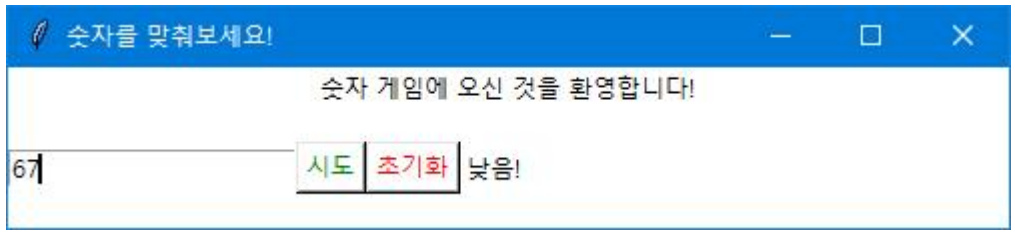
새내기 파이썬

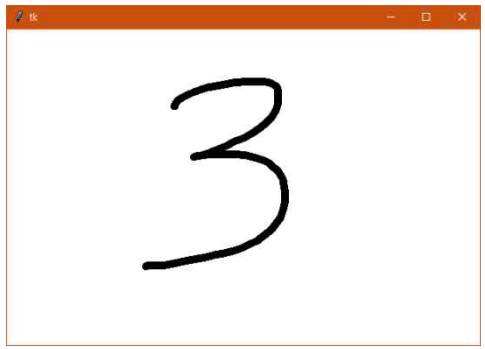
11장 TKINTER를 이용한 GUI 프로그래밍

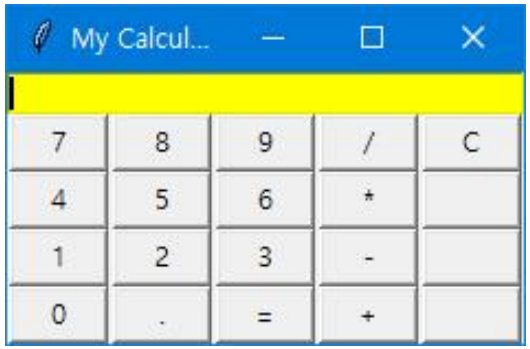
학습 목표

- GUI로 원하는 위치로 컴포넌트들을 배치할 수 있나요?
- 어떤 버튼을 눌렀을 때 원하는 작업을 할 수 있나요?
- 캔버스에 다양한 그림을 그릴 수 있나요?
- GUI로 그림판 프로그램을 작성할 수 있나요?

이번 장에서 만들 프로그램





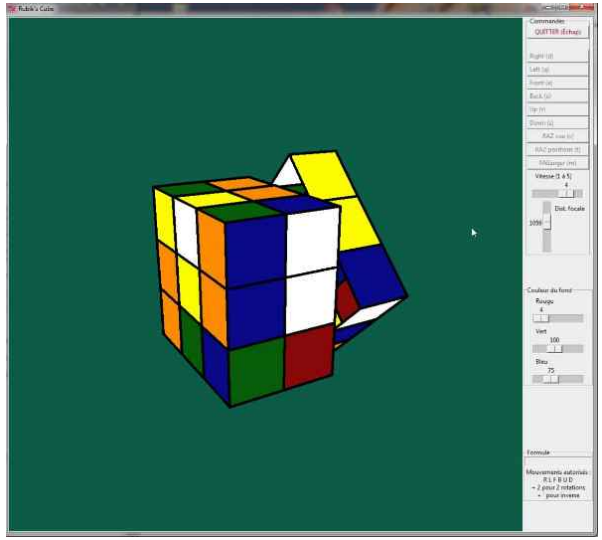


tkinter란?

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

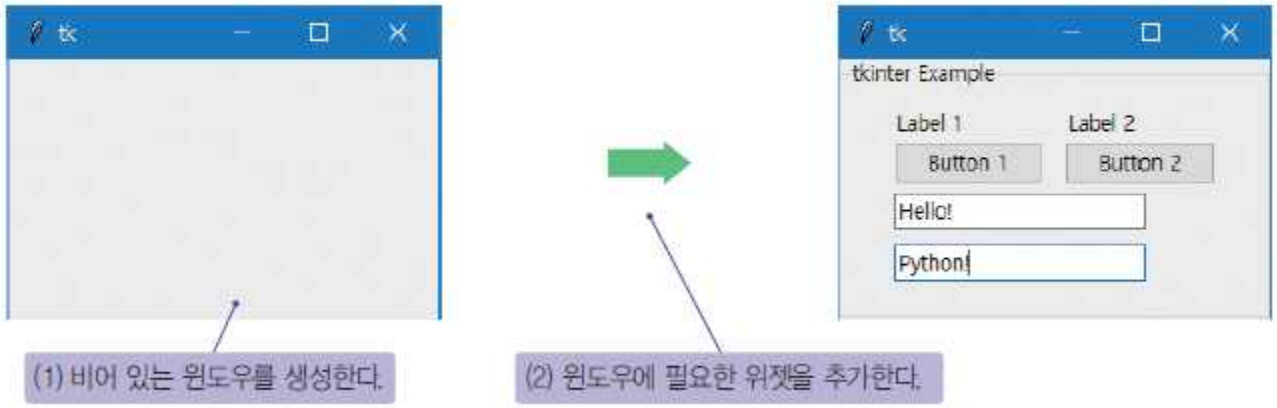
tkinter의 유래

- 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(root)를 생성하고 여기에 필요한 위젯들을 추가한다.



레이블이 있는 윈도우를 생성해보자

- 하나의 레이블이 있는 윈도우를 생성해보자

레이블이 있는 윈도우를 생성해보자

```
from tkinter import *                          # tkinter 모듈을 포함

root = Tk()                                    # 루트 윈도우를 생성
root.geometry("500x200")                       # Width x Height
label = Label(root, text="Hello tkinter")      # 레이블 위젯을 생성
label.pack()                                   # 레이블 위젯을 윈도우에 배치

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

분석

- from tkinter import *
  - tkinter 모듈을 포함시키는 것이다.
- root = Tk()
  - Tk()을 호출하면 Tk 클래스의 객체가 생성되면서 화면에 하나의 윈도우가 생성된다.
- root.geometry("500x200")
  - 루트 윈도우의 크기를 500×200으로 한다. 단위는 픽셀이다.
- label = Label(root, text="Hello tkinter")
  - 레이블 위젯을 생성한다.
- label.pack()
  - pack()은 압축 배치 관리자를 이용하여서 레이블을 컨테이너에 위치시킨다.
- root.mainloop()
  - mainloop()는 이벤트 처리 루프로서, 사용자로부터 오는 마우스나 키보드 이벤트를 처리한다.

기본 위젯들

- 위젯은 tkinter의 핵심이다. 위젯들을 이용하여 사용자가 프로그램과 상호 작용한다. 가장 기본이 되는 위젯들은 다음과 같다.

위젯	설명
Label	텍스트를 표시하는 데 사용되는 위젯
Button	버튼을 제공하는 위젯
Entry	한 줄의 텍스트를 입력받는 위젯
Text	여러 줄로 된 텍스트를 입력받는 위젯
Frame	컨테이너 위젯

위젯의 속성

- 레이블 위젯은 많은 속성을 가지고 있고, 생성할 때 이들 속성을 지정할 수 있다.
  - text: 출력할 텍스트이다.
  - font: 사용하는 폰트와 크기를 지정할 수 있다.
  - fg: foreground의 약자로 전경색(글자색)을 의미한다.
  - bg: background의 약자로서 배경색을 의미한다.
  - width, height: 위젯의 폭과 높이이다. 폭와 높이의 단위는 글자 개수이다.

레이블 위젯

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

label1 = Label(root,    text="안녕하세요?",              bg="yellow",    fg="blue",
                                width=80,    height=2    )
label2 = Label(root,    text="파이썬을 공부합니다.",font=("궁서체", 32))

label1.pack()
label2.pack()
root.mainloop()
```

버튼 위젯

- 버튼(Button) 위젯은 클릭 가능한 버튼을 표시하는 데 사용된다.

- 우리는 버튼에 이벤트를 처리하는 함수를 붙일 수 있다. 이벤트가 발생하였을 때 호출되는 이러한 함수를 콜백함수(callback function), 또는 핸들러(handler)라고 한다.

```
Button(root, text="Click", command=process)
```

콜백함수

버튼 위젯

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

def process():
                        label["text"] = "버튼이 클릭됨"

button = Button(root, text="클릭하세요!", command=process)
button.pack()

label = Label(root, text="버튼 클릭 안됨")
label.pack()
root.mainloop()
```

엔트리 위젯

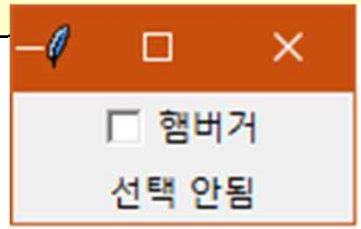
- 엔트리(Entry) 위젯은 사용자가 키보드로 입력한 내용을 우리에게 전달하는 위젯이다.
- get()을 사용하면 사용자의 입력을 가져올 수 있다. 사용자의 입력을 삭제하려면 delete()를 사용한다. 중간에 텍스트를 삽입하려면 insert()를 사용한다.

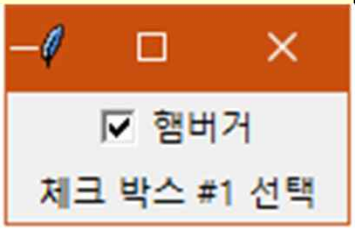
엔트리 위젯

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

def process():
                        label["text"] = entry.get()+"가 입력됨"

entry = Entry(root, fg="black", bg="yellow", width=20)
entry.pack()

button = Button(root, text="입력 후 클릭하세요!", command=process)
button.pack()
label = Label(root, text="아무 것도 입력 안됨!")
label.pack()
root.mainloop()
```

체크 버튼 위젯

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

def process():
        if var1.get() == 1:
                label["text"] = "체크 박스 #1 선택"
        else:
                label["text"] = "체크 박스 #1 선택 해제"

var1 = IntVar()
Checkbutton(root, text="햄버거", variable=var1, command=process).pack()

label = Label(root, text="선택 안됨")
label.pack()
root.mainloop()
```





라디오 버튼 위젯


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

def process():
        if var1.get() == 1:
                label["text"] = "햄버거 선택"
        elif var1.get() == 2:
                label["text"] = "피자 선택"
        else:
                label["text"] = "김밥 선택"

var1 = IntVar()
Radiobutton(root, text="햄버거", variable=var1, value=1, command=process).pack()
Radiobutton(root, text="피자", variable=var1, value=2, command=process).pack()
Radiobutton(root, text="김밥", variable=var1, value=3, command=process).pack()

label = Label(root, text="선택 안됨")
label.pack()
root.mainloop()
```

위젯의 속성 변경

- 위젯을 생성하기 위하여 생성자를 호출할 때, fg, bg, font, text, command와 같은 매개 변수에 값을 전달하여서 속성을 지정할 수 있다.

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


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

label.config(text = "World")                    # 레이블의 텍스트가 World로 변경
```

중간 점검

- 다음과 같은 텍스트가 들어 있는 tkinter 윈도우를 만드는 완전한 프로그램을 작성하라.

배치 관리자

- 파이썬에서는 위젯의 배치를 배치 관리자(layout manager)에게 맡긴다. 배치 관리자는 컨테이너 안에 존재하는 위젯의 크기와 위치를 자동적으로 관리하는 객체이다.

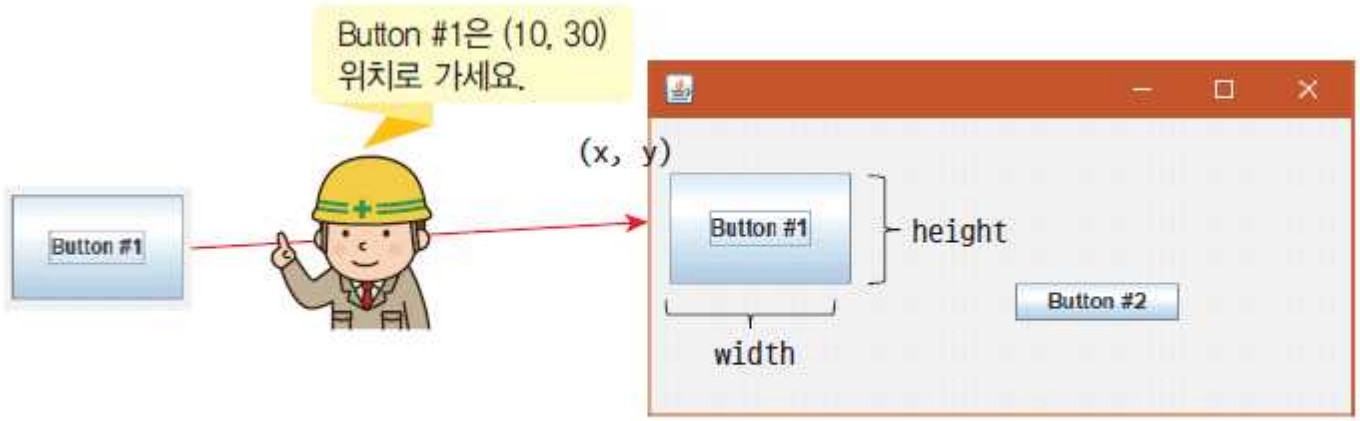
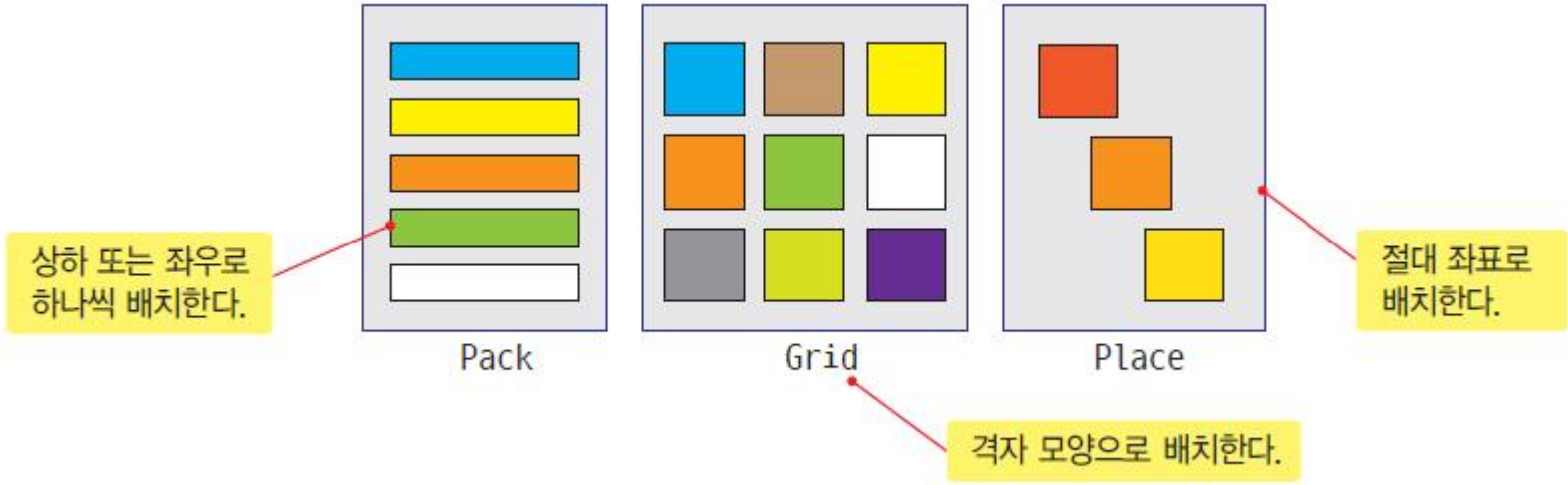


그림 11.1 배치 관리자

배치 관리자의 종류

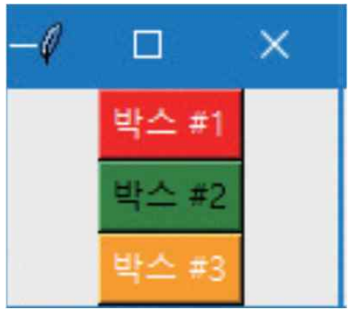
- 파이썬에서는 3가지의 기본 배치 관리자가 제공되며 같은 개수의 위젯을 가지고 있더라도 배치 관리자에 따라 상당히 달라 보일 수 있다. 파이썬은 다음과 같은 3종류의 배치 관리자를 제공한다.



압축 배치 관리자

- 격자 배치 관리자**(grid geometry manager)**는 위젯 (버튼, 레이블 등)을 테이블 형태로 배치한다.

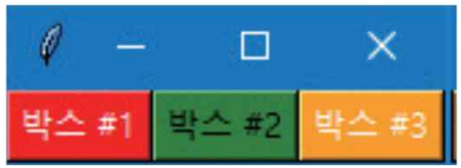
```
from Tkinter import *

root = Tk()

Label(root, text="박스 #1", bg="red", fg="white").pack()
Label(root, text="박스 #2", bg="green", fg="black").pack()
Label(root, text="박스 #3", bg="blue", fg="white").pack()

root.mainloop()
```



압축 배치 관리자

- 격자 배치 관리자**(grid geometry manager)**는 위젯 (버튼, 레이블 등)을 테이블 형태로 배치한다.

```
from Tkinter import *

root = Tk()

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

root.mainloop()
```



격자 배치 관리자

- 격자 배치 관리자(grid geometry manager)는 위젯을 테이블 형태로 배치한다. 격자 배치 관리자를 사용하면 부모 윈도우는 테이블의 셀로 분할되고, 각 위젯은 특정한 셀에 배치된다.

그림 11.2 격자 배치 관리자

격자 배치 관리자

- 격자 배치 관리자**(grid geometry manager)**는 위젯 (버튼, 레이블 등)을 테이블 형태로 배치한다.

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

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

root.mainloop()
```

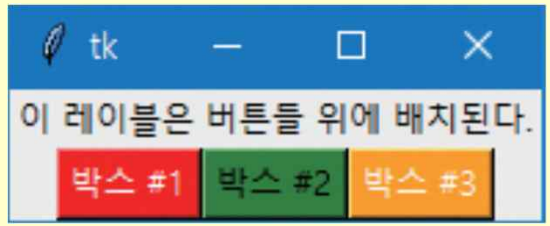
절대 위치 배치 관리자

- 격자 배치 관리자**(grid geometry manager)**는 위젯 (버튼, 레이블 등)을 테이블 형태로 배치한다.

```
from tkinter import *

root = Tk()

b1 = Button(root, text="박스 #1", bg="red", fg="white")
b1.place(x=0, y=0)
b2 = Button(root, text="박스 #2", bg="green", fg="black")
b2.place(x=20, y=30)
b3 = Button(root, text="박스 #3", bg="orange", fg="white")
b3.place(x=40, y=60)

root.mainloop()
```

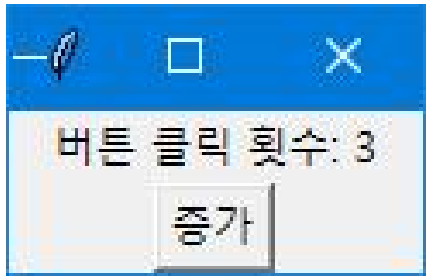
여러 배치 관리자 혼용하기

```
from tkinter import *

root = Tk()
f = Frame(root)

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(root, text="이 레이블은 버튼들 위에 배치된다.")

l.pack()
f.pack()

root.mainloop()
```



Lab: 카운터 만들기

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

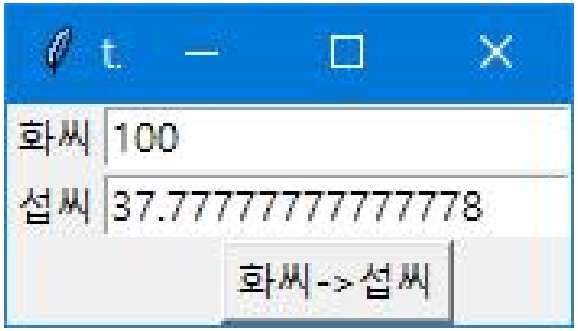


Sol:

```
from tkinter import *

root = Tk()


counter = 0

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

label = Label(root, text="아직 눌려지지 않음")
label.pack()
button = Button(root, text="증가", command=clicked)
button.pack()

root.mainloop()
```

Lab: 온도 변환기

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



Sol: 온도 변환기

```
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.

root  = Tk()

Label(root , text="화씨").grid(row=0, column=0)
Label(root, text="섭씨").grid(row=1, column=0)

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

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

숫자 추측 게임

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

실행결과

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:

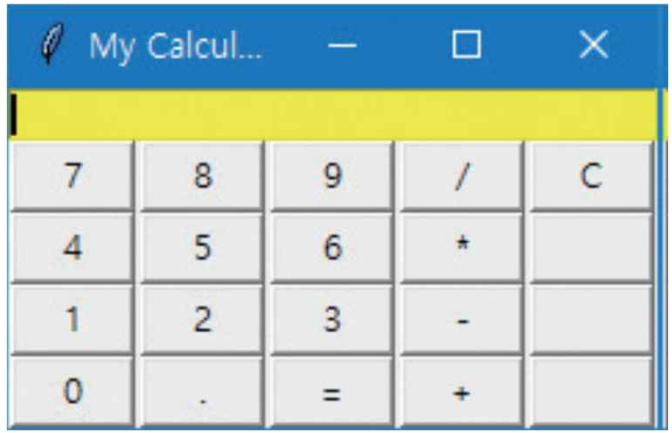
```
def reset(): # 정답을 다시 설정한다.
        global answer
        answer = random.randint(1,100)
        resultLabel["text"] = "다시 한번 하세요!"

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

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

Sol:

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

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

root.mainloop()
```

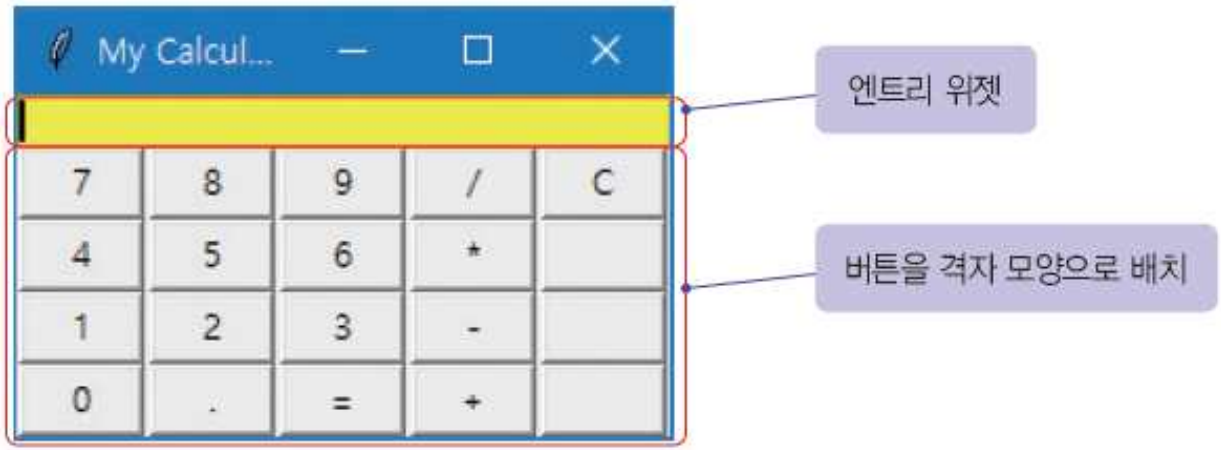
Lab: 계산기 프로그램

- 우리는 다음과 같은 계산기를 작성해보자.



사용자 인터페이스 작성

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



계산기 프로그램 #1

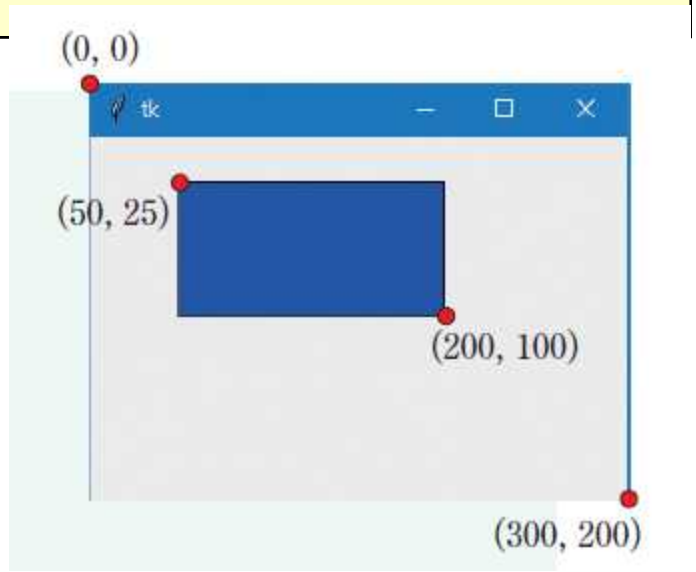
```
from tkinter import *

root = Tk()
root.title("My Calculator")
display = Entry(root, 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)
```

계산기 프로그램 #2

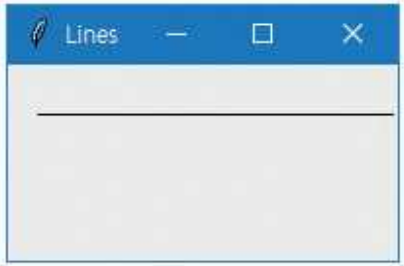
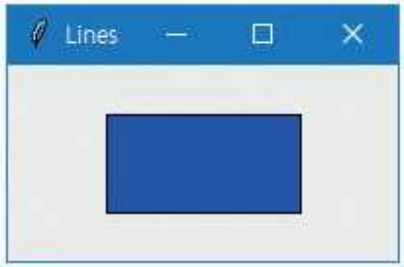
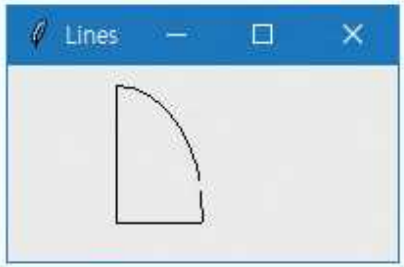
```
row_index = 1
col_index = 0
for button_text in button_list:
    Button(root, text=button_text, width=5,
                        command=lambda t=button_text: click(t)).grid(row=row_index,
                                column=col_index)
    col_index += 1
    if   col_index > 4:
        row_index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

화면에 그림 그리기

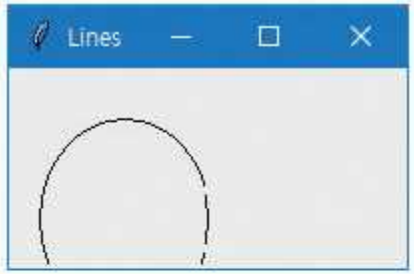
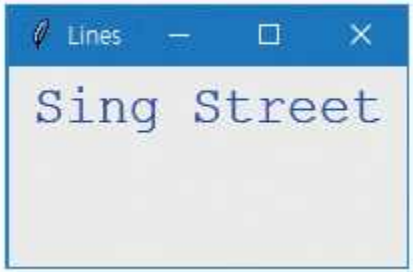
```
from tkinter import *

root = Tk()
w = Canvas(root, width=300, height=200)
w.pack()

w.create_rectangle(50, 25, 200, 100, fill="blue")
root.mainloop()
```



기초 도형 그리기

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.	

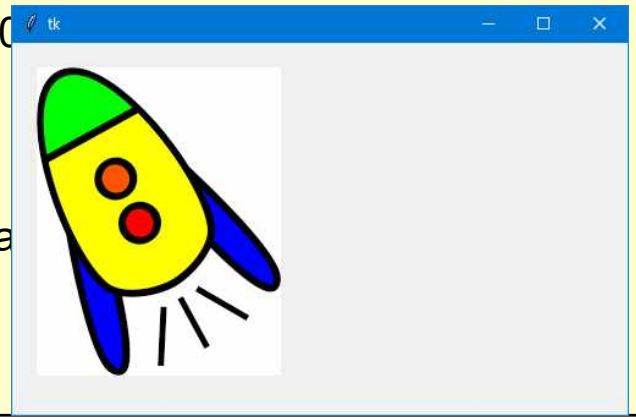
기초 도형 그리기

`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

- tkinter에서 이미지를 표시하려면 먼저 이미지를 적재하여야 한다. 파이썬이 접근할 수 있는 디렉토리에 이미지가 있어야 한다. tkinter가 읽을 수 있는 이미지 파일은 PNG 파일과 JPG 파일 뿐이다.
- create_image() 함수를 사용하여 캔버스에 그리면 된다. 현재 디렉토리에 있는 starship.png 이미지 파일을 읽어서 캔버스에 표시하는 예제는 다음과 같다.

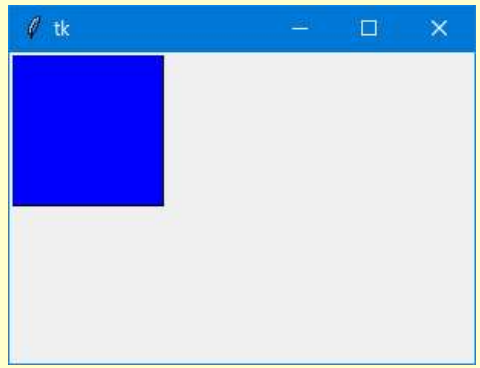
```
from tkinter import *
root = Tk()

canvas = Canvas(root,  width=500, height=300
canvas.pack()

img = PhotoImage(file="D:\\starship.png")
canvas.create_image(20, 20, anchor=NW, ima

root.mainloop()
```



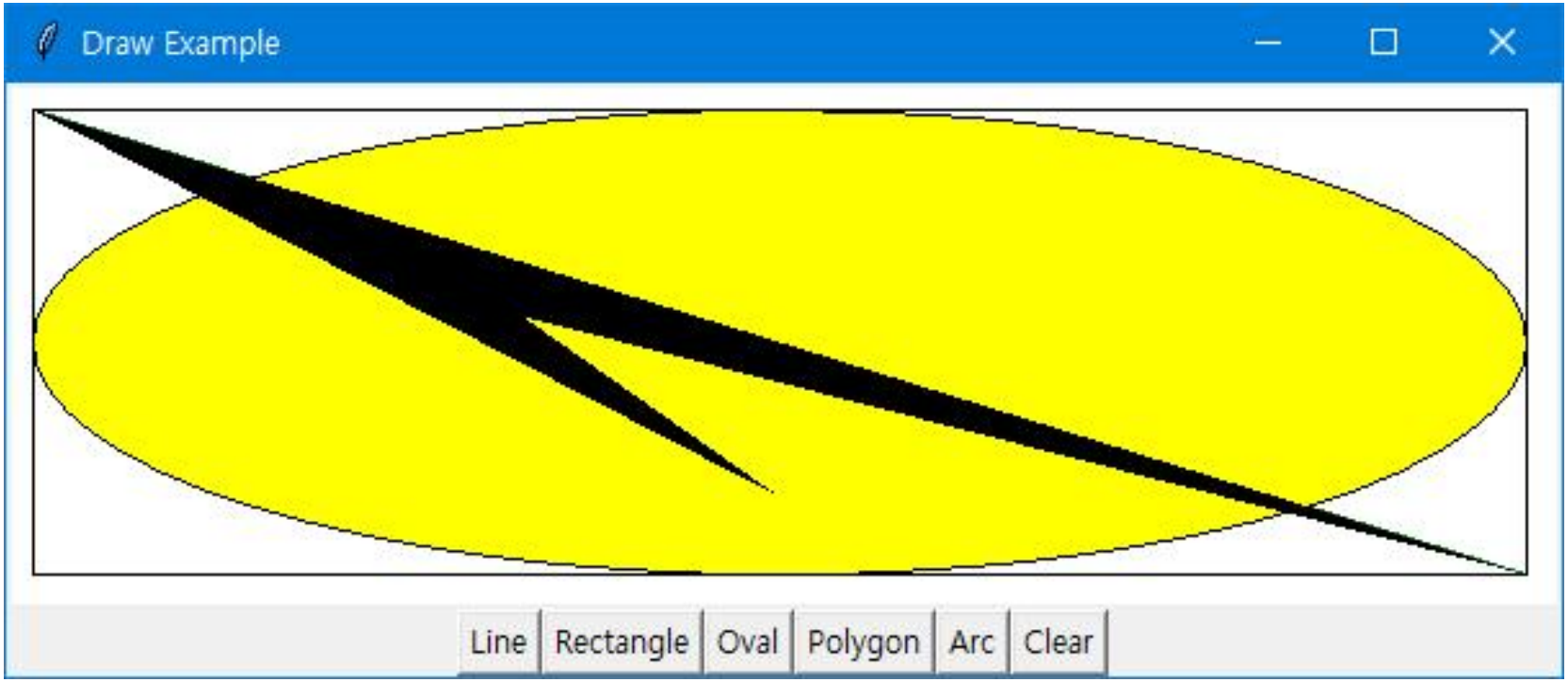
도형 관리

- 캔버스 위젯에 추가된 항목(선, 사각형, 원)들은 삭제하기 전까지 유지된다. 만약 그리기 속성을 변경하고 싶으면 coords(), itemconfig(), move()를 사용할 수 있다.

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

w = Canvas(root, 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.
root.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:

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

root.mainloop()
```

마우스 이벤트 처리

- tkinter 응용 프로그램은 대부분의 시간을 이벤트 루프에서 소모한다. 즉 mainloop()에서 이벤트를 기다리면서 반복 루프를 실행한다. 이것을 이벤트-구동 방식이라고 한다.

그림 11.3 tkinter에서의 이벤트 처리

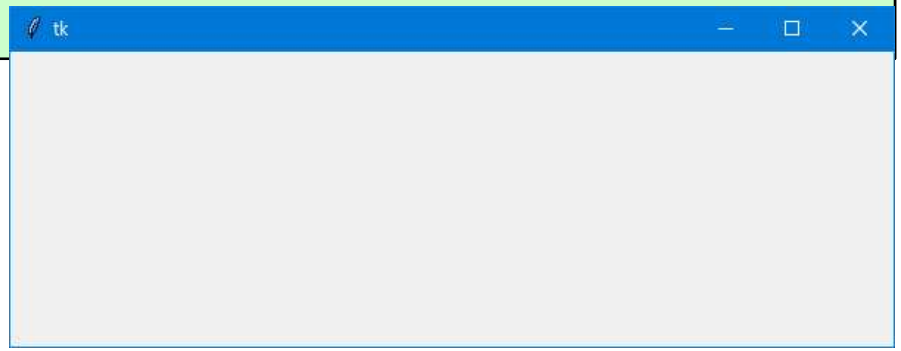
마우스 이벤트 처리

- tkinter는 이벤트를 처리하는 강력한 메카니즘을 가지고 있다. 각 위젯에 대하여 개발자는 파이썬 함수를 붙일 수 있다.

마우스 이벤트 처리

```
from tkinter import *

root = Tk()
root.geometry("600x200")

def callback(event):
    print(event.x, event.y, "에서 마우스 이벤트 발생")

root.bind("<Button-1>", callback)
root.mainloop()
```

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

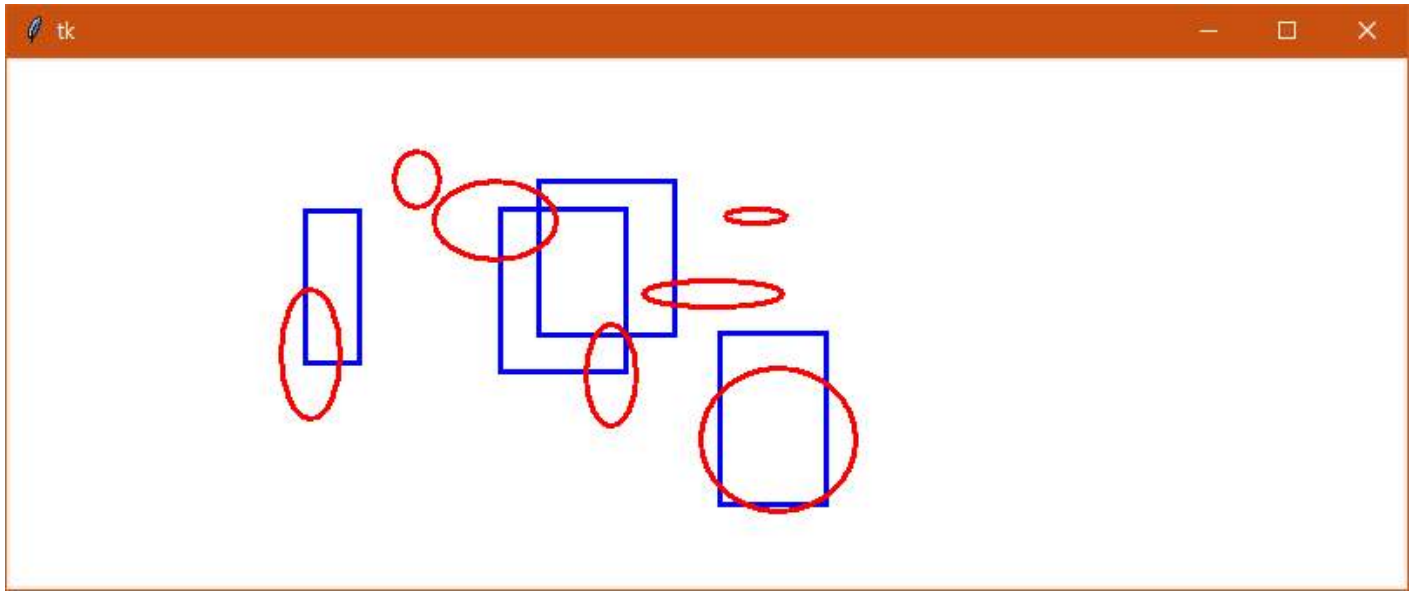


이벤트 지정자

이벤트	설명
〈Button〉	마우스 버튼이 눌려 졌을 때 발생하는 이벤트이다. 〈Button-1〉이 마우스의 왼쪽 버튼이고 〈Button-2〉이 중간 버튼, 〈Button-3〉이 오른쪽 버튼이다. 마우스 포인터의 현재 위치는 이벤트 객체의 x와 y 멤버에 저장된다. 위치는 위젯에 상대적이다.
〈Motion〉	마우스가 움직이면 발생한다. 버튼을 누르면서 움직이는 이벤트는 마우스 버튼의 위치에 따라 〈B1-Motion〉, 〈B2-Motion〉, 〈B3-Motion〉 등이 있다.
〈ButtonRelease〉	마우스 버튼을 놓을 때 발생한다. 〈ButtonRelease-1〉 등이 있다.
〈DoubleButton〉	버튼이 더블 클릭될때 발생한다.
〈Enter〉	마우스 포인터가 위젯으로 진입하였을 때 발생한다. 사용자가 엔터키를 눌렀다는 것이 아니다.
〈Leave〉	마우스 포인터가 위젯을 떠났을 때 발생한다.
〈return〉	사용자가 엔터키를 입력하면 발생한다.
〈Key〉	사용자가 어떤 키를 누르면 발생한다.
a	사용자가 "a"를 입력하였을 때 발생한다. 대부분의 인쇄 가능한 문자는 이런 식으로 이벤트를 연결할 수 있다.

Ex: 마우스로 도형 그리기

- 사용자가 마우스 왼쪽 버튼을 누르면 사각형이 그려지고 오른쪽 버튼을 누르면 원이 그려지는 프로그램을 작성하여 보자. 원의 크기와 사각형의 크기는 난수로 결정된다.

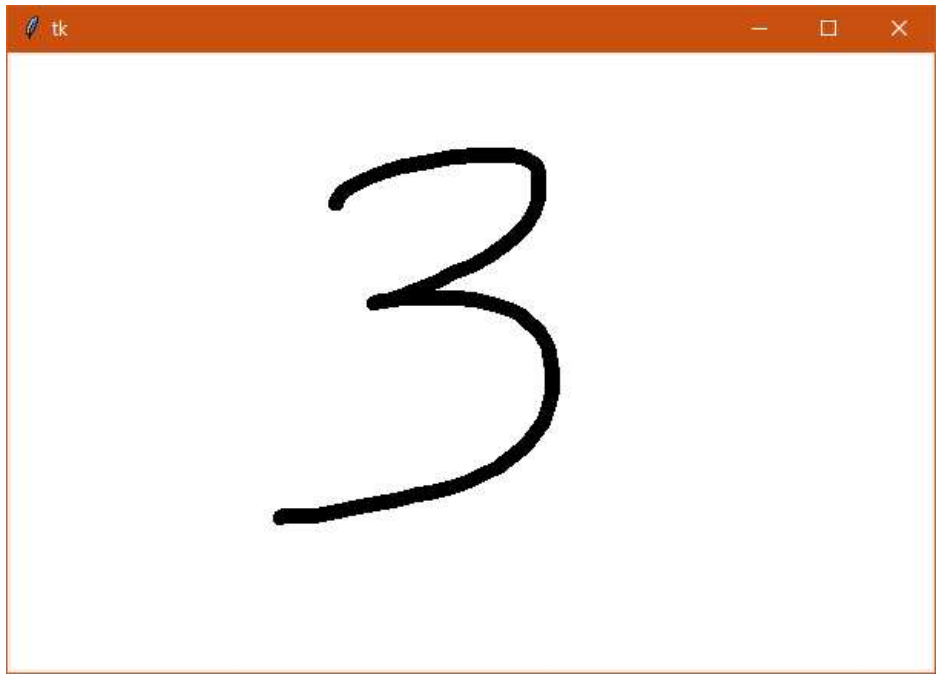


Ex: 마우스로 도형 그리기

```
from tkinter import *
import random

def drawRect(e):
        canvas.create_rectangle(e.x, e.y, e.x+random.randint(10, 100),
                                e.y+random.randint(5, 100), width=3,
                                outline="blue")

def drawCircle(e):
        canvas.create_oval(e.x, e.y, e.x+random.randint(10, 100),
                                e.y+random.randint(5, 100), width=3,
outline="red")

root = Tk()
canvas=Canvas(root, width=800, height=300, bg='white')
canvas.pack()

root.bind("<Button-1>", drawRect)
root.bind("<Button-3>", drawCircle)
root.mainloop()
```

Lab: 그림판 프로그램 만들기

- 캔버스에서는 2개의 이벤트를 처리한다. 첫 번째는 "<B1-Motion>"으로 왼쪽 버튼을 누른 채로 움직이면 발생하는 이벤트이다. 두 번째는 "<ButtonRelease-1>"로 버튼을 놓았을 때 발생하는 이벤트이다.



Sol:

```
from tkinter import *

mode = "pen"
old_x = None
old_y = None
mycolor = "black"

def paint(event):      # 이전 점과 현재 점 사이를 직선으로 연결한다.
    global mode, old_x, old_y
    fill_color = mycolor
    if   old_x and old_y:
        canvas.create_line(old_x, old_y, event.x, event.y,
                           capstyle=ROUND, width=10, 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

root = Tk()
```

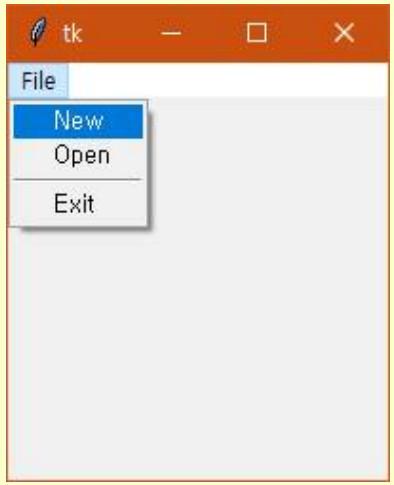
Sol:

```
canvas = Canvas(root, bg='white', width=600, height=400)
canvas.grid(row=1, columnspan=5)
canvas.bind('<B1-Motion>', paint)
canvas.bind('<ButtonRelease-1>', reset)

root.mainloop()
```

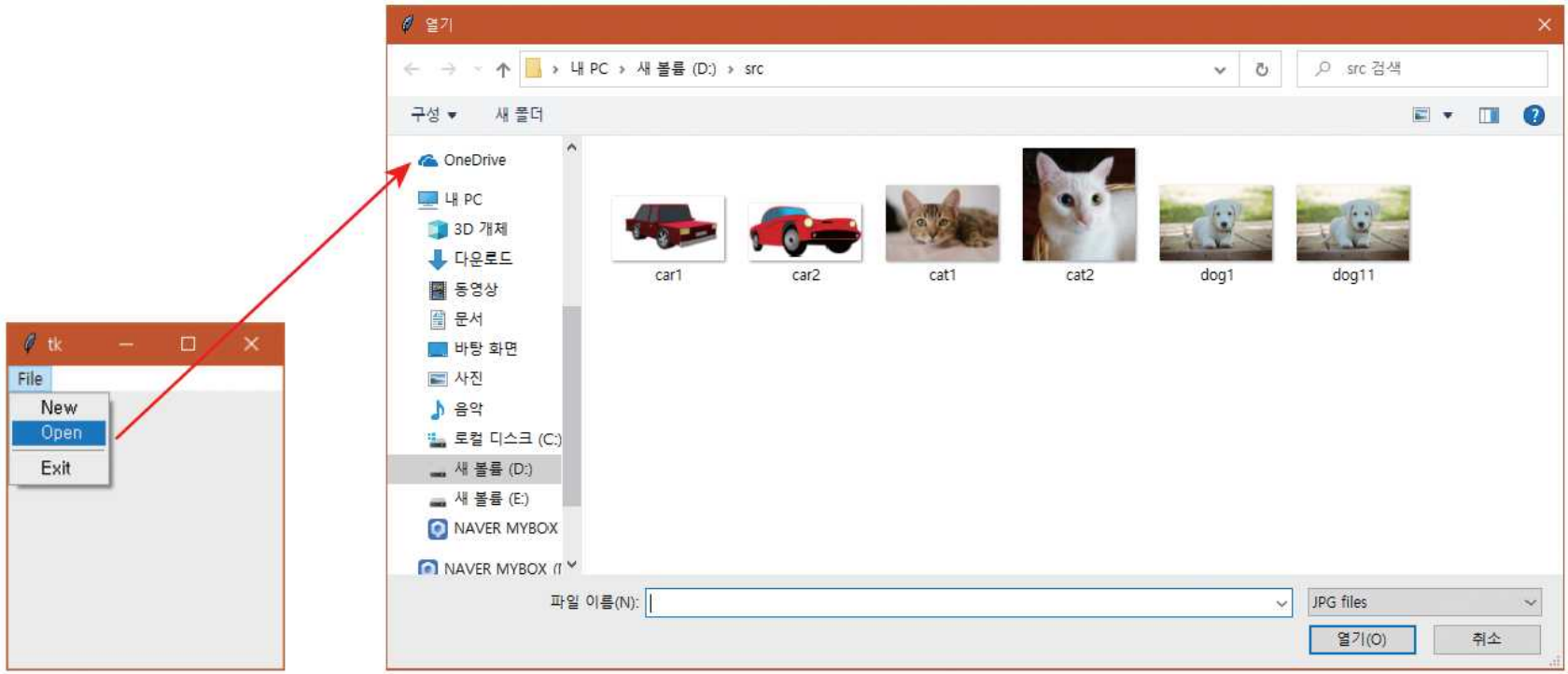
메뉴

- 제일 먼저 루트 윈도우에 메뉴바를 생성한다. 메뉴바 아래에 다시 메뉴 객체를 생성하고 메뉴 객체에 메뉴 항목들을 add_command() 함수를 호출하여서 추가한다.

```
from tkinter import *

def callback():
    print("메뉴가 선택되었음")

root = Tk()

menubar = Menu(root)

filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=callback)
filemenu.add_command(label="Open", command=callback)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)

menubar.add_cascade(label="File", menu=filemenu)

root.config(menu=menubar)
root.mainloop()
```



파일 열기 대화 상자

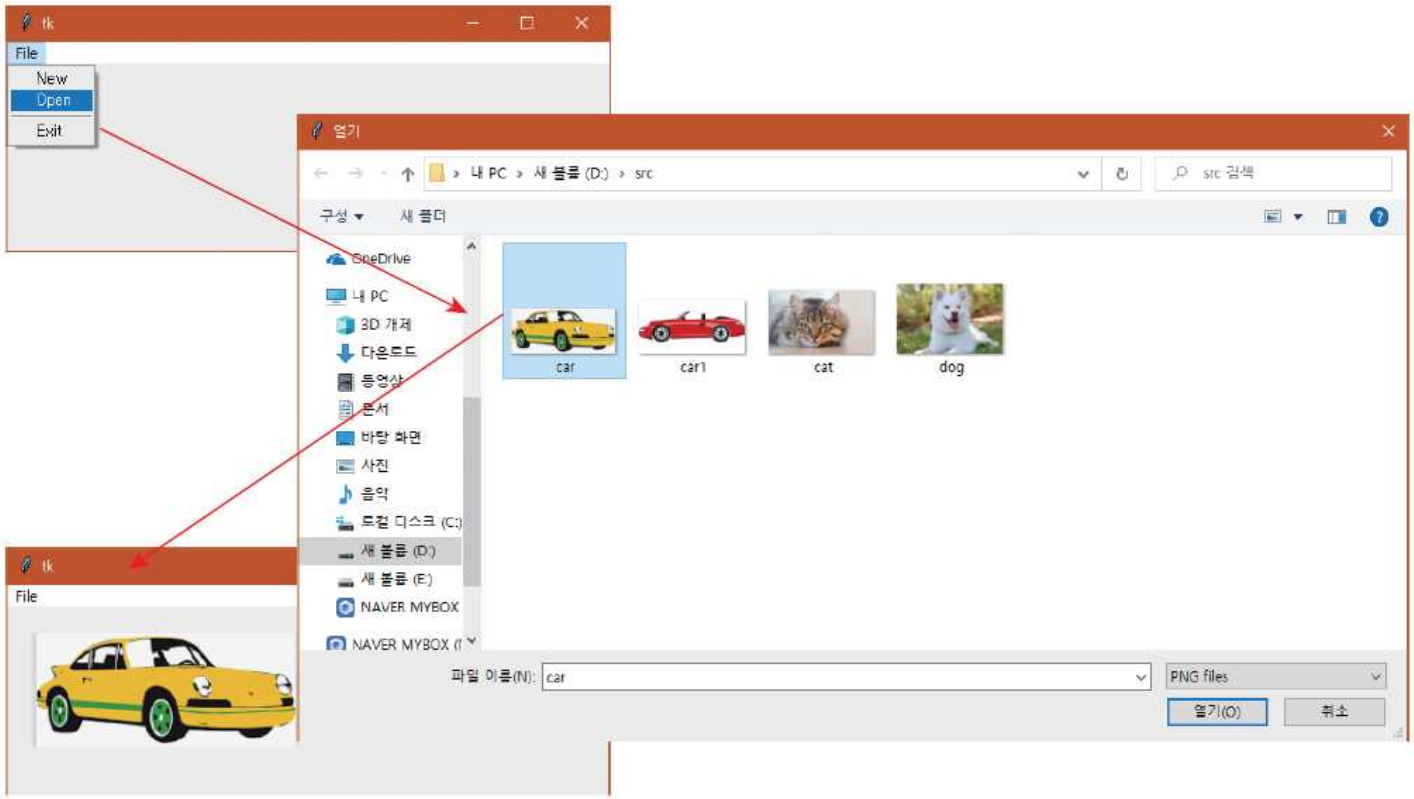

tk
File
New
Open
Exit
열기
내 PC › 새 볼륨 (D:) › src
src 검색
구성
새 폴더
OneDrive
내 PC
3D 개체
다운로드
동영상
문서
바탕 화면
사진
음악
로컬 디스크 (C:)
새 볼륨 (D:)
새 볼륨 (E:)
NAVER MYBOX
car1
car2
cat1
cat2
dog1
dog11
파일 이름(N):
JPG files
열기(O)
취소

파일 열기 대화 상자

```
from tkinter import *
from tkinter import filedialog

def FileOpen():
    filename = filedialog.askopenfilename(parent=root,
                                          filetypes=(("JPG files", "*.jpg"),
                                          ("all files", "*.*")))
    print(filename)

root = Tk()

menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New")
filemenu.add_command(label="Open", command=FileOpen)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)

root.config(menu=menubar)
root.mainloop()
```

이미지 그리기 프로그램

- 메뉴와 파일 열기 대화 상자를 이용하여서 이미지를 하나 선택하여서 화면에 그리는 프로그램을 작성해보자.



이미지 그리기 프로그램

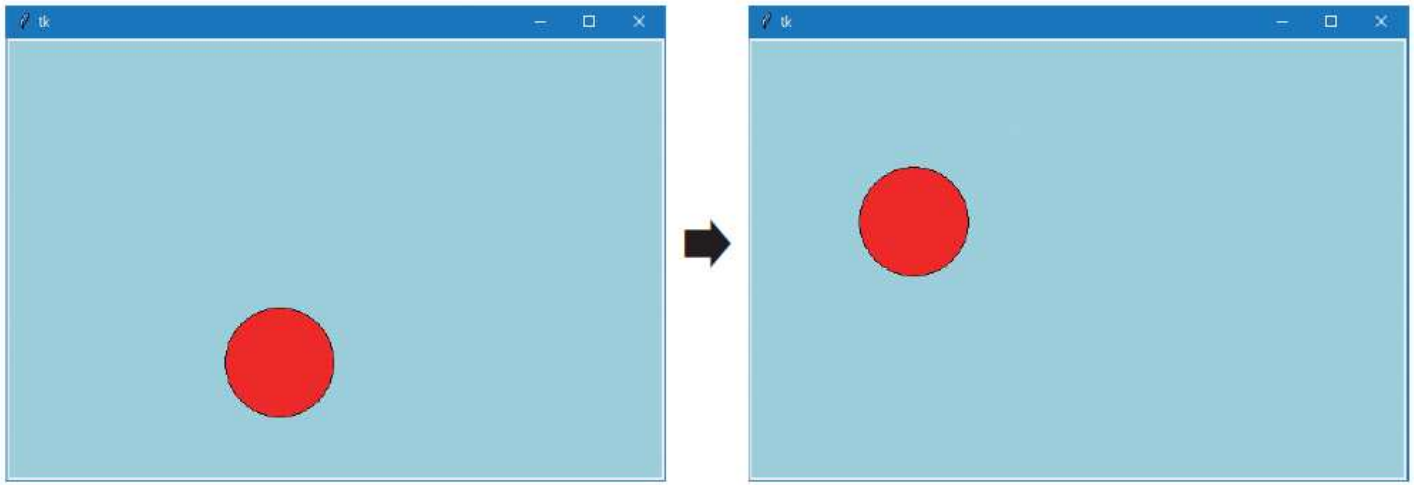
```
from tkinter import *
from tkinter import filedialog

def FileOpen():
    global img
    filename = filedialog.askopenfilename(parent=root,
                                          filetypes=(("PNG files", "*.png"),
                                          ("all files", "*.*")))
    img = PhotoImage(file=filename)
    canvas.create_image(20, 20, anchor=NW, image=img)

root = Tk()
canvas = Canvas(root,  width=500, height=150)
canvas.pack()
img = None                          # 이미지 객체를 가르키는 변수가 미리 생성되어 있어야 한다.
```

이미지 그리기 프로그램

```
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New")
filemenu.add_command(label="Open", command=FileOpen)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)

root.config(menu=menubar)
root.mainloop()
```

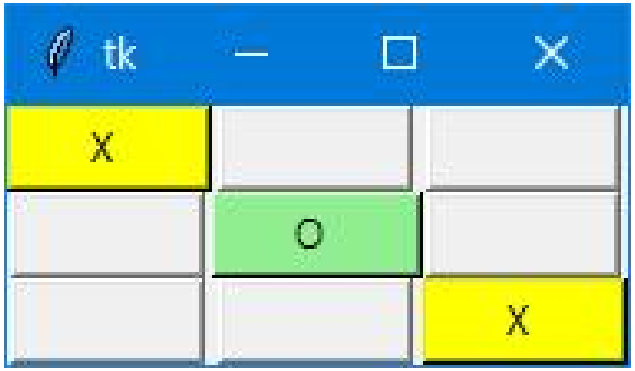
Mini Project: 애니메이션

- 파이썬을 이용하여 애니메이션을 작성하려면 일정한 시간 간격으로 조금씩 달라지는 그림을 화면에 그리면 된다. 예를 들어서 원이 화면에서 반사되면서 움직이는 애니메이션을 작성해보자.



Mini Project: TIC-TAC-TOE 게임

- Tic-Tac-Toe는 3×3칸을 가지는 게임판을 만들고, 경기자 2명이 동그라미 심볼(O)와 가위표 심볼(X)을 고른다. 경기자는 번갈아 가며 게임판에 동그라미나 가위표를 놓는다.



이번 장에서 배운 것

- tkinter에서는 먼저 루프 윈도우를 생성하고 레이블이나 버튼을 생성할 때 첫 번째 인수로 윈도우를 넘기면 된다.
- 파이썬은 3종류의 배치 관리자를 제공한다. 압축(pack) 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.

Q & A