

tkinter 윈도우 프로그래밍

# Tkinter



# 수업 내용

- tkinter 란?
- 배치 관리자
- 다양한 위젯들
  - 텍스트 입력 : Enter
  - 하나만 선택 : Radiobutton
  - 여러 개 선택 : Checkbutton
  - 위젯 모음 관리 : Frame
- 그래픽
- 마우스 입력 처리
- 메뉴 만들기

# tkinter ?

- 파이썬에서 GUI를 개발할 때 사용하는 모듈
- Tk interface의 약자
- Tcl/Tk 위에 객체지향을 입힌 것
- `window = Tk()`
  - 원도우 생성!
- 위젯(widget : window gadget)
  - GUI 기반 운용체제에서 사용하는 시각적인 구성 요소
  - 예 : 버튼, 레이블, 텍스트 필드, 메뉴

# Label

- 텍스트, 이미지를 화면에 출력하는 위젯
- 텍스트 출력
  - `lable = Lable(window, test='Hello World!!')`
- 이미지 출력
  - `photo = PhotoImage(file='img/cat.gif')`
  - `label = Label(window, image=photo)`

# 첫번째 프로그램 : window\_hello.py

```
from tkinter import *    # tkinter 모듈 포함
```

```
window = Tk()           # 루트 윈도우 생성
```

```
# 루트 윈도우 자식으로 레이블 위젯 생성
```

```
label = Label(window, text='Hello World!!')
```

```
# 텍스트 크기에 맞추어 레이블 위젯 크기 조정
```

```
label.pack()
```

→ 배치 관리자

```
window.mainloop()      # 이벤트 루프. 키보드/마우스 입력 처리 시작
```



# 다양한 텍스트 : window\_label.py

```
from tkinter import *
```

```
window = Tk()
```

```
window.title('윈도우 창 연습')
```

```
window.geometry("400x400+100+0") # 너비/높이 + X좌표 + Y좌표
```

```
window.resizable(True, False) # 창 크기 조절 방지 (X축, Y축)
```

```
label1 = Label(window, text='파이썬을')
```

```
label2 = Label(window, text='열심히', font=('함초롬바탕', 30), fg='#6799FF')
```

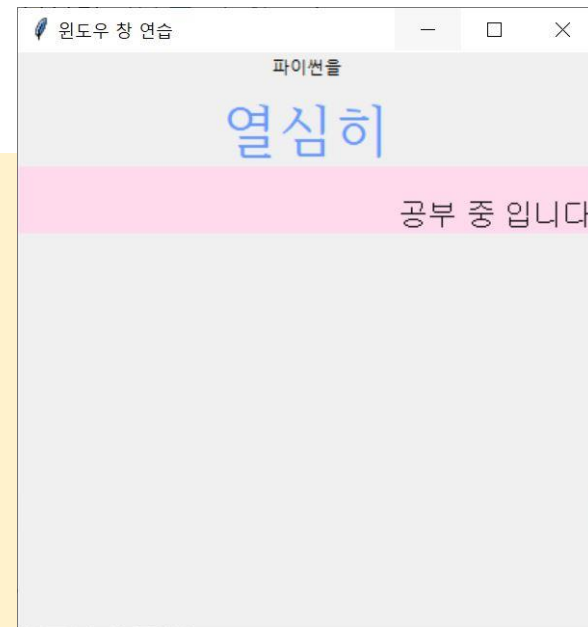
```
label3 = Label(window, text='공부 중 입니다', font=('맑은고딕', 15), bg='#FFD9EC',  
                width=100, height=2, anchor=SE)
```

```
label1.pack()
```

```
label2.pack()
```

```
label3.pack()
```

```
window.mainloop()
```



- Label 문자열 설정

이름	의미	기본값	속성
text	라벨에 표시할 문자열	-	-
textvariable	라벨에 표시할 문자열을 가져올 변수	-	-
anchor	라벨안의 문자열 또는 이미지의 위치	center	n, ne, e, se, s, sw, w, nw, center
justify	라벨의 문자열이 여러 줄 일 경우 정렬 방법	center	center, left, right
wraplength	자동 줄내림 설정 너비	0	상수

# 이미지 보여주기 : window\_photo.py

```
from tkinter import *

window = Tk()

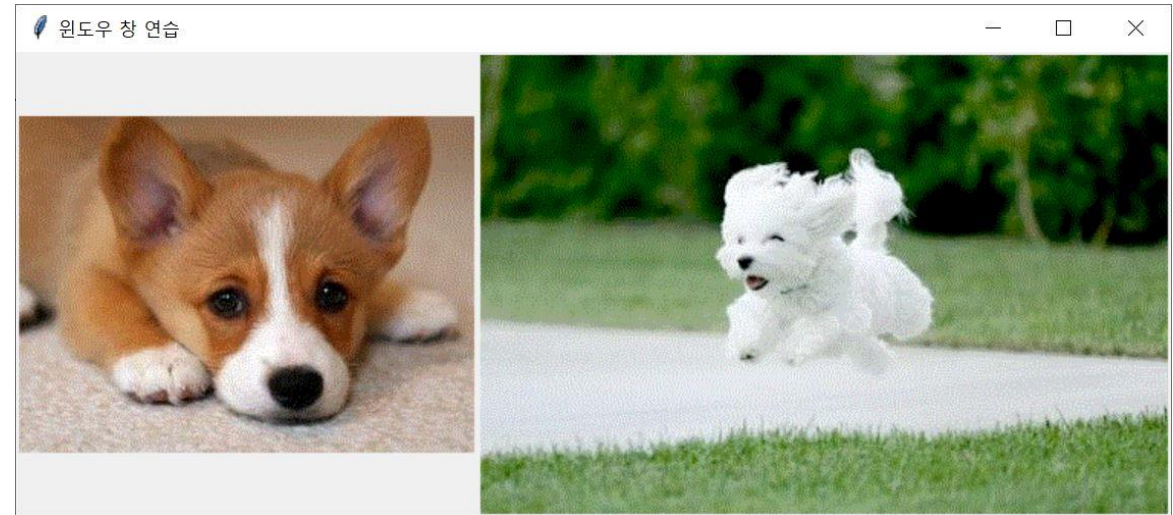
window.title("윈도우 창 연습")

photo =
PhotoImage(file="img/dog1.gif")
label1 = Label(window, image=photo)

photo2 =
PhotoImage(file="img/dog2.gif")
label2 = Label(window, image=photo2)

label1.pack(side=LEFT)
# label1.pack(side=RIGHT)
label2.pack()

window.mainloop()
```





# 버튼과 이벤트 처리

```
from tkinter import *
```

```
window = Tk()
```

```
def myFun() :  
    print('clicked button')
```

```
b1 = Button(window, text='파이썬 버튼', command=myFun)  
b1.pack()
```

```
window.mainloop()
```

- 콜백 함수(callback function) 혹은 핸들러(handler)
  - 이벤트가 발생 했을 때 호출되는 함수
  - Button(window, text='버튼', command= 함수 이름)

버튼을 클릭하면  
이 함수 실행

```
from tkinter import *
from tkinter import messagebox
```

```
## 함수 정의
```

```
def myFunc() :
    button2.configure(text='와우~')
    messagebox.showinfo("강아지 버튼", "강아지가 귀엽조 ^^")
```

```
## 메인 코드
```

```
window = Tk()
```

```
window.title("윈도우 창 연습")
```

```
photo = PhotoImage(file="img/dog1.gif")
button1 = Button(window, image=photo, command=myFunc)
```

```
photo2 = PhotoImage(file="img/dog2.gif")
label2 = Label(window, image=photo2)
```

```
button2 = Button(window, text="종료", command=window.destroy, width=10, height=2)
```

```
button1.pack(side=LEFT)
label2.pack()
button2.pack()
```

```
window.mainloop()
```

button1

label2

button2

종료

와우~

### 버튼의 option

- width : 가로 길이
- height : 세로 길이
- bd : 가장자리
- img : 그림 지정
- compound : 그림과 텍스트 함께 사용
- padx, pady : 여백 지정

# 배치 관리자

- 모든 위젯은 배치를 해야 화면에 보임!!!
- 종류
  - grid()
    - 격자 배치 관리자(grid geometry manager)
    - 위젯을 행과 열로 구성된 2차원적인 격자에 배치
  - pack()
    - 압축 배치 관리자(pack geometry manager)
    - 위젯들이 부모 위젯 안에 압축. 자식 위젯들은 사각형 블록으로 간주하여 배치
  - place()
    - 절대 배치 관리자(Place geometry manager)
    - 주어진 위치에 위젯 배치

# 배치 관리자 : pack

- 위젯을 최대한 붙여서 배치
- padx, pady, ipadx, ipady : x, y 여백 지정
- grid()와 같이 사용 불가, place()와 같이 사용 가능

```
# window_pack.py
from tkinter import *
```

```
window = Tk()
```

```
Label(window, text="박스 #1", bg='red', fg='white').pack()
Label(window, text="박스 #2", bg='green', fg='white').pack()
Label(window, text="박스 #3", bg='blue', fg='white').pack(fill=X)
```

```
window.mainloop()
```



## pack(fill=X)

X = 수평으로만 늘리기  
Y = 수직으로만 늘리기  
BOTH = 수평, 수직 모두 늘리기  
NONE = 늘리지 않기

# pack 인자

이름	의미	기본값	속성
side	특정 위치로 공간 할당	top	top, bottom, left, right
anchor	할당된 공간 내에서 위치 지정	center	center, n, e, s, w, ne, nw, se, sw
fill	할당된 공간에 대한 크기 맞춤	none	none, x, y, both
expand	미사용 공간 확보	False	Boolean
ipadx	위젯에 대한 x 방향 내부 패딩	0	상수
ipady	위젯에 대한 y 방향 내부 패딩	0	상수
padx	위젯에 대한 x 방향 외부 패딩	0	상수
pady	위젯에 대한 y 방향 외부 패딩	0	상수

# 배치 관리자 : grid

- grid(격자) 배치 관리자
  - 위젯을 테이블 형태로 배치
  - 부모 윈도우를 행, 열로 분할하여 각 위젯은 지정한 셀에 배치됨

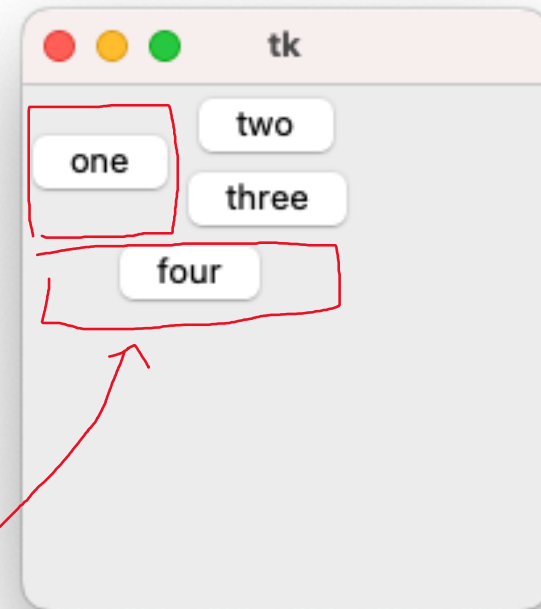
```
from tkinter import *
```

```
win = Tk()  
win.geometry('200x200')
```

```
b1 = Button(win, text='one')  
b2 = Button(win, text='two')  
b3 = Button(win, text='three')  
b4 = Button(win, text='four')
```

```
b1.grid(row=0, column=0, rowspan=2)  
b2.grid(row=0, column=1)  
b3.grid(row=1, column=1)  
b4.grid(row=2, column=0, colspan=2)
```

```
win.mainloop()
```



# grid 인자

이름	의미	기본값	속성
row	행 위치	0	상수
column	열 위치	0	상수
rowspan	행 위치 조정	1	상수
columnspan	열 위치 조정	1	상수
sticky	할당된 공간 내에서의 위치 조정	-	n, e, s, w, nw, ne, sw, se
ipadx	위젯에 대한 x 방향 내부 패딩	0	상수
ipady	위젯에 대한 y 방향 내부 패딩	0	상수
padx	위젯에 대한 x 방향 외부 패딩	0	상수
pady	위젯에 대한 y 방향 외부 패딩	0	상수



```
from tkinter import * # window_gride2.py
```

```
window = Tk()
window.title('grid test')
```

```
data = IntVar()
data.set(20)
```

```
h_label = Label(window, text='Height: ', font=('맑은고딕', 15), fg='#1F50B5')
h_entry = Entry(window, font=('맑은고딕', 15), width=10)
```

```
w_label = Label(window, text='Width: ', font=('맑은고딕', 15), fg='#1F50B5')
w_entry = Entry(window, font=('맑은고딕', 15), width=10, bd=3, textvariable=data)
```

```
raw_image = PhotoImage(file='img/dog2.gif')
image = Label(window, image=raw_image)
```

```
button1 = Button(window, text='Zoom in')
button2 = Button(window, text='Zoom out')
```

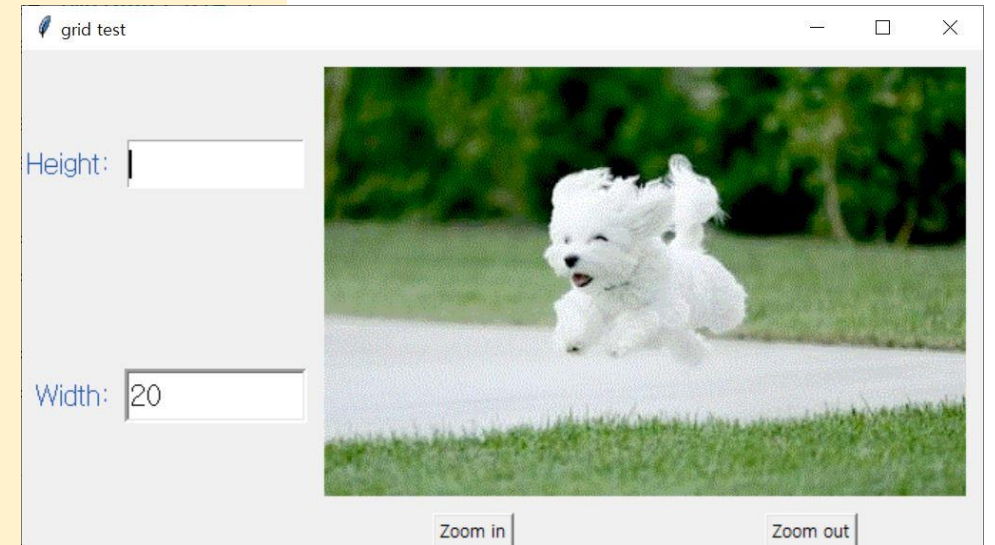
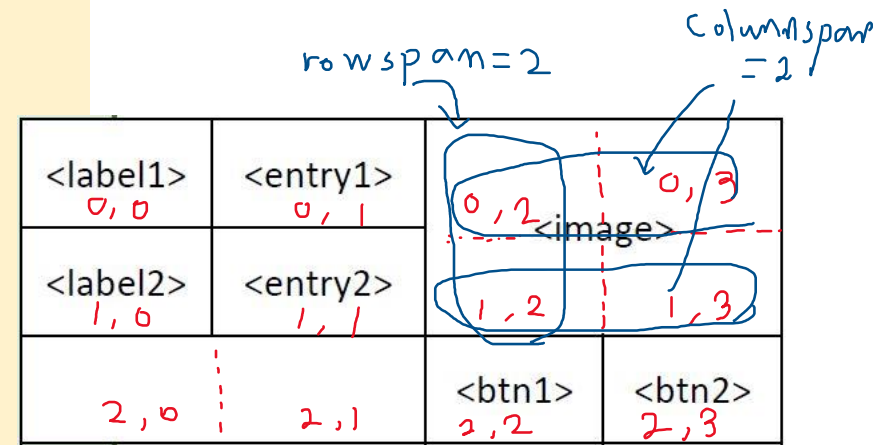
```
h_label.grid(row=0, column=0, sticky=E)
w_label.grid(row=1, column=0, sticky=E)
```

```
h_entry.grid(row=0, column=1, ipadx=5, ipady=5)
w_entry.grid(row=1, column=1, ipadx=5, ipady=5)
```

```
image.grid(row=0, column=2, rowspan=2, colspan=2, padx=10, pady=10)
```

```
button1.grid(row=2, column=2)
button2.grid(row=2, column=3)
```

```
window.mainloop()
```





# 배치 관리자 : place

- 절대 위치를 사용하여 위젯 배치
- x, y 값 필요



```
# window_place.py
from tkinter import *

window = Tk()

Label(window, text="박스 #1", bg='red', fg='white').place(x=0, y=0)
Label(window, text="박스 #2", bg='green', fg='white').place(x=20, y=20)
Label(window, text="박스 #3", bg='blue', fg='white').place(x=40, y=40)

window.mainloop()
```

# place 인자

이름	의미	기본값	속성
x	x좌표 배치	0	상수
y	y좌표 배치	0	상수
relx	x좌표 배치 비율	0	0 ~ 1
rely	y좌표 배치 비율	0	0 ~ 1
width	위젯의 너비	0	상수
height	위젯의 높이	0	상수
relwidth	위젯의 너비 비율	0	0 ~ 1
relheight	위젯의 높이 비율	0	0 ~ 1
anchor	위젯의 기준 위치	nw	n, e, w, s, ne, nw, se, sw

```
from tkinter import * # window_place2.py

window = Tk()
window.title('pack test')
window.geometry('450x200')
data = IntVar()
data.set(20)

h_label = Label(window, text='Height: ', font=('맑은고딕', 15), fg='#1F50B5')
h_entry = Entry(window, font=('맑은고딕', 15), width=10)

w_label = Label(window, text='Width: ', font=('맑은고딕', 15), fg='#1F50B5')
w_entry = Entry(window, font=('맑은고딕', 15), width=10, bd=3, textvariable=data)

raw_image = PhotoImage(file='img/cat.gif')

#raw_image = raw_image.zoom(2)
raw_image = raw_image.subsample(3)

image = Label(window, image=raw_image)

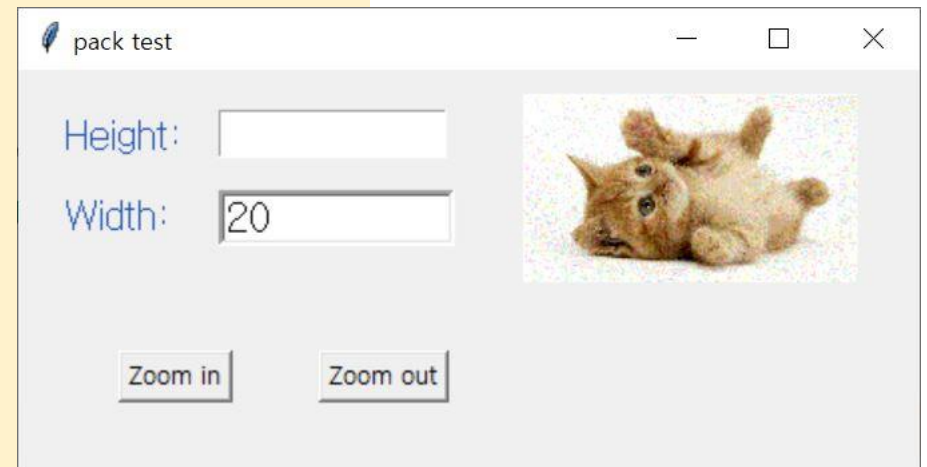
button1 = Button(window, text='Zoom in')
button2 = Button(window, text='Zoom out')

h_label.place(x=20, y=20)
w_label.place(x=20, y=60)

h_entry.place(x=100, y=20)
w_entry.place(x=100, y=60)

image.place(x=250, y=10)
button1.place(x=50, y=140)
button2.place(x=150, y=140)

window.mainloop()
```



# 배치 관리자 : 예제

```
from tkinter import * # window_photoButtonGrid.py
from tkinter import messagebox

def myFunc() :
    messagebox.showinfo("강아지 버튼", "강아지가 귀엽죠 ^^")
    button2.configure(text = '와우~')

window = Tk()

window.title("윈도우 창 연습")

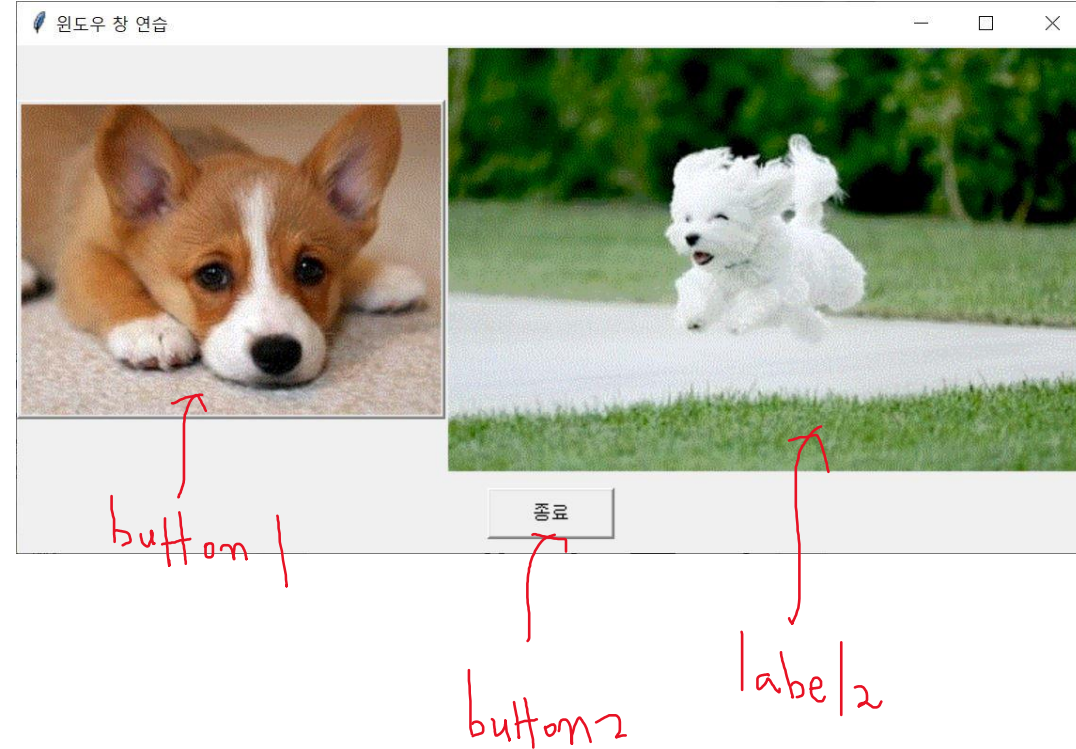
photo = PhotoImage(file="img/dog1.gif")
button1 = Button(window, image=photo, command=myFunc)

photo2 = PhotoImage(file="img/dog2.gif")
label2 = Label(window, image=photo2)

button2 = Button(window, text="종료", font=('맑은고딕', 10),
                  command=quit, width=10, height=2)

button1.grid(row=0, column=0)
label2.grid(row=0, column=1)
button2.grid(row=1, column=0, columnspan=2,
             padx=3, pady=10)

window.mainloop()
```



# tkinter의 위젯들

위젯 클래스	
Button	간단한 버튼으로 명령 수행
Canvas	화면에 그리기
Checkbutton	켜기/끄기
Entry	한 줄의 텍스트 입력 받는 필드
Frame	컨테이너 클래스. 경계선과 배경을 가지고 다른 위젯들을 그룹핑하는데 사용
Label	텍스트나 이미지 표시
Listbox	선택 사항 표시
Menu	메뉴 표시. 풀다운/팝업 메뉴
Menubutton	메뉴 버튼. 풀다운 메뉴 가능
Message	텍스트 표시. 레이블 위젯과 비슷. 자동적으로 주어진 크기로 텍스트 축소
Radiobutton	여러 값 중 하나 선택
Scale	슬라이더를 끌어서 수치값 입력에 사용
Scrollbar	캔버스, 엔트리, 리스트 박스, 텍스트 위젯을 위한 스크롤바
Text	형식을 가지는 텍스트 표시. 스타일, 속성 지정 가능
Toplevel	최상위 윈도우로 표시되는 독립적인 컨테이너 위젯
LabelFrame	경계선과 제목을 가지는 프레임 위젯의 변형
PanedWindow	자식 위젯들을 크기 조절이 가능한 패널로 관리하는 컨테이너 위젯
Spinbox	특정한 범위에서 값을 선택하는 엔트리 위젯의 변형

# 텍스트 입력 : Entry

- 사용자 입력 받을 때 사용

```
# window_entry.py
from tkinter import *

window = Tk()
window.geometry('300x100')
Label(window, text="이름").place(x=20, y=20)
Label(window, text="나이").place(x=20, y=50)

e1 = Entry(window)
e2 = Entry(window)

e1.place(x=60, y=20)
e2.place(x=60, y=50)

window.mainloop()
```



## Entry option

font : 입력 글자 폰트 지정  
fg : 글씨 색상  
bd : 경계 두께  
show : 입력 글자 대신 표시할 문자 지정  
textvariable : entry 필드와 연결할 변수 지정

```

from tkinter import * # window_entry2.py

def show() :
    print ("이름 : %s \n나이 : %s" % (e1.get(), e2.get()))
    e1.insert(END, 3.5)

window = Tk()
window.geometry("300x150")

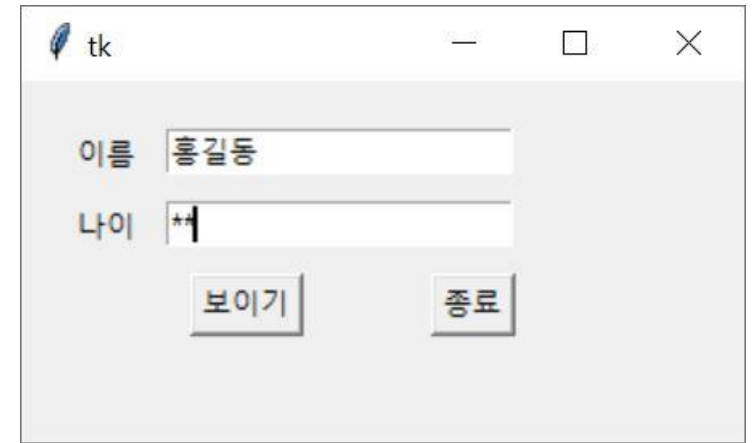
Label(window, text="이름").place(x=20, y=20)
Label(window, text="나이").place(x=20, y=50)

e1 = Entry(window)
e2 = Entry(window, show='*')

e1.place(x=60, y=20)
e2.place(x=60, y=50)

Button(window, text="보이기", command=show).place(x=70, y=80)
Button(window, text="종료", command=quit).place(x=170, y=80)
window.mainloop()

```





# 키 연결하기

```
# window_calculator.py
from tkinter import *
from math import *

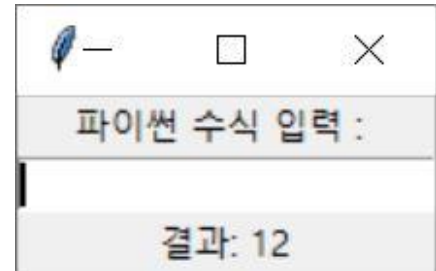
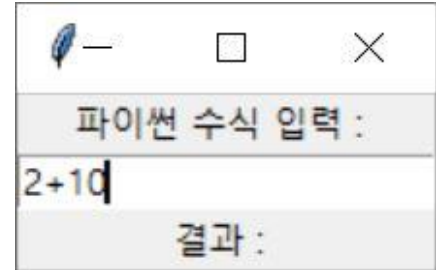
def calculate(event) :
    label.configure(text="결과: " + str(eval(entry.get())))
    entry.delete(0, END)

window = Tk()

Label(window, text="파이썬 수식 입력 : ").pack()
entry = Entry(window)
entry.bind("<Return>", calculate)
entry.pack()

label = Label(window, text="결과 : ")
label.pack()

window.mainloop()
```





# Entry 메소드

이름	의미
insert(index, "문자열")	<code>index</code> 위치에 문자열 추가
delete(start_index, end_index)	<code>start_index</code> 부터 <code>end_index</code> 까지의 문자열 삭제
get()	기입창의 텍스트를 문자열로 반환
index(index)	<code>index</code> 에 대응하는 위치 획득
icursor(index)	<code>index</code> 앞에 키보드 커서 설정
select_adjust(index)	<code>index</code> 위치까지의 문자열을 블록처리
select_range(start_index, end_index)	<code>start_index</code> 부터 <code>end_index</code> 까지 블록처리
select_to(index)	키보드 커서부터 <code>index</code> 까지 블록처리
select_from(index)	키보드 커서의 색인 위치를 <code>index</code> 위치에 문자로 설정하고 선택
select_present()	블록처리 되어있는 경우 <code>True</code> , 아닐 경우 <code>False</code>
select_clear()	블록처리 해제
xview()	가로스크롤 연결
xview_scroll(num, str)	가로스크롤의 속성 설정

# Frame

- 다른 위젯을 담을 수 있는 사각형 영역
- 용도
  - 레이아웃이 복잡할 때 위젯들을 그룹으로 나눔
  - 위젯들 정렬, 위젯 사이의 여백을 부여하기 편함

```
# window_frame.py
from tkinter import *

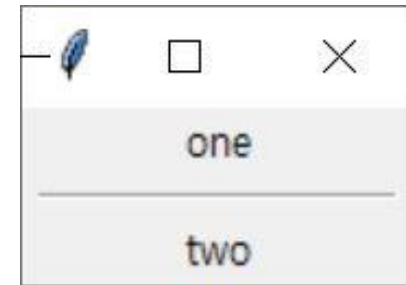
window = Tk()

Label(window, text="one").pack()

separator = Frame(window, height=2, bd=1, relief=SUNKEN)
separator.pack(fill=X, padx=5, pady=5)

Label(window, text="two").pack()

window.mainloop()
```



- bd : 경계선 두께
- relief : frame의 경계선 모양 (FLAT, RAISED, GROOVE, RIDGE)

```
from tkinter import * # window_entry3.py

def show() :
    print ("이름 : %s \n나이 : %s" % (e1.get(), e2.get()))
    d_frame.destroy()
    b_frame.destroy()
```

```
window = Tk()
window.geometry("400x200")
```

```
u_frame = Frame(window, height=30)
u_frame.grid(row=0)
```

```
d_frame = Frame(window)
d_frame.grid(row=1)
```

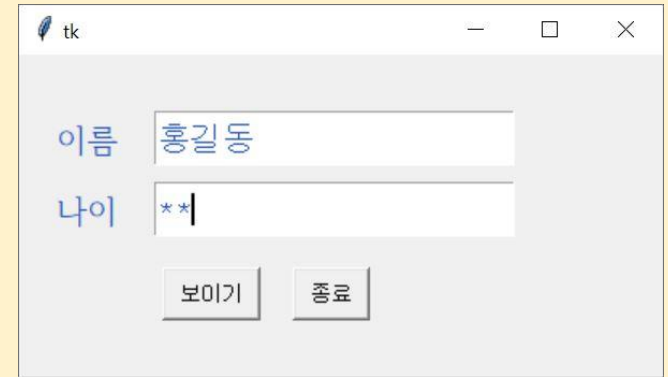
```
b_frame = Frame(window, height=10)
b_frame.grid(row=2)
```

```
Label(d_frame, text="이름", font=('함초롬바탕', 15), fg='#1F50B5').grid(row=0, column=0, padx=20)
Label(d_frame, text="나이", font=('함초롬바탕', 15), fg='#1F50B5').grid(row=1, column=0)
```

```
e1 = Entry(d_frame, font=('맑은고딕', 15), fg='#1F50B5')
e2 = Entry(d_frame, font=('맑은고딕', 15), fg='#1F50B5', show='*')
```

```
e1.grid(row=0, column=1, ipady=5, pady=5)
e2.grid(row=1, column=1, ipady=5, pady=5)
```

```
Button(b_frame, text="보이기", font=('맑은고딕', 10), command=show).grid(row=0, column=0,
                                                                 padx=10, pady=14, ipadx=5, ipady=5)
Button(b_frame, text="종료", font=('맑은고딕', 10), command=quit).grid(row=0, column=1,
                                                                 padx=10, pady=14, ipadx=5, ipady=5)
window.mainloop()
```



```
이름 : 홍길동
나이 : 32
```

# frame 형태 설정

이름	의미	기본값	속성
width	프레임의 너비	0	상수
height	프레임의 높이	0	상수
relief	프레임의 테두리 모양	flat	flat, groove, raised, ridge, solid, sunken
background=bg	프레임의 배경 색상	SystemButtonFace	color
padx	프레임의 테두리와 내용의 가로 여백	1	상수
pady	프레임의 테두리와 내용의 세로 여백	1	상수

# frame 형식 설정

이름	의미	기본값	속성
cursor	프레임의 마우스 커서 모양	-	커서 속성
class_	클래스 설정	-	-
visual	시각적 정보 설정	-	-
colormap	256 색상을 지정하는 색상 맵 설정	-	new

# frame 하이라이트 설정

이름	의미	기본값	속성
highlightcolor	프레임이 선택되었을 때 색상	SystemWindowFrame	color
highlightbackground	프레임이 선택되지 않았을 때 색상	SystemButtonFace	color
highlightthickness	프레임이 선택되었을 때 두께 (두께 설정)	0	상수

# frame 동작 설정

이름	의미	기본값	속성
takefocus	Tab 키를 이용하여 위젯 이동 허용 여부	False	Boolean
container	응용 프로그램이 포함될 컨테이너로 사용	False	Boolean

# 실습 01 : 로그인 화면 만들기

- 아이디와 패스워드를 입력 받아 로그인 하는 화면을 만드시오.
  - 화면 설계
  - 기능 설계
- 위젯 배치
- 기능 구현



The image shows a simple login window with a title bar that says '로그인'. Inside the window, there are two labels, 'ID:' and 'PASSWORD:', each followed by a rectangular input field. To the right of these input fields is a button with the text 'login' on it.

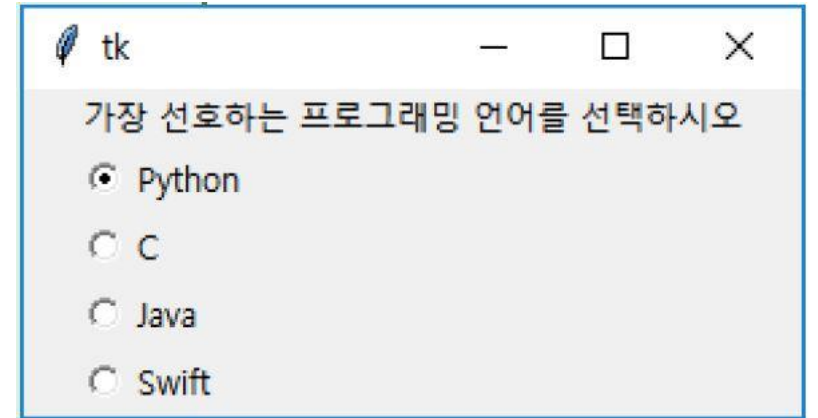
# 라디오 버튼

- 그룹 안에서 한 개의 버튼만 선택 가능

```
from tkinter import *

window = Tk()
choice = IntVar()
Label(window,
      text = "가장 선호하는 프로그래밍 언어를 선택하시오",
      justify = LEFT,
      padx= 20).pack()

Radiobutton(window, text="Python", padx=20,
            variable=choice, value=1).pack(anchor=W)
Radiobutton(window, text="C", padx=20,
            variable=choice, value=2).pack(anchor=W)
Radiobutton(window, text="Java", padx=20,
            variable=choice, value=3).pack(anchor=W)
Radiobutton(window, text="Swift", padx=20,
            variable=choice, value=4).pack(anchor=W)
window.mainloop()
```





# 라디오 버튼 - 수정

- 그룹 안에서 한 개의 버튼만 선택 가능

```
from tkinter import *
window = Tk()
select = IntVar()
select.set(1)

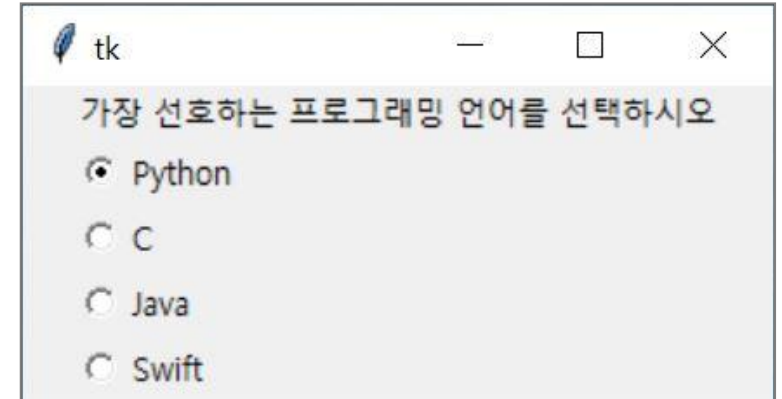
list = [("Python", 1), ("C", 2), ("Java", 3),
        ("Swift", 4)]

def PrintChoice() :
    print(select.get())

Label(window,
      text = "가장 선호하는 프로그래밍 언어를  
선택하십시오",
      justify = LEFT,
      padx = 20).pack()

for txt, val in list:
    Radiobutton(window,
                text = txt,
                padx = 20,
                variable = select,
                command = PrintChoice,
                value = val).pack(anchor=W)

window.mainloop()
```



1  
2  
3  
4

# tkinter 위젯과 연결된 변수

- 위젯 값이 변하면 변수 값도 변함, 변수 값이 변해도 위젯 값 변함

- **StringVar**

- String 변수 선언

- **IntVar**

- Integer(정수) 변수 선언

- **DoubleVar**

- Float(실수) 변수 선언

- **BooleanVar**

- True/ False 변수 선언

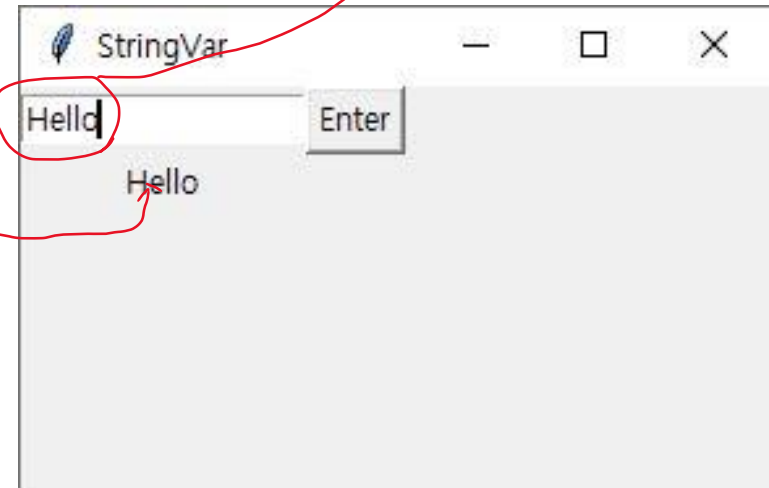
```
from tkinter import *  
window = Tk()
```

```
def click():  
    txt_label.configure(text = input_text.get())
```

```
# input_text = ""  
input_text = StringVar()
```

```
input_text_enterded = Entry(window, width=15, textvariable=input_text)  
input_text_enterded.grid(column=0, row=1)  
action = Button(window, text='Enter', command=click)  
action.grid(column=1, row=1)  
txt_label = Label(window, text="")  
txt_label.grid(column=0, row=2)
```

```
window.title('StringVar')  
window.geometry("200x100")  
window.mainloop()
```



# 체크 박스

- 사용자가 클릭하여 Yes/No를 선택할 수 있게 하는 위젯
- 여러개 동시 선택가능

```
from tkinter import *

window = Tk()

Label(window, text="선호하는 언어를 모두 선택하시오: ").grid(row=0, sticky=W)

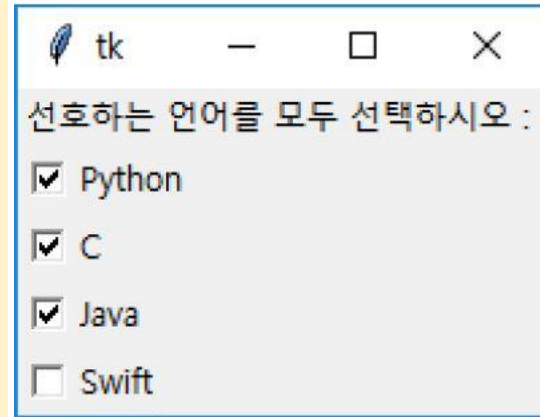
value1 = IntVar()
Checkbutton(window, text="Python", variable=value1).grid(row=1, sticky=W)

value2 = IntVar()
Checkbutton(window, text="C", variable=value2).grid(row=2, sticky=W)

value3 = IntVar()
Checkbutton(window, text="Java", variable=value3).grid(row=3, sticky=W)

value4 = IntVar()
Checkbutton(window, text="Swift", variable=value4).grid(row=4, sticky=W)

window.mainloop()
```



```
from tkinter import *
from tkinter import messagebox

window = Tk()

def showCheckButton() :
    if value1.get() == 1 :
        messagebox.showinfo('', '파이썬을 선택하셨습니다!')
    else :
        messagebox.showinfo('', '파이썬 선택을 취소하셨습니다...ㅠㅠ ')

Label(window, text="선호하는 언어를 모두 선택하십시오 : ").grid(row=0, sticky=W)

value1 = IntVar()
Checkbutton(window, text="Python", variable=value1,
             command=showCheckButton).grid(row=1, sticky=W)

value2 = IntVar()
Checkbutton(window, text="C", variable=value2).grid(row=2, sticky=W)

value3 = IntVar()
Checkbutton(window, text="Java", variable=value3).grid(row=3, sticky=W)

value4 = IntVar()
Checkbutton(window, text="Swift", variable=value4).grid(row=4, sticky=W)

window.mainloop()
```

## 실습 02

- 라디오 버튼을 이용하여 사용자가 선택한 선호 언어를 입력받고, 확인 버튼을 누르면 그 결과를 윈도우에 출력하시오

# 그래픽

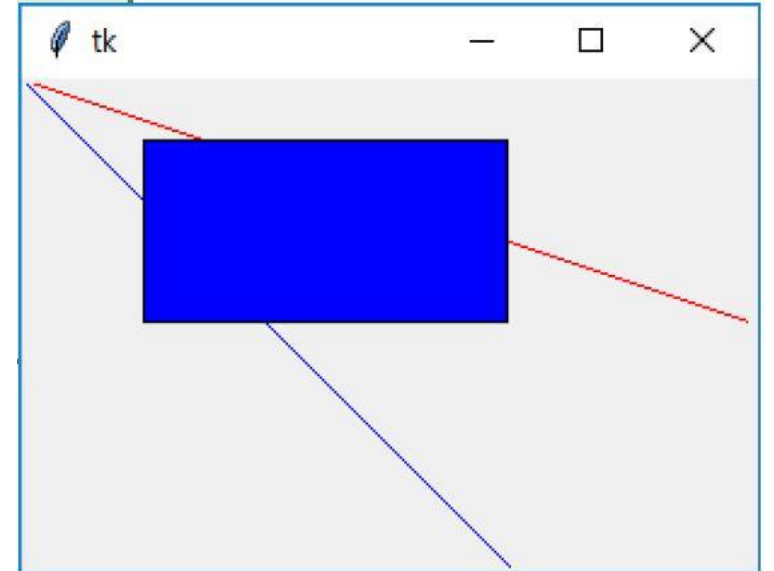
```
from tkinter import *

window = Tk()

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

i = w.create_line(0,0,300,200)
w.create_line(0, 0, 300, 100, fill="red")
w.create_rectangle(50, 25, 200, 100, fill='blue')
w.coords(i, 0, 0, 300, 300)      # 좌표를 변경
w.itemconfig(i, fill='green')      # 색상을 변경

#w.delete(i)      # 삭제
#w.delete(ALL)    # 모든 항목 삭제
window.mainloop()
```



# 그래픽 : 기타

- 타원그리기
  - `canvas.create_oval(x0, y0, x1, y1)`
- 호그리기
  - `create_arc(x0, y0, x1, y1, extent=90, style=ARC)`
- 다각형그리기
  - `create_polygon(x0, y0, x1, y1, x2, y2, outline='red', fill='yellow', width=5)`
- 텍스트표시
  - `create_text(x, y, text='파이썬최고!!', font=('Times', 30), fill='AliceBlue')`
- 이미지표시
  - `img= PhotoImage(file="img/cat.jpg")`
  - `create_image(x, y, anchor=NW, image=img)`

# 마우스 이벤트 처리 01

```
from tkinter import *

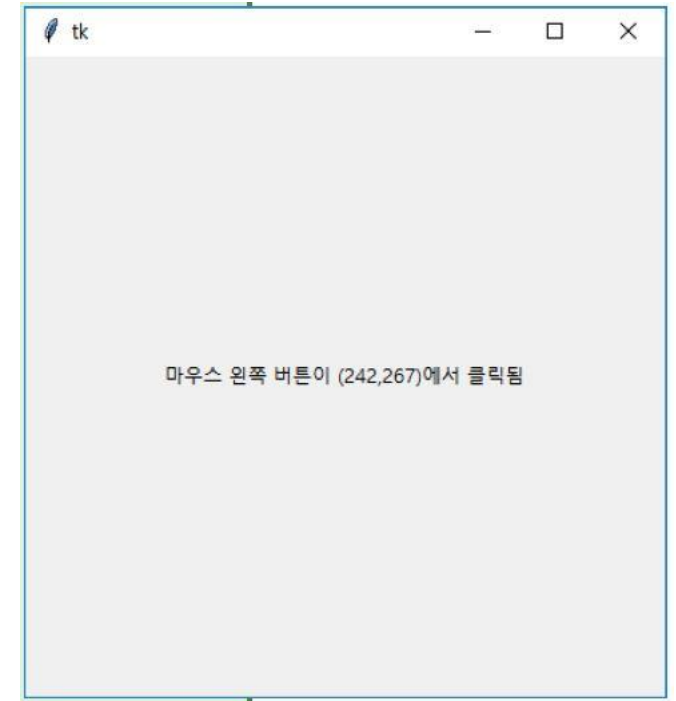
def clickMouse(event) :
    print(event.num)
    txt = ""
    if event.num == 1 :
        txt += "마우스 왼쪽 버튼이 ("
    elif event.num == 3 :
        txt += "마우스 오른쪽 버튼이 ("

    txt += str(event.x) + "," + str(event.y) + ")에서 클릭됨"
    label1.configure(text=txt)

window = Tk()
window.geometry("400x400")
window.bind("<Button>", clickMouse)

label1 = Label(window, text = "이곳이 바뀜")
label1.pack(expand=1, anchor=CENTER)

window.mainloop()
```





# 마우스 이벤트 처리 02

```
from tkinter import *
from tkinter import messagebox

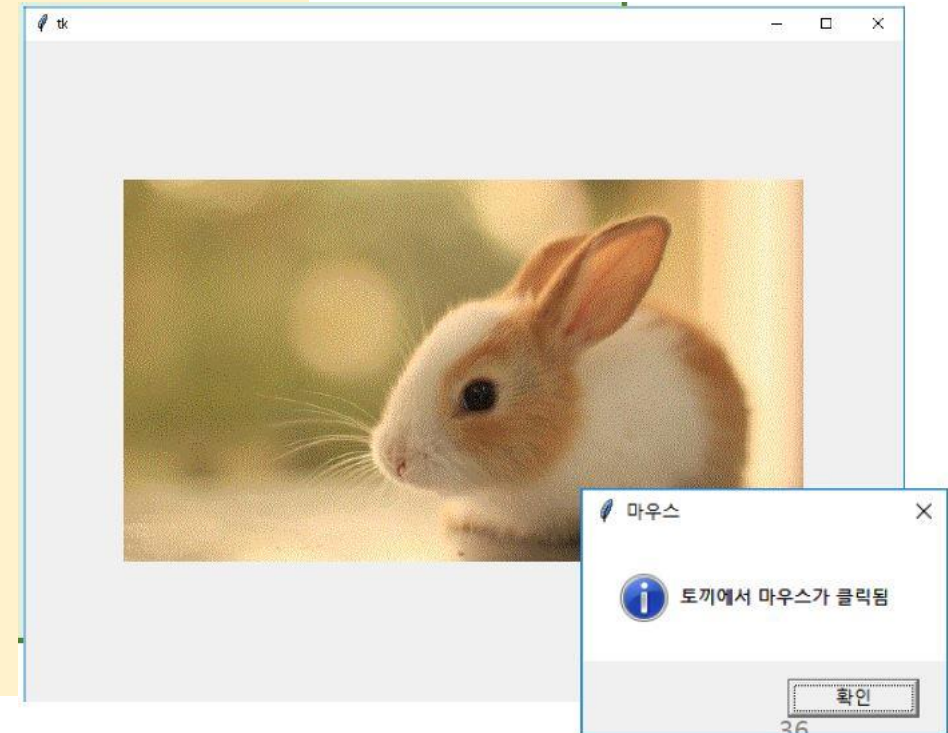
def clickImage(event) :
    messagebox.showinfo("마우스", "토끼에서 마우스가 클릭됨")

window = Tk()
window.geometry("800x600")

photo = PhotoImage(file="img/rabbit.gif")
label1 = Label(window, image=photo)

label1.bind("<Button-1>", clickImage)

label1.pack(expand=1, anchor=CENTER)
window.mainloop()
```



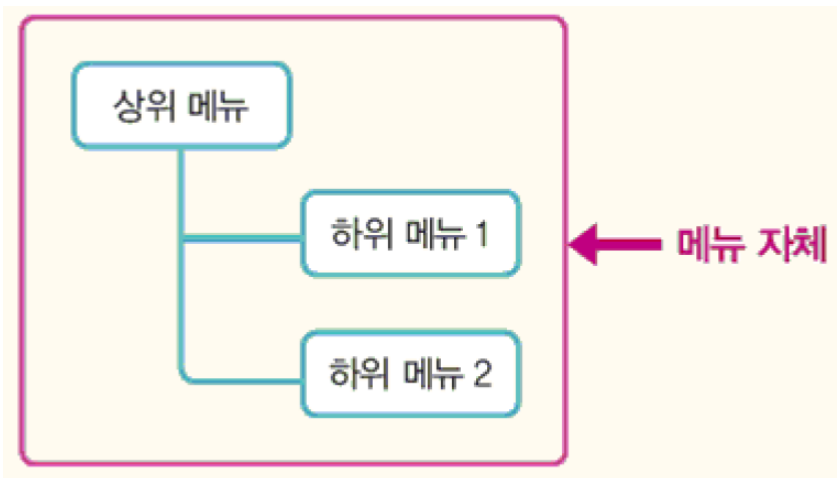
# 마우스 이벤트

마우스 작동	관련 마우스 버튼	이벤트 코드
클릭할 때	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〉
더블클릭할 때	모든 버튼 공통	〈Double-Button〉
	왼쪽 버튼	〈Double-Button-1〉
	가운데 버튼	〈Double-Button-2〉
	오른쪽 버튼	〈Double-Button-3〉
드래그할 때	왼쪽 버튼	〈B1-Motion〉
	가운데 버튼	〈B2-Motion〉
	오른쪽 버튼	〈B3-Motion〉

마우스 작동	이벤트 코드
마우스 커서가 위젯 위로 올라왔을 때	〈Enter〉
마우스 커서가 위젯에서 떠났을 때	〈Leave〉

# 메뉴 만들기

- 메뉴 구성



- 만들기

```
메뉴 자체 = Menu(부모 윈도우)
```

```
부모 윈도우.config(menu = 메뉴 자체)
```

```
상위 메뉴 = Menu(메뉴 자체)
```

```
메뉴 자체.add_cascade(label = "상위 메뉴텍스트, menu = 상위 메뉴)
```

```
상위 메뉴.add_command(label = "하위 메뉴1", command = 함수1)
```

```
상위 메뉴.add_command(label = "하위 메뉴2", command = 함수2)
```

```

from tkinter import *
from tkinter.filedialog import *

def func_open() :
    global photo
    filename = askopenfilename(parent=window, filetypes=(("GIF 파일", "*.gif"), ("모든 파일", "*.*")))
    photo = PhotoImage(file=filename)
    pLabel.configure(image=photo)
    pLabel.image = photo

def func_save() :
    # define options for opening or saving a file
    file_opt = options = {}
    #options['defaultextension'] = '.gif'
    options['filetypes'] = [('gif files', '.gif'), ('png files', '.png'), ('all files', '*.*')]
    options['initialdir'] = 'img/'
    #options['initialfile'] = 'cat.gif'
    options['parent'] = window
    options['title'] = 'My tkinter Example'

    saveName = asksaveasfilename(**file_opt)
    if saveName :
        photo.write(saveName)

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

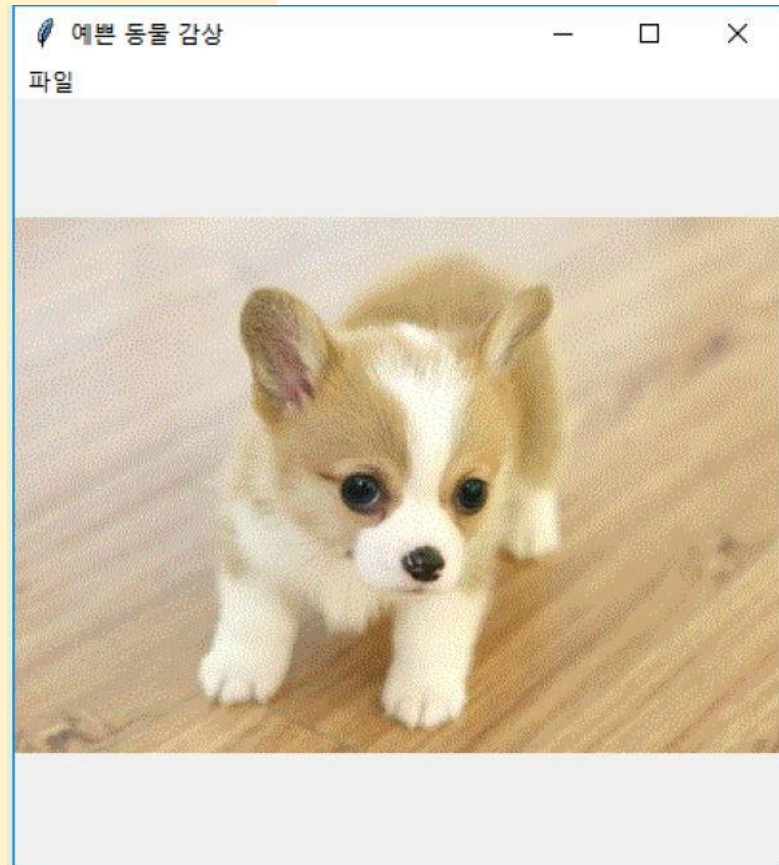
window=Tk()
window.geometry("400x400")
window.title("예쁜 동물 감상")

photo = PhotoImage()
pLabel = Label(window, image=photo)
pLabel.pack(expand=1, anchor=CENTER)

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

fileMenu = Menu(mainMenu, tearoff=0)
mainMenu.add_cascade(label="파일", menu=fileMenu)
fileMenu.add_command(label="열기", command=func_open)
fileMenu.add_command(label="저장", command=func_save)
fileMenu.add_separator()
fileMenu.add_command(label="종료", command=func_exit)

```

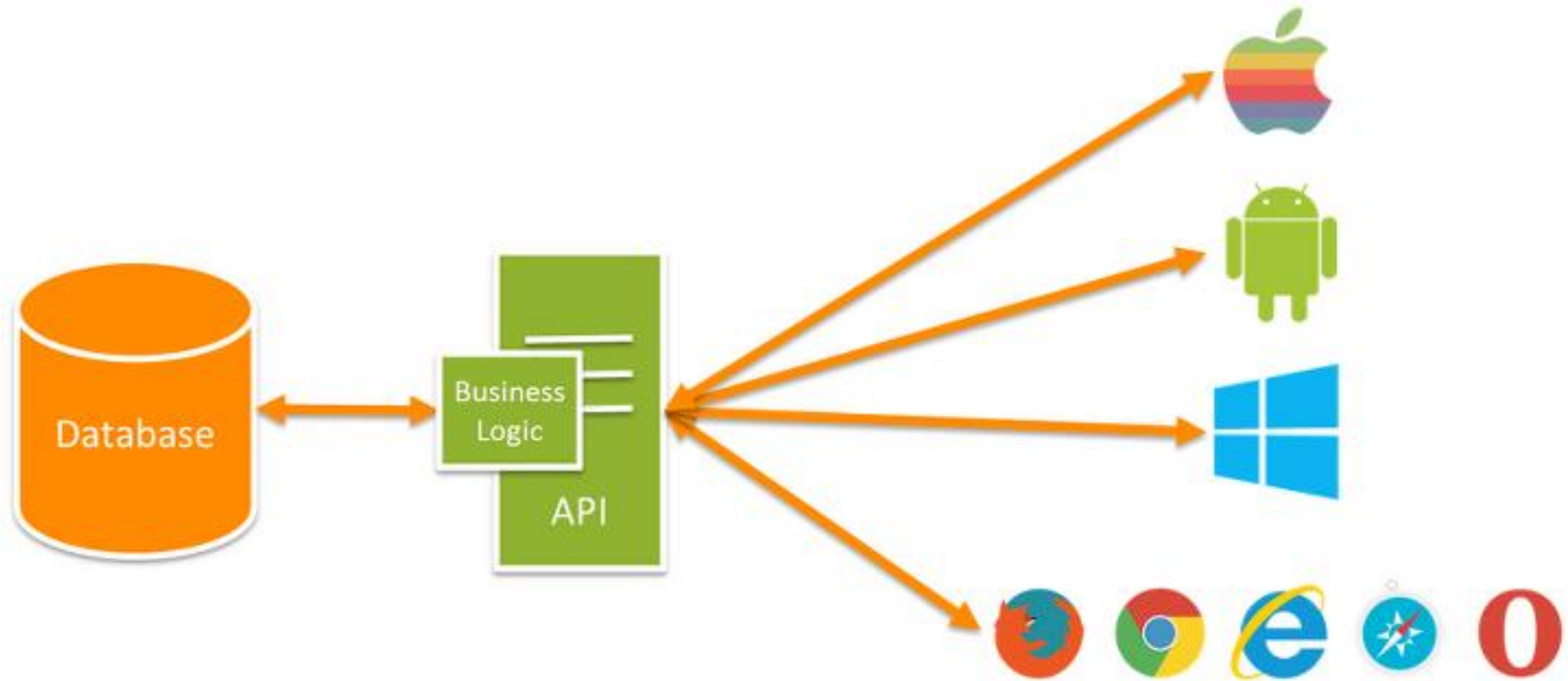


# Weather App 제작

- OpenWeather 사이트 가입
  - <https://openweathermap.org/>
- Current weather data
  - <https://openweathermap.org/current>
- My API keys
  - [https://home.openweathermap.org/api\\_keys](https://home.openweathermap.org/api_keys)
- Weather icon list
  - <https://openweathermap.org/weather-conditions>
- 참고 학습 사이트
  - <https://www.askpython.com/python/examples/gui-weather-app-in-python>

# REST API

- 웹에 존재하는 모든 자원(이미지, 동영상, DB 자원)에 고유한 URI를 부여해 활용"하는 것으로, 자원을 정의하고 자원에 대한 주소를 지정하는 방법
- XML, JSON 형식으로 데이터 전달



# REST API 연산

**“REST”**

GET	/movies	Get list of movies
GET	/movies/:id	Find a movie by its ID
POST	/movies	Create a new movie
PUT	/movies	Update an existing movie
DELETE	/movies	Delete an existing movie

# XML, JSON

- 서로 다른 컴퓨터 시스템 간 통신을 통해 표준화 된 방법으로 데이터를 전달하기 위한 형식

- XML

```
<dog>
  <name>식빵</name>
  <family>웰시코기</family>
  <age>1</age>
  <weight>2.14</weight>
</dog>
```

- JSON

```
{
  "name": "식빵",
  "family": "웰시코기",
  "age": 1,
  "weight": 2.14
}
```

- XML

- <https://www.w3schools.com/xml/>

- JSON

- [https://www.w3schools.com/js/js\\_json\\_intro.asp](https://www.w3schools.com/js/js_json_intro.asp)



# JSON 데이터 파싱하기

- JSON 인코딩
  - 파이썬 객체(Dictionary, List, tuple 등)을 JSON 문자열(str)로 변환
  - `pip install json` : JSON 인코딩/ 디코딩 표준 라이브러리

```
import json

# 테스트용 Python Dictionary
customer = {
    'id': 152352,
    'name': '강진수',
    'history': [
        {'date': '2015-03-11', 'item': 'iPhone'},
        {'date': '2016-02-23', 'item': 'Monitor'},
    ]
}

# JSON 인코딩
jsonString = json.dumps(customer, indent=4)

# 문자열 출력
print(jsonString)
print(type(jsonString))    # class str
```

```
{
    "history": [
        {
            "date": "2015-03-11",
            "item": "iPhone"
        },
        {
            "date": "2016-02-23",
            "item": "Monitor"
        }
    ],
    "id": 152352,
    "name": "\uac15\uc9c4\uc218"
}
```

- JSON 디코딩

- JSON 문자열(str)을 파이썬 객체(Dictionary, List, Tuple 등)으로 변환

```
import json

# 테스트용 JSON 문자열
jsonString = '{"name": "강진수", \
              "id": 152352, \
              "history": [{"date": "2015-03-11", \
                           "item": "iPhone"}, \
                           {"date": "2016-02-23", \
                           "item": "Monitor"}]}'

# JSON 디코딩
dict = json.loads(jsonString)

# Dictionary 데이터 출력
print(dict['name'])
for h in dict['history']:
    print(h['date'], h['item'])
```

결과값 :

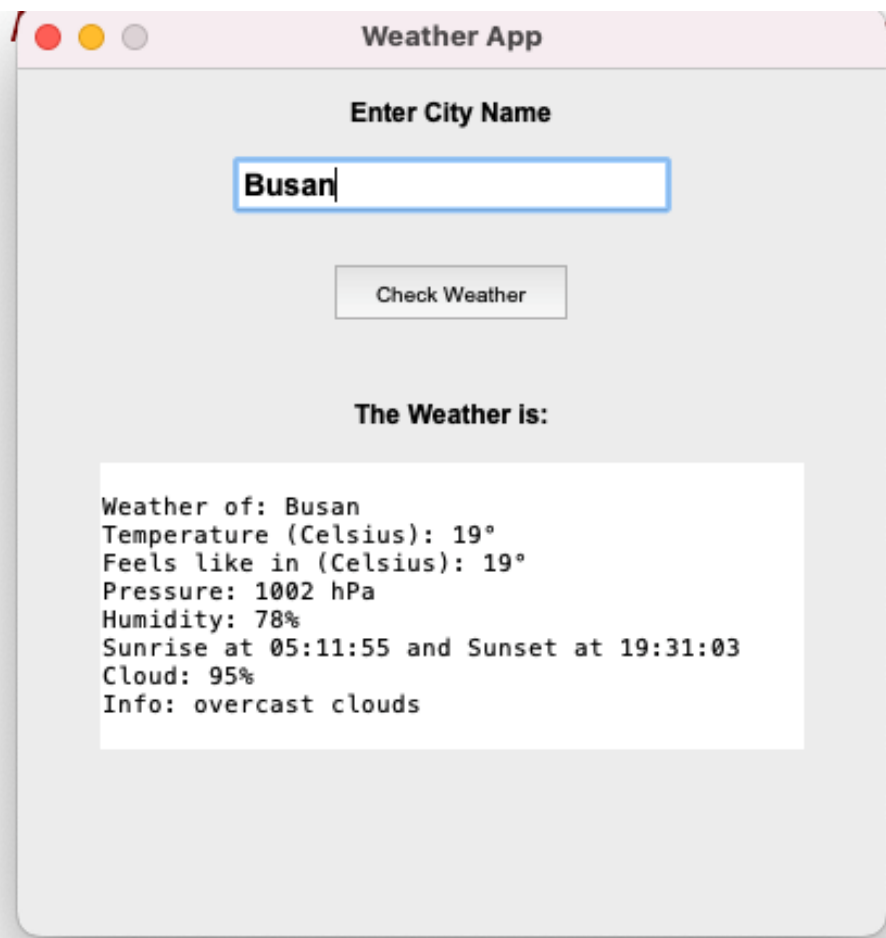
강진수  
2015-03-11 iPhone  
2016-02-23 Monitor

# requests 설치

- requests
  - HTTP 클라이언트
    - 내부 라이브러리 urllib를 제공하나, requests 문법이 간결함
  - `pip install requests`

2022-05-30 21:19:43.583 python[9654:66457] TSM /

```
{
  "coord": {
    "lon": 129.0403,
    "lat": 35.1028
  },
  "weather": [
    {
      "id": 804,
      "main": "Clouds",
      "description": "overcast clouds",
      "icon": "04n"
    }
  ]
}
```




The screenshot shows a macOS-style window titled "Weather App". Inside the window, there is a section titled "Enter City Name" with a text input field containing the word "Busan". Below the input field is a button labeled "Check Weather". Underneath the button, there is a section titled "The Weather is:". Below this title, a white box contains the following text: "Weather of: Busan", "Temperature (Celsius): 19°", "Feels like in (Celsius): 19°", "Pressure: 1002 hPa", "Humidity: 78%", "Sunrise at 05:11:55 and Sunset at 19:31:03", "Cloud: 95%", and "Info: overcast clouds".

Weather Application

Seoul,KR

Weather Report


2022-05-30




City or Country Name

Busan


Weather Report




Clouds



20°C  
68.0°F



62



1003