0 | 1 | 2 | 3 |
|---|--------|--------|--------|--------|
| 0 | label | entry1 | image | |
| 1 | label | entry2 | | |
| 2 | button | | button | button |

```
from tkinter import *  
help(PhotoImage)
```

버튼 이벤트 처리



이벤트가 발생하면 지정된 함수가 호출되는 개념입니다.



Syntax: 버튼 이벤트 처리

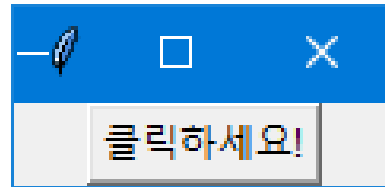
형식 Button = Button(window, command=함수 이름)|

예 Button(window, text="클릭하세요!", command=process)

버튼에서 이벤트가 발생하면
호출되는 함수 등록

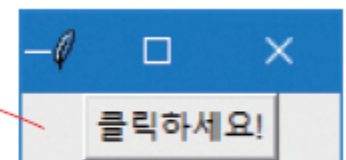
예제

```
from tkinter import *  
  
def process():  
    print("버튼이 클릭되었습니다.")  
  
window = Tk()  
button = Button(window, text="클릭하세요!", command=process)  
button.pack()  
  
window.mainloop()
```



실행결과

버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.



Lab: 카운터 만들기

- 레이블과 버튼을 사용하여 간단한 카운터를 작성하여 보자. 증가 버튼을 누르면 카운터가 1 증가된다.

```
from tkinter import *
```

```
window = Tk()
```

```
counter = 0
```

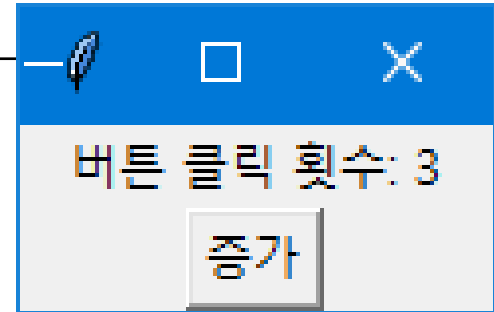
```
def clicked():  
    global counter  
    counter += 1  
    label['text'] = '버튼 클릭 횟수: ' + str(counter)
```

```
label = Label(window, text="아직 눌러지지 않음")
```

```
label.pack()
```

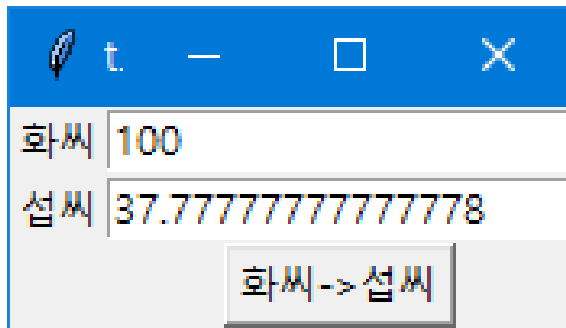
```
button = Button(window, text="증가", command=clicked).pack()
```

```
window.mainloop()
```



Lab: 온도 변환기 #2

- 온도 변환 프로그램의 위젯들을 다음과 같이 배치하고 "화씨->섭씨" 버튼을 누르면 입력한 화씨 온도가 섭씨온도로 변환되어서 나타나게 하라.



| | |
|----------------------------|-------------------|
| 화씨 | 100 |
| 섭씨 | 37.77777777777778 |
| <button>화씨->섭씨</button> | |

Sol:

```
from tkinter import *
```

```
# 이벤트 처리 함수를 정의한다.
```

```
def process():
```

```
    tf = float(e1.get())
```

```
    tc = (tf-32.0)*5.0/9.0
```

```
    e2.delete(0, END)
```

```
    e2.insert(0, str(tc))
```

```
# e1에서 문자열을 읽어서 부동소수점으로 변경
```

```
# 화씨 온도를 섭씨 온도로 변환한다.
```

```
# 처음부터 끝까지 지운다.
```

```
# tc 변수의 값을 문자열로 변환하여 추가한다.
```

```
window = Tk()
```

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

```
Label(window, text="섭씨").grid(row=1, column=0)
```

```
e1 = Entry(window)
```

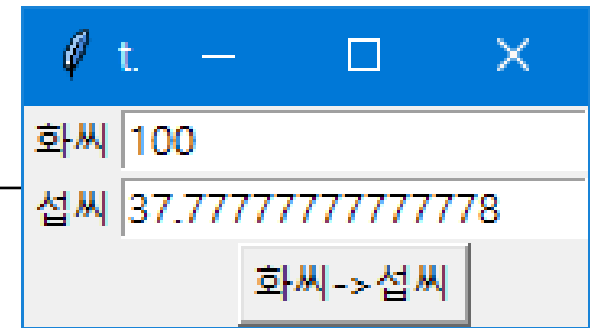
```
e2 = Entry(window)
```

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

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

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

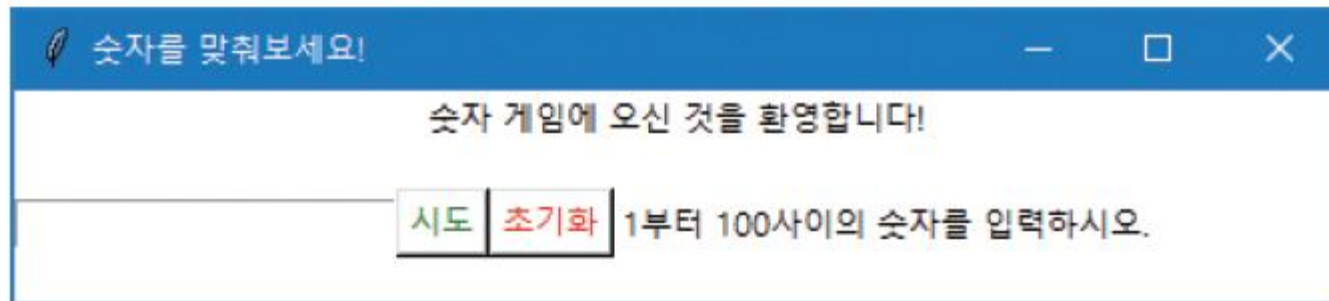
```
window.mainloop()
```



숫자 추측 게임

- 사용자가 컴퓨터가 생성한 숫자(1부터 100사이의 난수)를 알아맞히는 게임을 그래픽 사용자 인터페이스를 사용하여 제작해보자.

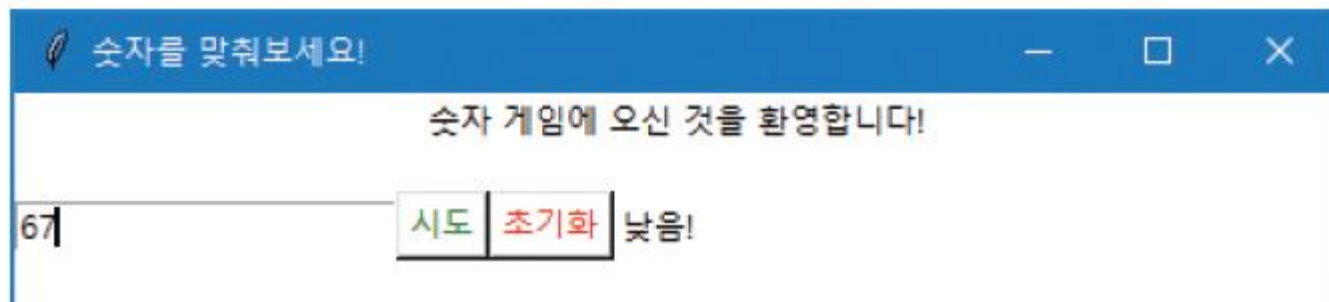
📌 실행결과



숫자를 맞춰보세요!

숫자 게임에 오신 것을 환영합니다!

시도 초기화 1부터 100사이의 숫자를 입력하십시오.

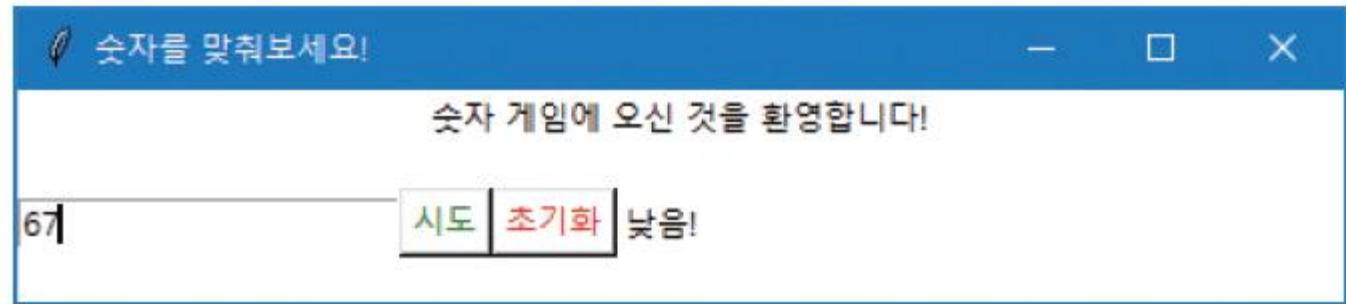


숫자를 맞춰보세요!

숫자 게임에 오신 것을 환영합니다!

67 시도 초기화 낮음!

Sol:



```
from tkinter import *  
import random
```

```
answer = random.randint(1,100)          # 정답을 1에서 100 사이의 난수로 설정한다.
```

```
def guessing():  
    guess = int(guessField.get()) # 텍스트 필드에서 사용자가 입력한 값을  
                                # 가져온다.
```

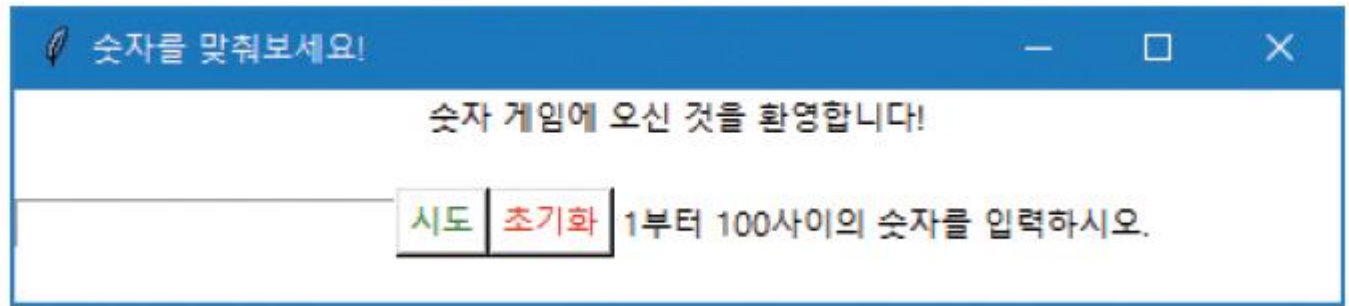
```
    if guess > answer:  
        msg = "높음!"
```

```
    elif guess < answer:  
        msg = "낮음!"
```

```
    else:  
        msg = "정답!"
```

```
    resultLabel["text"] = msg          # 메시지를 출력한다.  
    guessField.delete(0, 5)
```

Sol:

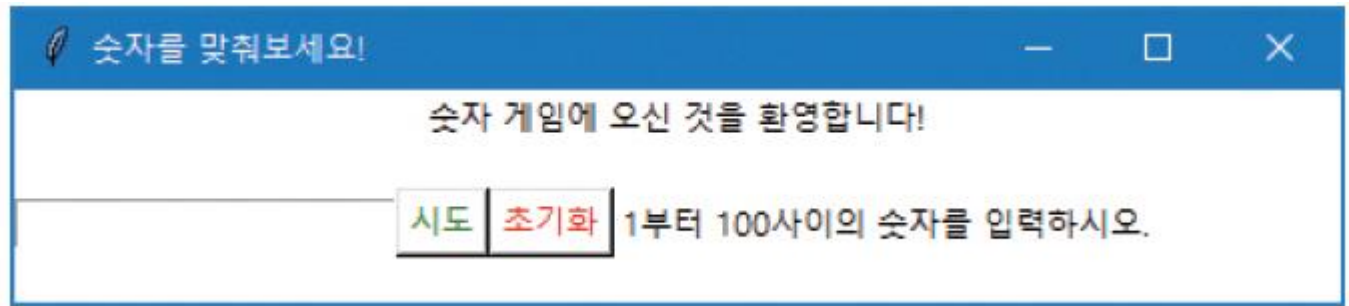


```
def reset():                                     # 정답을 다시 설정한다.
    global answer
    answer = random.randint(1,100)
    resultLabel["text"] = "다시 한번 하세요!"

window = Tk()
window.configure(bg="white")
window.title("숫자를 맞춰보세요!")

window.geometry("500x80")
titleLabel = Label(window, text="숫자 게임에 오신 것을 환영합니다!", bg="white")
titleLabel.pack()
```

Sol:



```
guessField = Entry(window)
guessField.pack(side="left")
tryButton = Button(window, text="시도", fg="green", bg="white", command=guessing )
tryButton.pack(side="left")

resetButton = Button(window, text="초기화", fg="red", bg="white", command=reset)
resetButton.pack(side="left")
resultLabel = Label(window, text="1부터 100사이의 숫자를 입력하시오.", bg="white")
resultLabel.pack(side="left")

window.mainloop()
```

Lab: 가위, 바위, 보 게임



Sol:

```
import random
from tkinter import *

window = Tk()
Label(window, text="선택하세요", font=("Helvetica", "16")).pack()
frame = Frame(window)

# 이미지를 d 드라이브로 복사하여 주세요!
rock_image = PhotoImage(file="d:\\rock.png")
paper_image = PhotoImage(file="d:\\paper.png")
scissors_image = PhotoImage(file="d:\\scissors.png")

# 초판 교재 오타!!
def pass_s():
    decide("가위")
def pass_r():
    decide("바위")
def pass_p():
    decide("보")
```

Sol:

```
rock = Button(frame, image=rock_image, command=pass_r)
rock.pack(side="left")
paper = Button(frame, image=paper_image, command=pass_p)
paper.pack(side="left")
scissors = Button(frame, image=scissors_image, command=pass_s)
scissors.pack(side="left")

frame.pack()
Label(window, text="컴퓨터는 다음을 선택하였습니다.", font=("Helvetica", "16")).pack()

computer_image = Label(window, image=rock_image)
computer_image.pack()
output = Label(window, text="", font=("Helvetica", "16"))
output.pack()
```



Sol:

```
def decide(human):
    computer = random.choice(["가위", "바위", "보"])
    if computer == "바위":
        computer_image["image"] = rock_image
    elif computer == "보":
        computer_image["image"] = paper_image
    else:
        computer_image["image"] = scissors_image

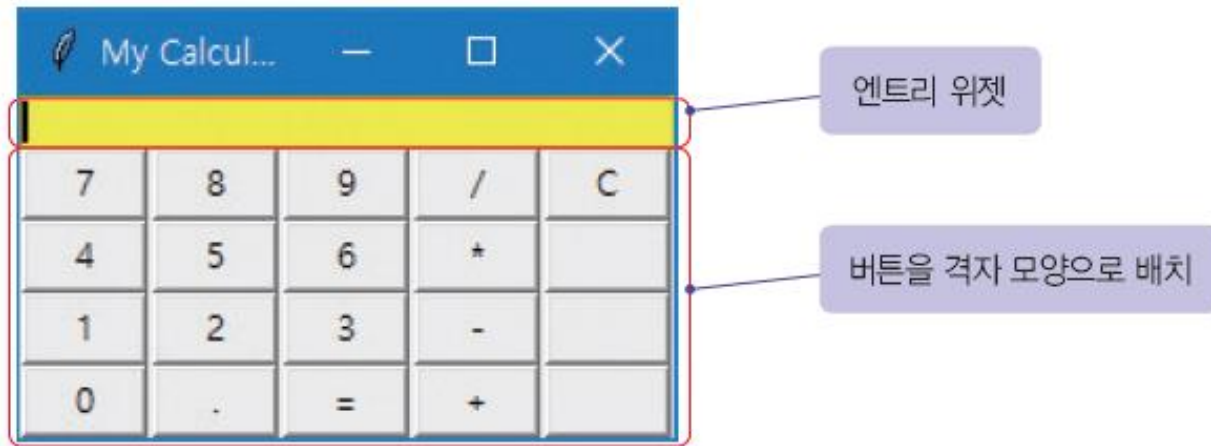
    if (computer == "바위" and human == "보") or (computer == "보" and human == "가위") \
        or (computer == "가위" and human == "바위"):
        result = "인간 승리!"
    elif computer == human:
        result = "비겼습니다."
    else:
        result = "컴퓨터 승리!"
    output.config(text="인간: " + human + "   컴퓨터:" + computer + "   " + result)

window.mainloop()
```



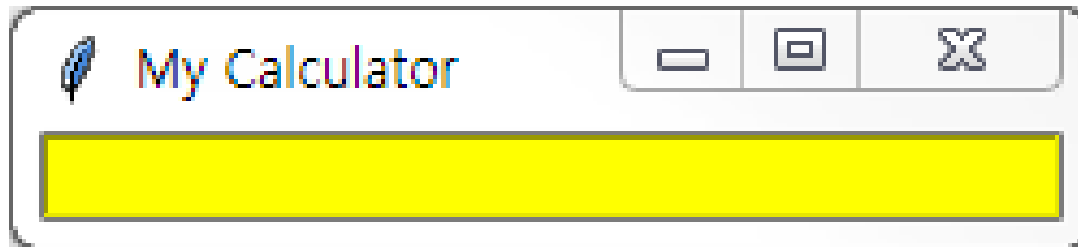
Lab: 계산기 프로그램

- 계산기는 격자 배치 관리자를 사용 하면 될 것이다.
버튼과 엔트리 위젯만 있으면 된다.



계산기 프로그램 #1

```
from tkinter import *  
  
window = Tk()  
window.title("My Calculator")  
display = Entry(window, width=33, bg="yellow")  
display.grid(row=0, column=0, columnspan=5)
```

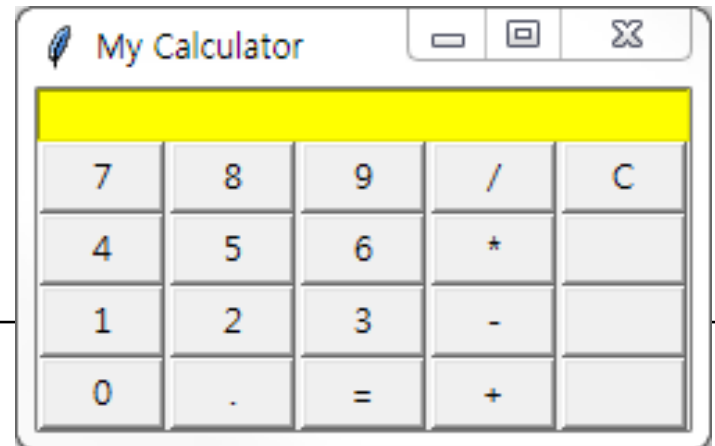


계산기 프로그램 #2

```
button_list = [  
'7', '8', '9', '/', 'C',  
'4', '5', '6', '*', '',  
'1', '2', '3', '-', '',  
'0', '.', '=', '+', '']
```

```
row_index = 1  
col_index = 0
```

```
for button_text in button_list:  
    Button(window, text=button_text, width=5).grid(row=row_index, column=col_index)  
    col_index += 1  
    if col_index > 4:  
        row_index += 1  
        col_index = 0
```

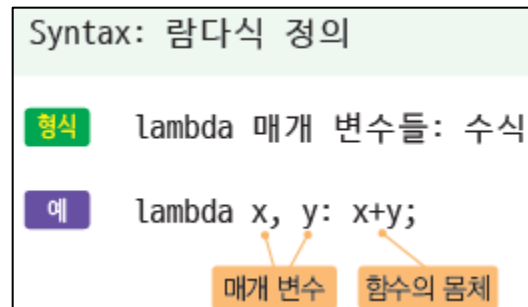
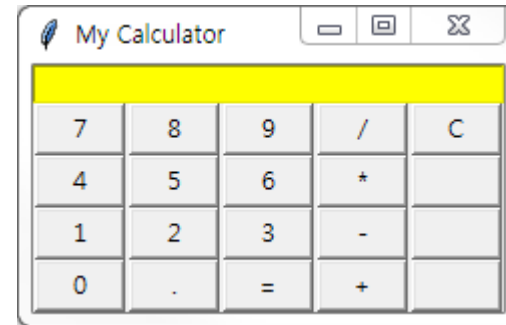


계산기 프로그램 #3

```
from tkinter import *
window = Tk(); window.title("My Calculator")
display = Entry(window, width=33, bg="yellow"); display.grid(row=0, column=0, columnspan=5)
button_list = ['7', '8', '9', '/', 'C', '4', '5', '6', '*', '1', '2', '3', '-', '0', '.', '=', '+', '']

def click(key):
    if key == "=":
        result = eval(display.get())
        s = str(result)
        display.insert(END, "=" + s)
    else:
        display.insert(END, key)

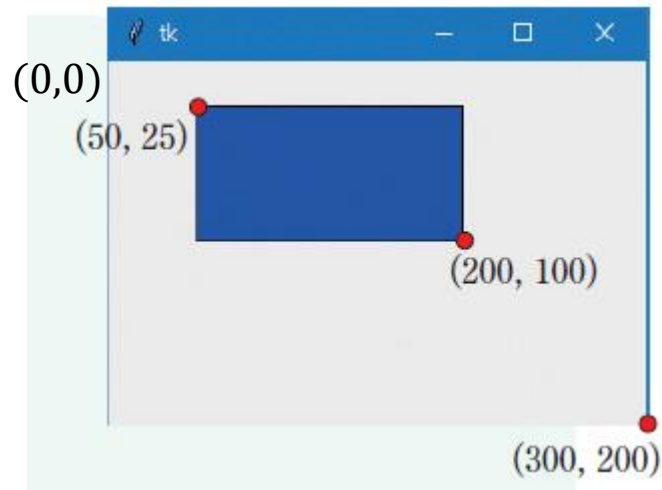
row_index = 1
col_index = 0
for button_text in button_list:
    Button(window, text=button_text, width=5,
           command=lambda t=button_text: click(t)).grid(row=row_index, column=col_index)
    col_index += 1
    if col_index > 4:
        row_index += 1
        col_index = 0
window.mainloop()
```



def lambda (t = button_text):

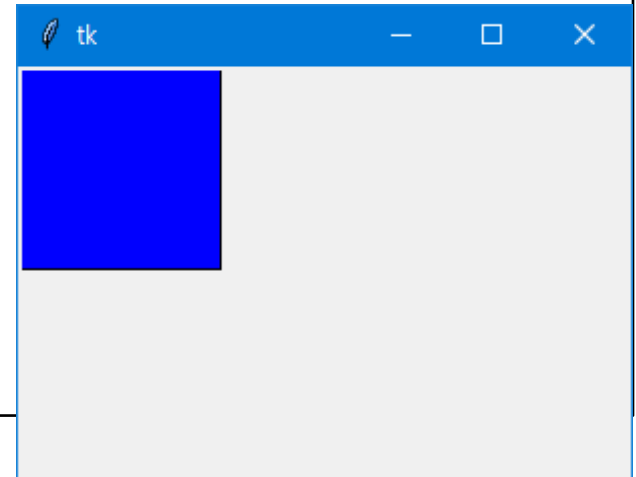
화면에 그림 그리기

```
from tkinter import *  
  
window = Tk()  
w = Canvas(window, width=300, height=200)  
w.pack()  
  
w.create_rectangle(50, 25, 200, 100, fill="blue")  
window.mainloop()
```



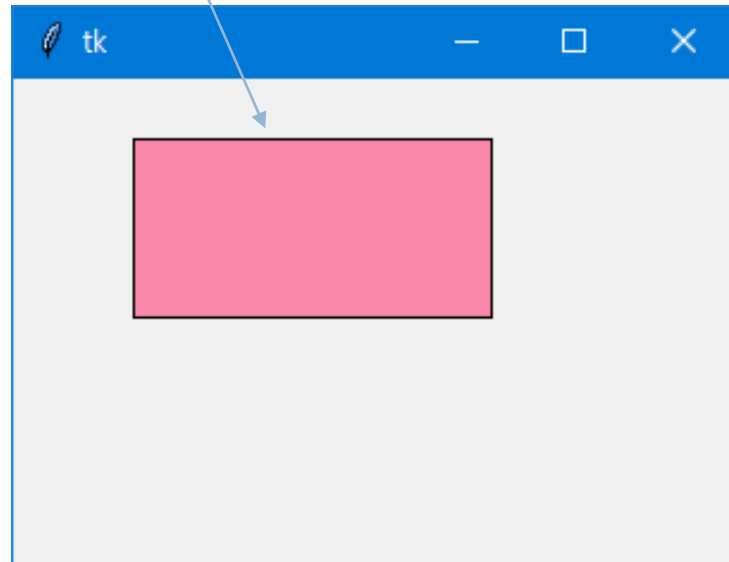
도형 관리

```
from tkinter import *  
  
window = Tk()  
  
w = Canvas(window, width=300, height=200)  
w.pack()  
  
i = w.create_rectangle(50, 25, 200, 100, fill="red")  
  
w.coords(i, 0, 0, 100, 100)    # 좌표를 변경한다.  
w.itemconfig(i, fill="blue")    # 색상을 변경한다.  
  
#w.delete(i)                    # 삭제한다.  
#w.delete(ALL)                  # 모든 항목을 삭제한다.  
window.mainloop()
```



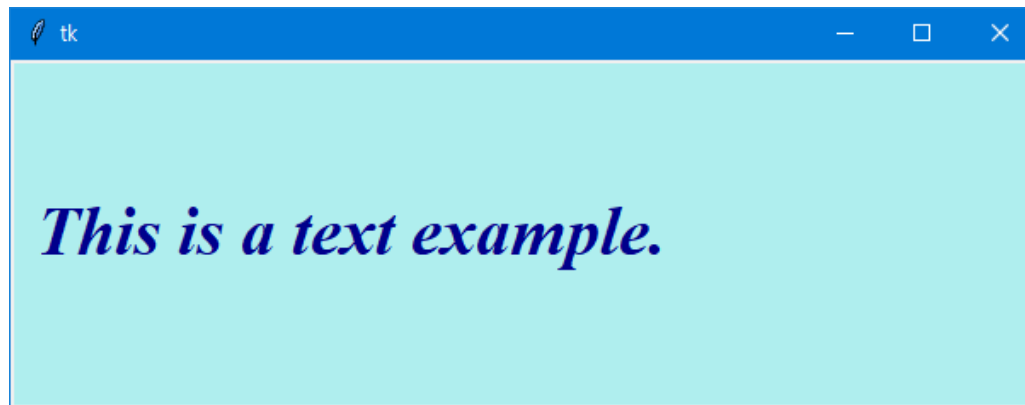
색상 설정

```
w.create_rectangle(50, 25, 200, 100, fill="#FA88AB")
```



폰트

```
from tkinter import *  
  
window = Tk()  
canvas = Canvas(window, width=600, height=200, bg = '#afeeee')  
canvas.create_text(200, 100, fill="darkblue", font="Times 30 italic bold",  
                  text="This is a text example.")  
canvas.pack()  
window.mainloop()
```



기초 도형 그리기

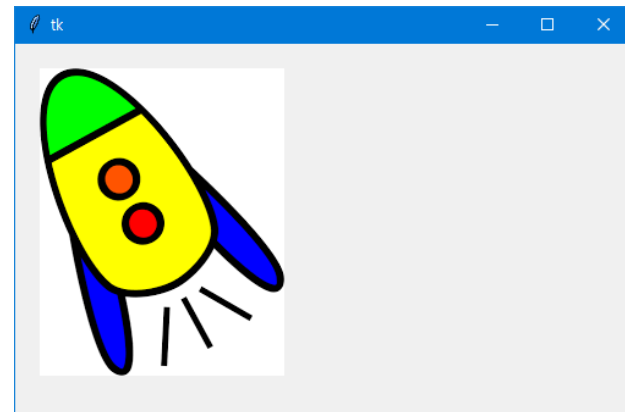
| 도형의 종류 | 설명 | 그림 |
|--|-----------------------------------|--|
| <code>canvas.create_line(15, 25, 200, 25)</code> | 직선을 그리는 메소드 |  |
| <code>canvas.create_rectangle(50, 25, 150, 75, fill="blue")</code> | 사각형을 그리는 메소드 |  |
| <code>canvas.create_arc(10, 10, 100, 150, extent=90)</code> | 사각형에 내접한 원이 그려지고 원 중에서 90도만 그려진다. |  |

기초 도형 그리기

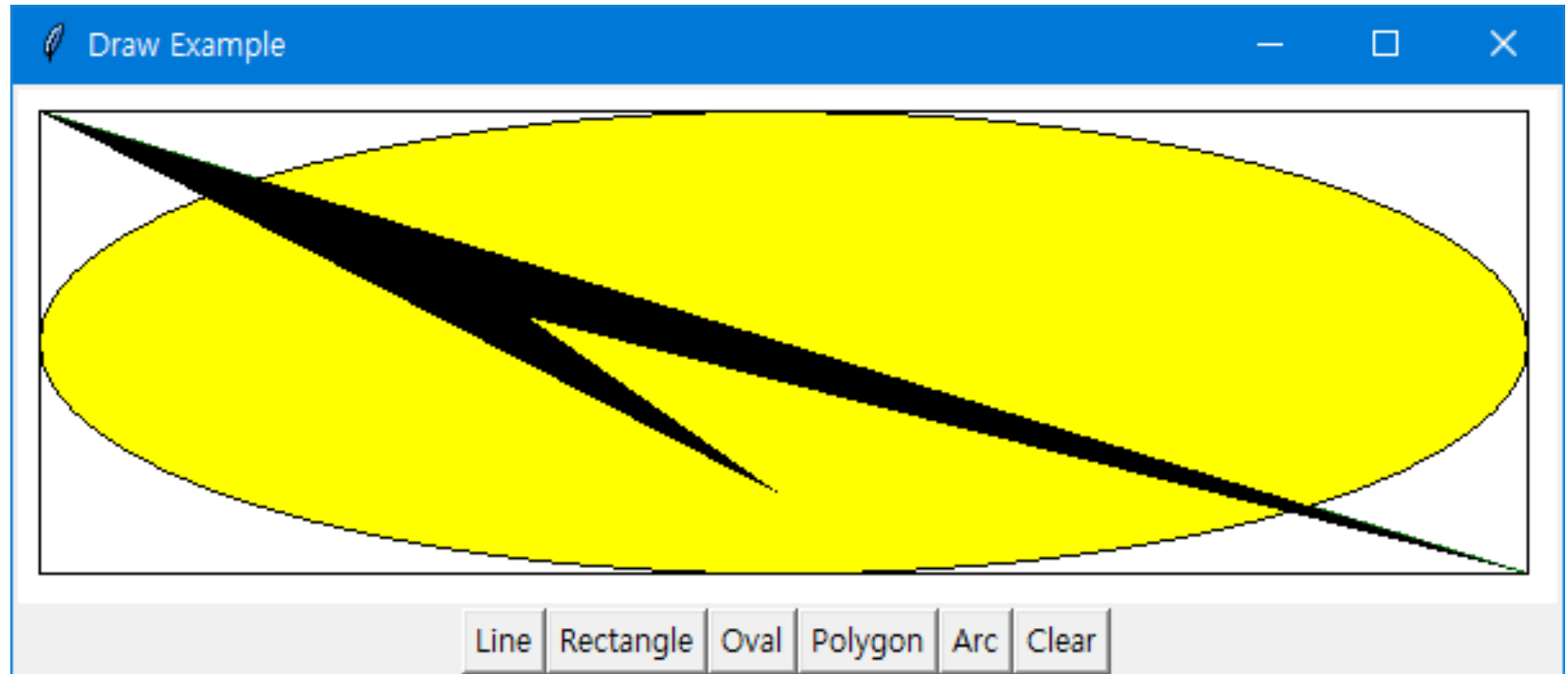
| | | |
|---|--|---|
| <pre>canvas.create_oval(15, 25, 100, 125)</pre> | 타원은 지정된 사각형 안에 그려진다. |  |
| <pre>canvas.create_polygon(10, 10, 150, 110, 250, 20, fill="yellow")</pre> | (10, 10)에서 출발하여서 (150, 110)으로 가고 최종적으로 (250, 20)에서 종료된다. |  |
| <pre>canvas.create_text(100, 20, text='Sing Street', fill='blue', font=('Courier', 20))</pre> | 텍스트의 중앙 위치를 나타내는 (x, y) 좌표, 색상을 표시하는 매개 변수 fill, 폰트를 나타내는 매개 변수 font |  |

이미지 표시하기

```
from tkinter import *  
window = Tk()  
  
canvas = Canvas(window, width=500, height=300)  
canvas.pack()  
  
img = PhotoImage(file="D:\\starship.png")  
canvas.create_image(20, 20, anchor=NW, image=img)  
  
window.mainloop()
```



Lab: 도형 그리기



Sol:

```
from tkinter import *  
  
WIDTH = 600  
HEIGHT = 200  
def displayRect():  
    canvas.create_rectangle(10,10,WIDTH-10,HEIGHT-10)  
def displayOval():  
    canvas.create_oval(10,10,WIDTH-10,HEIGHT-10, fill="yellow")  
def displayArc():  
    canvas.create_arc(10,10,WIDTH-10,HEIGHT-10,start=0,  
                      extent=120,width=10,fill='blue')  
def displayPolygon():  
    canvas.create_polygon(10,10,WIDTH-10,HEIGHT-10,200,90,300, 160)  
def displayLine():  
    canvas.create_line(10,10,WIDTH-10,HEIGHT-10,fill='green')  
  
def clearCanvas():  
    canvas.delete(ALL)
```

Sol:

```
window=Tk()
canvas=Canvas(window, width=WIDTH, height=HEIGHT, bg='white')
canvas.pack()
frame=Frame(window)
frame.pack()

btRectangle=Button(frame, text="Rectangle",
command=displayRect).grid(row=1,column=2)
btOval=Button(frame,text="Oval",command=displayOval).grid(row=1,column=3)
btArc=Button(frame, text="Arc",command=displayArc).grid(row=1,column=5)
btPolygon=Button(frame, text="Polygon",command=displayPolygon).grid(row=1,column=4)
btLine=Button(frame, text="Line",command=displayLine).grid(row=1,column=1)
btClear=Button(frame,text="Clear",command=clearCanvas).grid(row=1,column=7)

window.mainloop()
```

키보드와 마우스 이벤트 처리

Syntax: tkinter 이벤트 처리

형식 위젯.bind(이벤트지정자 , 콜백함수)

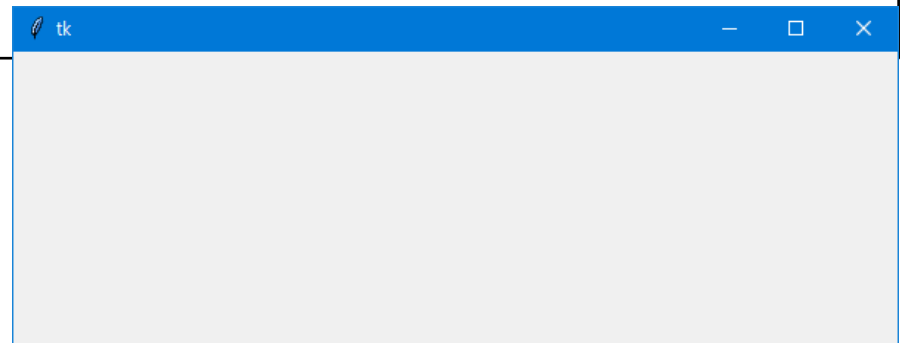
예 frame.bind("<Button-1>", callback)

첫번째 마우스 버튼이 눌리면
callback 함수가 호출된다

마우스 이벤트 처리

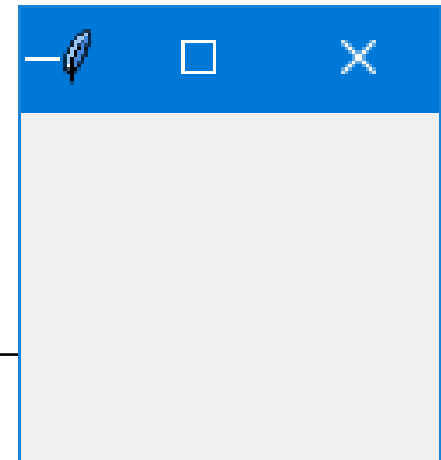
```
from tkinter import *  
  
window = Tk()  
window.geometry("600x200")  
  
def callback(event):  
    print(event.x, event.y, "에서 마우스 이벤트 발생")  
  
window.bind("<Button-1>", callback)  
window.mainloop()
```

32 44 에서 마우스 이벤트 발생
6 52 에서 마우스 이벤트 발생



마우스 이벤트 처리

```
from tkinter import *  
  
window = Tk()  
  
def key(event):  
    print ( repr(event.char), "가 눌렸습니다. ")  
  
def callback(event):  
    frame.focus_set()  
    print(event.x, event.y, "에서 마우스 이벤트 발생")  
  
frame = Frame(window, width=100, height=100)  
frame.bind("<Key>", key)  
frame.bind("<Button-1>", callback)  
frame.pack()  
  
window.mainloop()
```



66 31 에서 마우스 이벤트 발생
'k' 가 눌렸습니다.

Lab: 그림판 프로그램 만들기



Sol:

```
from tkinter import *
from tkinter.colorchooser import askcolor

DEFAULT_PEN_SIZE = 1.0
DEFAULT_COLOR = "black"

mode = "pen"; old_x = None; old_y = None
mycolor = DEFAULT_COLOR
erase_on = False

def use_pen():      # 펜 버튼이 선택되면 모드를 "pen"으로 바꾼다.
    global mode
    mode = "pen"

def use_brush():    # 브러쉬 버튼이 선택되면 모드를 "brush"으로 바꾼다.
    global mode
    mode = "brush"

def choose_color(): # 색상 버튼이 선택되면 사용자에게 색상을 요구한다.
    global mycolor
    mycolor = askcolor(color=mycolor)[1]

def use_eraser():   # 지우개 버튼이 선택되면 모드를 "erase"으로 바꾼다.
    global mode
    mode = "erase"
```

Sol:

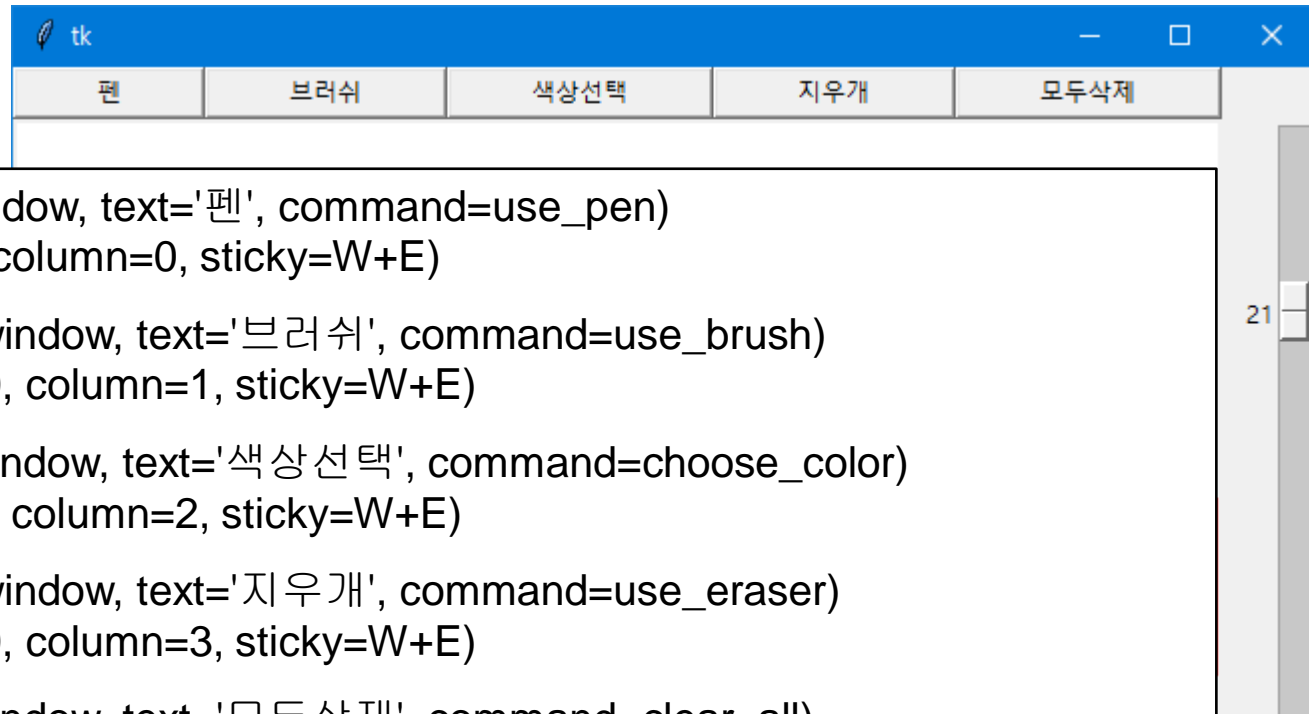
```
def paint(event):    # 이전 점과 현재 점 사이를 직선으로 연결한다.
    global var, erase_on, mode, old_x, old_y
    fill_color = 'white' if mode == "erase" else mycolor
    if old_x != None and old_y != None:
        canvas.create_line(old_x, old_y, event.x, event.y,
                           capstyle=ROUND, width=var.get(), fill=fill_color)
    old_x = event.x
    old_y = event.y

def reset(event):    # 사용자가 마우스 버튼에서 손을 떼면 이전 점을 삭제한다.
    global old_x, old_y
    old_x, old_y = None, None

def clear_all():     # 캔버스에 그려진 모든 그림을 삭제한다.
    global canvas
    canvas.delete('all')

window = Tk()
var = DoubleVar()    # 슬라이더와 연결될 객체를 생성한다.
```

Sol:



```
penButton = Button(window, text='펜', command=use_pen)
penButton.grid(row=0, column=0, sticky=W+E)

brushButton = Button(window, text='브러쉬', command=use_brush)
brushButton.grid(row=0, column=1, sticky=W+E)

colorButton = Button(window, text='색상선택', command=choose_color)
colorButton.grid(row=0, column=2, sticky=W+E)

eraseButton = Button(window, text='지우개', command=use_eraser)
eraseButton.grid(row=0, column=3, sticky=W+E)

clearButton = Button(window, text='모두삭제', command=clear_all)
clearButton.grid(row=0, column=4, sticky=W+E)

scale = Scale(window, variable=var, orient=VERTICAL)      # var = DoubleVar()
scale.grid(row=1, column=5, sticky=N+S)

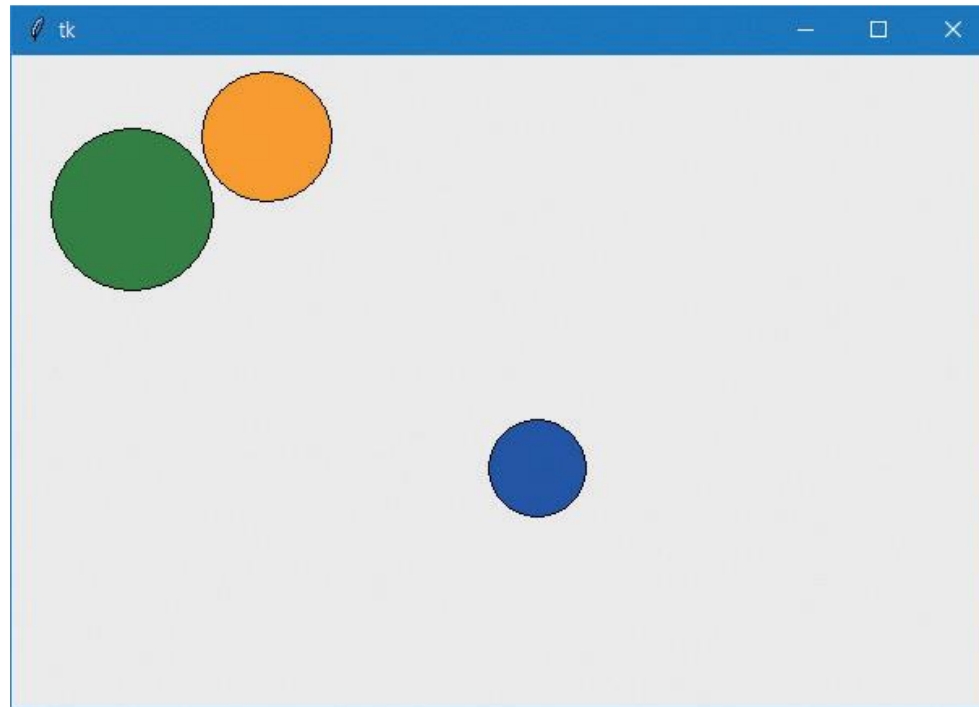
canvas = Canvas(window, bg='white', width=600, height=400)
canvas.grid(row=1, columnspan=5)

canvas.bind('<B1-Motion>', paint)
canvas.bind('<ButtonRelease-1>', reset)

window.mainloop()
```

Lab: 공 애니메이션 I

- 다음과 같이 크기와 색상이 다른 3개의 공이 캔버스에서 반사되는 응용 프로그램을 작성해보자.



Sol:

```
from tkinter import *
import time
import random

window = Tk()
canvas=Canvas(window, width=600,height=400)
canvas.pack()

class Ball():
    def __init__(self,color,size):
        self.id=canvas.create_oval(0, 0, size, size,fill=color)
        self.dx=random.randint(1,10)
        self.dy=random.randint(1,10)

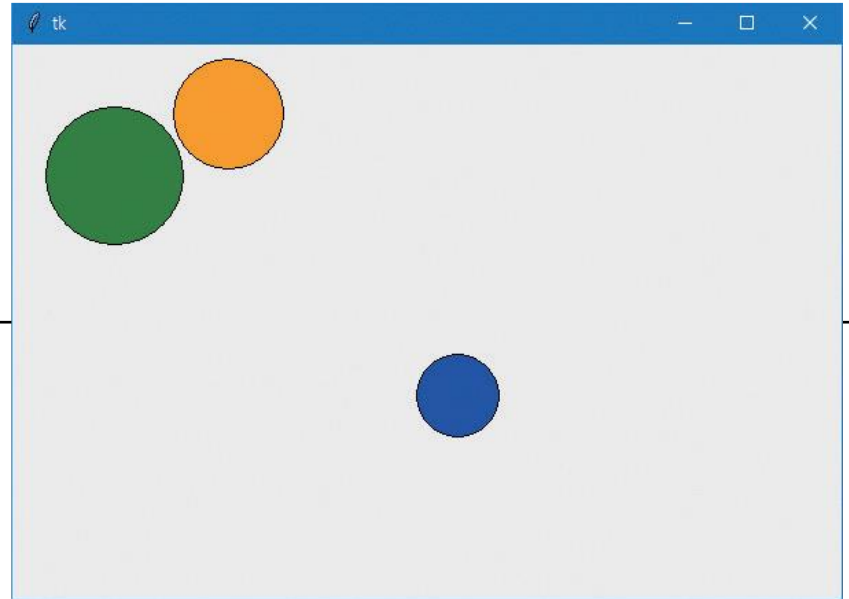
    def move(self):
        canvas.move(self.id,self.dx,self.dy)
        x0, y0, x1, y1 = canvas.coords(self.id)
        if y1 > canvas.winfo_height() or y0 < 0: # 원이 위쪽이나 아래쪽으로 벗어났으면
            self.dy = -self.dy                # dy의 부호를 반전시킨다.
        if x1 > canvas.winfo_width() or x0 < 0: # 원이 왼쪽이나 오른쪽으로 벗어났으면
            self.dx = -self.dx                # dx의 부호를 반전시킨다.
```


Sol:

```
ball1=Ball("blue", 60)  
ball2=Ball("green",100)  
ball3=Ball("orange",80)
```

```
while True:  
    ball1.move()  
    ball2.move()  
    ball3.move()  
    window.update()  
    time.sleep(0.05)
```

```
window.mainloop()
```



Lab: 공 애니메이션 II

- 공을 30개 정도 만들어서 움직이려면 어떻게 해야 할까? 이럴 때는 개별 변수를 사용하여 각 객체를 참조하는 것은 거의 불가능하다. 이런 경우에는 리스트에 객체를 저장하여야 한다.

```
colors = ["red", "orange", "yellow", "green", "blue", "indigo", "violet"]
```

```
ballList = [Ball(random.choice(colors), 60) for _ in range(30)]
```

```
while True:  
    for ball in ballList:  
        ball.move()  
        window.update()  
        time.sleep(0.05)  
window.mainloop()
```

