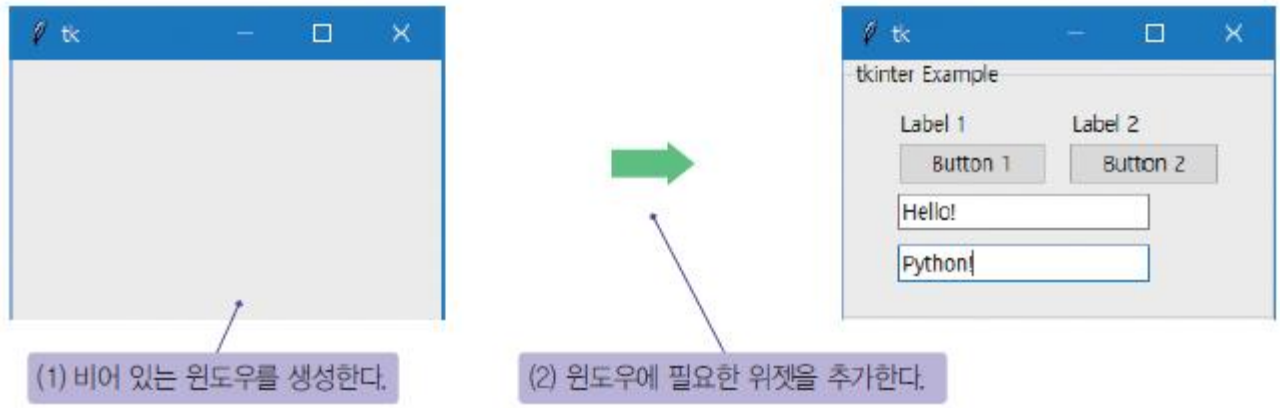
11장 TKINTER를 이용한 GUI 프로그래밍

tkinter 시작하기. p377

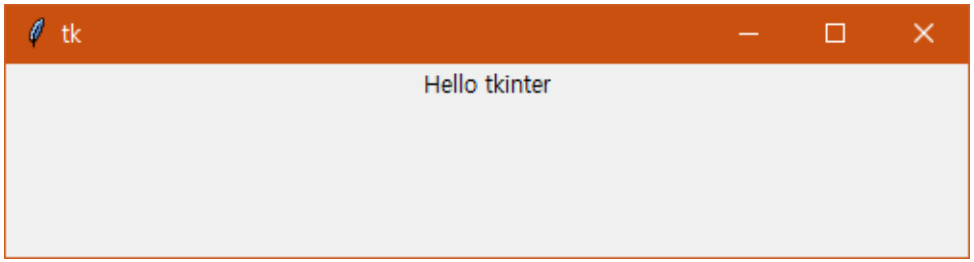
- tkinter : 파이썬에서 그래픽 사용자 인터페이스(GUI: graphical user interface)를 개발할 때 필요한 모듈

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



Hello tkinter 프로그램. p377

- “Hello tkinter”라는 레이블만 가지는 윈도우를 생성해보자




```
from tkinter import *                          # tkinter 모듈을 포함

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

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

기본 위젯들. p378

- 위젯들을 이용하여 사용자가 프로그램과 상호 작용한다.

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

레이블 위젯. p379

- 레이블 위젯의 속성
  - 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. 단위는 글자 개수이다.

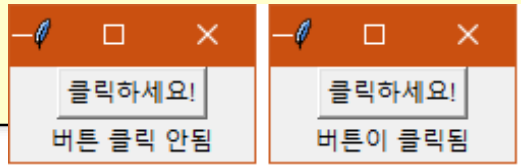
```
from tkinter import *
root = Tk()

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. p380

- 버튼에는 이벤트를 처리하는 함수를 붙일 수 있다.
- 이벤트가 발생했을 때 호출되는 함수를 콜백함수(callback function), 또는 핸들러(handler)라고 한다. Command 키워드에 함수의 이름을 전달한다.

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

```
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. p380

- 엔트리(Entry) 위젯 : 사용자가 키보드로 입력한 내용을 전달
  - get() : 사용자의 입력을 가져온다.
  - delete() : 사용자의 입력을 삭제
  - insert() : 중간에 텍스트를 삽입


```
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. p381

- 체크 버튼에 변수를 연결해야 하며 IntVar()을 호출해서 정수형 변수를 동적으로 생성한다(참고. 뒷면). 이 변수에 1이 반환되면 체크된 상태.

```
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()
```

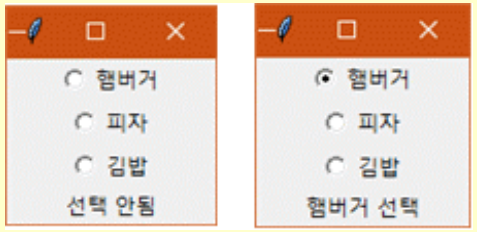
체크 버튼 위젯. p381

- (참고) tkinter에서는 변수선언시 일반코딩하듯 변수 선언을 하면 에러가 발생한다. Tkinter에서 제공하는 함수를 사용해서 선언해야 한다.
  - StringVar : string 변수 선언
  - IntVar : 정수 변수 선언
  - DoubleVar : 실수 변수 선언
  - BooleanVar : True, False 변수 선언

라디오 버튼 위젯. p382

- 여러 개 중에서 하나만 선택할 때 사용

```
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. p382

- 위젯을 생성(생성자를 호출)할 때는 fg, bg, font, text, command와 같은 매개 변수에 값을 전달하여 속성을 지정.

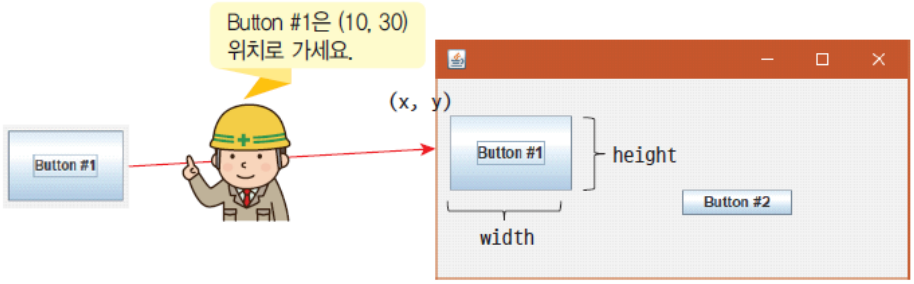
```
label = Label(root,  text="Times Font 폰트와 빨강색을 사용합니다.", fg = "red", font = "Times 32 bold italic")
```

- 위젯이 생성되고 나면 딕셔너리 형식을 사용하여 위젯 속성을 변경

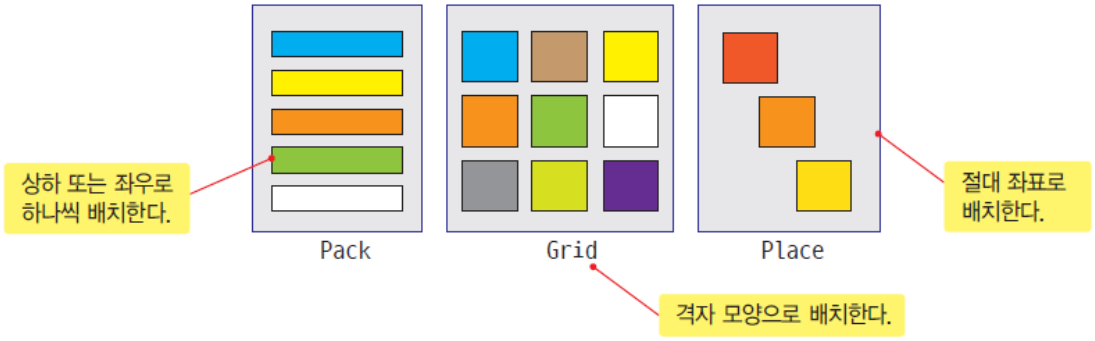
```
label["text"] = "This is a label."
label["fg"] = "blue"
label["bg"] = "#FF00AA"

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

배치 관리자. p383

- 배치 관리자(layout manager) : 컨테이너 안에 존재하는 위젯의 크기와 위치를 자동적으로 관리하는 객체



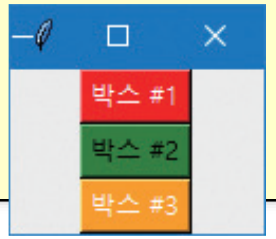
- 3가지의 기본 배치 관리자가 제공되며 같은 개수의 위젯을 가지고 있더라도 배치 관리자에 따라 상당히 달라 보일 수 있다



압축 배치 관리자. p384

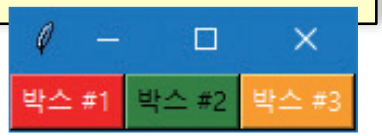
- 압축 배치 관리자(pack geometry manager) : 제일 간단한 배치 관리자로 위젯을 상하 또는 좌우로 배치

```
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()
```



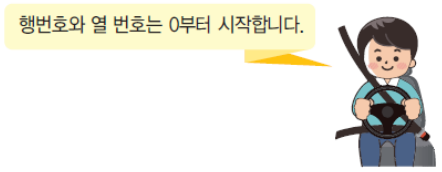
- side=LEFT 등을 붙여 좌우로 배치할 수도 있다.

```
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)
```



격자 배치 관리자. p384

- 격자 배치 관리자(grid geometry manager) : 위젯을 테이블 형태로 배치

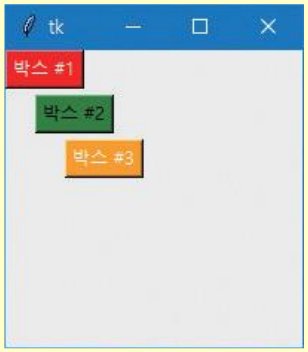



```
from tkinter import *
root = Tk()

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. p385

- 절대 위치 배치 관리자(place geometry manager) : 절대 위치를 사용하여 위젯을 배치

```
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 활용하기. p386

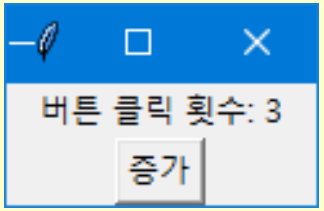
- 하나의 컨테이너 안에 다른 컨테이너를 배치하고 컨테이너마다 배치 관리자를 다르게 할 수 있다. 컨테이너로 가장 많이 사용되는 것은 프레임(Frame)이다.

```
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. p387

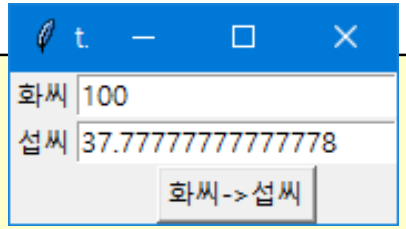
- 레이블과 버튼을 사용하여 간단한 카운터를 작성하여 보자. 증가 버튼을 누르면 카운터가 1씩 증가된다.

```
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. p388

- 온도 변환 프로그램의 위젯들을 다음과 같이 배치하고 “화씨->섭씨” 버튼을 누르면 입력한 화씨 온도가 섭씨온도로 변환되어서 나타나게 해보자.




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

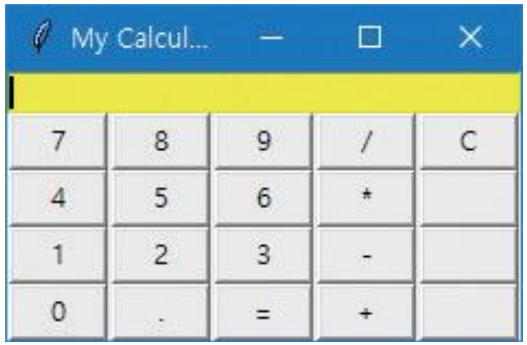
- (Quiz) 기능을 추가하자.

Lab 숫자 추측 게임. p389

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

실행결과

Lab 계산기 프로그램. p391

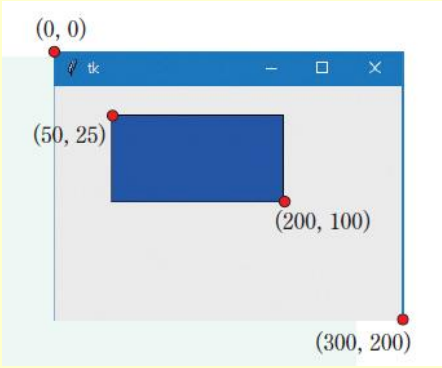


- (Quiz) 기능을 추가하자.
  - ‘C’
  - ‘<-’

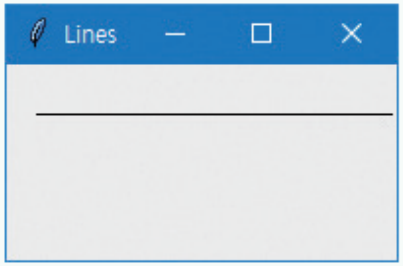
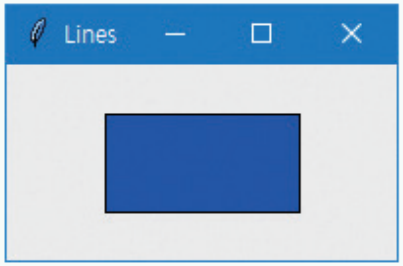
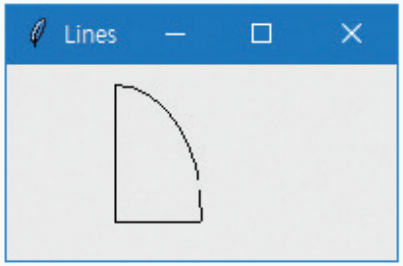
화면에 그림 그리기. p393

- 캔버스(canvas) 위젯을 윈도우 위에 생성한 후에 캔버스에 그림을 그린다. 그래픽에서는 왼쪽 상단이 (0, 0)이 되는 좌표계를 사용한다.
- 캔버스에 사각형을 그리는 코드

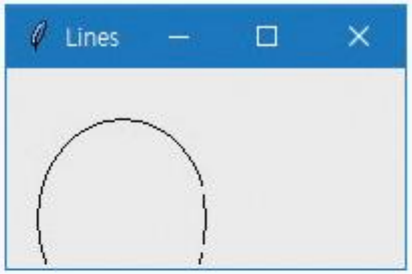
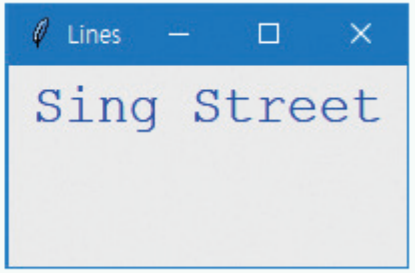
```
from tkinter import *

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. p393

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. p394

`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. p394

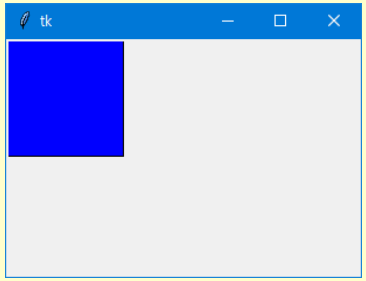
- tkinter가 읽을 수 있는 이미지 파일은 PNG 파일과 JPG 파일
- PhotoImage()에 의해 이미지를 읽어 변수에 저장한 후 create_image() 함수를 사용하여 캔버스에 그린다.

```
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, image=img)

root.mainloop()
```

도형 관리. p395

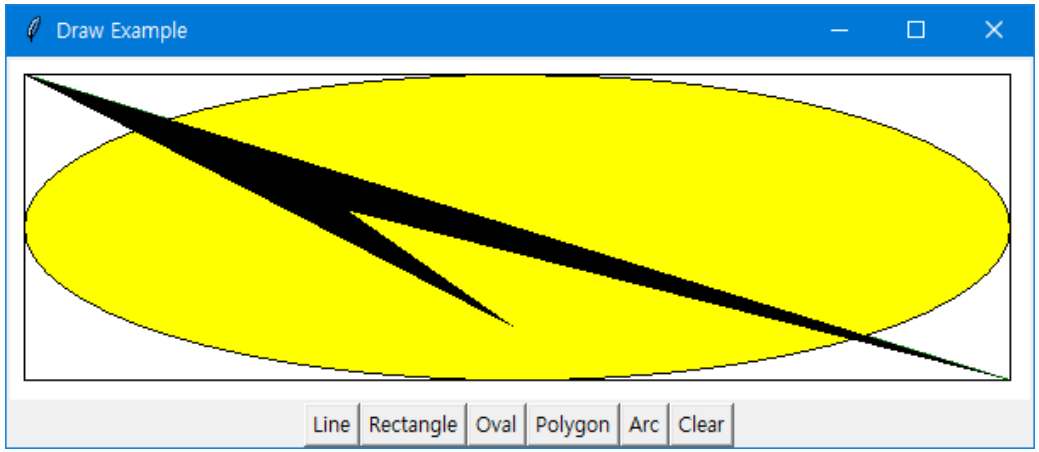
- coords() : 좌표를 변경한다
- itemconfig() : 도형의 속성을 변경한다
- delete() : 도형을 삭제한다

```
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. p396

- 사용자가 버튼을 클릭하면 해당 도형을 캔버스 위에 그리는 프로그램을 작성해보자.



마우스 이벤트 처리. p398

- 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

- 키보드의 키를 눌러도 이벤트가 발생하고 마우스 버튼을 눌러도 이벤트가 발생한다. tkinter는 이벤트를 처리하는 강력한 메카니즘을 가지고 있다.

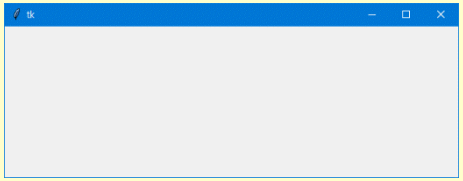
```
root.bind("<Button-1>", callback)
```

위젯 | 이벤트 지정자 | 이벤트 처리 함수

마우스 이벤트 처리. p398

- 루트 윈도우에서 마우스 버튼을 누르면 콘솔에 “32 44에서 마우스 이벤트 발생”과 같은 메시지를 출력

```
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()
```



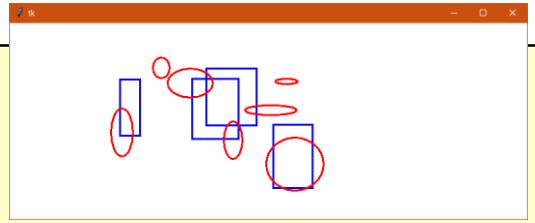
```
32 44 에서 마우스 이벤트 발생
6 52 에서 마우스 이벤트 발생
```

이벤트 지정자. p399

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.

Example 마우스로 도형 그리기. p399

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

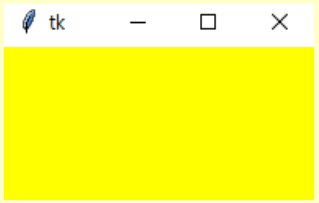


```
from tkinter import *
import random

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

Example 키보드 이벤트. p399

- (추가) 교재에 없는 예제 - 키보드 이벤트를 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=200, height=100, ba
frame.bind("<Key>", key)
frame.bind("<Button-1>", callback)
frame.pack()

window.mainloop()
```

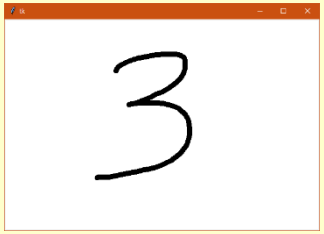



```
168 62 에서 마우스 이벤트 발생
'a' 가 눌렸습니다.
'2' 가 눌렸습니다.
'-' 가 눌렸습니다.
'/' 가 눌렸습니다.
'g' 가 눌렸습니다.
'7' 가 눌렸습니다.
```

Lab 그림판 프로그램 만들기. p401

- 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.

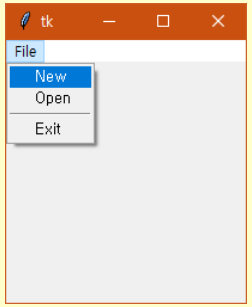
```
..
...
def paint(event):      # 이전 점과 현재 점 사이를 직선으로 연결한다.
    global mode, old_x, old_y
    fill_color = mycolor
    if   old_x and old_y:
        canvas.create_line(old_x, old_y, event.x, event.y,
                           capstyle=ROUND, width=10, fill=fill_color)
    old_x = event.x
    old_y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


…
…
root = Tk()
```



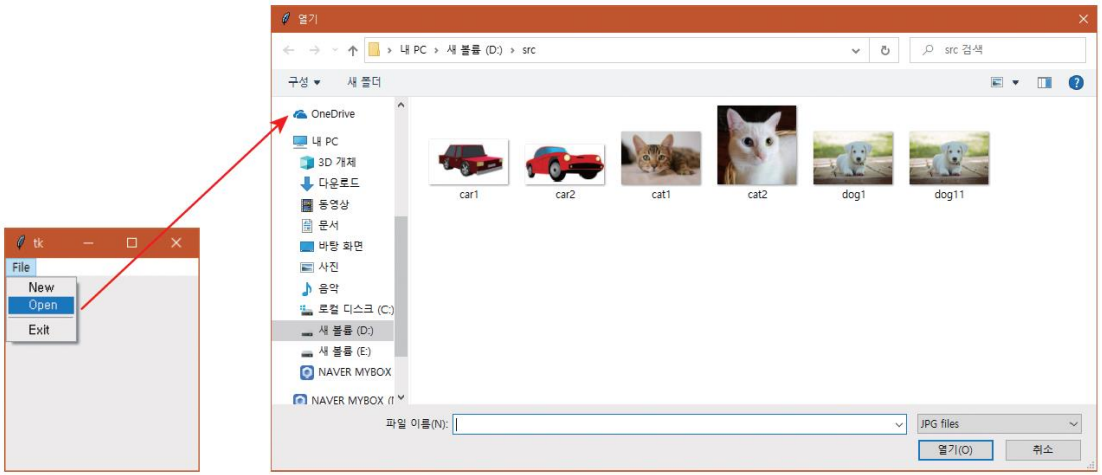
메뉴. p403

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

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

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

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. p403



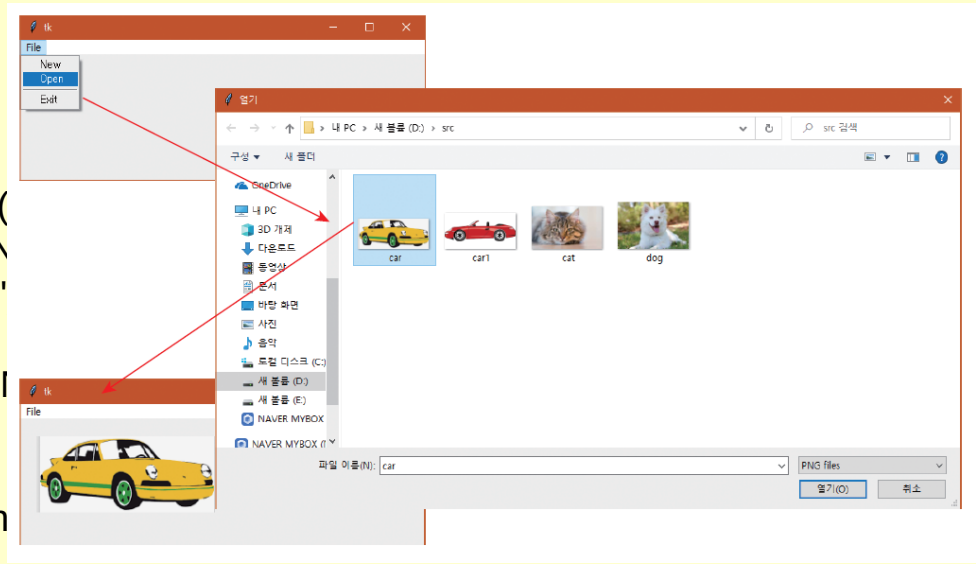
파일 열기 대화 상자. p404

```
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. p404

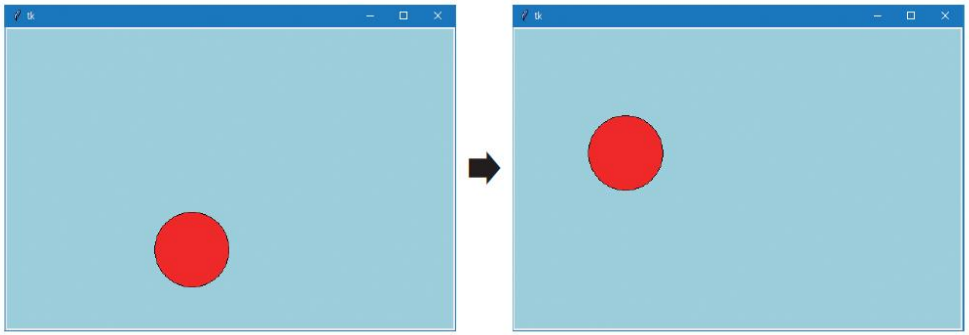
- 메뉴와 파일 열기 대화 상자를 이용하여서 이미지를 하나 선택하여 화면에 그리는 프로그램을 작성해보자.

```
from tkinter import *
from tkinter import filedialog

def FileOpen():
    global img
    filename = filedialog.askopenfilename(
                                filetypes=(("PN
                                ("all files", "*.*"
    img = PhotoImage(file=filename)
    canvas.create_image(20, 20, anchor=I

root = Tk()
canvas = Canvas(root,  width=500, heigh
canvas.pack()
img = None                  # 이미지 객체를 가르키는 변수가 미리 생성되어 있어야 한다.
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
…
…
```



Mini Project 애니메이션. p406

- 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. p407

연습문제. p409

Programming. p411