

Chapter 11

Tkinter와 GUI 프로그래밍

임경태

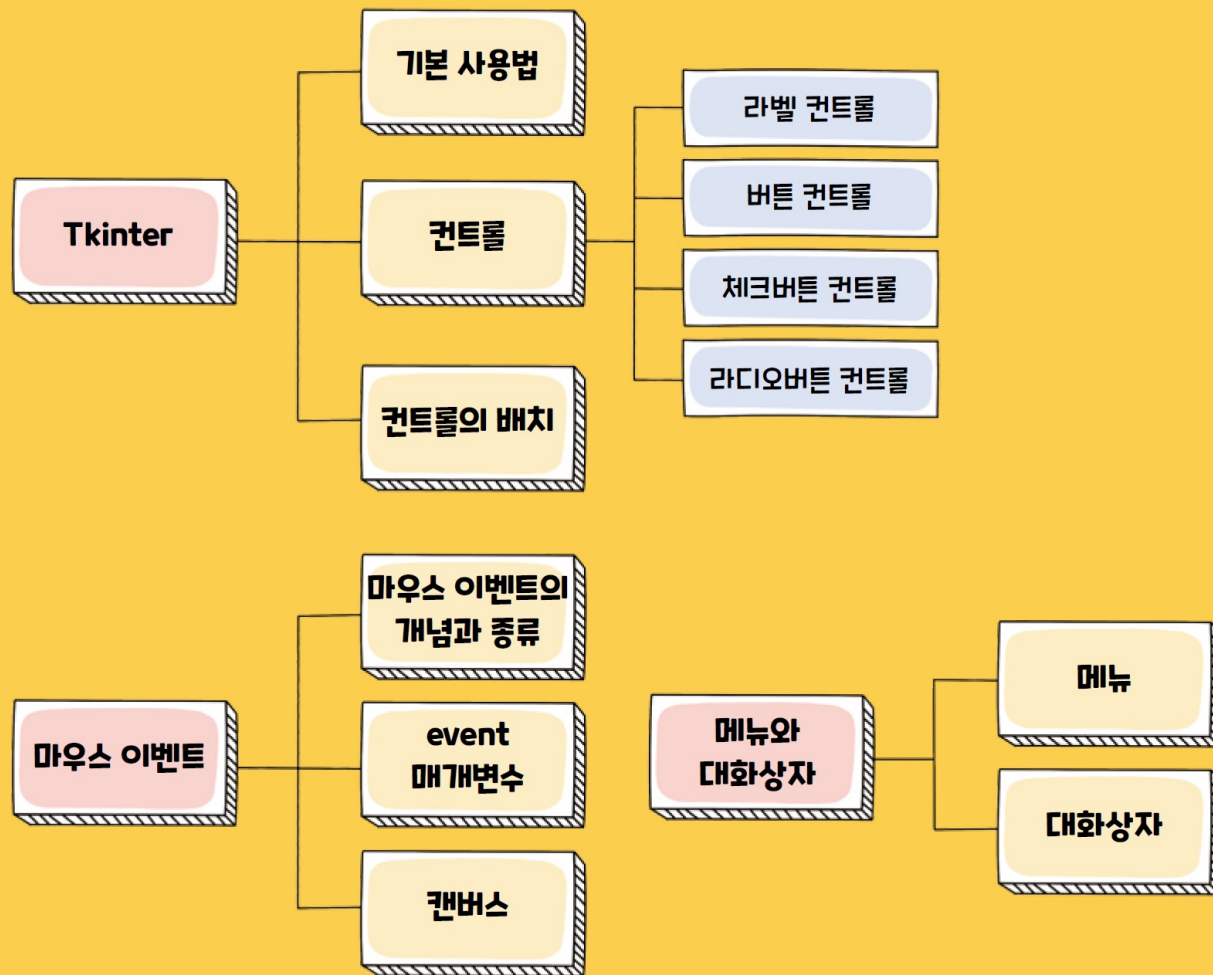


목차

1. Tkinter 라이브러리
2. 마우스 이벤트 처리하기
3. 메뉴와 대화상자 만들기

[실전 예제] GUI로 편하게 거북이 그리기

Preview



학습목표

- 윈도우 창이 나오는 GUI 프로그램을 작성합니다.
- 다양한 컨트롤의 활용법을 학습합니다.
- 마우스 클릭을 처리하는 프로그램을 작성합니다.
- 메뉴 및 대화상자를 만드는 방법을 학습합니다.

Section 01

Tkinter 라이브러리



■ GUI 프로그램

- 윈도우 창이 나오는 프로그램
- GUI(Graphical User Interface)
- 윈도우 창을 쉽게 만들 수 있는 Tkinter 라이브러리

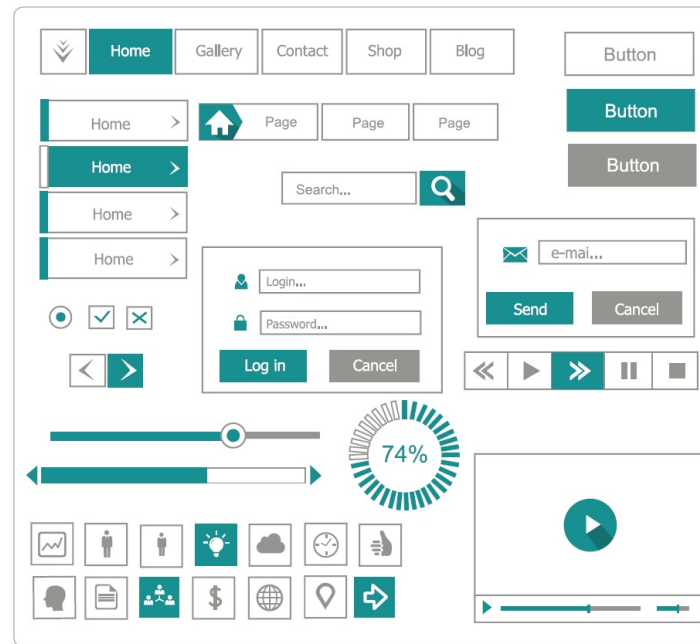


그림 12-1 GUI 프로그램



■ 컨트롤(Control)

- Tkinter의 핵심은 컨트롤을 다루는 것임
- 컨트롤(Control)은 윈도우 창에 나올 수 있는 문자, 버튼, 체크박스, 라디오버튼 등을 의미함

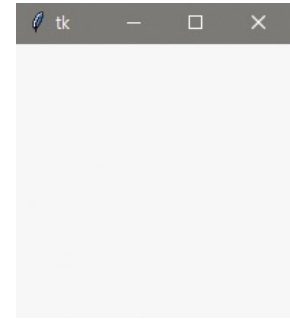


■ 빈 윈도우 창 만들기

- Tk() : 기본 윈도우 창을 반환하며 이를 루트 윈도우(Root Window)라고도 부름
- 루트 윈도우 : 제일 아래에 깔린 윈도우 창

[코드 12-1]

```
from tkinter import *  
root = Tk()  
  
# 이 부분에서 컨트롤을 배치  
  
root.mainloop()
```





■ 윈도우 창 조절하기

- 윈도우 창에 제목을 달고, 크기를 지정할 수 있음
- 또한, 윈도우 창의 크기가 변경되지 않도록 고정시킬 수 있음

[코드 12-2]

```
from tkinter import *  
root = Tk()  
  
root.title("창 조절 연습")  
root.geometry("500x200")  
root.resizable(width=FALSE, height=FALSE)  
root.mainloop()
```





확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

파이썬에서 윈도우 창을 사용하기 위해서 임포트해야 하는 라이브러리는 이다.

2. 다음 빈칸에 들어갈 윈도우 창의 크기를 지정하기 위한 함수를 고르시오.

root. ("가로x세로")

① title()

② geometry()

③ resizable()

④ mainloop()

정답

Click!



■ 라벨

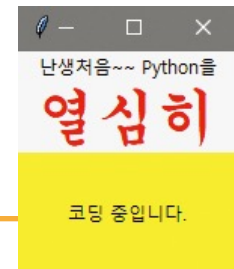
- 문자를 표현할 수 있는 컨트롤을 의미함
- Label(루트 윈도우, 옵션...) 형태로 사용함
 - 옵션에서 컨트롤에 대한 다양한 설정을 해줄 수 있음
- 컨트롤은 pack() 함수를 호출해야 화면에 나타남

[코드 12-3]

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

label1 = Label(root, text="난생처음~~ Python을")
label1.pack()
label2 = Label(root, text="열심히", font=("궁서체", 30), fg="red")
label2.pack()
label3 = Label(root, text="코딩 중입니다.", bg="yellow", width=20,
               height=5, anchor=CENTER)
label3.pack()

root.mainloop()
```





■ 버튼

- 마우스로 클릭하면 눌리는 효과와 함께 어떤 작업이 실행되는 컨트롤
- Button(루트 윈도우, 옵션...) 형태로 사용함
- 라벨은 주로 설명을 위한 용도로 사용되는 반면, 버튼은 클릭을 통한 실행을 위해 사용함
 - 따라서 버튼은 라벨과 달리 버튼이 눌렸을 때 어떤 작업이 발생해야 합니다.
- 버튼을 누르면 실행될 내용을 함수로 만드는 것이 일반적이며, command 옵션을 통해 버튼을 눌렀을 때 지정한 작업을 처리합니다

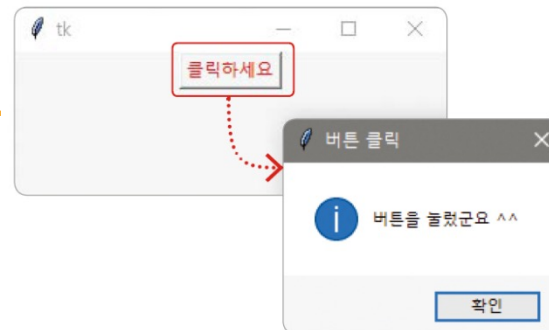


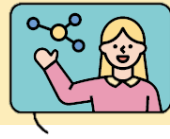
■ 버튼

- 버튼을 누르면 간단한 메시지 창이 나오는 프로그램

[코드 12-4]

```
from tkinter import *  
root = Tk()  
  
## 함수 선언부  
def myFunc() :  
    messagebox.showinfo("버튼 클릭", "버튼을 눌렀군요 ^^")  
  
## 메인 코드부  
root = Tk()  
root.geometry('300x100')  
  
button1 = Button(root, text="클릭하세요", fg="red", command=myFunc)  
button1.pack()  
  
root.mainloop()
```





■ 체크버튼

- 네모박스가 켜지거나 꺼지는 컨트롤
- Checkbutton(루트 윈도우, 옵션...) 형태로 사용함

[코드 12-5]

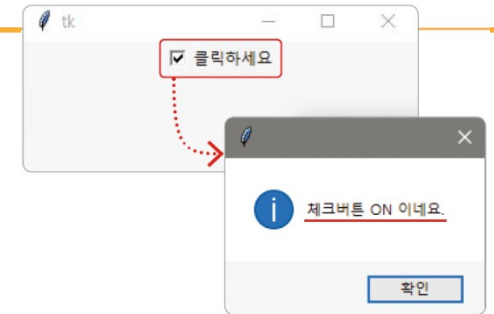
```
from tkinter import *
root = Tk()

## 함수 선언부
def myFunc() :
    if chk.get() == 0 :
        messagebox.showinfo("", "체크버튼 OFF 네요.")
    else :
        messagebox.showinfo("", "체크버튼 ON 이네요.")

## 메인 코드부
root = Tk()
root.geometry('300x100')

chk = IntVar()
cb1 = Checkbutton(root, text="클릭하세요", variable=chk, command=myFunc)
cb1.pack()

root.mainloop()
```





■ 라디오버튼 컨트롤

- 여러 개 중에서 하나를 선택할 때 사용하는 컨트롤
- Radio button(루트 윈도우, 옵션...) 형태로 사용함

[코드 12-6]

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

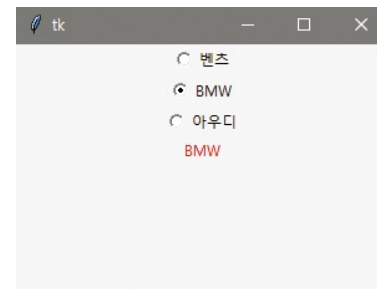
## 함수 선언부
def myFunc() :
    if myVar.get() == 1 :
        label1.configure(text="벤츠")
    elif myVar.get() == 2 :
        label1.configure(text="BMW")
    else :
        label1.configure(text="아우디")

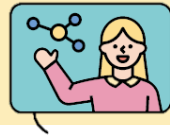
## 메인 코드부
root = Tk()
root.geometry('300x200')
```



■ 라디오버튼 컨트롤

```
myVar = IntVar()  
rb1 = Radiobutton(root, text="벤츠", variable=myVar, value=1, command=myFunc)  
rb1.pack()  
rb2 = Radiobutton(root, text="BMW", variable=myVar, value=2, command=myFunc)  
rb2.pack()  
rb3 = Radiobutton(root, text="아우디", variable=myVar, value=3,  
command=myFunc)  
rb3.pack()  
  
label1 = Label(root, text="선택한 차량 : ", fg="red")  
label1.pack()  
  
root.mainloop()
```





확인문제

다음 빈칸에 해당하는 단어를 <보기>에서 골라 채우시오.

[보기]

㉠ Label

㉡ command

㉢ variable

㉣ configure

- (1) 버튼의 옵션 중, 함수 이름을 지정하는 옵션은 이다.
- (2) 글자를 표현하는 컨트롤은 이다.
- (3) 컨트롤의 옵션값을 변경시키려면 컨트롤명. (옵션=값) 형식을 사용한다.
- (4) 체크버튼의 옵션 중에서 변수를 설정하는 것은 이다.

정답

Click!



■ 수평으로 정렬하기

- pack() 함수의 옵션 중 side=LEFT, RIGHT 방식을 사용함
 - LEFT : 왼쪽부터 정렬
 - RIGHT : 오른쪽부터 정렬

[코드 12-7]

```
from tkinter import *  
root = Tk()  
  
button1 = Button(root, text="버튼1")  
button2 = Button(root, text="버튼2")  
button3 = Button(root, text="버튼3")  
  
button1.pack(side=LEFT)  
button2.pack(side=LEFT)  
button3.pack(side=LEFT)  
  
root.mainloop()
```





■ 수평으로 정렬하기

- [코드 12-7]의 8~10행을 LEFT에서 RIGHT로 변경하고 실행하면 버튼1이 가장 오른쪽으로 감

[코드 12-8]

```
from tkinter import *  
root = Tk()  
  
button1 = Button(root, text="버튼1")  
button2 = Button(root, text="버튼2")  
button3 = Button(root, text="버튼3")  
  
button1.pack(side=RIGHT)  
button2.pack(side=RIGHT)  
button3.pack(side=RIGHT)  
  
root.mainloop()
```





■ 수직으로 정렬하기

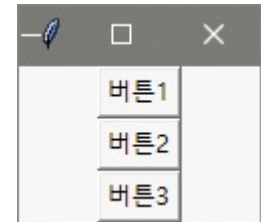
- pack() 함수의 옵션 중 side=TOP, BOTTOM 방식을 사용함
 - TOP : 위에서부터 정렬
 - BOTTOM : 아래에서부터 정렬

[코드 12-9]

...생략([코드 12-7]의 1~6행)...

```
button1.pack(side=TOP)
button2.pack(side=TOP)
button3.pack(side=TOP)

root.mainloop()
```

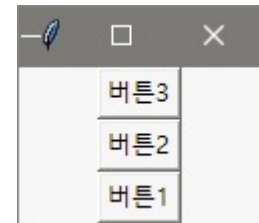


[코드 12-10]

...생략([코드 12-7]의 1~6행)...

```
button1.pack(side=BOTTOM)
button2.pack(side=BOTTOM)
button3.pack(side=BOTTOM)

root.mainloop()
```





■ 컨트롤 사이에 여백 주기

- pack() 함수의 옵션 중 padx=픽셀값 또는 pady=픽셀값 방식을 사용함
- padx=10 : 컨트롤의 왼쪽과 오른쪽에 각각 10px씩 여백이 생김
 - 생각한 것 보다 더 많은 여백이 생길 수 있으므로 주의해야 함

[코드 12-10]

...생략([코드 12-7]의 1~6행)...

```
button1.pack(side=TOP, fill=X, padx=10, pady=10)
button2.pack(side=TOP, fill=X, padx=10, pady=10)
button3.pack(side=TOP, fill=X, padx=10, pady=10)

root.mainloop()
```





확인문제

다음 빈칸에 들어갈 단어의 순서가 올바른 것을 고르시오.

pack() 함수의 side 옵션에서 컨트롤을 왼쪽부터 정렬하려면 , 오른쪽부터 정렬하려면 , 위쪽부터 정렬하려면 , 아래쪽부터 정렬하려면 (을)를 사용한다.

- ① LEFT - TOP - RIGHT - BOTTOM
- ② RIGHT - LEFT - BOTTOM - TOP
- ③ LEFT - RIGHT - TOP - BOTTOM
- ④ RIGHT - TOP - LEFT - BOTTOM

정답

Click!

Section 02

마우스 이벤트 처리하기



■ 마우스를 클릭했을 때 처리되는 형식

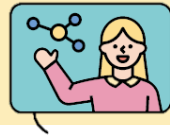
- GUI 환경에서 마우스를 클릭하는 것을 이벤트(Event)라고 부름
- root.mainloop() 함수
 - 이러한 이벤트가 발생하는 것을 기다리는 함수

```
def 마우스클릭처리함수(event) :  
    # 마우스 클릭시 작동할 내용 코딩  
  
root.bind("마우스 클릭 종류" , 이벤트처리함수)
```



매개변수 이름 event는 임의로 지어준 것입니다. 그냥 간단히 e라고 쓰거나, 아무거나 써도 됩니다.
event 매개변수의 용도는 잠시 후에 확인해 보겠습니다.

마우스 클릭 이벤트의 개념



■ 마우스 왼쪽 버튼을 클릭했을 때 처리하는 방법

[코드 12-13]

```
from tkinter import *
from tkinter import messagebox

# 함수 선언부
def clickLeft(event) :
    messagebox.showinfo("마우스", "왼쪽 마우스가 클릭됨")

# 메인 코드부
root = Tk()

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





■ 마우스 이벤트의 종류

- 마우스 이벤트는 마우스를 클릭할 때, 떼었을 때, 더블 클릭할 때, 드래그할 때 등의 다양한 이벤트가 발생할 수 있음

표 12-1 마우스 이벤트의 종류

마우스 작동	관련 마우스 버튼	이벤트 코드
클릭	모든 버튼 공통	〈Button〉
	왼쪽 버튼	〈Button-1〉
	가운데 버튼	〈Button-2〉
	오른쪽 버튼	〈Button-3〉
떼었을 때	모든 버튼 공통	〈ButtonRelease〉
	왼쪽 버튼	〈ButtonRelease-1〉
	가운데 버튼	〈ButtonRelease-2〉
	오른쪽 버튼	〈ButtonRelease-3〉
더블 클릭	모든 버튼 공통	〈Double-Button〉
	왼쪽 버튼	〈Double-Button-1〉
	가운데 버튼	〈Double-Button-2〉
	오른쪽 버튼	〈Double-Button-3〉
드래그	왼쪽 버튼	〈B1-Motion〉
	가운데 버튼	〈B2-Motion〉
	오른쪽 버튼	〈B3-Motion〉



확인문제

※ 다음 빈칸에 들어갈 단어를 채우시오.

1. Tkinter 윈도우 프로그램의 마우스 이벤트를 처리하기 위해서 root. 함수를 사용한다.
2. 모든 마우스 버튼을 클릭했을 때 처리하는 이벤트는 이다.

정답

Click!



■ event 매개변수

- 마우스를 클릭했을 때 처리하는 함수에는 event 매개변수를 받음
- event 매개변수에는 마우스와 관련된 다양한 정보를 포함하고 있음
 - 마우스를 클릭한 위치의 좌표(x, y), 마우스 버튼의 번호(num) 등



■ event 매개변수

[코드 12-14]

```
from tkinter import *

# 함수 정의 부분
def clickMouse(event) :
    if event.num == 1 :
        txt = "왼쪽 버튼 : (" + str(event.x) + "," + str(event.y) + ")"
    elif event.num == 3 :
        txt = "오른쪽 버튼 : (" + str(event.x) + "," + str(event.y) + ")"

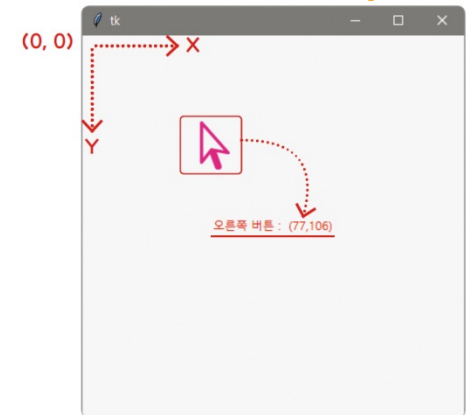
    label1.configure(text=txt)

# 메인 코드 부분
root = Tk()
root.geometry("400x400")

label1 = Label(root, text="여기가 바뀝니다.", fg="red" )
label1.pack(expand=1, anchor=CENTER)

root.bind("<Button>",clickMouse)

root.mainloop()
```





■ 캔버스

- 선, 원, 사각형 등의 그림이 그려지는 컨트롤

- 선

- create_line() 함수 : width로 선의 두께 설정, fill로 선의 색상 설정

```
canvas.create_line(시작x, 시작y, 끝x, 끝y, width=선두께, fill="선색상")
```

- 타원

- create_oval() 함수 : 두 점에 포함되는 크기의 타원을 그림
outline은 타원의 테두리 색상을 지정함

```
canvas.create_oval(시작x, 시작y, 끝x, 끝y, outline="테두리색상")
```

- 사각형

- create_rectangle() 함수 : 두 점 사이의 사각형을 그림
fill은 사각형의 내부색상을 지정함

```
canvas.create_rectangle(시작x, 시작y, 끝x, 끝y,  
                        outline="테두리색상", fill="내부색상")
```



■ 캔버스에 선, 원, 직사각형 그리기

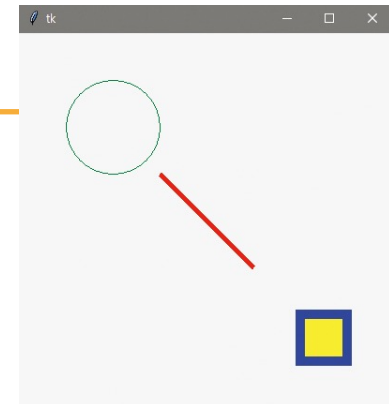
[코드 12-15]

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

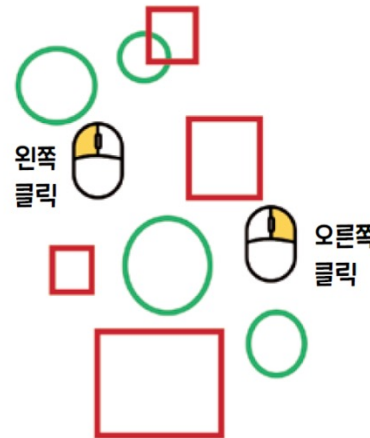
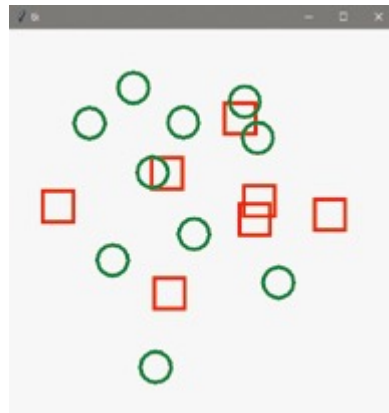
# 함수 정의 부분
canvas = Canvas(root, height=400, width=400)
canvas.pack()

# 메인 코드 부분
canvas.create_line(150, 150, 250, 250, width=5, fill="red")
canvas.create_oval(50, 50, 150, 150, outline="green")
canvas.create_rectangle(300, 300, 350, 350, width=10, outline="blue",
                       fill="yellow")

root.mainloop()
```



마우스 왼쪽 버튼을 클릭하면 클릭한 점을 중심으로 원이 그려지고, 오른쪽 버튼을 클릭하면 사각형이 그려지는 프로그램을 작성해 봅시다.



1. lab12-01.py 파일을 만들고, tkinter를 임포트하기

```
from tkinter import *
```

2. clickLeft()를 정의하기

- 마우스 왼쪽 버튼을 클릭하면 클릭한 점을 중심으로 왼쪽으로 20, 오른쪽으로 20 범위의 원을 그리는 함수

```
# 함수 선언부
def clickLeft(e) :
    canvas.create_oval(e.x-20, e.y-20, e.x+20, e.y+20, width=5,
                       outline="green")
```

3. clickRight()를 정의하기

- 2와 같은 방식으로 마우스 오른쪽 버튼을 클릭하면 사각형을 그리는 함수

```
def clickRight(e) :
    canvas.create_rectangle(e.x-20, e.y-20, e.x+20, e.y+20,
                            width=5, outline="red")
```

4. 폭과 높이가 500인 캔버스를 생성하기

```
# 메인 코드부
root = Tk()

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

5. 왼쪽 및 오른쪽 마우스 버튼을 클릭하면 실행될 함수를 지정하기

```
canvas.bind("<Button-1>", clickLeft)
canvas.bind("<Button-3>", clickRight)

root.mainloop()
```

6. <Ctrl> + <S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

Section 03

메뉴와 대화상자 만들기



■ 메뉴

■ 윈도우 창 상단의 메뉴의 대표적인 예

- 파이썬 IDLE 상단 메뉴에서 [파일]-[열기]를 선택하면 파일을 선택하는 대화 상자가 나옴
- 이때 [파일]이 상위 메뉴, [열기]가 하위 메뉴에 해당됨

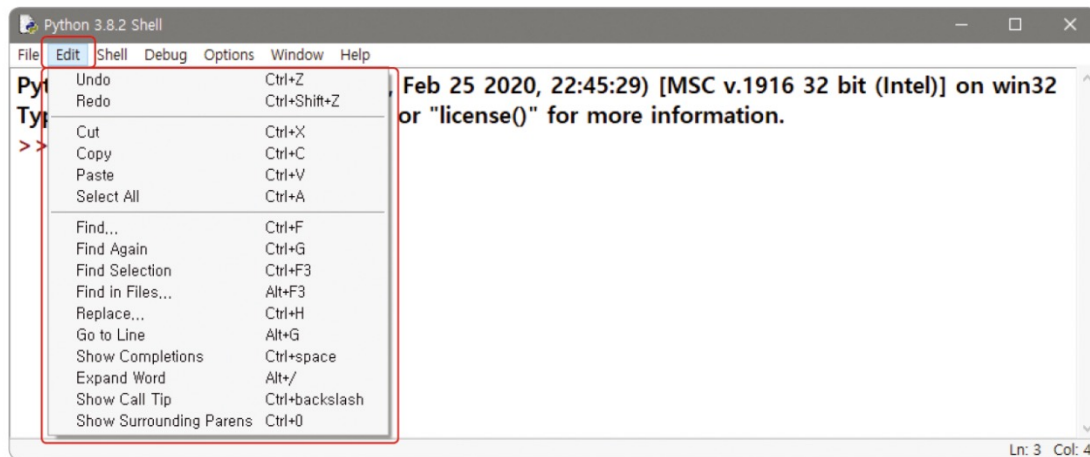


그림 12-2 파이썬 IDLE 상단의 메뉴



■ 메뉴의 구성 개념과 형식

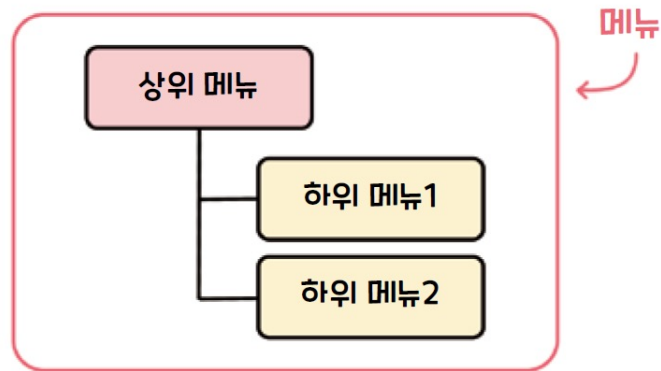


그림 12-3 메뉴의 구성 개념도

```
메뉴 = Menu(루트 윈도우)  
루트윈도우.config(menu=메뉴)
```

```
상위메뉴 = Menu(메뉴)  
메뉴.add_cascade(label="상위 메뉴 텍스트", menu=상위 메뉴)  
상위메뉴.add_command(label="하위 메뉴1", command=함수1)  
상위메뉴.add_command(label="하위 메뉴2", command=함수2)
```



■ 메뉴의 겉모양 생성

[코드 12-16]

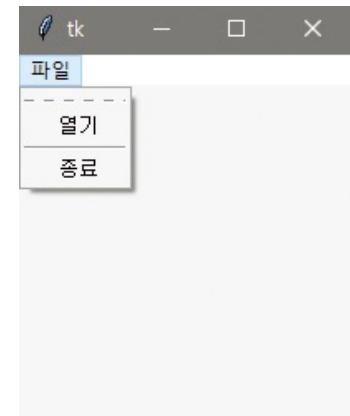
```
from tkinter import *
from tkinter import messagebox

root = Tk()

mainMenu = Menu(root)
root.config(menu=mainMenu)

fileMenu = Menu(mainMenu)
mainMenu.add_cascade(label="파일", menu=fileMenu)
fileMenu.add_command(label="열기")
fileMenu.add_separator()
fileMenu.add_command(label="종료")

root.mainloop()
```





■ 메뉴를 선택하면 작동하도록 코드 수정하기

[코드 12-17]

```
from tkinter import *
from tkinter import messagebox

# 함수 정의 부분
def func_open() :
    messagebox.showinfo("메뉴선택", "열기 메뉴를 선택했습니다")

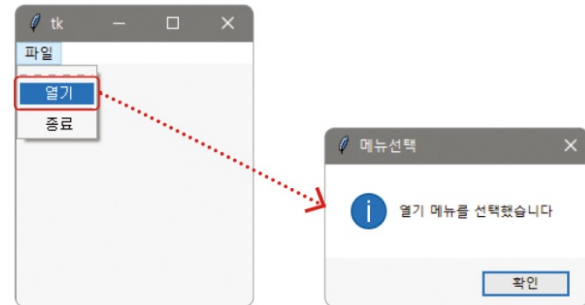
def func_exit() :
    root.quit()
    root.destroy()

# 메인 코드 부분
root = Tk()

mainMenu = Menu(root)
root.config(menu=mainMenu)

fileMenu = Menu(mainMenu)
mainMenu.add_cascade(label="파일", menu=fileMenu)
fileMenu.add_command(label="열기", command=func_open)
fileMenu.add_separator()
fileMenu.add_command(label="종료", command=func_exit)

root.mainloop()
```





■ 대화상자

- 사용자에게 정보를 보여주거나 사용자로부터 입력을 요청하는 등의 상호작용을 위한 창을 의미함

■ 숫자나 문자 입력 대화상자

- `tkinter.simpledialog` 모듈을 임포트해야 함
- `askinteger()` 함수 : 정수 입력받는 함수
- `askfloat()` 함수 : 실수 입력받는 함수
- `askstring()` 함수 : 문자열을 입력받는 함수



■ 숫자나 문자 입력 대화상자

[코드 12-18]

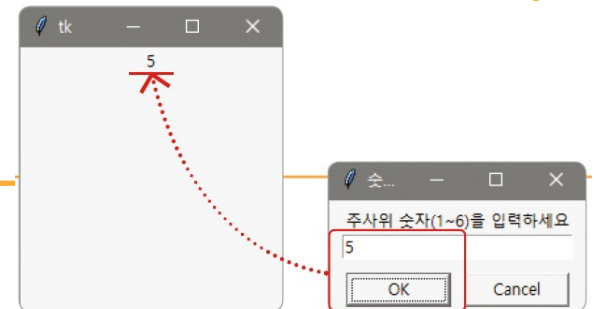
```
from tkinter import *
from tkinter.simpledialog import *

root = Tk()
root.geometry('200x200')

label1 = Label(root, text="입력된 값")
label1.pack()

value = askinteger("숫자입력", "주사위 숫자(1~6)을 입력하세요",
                  minvalue=1, maxvalue=6)
label1.configure(text=str(value))

root.mainloop()
```





■ 파일 열기 및 저장 대화상자

- tkinter.filedialog 모듈을 임포트해야 함
- askopenfilename() / asksaveasfilename() 함수 : 파일 열기 대화상자 함수

[코드 12-19]

```
from tkinter import *
from tkinter.filedialog import *

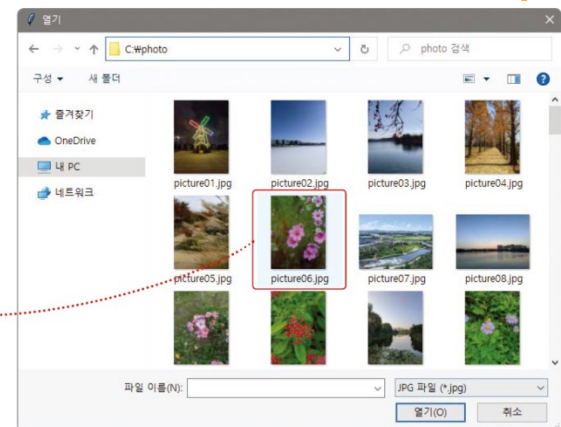
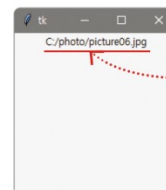
root = Tk()
root.geometry('200x200')

label1 = Label(root, text="선택된 파일이름")
label1.pack()

filename = askopenfilename(parent=root, filetypes=(("JPG 파일",
                                                    "*.jpg"), ("모든 파일", "*. *.*")))

label1.configure(text=filename)

root.mainloop()
```





■ 파일 열기 및 저장 대화상자

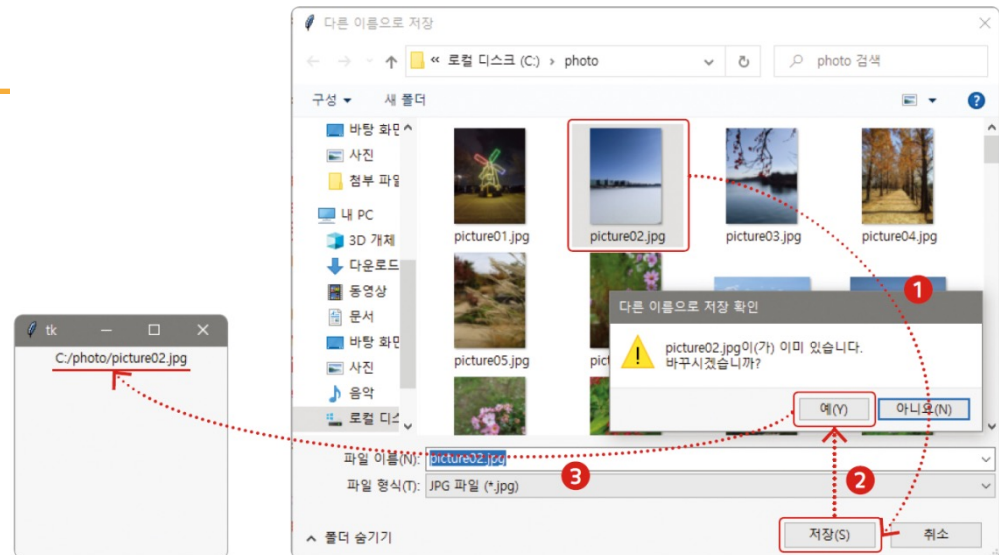
- `asksaveasfilename()` 함수 : 파일을 저장하기 위한 대화상자 함수
- [코드 12-19]의 10~12행을 수정하기

[코드 12-20]

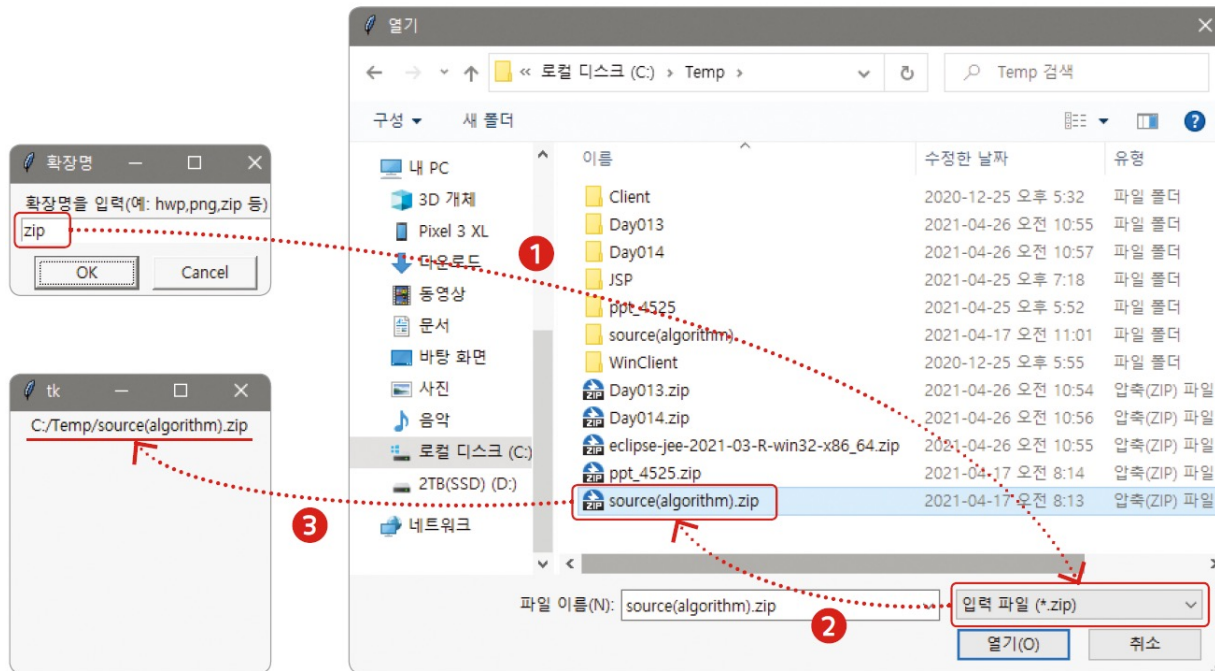
...생략([코드 12-19]의 1~8행)...

```
savefname = asksaveasfilename(parent=root, filetypes=(("JPG 파일",  
".*.jpg"), ("모든 파일", "*. *.*")))  
label1.configure(text=savefname)
```

```
root.mainloop()
```



확장명을 입력받은 후에, 입력한 확장명의 파일만 선택하는 코드를 작성해 봅시다



1. lab12-02.py 파일을 만들고, 입력을 받기 위한 대화상자 모듈과 파일 열기 대화상자 모듈을 임포트하기

```
from tkinter import *  
from tkinter.filedialog import *  
from tkinter.simpledialog import *
```

2. tkinter 화면을 생성하고, 선택된 파일이름을 보여줄 라벨도 생성하기

```
root = Tk()  
root.geometry('200x200')  
  
label1 = Label(root, text="선택된 파일이름")  
label1.pack()
```

4. 확장명을 문자열로 입력받기

```
extName = askstring("확장명", "확장명을 입력(예: hwp,png,zip 등)")
```

5. filetypes 옵션을 "*.입력글자" 형식으로 만들어서 파일 선택 대화상자를 열고 선택한 파일이름을 라벨에 출력하기

```
filename = askopenfilename(parent=root, filetypes=(("입력 파일",  
                                                    ".*."+extName), ("모든 파일", " *.*") ))  
label1.configure(text=str(filename))  
  
root.mainloop()
```

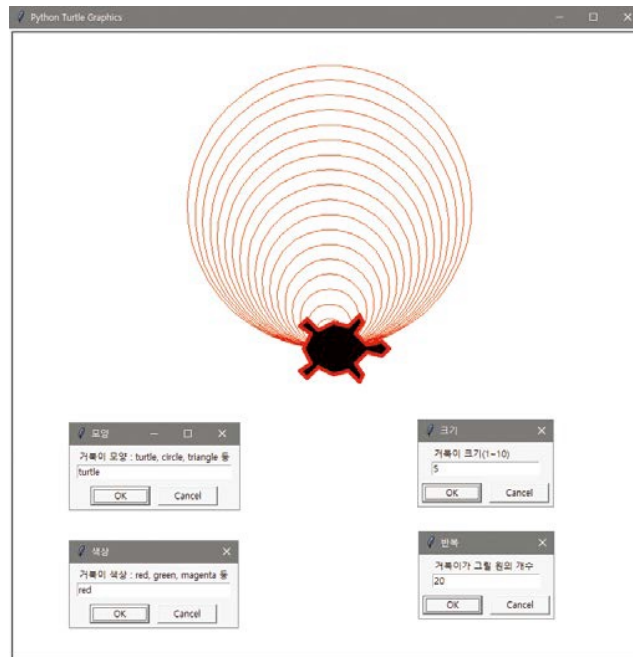
6. <Ctrl> + <S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

[실전 예제] GUI로 편하게 거북이 그리기

실전 예제 GUI로 편하게 거북이 그리기

[문제]

- 거북이의 모양, 색상, 크기 및 거북이가 그릴 원의 개수를 GUI로 입력받기
- 입력받은 값을 기반으로 거북이가 원을 그리기



실전 예제 GUI로 편하게 거북이 그리기

[해결]

```
import turtle
import random
from tkinter.filedialog import *
from tkinter.simpledialog import *

root = Tk( )
shape = askstring("모양", "거북이 모양 : turtle, circle, triangle 등")
color = askstring("색상", "거북이 색상 : red, green, magenta 등")
size = askinteger("크기", "거북이 크기(1~10)", minvalue=1, maxvalue=10)
repeat = askinteger("반복", "거북이가 그릴 원의 개수")
root.destroy( )

turtle.shape(shape)
turtle.pencolor(color)
turtle.turtlesize(size, size, size)

turtle.setup(850, 850)
turtle.screensize(800, 800)

for i in range(repeat) :
    turtle.circle(i*10)

turtle.done( )
```

Preview

감사합니다 :)