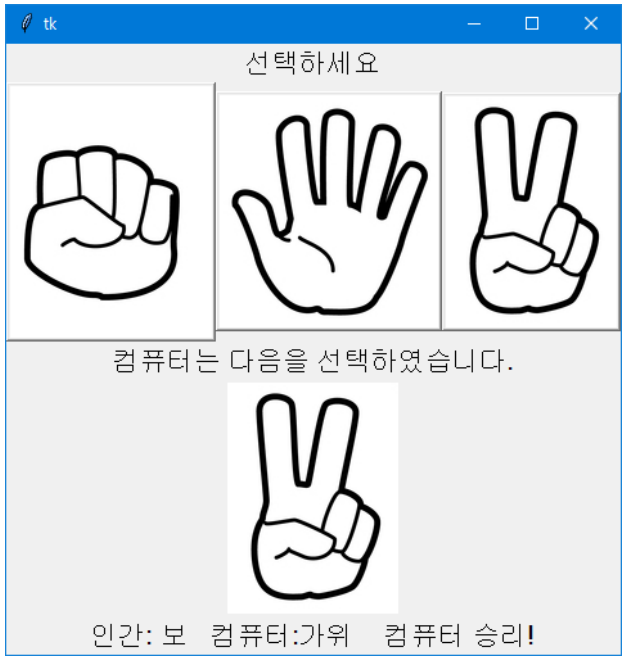
파이썬 Express

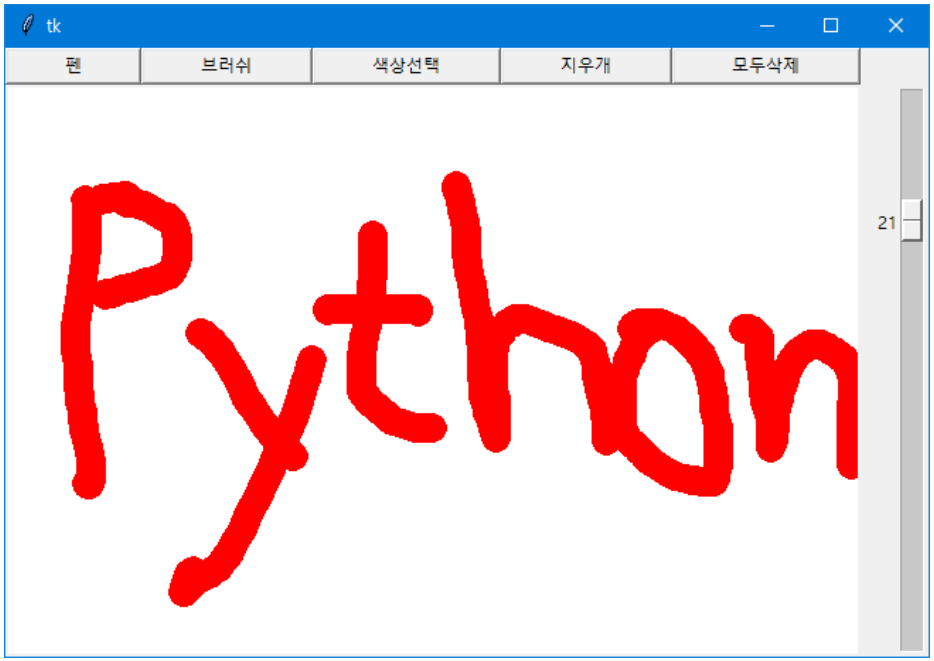
9장 GUI 프로그래밍

학습 목표

- tkinter를 이용하여 간단한 GUI 프로그램을 작성할 수 있다.
- GUI의 일반적인 구조를 이해할 수 있다.
- 배치 관리자를 사용할 수 있다.
- 위젯의 콜백 함수를 이용하여 이벤트를 처리할 수 있다.
- 캔버스에 다양한 도형을 그릴 수 있다.
- 애니메이션을 만들 수 있다.

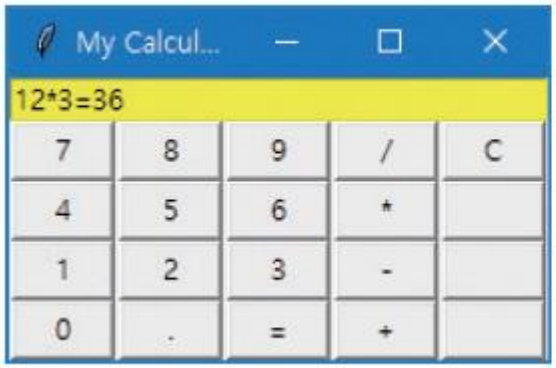
이번 장에서 만들 프로그램





tkinter란

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

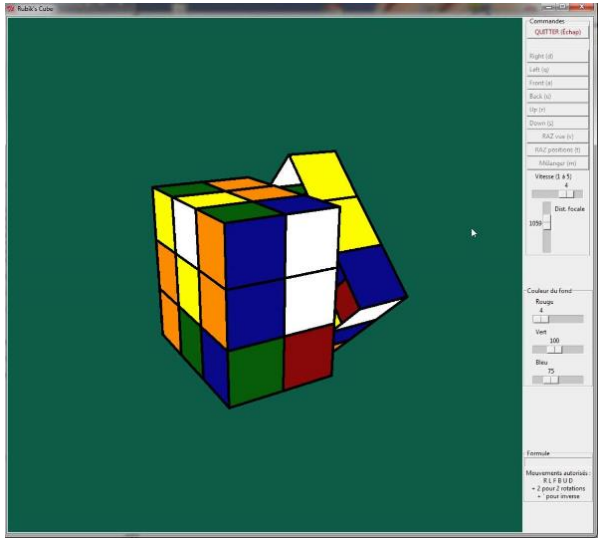



```
첫 번째 수를 입력하시오:45

두 번째 수를 입력하시오:6

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

- 기존에 그림을 그릴 때 사용했던 turtle 모듈은 이해하고 쉽고 사용하기 쉽지만 약점이 있다. 다양한 그림을 그리기 힘들고 속도가 느리다.

tkinter의 유래

- tkinter는 예전부터 유닉스 계열에서 사용되던 Tcl/Tk 위에 객체 지향 계층을 입힌 것이다. Tk는 John Ousterhout에 의하여 Tcl 스크립팅 언어를 위한 GUI 확장으로 개발



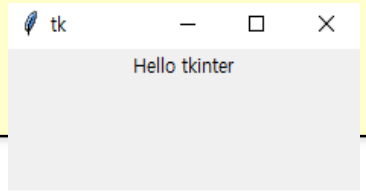
Tkinter 시작하기

- 윈도우(window)를 생성하고 여기에 필요한 위젯들을 추가한다.
- 위젯(widget: window gadget)은 GUI 기반의 시스템에서 사용하는 각종 시각적인 요소. 버튼, 레이블, 슬라이더, 콤보 박스 등

Tkinter 시작하기

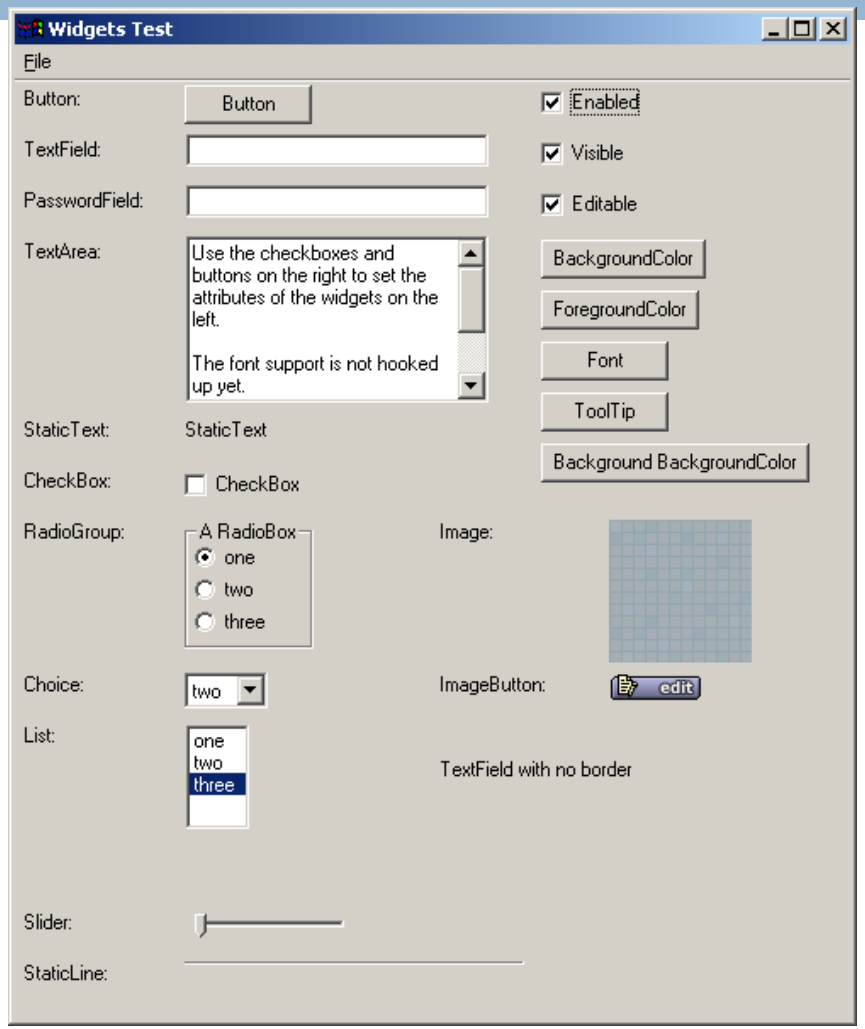
```
from tkinter import *                                  # tkinter 모듈을 포함

window = Tk()                                          # 루트 윈도우를 생성
label = Label(window, text="Hello tkinter")            # 레이블 위젯을 생성
label.pack()                                           # 레이블 위젯을 윈도우에 배치

window.mainloop()                                      # 윈도우가 사용자 동작을 대기
```

p409.py



- 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") : 부모 컨테이너 window에 레이블 위젯 Label을 생성한다.
- label.pack() : pack()은 압축 배치 관리자를 이용하여서 레이블을 컨테이너에 위치시킨다.
- window.mainloop() : 반드시 맨 끝에 이 문장을 가져야 한다.

기본 위젯들

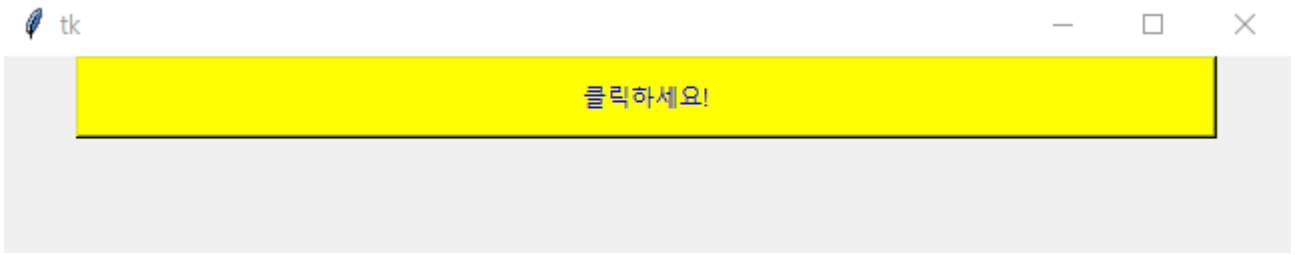

Widgets Test
File
Button:
Button
TextField:
PasswordField:
TextArea:
Use the checkboxes and buttons on the right to set the attributes of the widgets on the left.
The font support is not hooked up yet.
StaticText:
StaticText
CheckBox:
CheckBox
RadioGroup:
A RadioBox
one
two
three
Choice:
two
List:
one
two
three
Slider:
StaticLine:
Enabled
Visible
Editable
BackgroundColor
ForegroundColor
Font
ToolTip
Background BackgroundColor
Image:
ImageButton:
edit
TextField with no border

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

p411.py

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

button = Button(window,    text="클릭하세요!",
                bg="yellow",    fg="blue",          # 전경색과 배경색 설정
                width=80,    height=2               # 크기 설정(글자수)
)

button.pack()
window.mainloop()
```



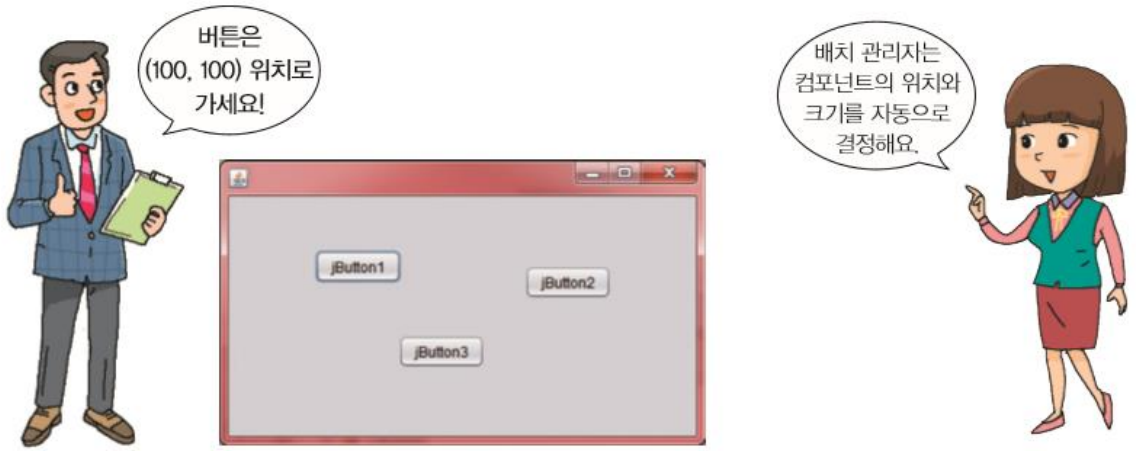
엔트리 위젯

```
from tkinter import *                                                    p412.py
window = Tk()

entry = Entry(window, fg="black", bg="yellow", width=80)

entry.pack()
window.mainloop()
```

배치 관리자

- 컨테이너 안에 존재하는 위젯의 크기와 위치를 자동적으로 관리하는 객체

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



압축 배치 관리자

- 위젯들을 최대한 압축하여 상하로 배치

p414.py

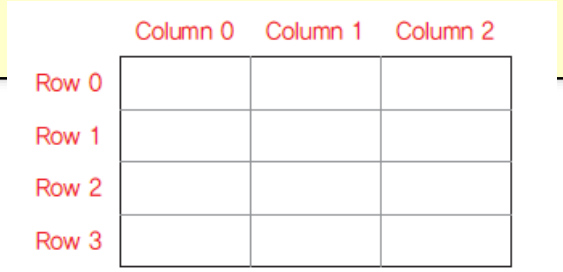
```
from tkinter import *
window = Tk()

Button(window, text="박스 #1", bg="red", fg="white").pack()
Button(window, text="박스 #2", bg="green", fg="black").pack()
Button(window, text="박스 #3", bg="orange", fg="white").pack()

# 박스 배치 방향을 지정하려면 side= 사용
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=RIGHT)

window.mainloop()
```

격자 배치 관리자

- 위젯을 테이블 형태로 배치

p415_grid.py

```
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()
```

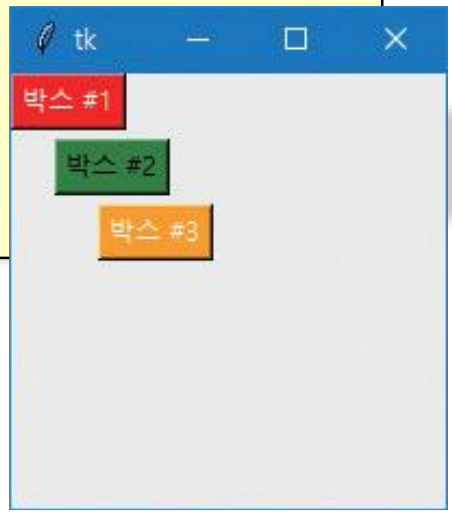


절대 위치 배치 관리자

- 절대 위치를 사용하여 위젯을 배치

p415_place.py

```
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 활용하기

- 위젯을 프레임에 담았다가 원하는 시점에 프레임을 pack하여 배치

p416_01.py

```
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

p416_02.py

```
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()
```

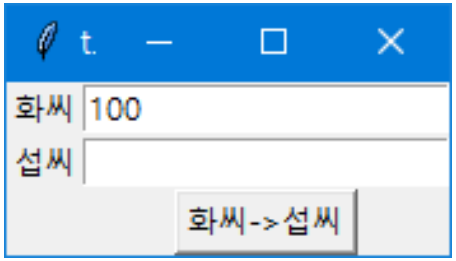
가로크기=600 픽셀, 세로크기=100 픽셀

가로크기=10글자, 세로크기=2글자

Lab: 온도 변환기

p417.py

```
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()
```

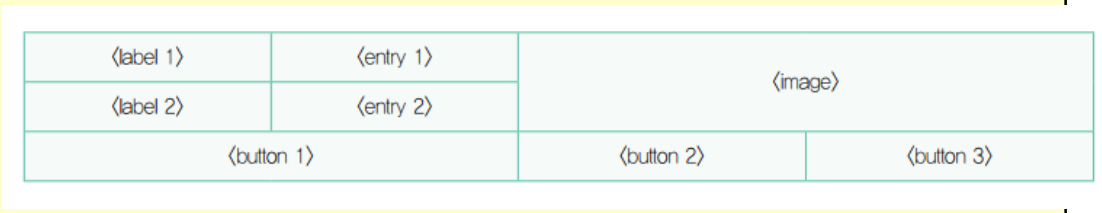


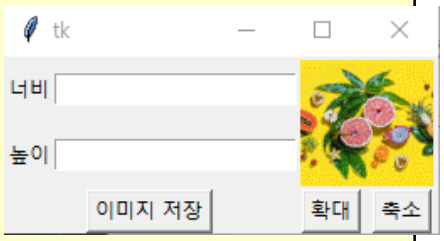
--> 버튼에 기능을 넣은 코드는 p.422 에서.

Lab: 격자 배치 관리자 실습

p419.py

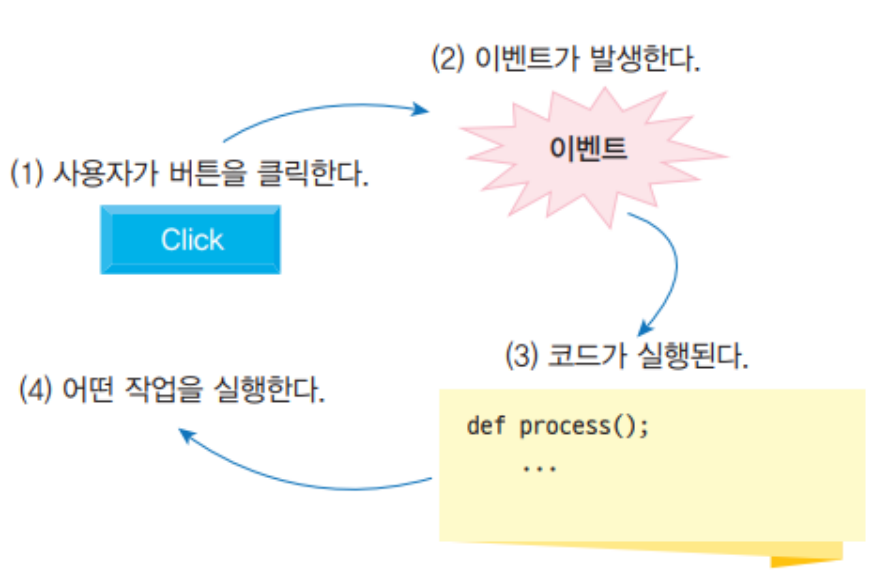
```
from tkinter import *

window  = Tk()

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

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( )
```

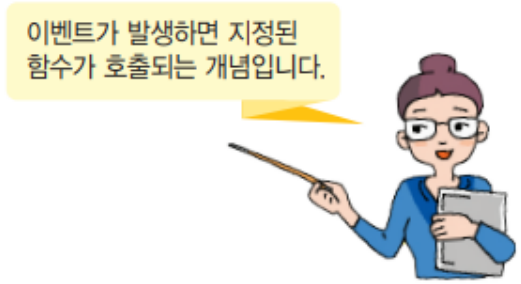




버튼 이벤트 처리

- 이벤트 구동 프로그래밍(event driven programming) : 이벤트가 프로그램의 실행 순서를 결정하는 프로그래밍
- 버튼에 이벤트를 처리하는 함수를 붙일 수 있다. 버튼에 이벤트를 처리하는 함수가 연결되어 있으면 이벤트가 발생했을 때 그 함수가 호출된다. 이벤트가 발생했을 때 호출되는 함수를 콜백함수(callback function) 또는 핸들러(handler)라고 한다.





버튼 이벤트 처리

- 버튼에 콜백함수를 등록하려면 버튼의 생성자를 호출할 때, command 매개변수에 이벤트를 처리하는 함수의 이름을 지정하면 된다.

Syntax: 버튼 이벤트 처리

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

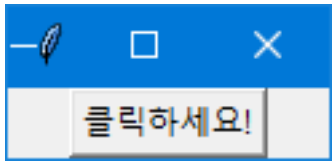
예 `Button(window, text="클릭하세요!", command=process)`

버튼에서 이벤트가 발생하면 호출되는 함수 등록

버튼 이벤트 처리

p421.py

```
from tkinter import *

def process():
        print("버튼이 클릭되었습니다.")

window = Tk()
button = Button(window, text="클릭하세요!", command=process)
button.pack()
window.mainloop()
```



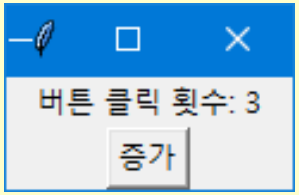

```
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
버튼이 클릭되었습니다.
```

Lab: 카운터 만들기

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

p422.py

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

window.mainloop()
```

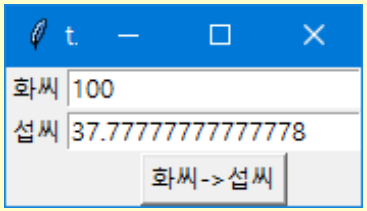


Lab: 온도 변환기 #2

- "화씨->섭씨" 버튼을 누르면 입력한 화씨 온도가 섭씨온도로 변환되어서 나타나게.

p423.py

```
from tkinter import *

# 이벤트 처리 함수를 정의한다.
def process():
    tf = float(e1.get())          # e1에서 문자열을 읽어서 부동소수점형으로 변경
    tc = (tf-32.0)*5.0/9.0        # 화씨 온도를 섭씨 온도로 변환한다.
    e2.delete(0, END)             # 처음부터 끝까지 지운다.
    e2.insert(0, str(tc))         # 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)
…
…
```



위젯의 속성 변경

- 위젯의 생성자를 호출할 때 fg, bg, font, text, command 와 같은 매개변수에 값을 전달하여서 속성을 지정할 수 있다.

```
label = Label(window, text="Times Font 폰트와 빨강색을 사용합니다.", fg="red",
font="Times 32 bold italic").pack()

label["text"] = "This is a label"
label["fg"] = "blue"
label["bg"] = "#FF00AA"

label.config(text = "World")
```

숫자 추측 게임

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

p425.py

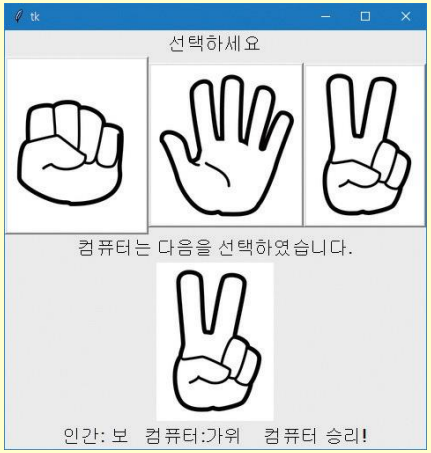
```
from tkinter import *
import random

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!"
…
…
```

실행결과

Lab: 가위, 바위, 보 게임

p427.py

```
import random
from tkinter import *

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

rock_image = PhotoImage(file="rock.png")
paper_image = PhotoImage(file="paper.png")
scissors_image = PhotoImage(file="scissors.png")

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



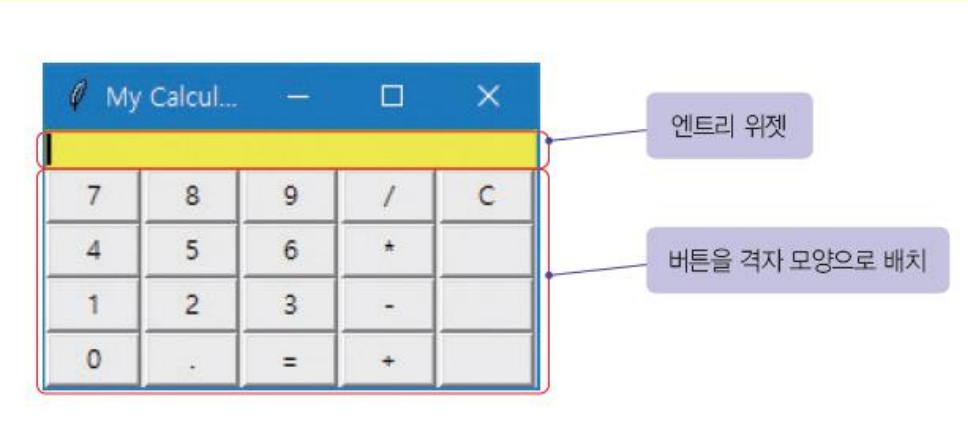
Lab: TIC-TAC-TOE 게임

- Tic-Tac-Toe는 3×3칸을 가지는 게임판을 만들고, 경기자 2명이 동그라미 심볼(O)와 가위표 심볼(X)을 고른다

Lab: 계산기 프로그램

p432.py

```
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)
…
…
```

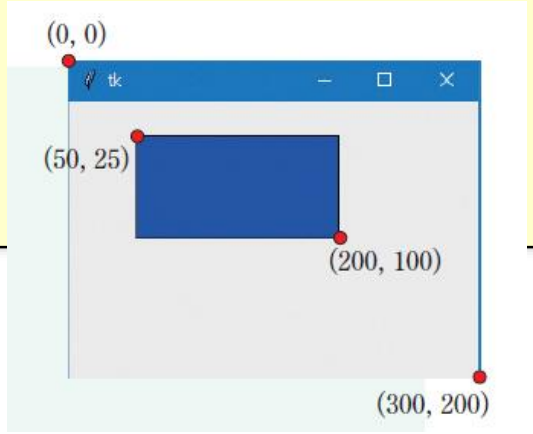


화면에 그림 그리기

- 캔버스(canvas) 위젯을 윈도우에 생성한 후 캔버스에 그림을 그린다.
- 캔버스 위젯을 사용하면 많은 그래픽 기능을 사용할 수 있다. 그래프를 그린다거나 그래픽 에디터를 작성할 수도 많은 종류의 커스텀 위젯을 작성할 수 있다.

p434_canvas1.py

```
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()
```

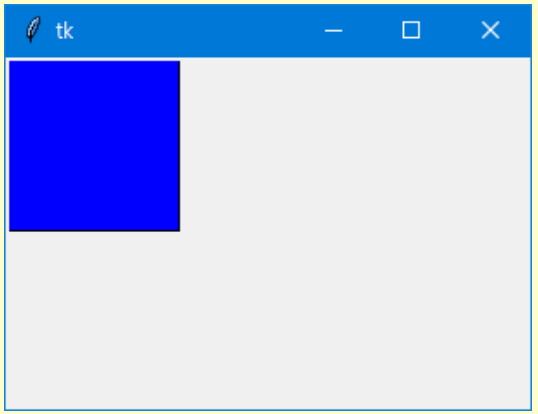


도형 관리

- coords() : 좌표를 변경, itemconfig() : 도형의 속성을 변경
- create_rectangle() : 식별자를 반환, delete() : 도형을 삭제

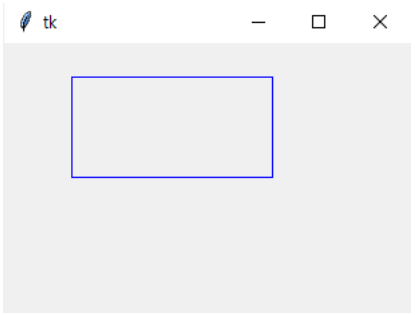
p434_canvas2.py

```
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()
```

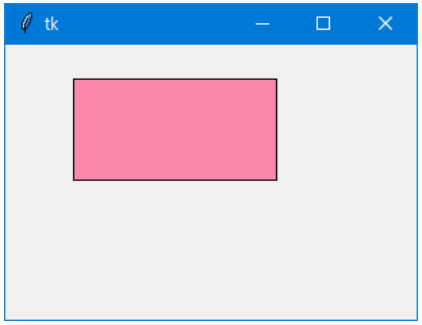


색상 설정

```
w.create_rectangle(50, 25, 200, 100, outline="blue")
```




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

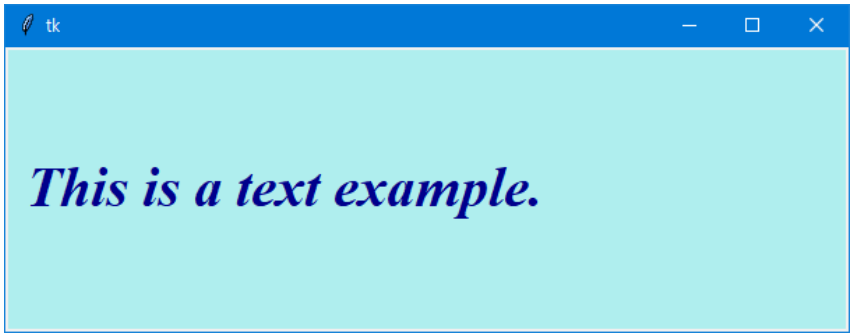


폰트

- (폰트 이름, 폰트 크기, 폰트 스타일) 형식

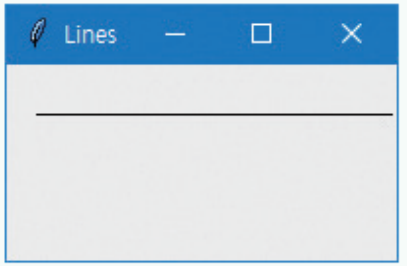
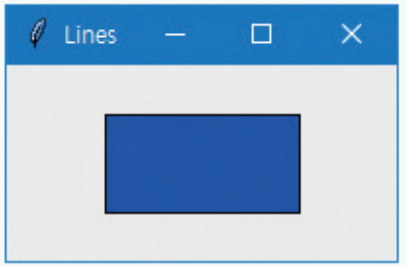
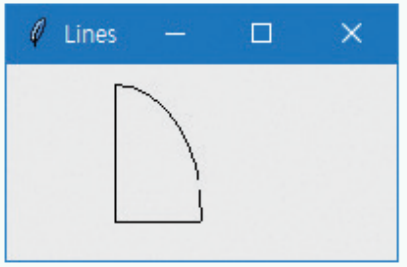
p436_font.py

```
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()
```

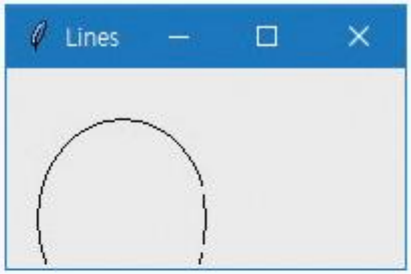
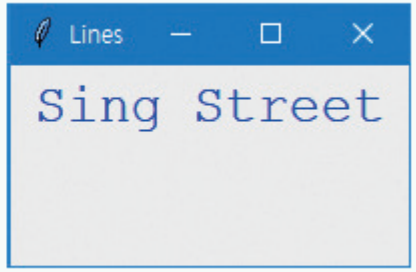


기초 도형 그리기

- 예제는 p. 438 에서.

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

기초 도형 그리기

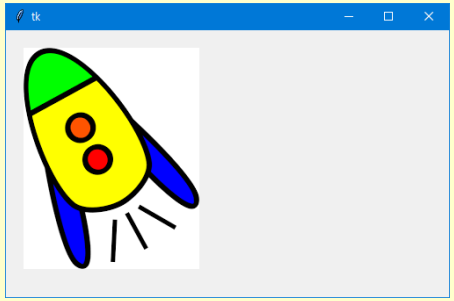
`canvas.create_oval(15, 25, 100, 125)`	타원은 지정된 사각형 안에 그려진다.	
`canvas.create_polygon(10, 10, 150, 110, 250, 20, fill="yellow")`	(10, 10)에서 출발하여서 (150, 110)으로 가고 최종적으로 (250, 20)에서 종료된다.	
`canvas.create_text(100, 20, text='Sing Street', fill='blue', font=('Courier', 20))`	텍스트의 중앙 위치를 나타내는 (x, y) 좌표, 색상을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

이미지 표시하기

- tkinter가 읽을 수 있는 이미지 파일은 PNG 파일과 JPG 파일
- PhotoImage로 이미지를 지정한 후 create_image() 함수로 캔버스에 그리면 된다.

p437.py

```
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()
```

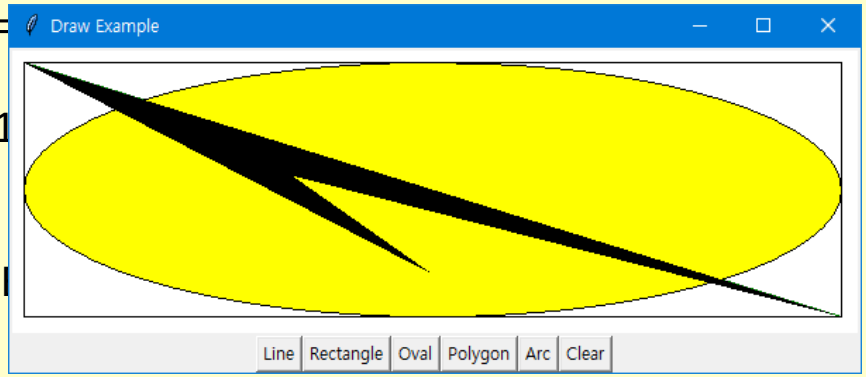


이미지 왼쪽 상단을 좌표(20, 20)의 기준점으로 사용

Lab: 도형 그리기

p438.py

```
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=
def displayPolygon():
    canvas.create_polygon(10,10,WIDTH-1

def displayLine():
    canvas.create_line(10,10,WIDTH-10,HI
…
…
```



참고:

이름	의미	설명
create_line(x1, y1, x2, y2, ⋯ , xn, yn, option)	선	(x1, y1), (x2, y2), ⋯, (xn, yn) 까지 연결되는 선 생성
create_line(x1, y1, x2, y2, option)	사각형	(x1, y1)에서 (x2, y2)의 크기를 갖는 사각형 생성
create_polygon(x1, y1, x2, y2, ⋯ , xn, yn, option)	다각형	(x1, y1), (x2, y2), ⋯, (xn, yn) 의 꼭지점을 갖는 다각형 생성
create_oval(x1, y1, x2, y2, option)	원	(x1, y1)에서 (x2, y2)의 크기를 갖는 원 생성
create_arc(x1, y1, x2, y2, start, extent, option)	호	(x1, y1)에서 (x2, y2)의 크기를 가지며 `start` 각도부터 `extent`의 각을 지니는 호 생성
create_image(x, y, image, option)	이미지	(x, y) 위치의 `image` 생성

- option
  - `fill` : 배경 색상
  - `outline` : 두께 색상
  - `width` : 두께
  - `fill` : 배경 색상
  - `anchor` : 위치 지정

키보드와 마우스 이벤트 처리

- 위젯에서 이벤트 지정자와 일치하는 이벤트가 발생하면 주어진 이벤트 처리 함수가 이벤트를 설명하는 객체와 함께 호출된다.

Syntax: tkinter 이벤트 처리

형식 위젯.bind(이벤트지정자 , 콜백함수)

예 frame.bind("<Button-1>", callback)

첫번째 마우스 버튼이 눌리면 callback 함수가 호출된다

p440.py

```
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()
```

이벤트 지정자

- <Button-1> : 마우스 버튼을 눌렀을 때. 1-왼쪽, 2-중간, 3-오른쪽
- <B1-Motion> : 마우스 버튼 1이 눌려진 채로 움직일 때
- <ButtonRelease-1> : 마우스 버튼 1에서 손을 뗄 때
- <Double-Button-1> : 마우스 버튼 1을 더블클릭할 때
- <Enter> : 마우스 포인터가 위젯으로 진입했을 때
- <Leave> : 마우스 포인터가 위젯을 떠났을 때
- <FocusIn> : 포커스가 현재의 위젯으로 이동했을 때
- <FocusOut> : 포코스가 현재의 위젯에서 다른 위젯으로 이동했을 때
- <return> : 엔터키를 눌렀을 때. Cancel(Break), BackSpace, Tab, Shift_L, Control_L, Alt_L 등도 이런 식으로 연결
- <Key> : 어떤 키라도 눌렀을 때
- a : 글자 'a'를 입력했을 때. 다른 문자도 이런 식으로 연결
- <Shift-Up> : 시프트 키를 누른 상태에서 위쪽 화살표를 눌렀을 때
- <Configure> : 위젯의 크기, 위치, 플랫폼 등을 변경하였을 때

마우스 이벤트 처리

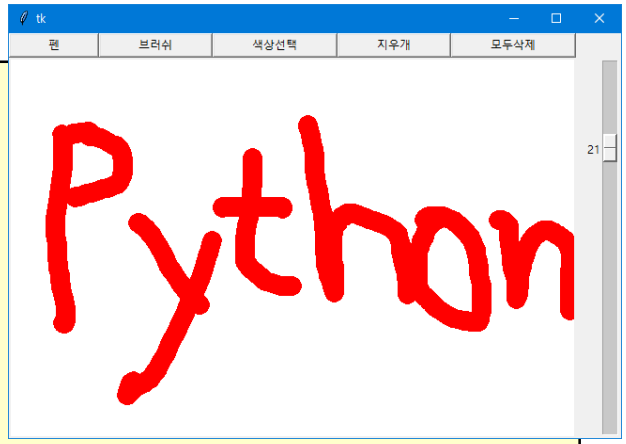
p442.py

```
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, background="yellow")
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

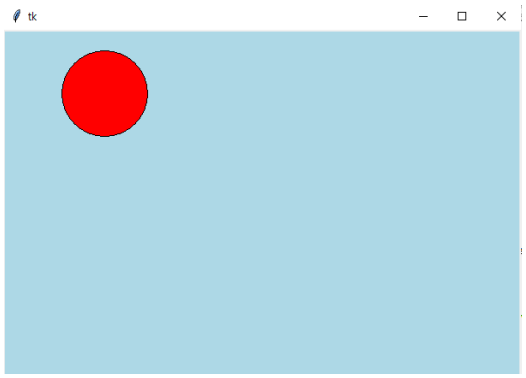
p445.py



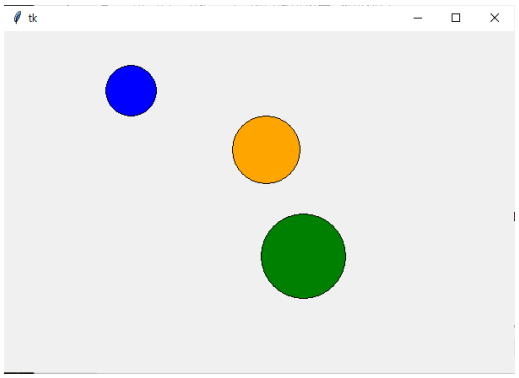

```
from tkinter import *
from tkinter.colorchooser import askcolor

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"
…
…
```

Lab: 공 애니메이션 I, II

- 한 개의 공 애니메이션에서 코드를 분석해보고 생각을 키워가자.

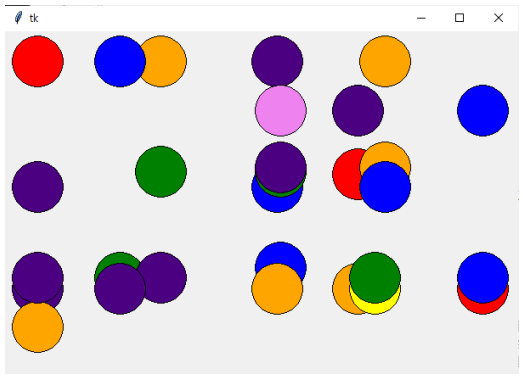


p447_bouncing_ball1.py



p447_bouncing_ball2.py

객체를 사용



p447_bouncing_ball3.py

리스트를 사용하여 객체를 저장

이번 장에서 배운 것

- tkinter에서는 먼저 루프 윈도우를 생성하고 레이블이나 버튼을 생성할 때 첫 번째 인수로 윈도우를 넘기면 된다.
- 파이썬은 3종류의 배치 관리자를 제공한다. 배치 관리자, 격자(grid) 배치 관리자. 절대(place) 배치 관리자가 바로 그것이다.
- 위젯에 이벤트를 처리하는 함수를 연결하려면 bind() 메소드를 사용한다. 예를 들면 widget.bind('<Button-1>', sleft)와 같이 하면 된다.