LESSON

GUI(Graphic User Interface) 프로그래밍

tkinter

참고할 사이트

https://docs.python.org/ko/3/library/tkinter.html

https://076923.github.io/posts/Python-tkinter-1/

창(window) 생성 및 설정

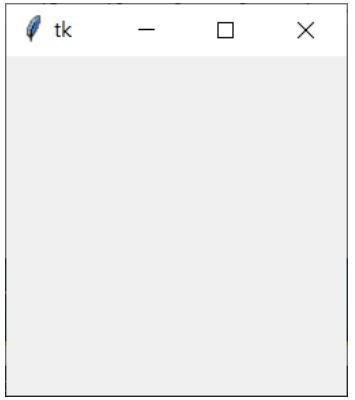
- 패키지 import
```
from tkinter import *
```

- 창 만들기
```
창변수=Tk()

창변수.mainloop()
```

- 창 설정하기
```
 제목: 창변수.title()
 크기: 창변수.geometry('가로x세로+x좌표+y좌표')
 변경: 창변수.resizable(width=False,height=False)
```

창(window) 만들기

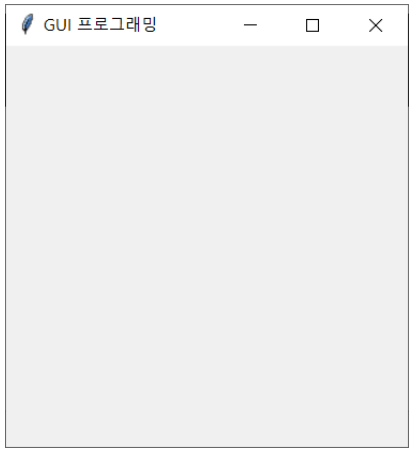

```
# GUI 모듈 import
from tkinter import *

# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
     w = Tk()

    # 창이 닫히지 않게 대기 상태로 만들기
     w.mainloop()
```




```
# 창을 생성
w = Tk()

# 창이 닫히지 않게 창을 대기 상태로 만든다.
w.mainloop()
```

창(window) 크기, 제목 설정하기


```
# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
     w = Tk()

    # 창 크기 및 제목 설정하기
     w.title("GUI 프로그래밍")
    w.geometry("300x300")

    # 창이 닫히지 않게 대기 상태로 만들기
     w.mainloop()
```



이미지 객체 생성

- Photoimage로 이미지 로드하기  png 형식의 파일만 지정 가능.

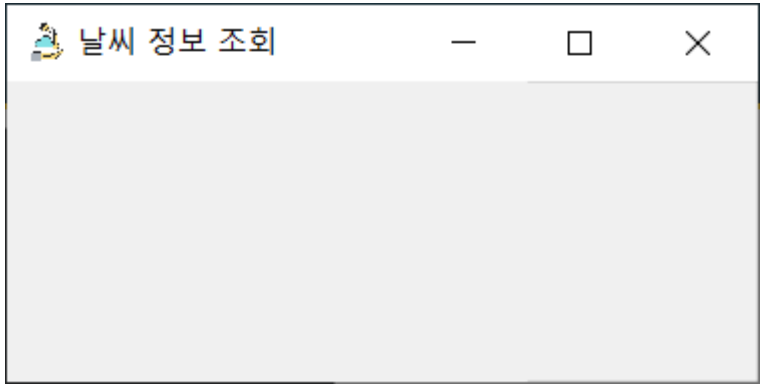
  변수=PhotoImage(file='파일경로명')

- 파일 형식에 관계없이 이미지 로드하기  png, jpg, gif...

  1. pillow 패키지 설치 : pip install Pillow
  2. 패키지 import 하기 : from PIL import ImageTk
  3. 이미지 객체 만들기 : ImageTk.PhotoImage(file='파일명')

창 아이콘 변경하기


```
from tkinter import *

w = Tk()
img=PhotoImage(file='weather.png')
w.iconphoto(True,img)
w.title('날씨 정보 조회')
w.geometry('300x120')

w.mainloop()
```



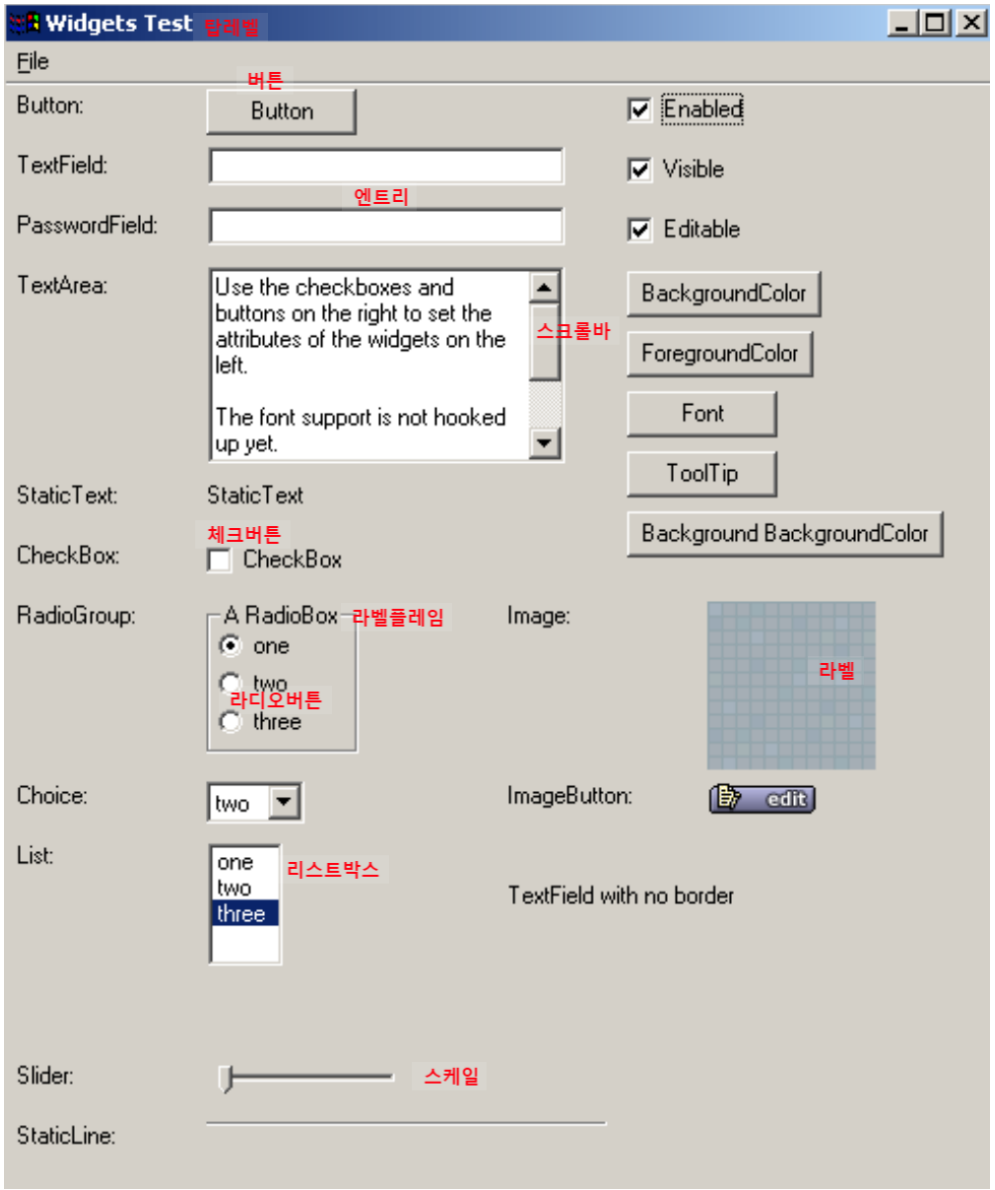
PhotoImage(file=파일명)
Tkinter 라이브러리에서 사용되는 클래스로, 이미지를 표시하기 위한 이미지 객체를 생성한다.

iconphoto()
이미지를 메인 창의 아이콘으로 설정한다. **True** 는 이 이미지를 메인 창의 아이콘으로 지정한다.

위젯(widget)의 종류

위젯(widget)

사용자 인터페이스를 작성하기 위해 제공되는 도구들



Label 위젯 생성

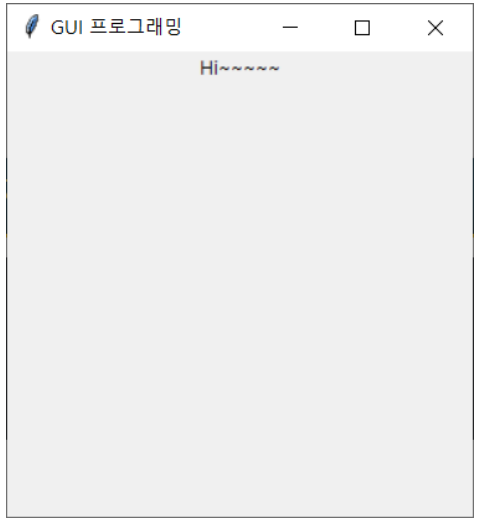
- Label 위젯 생성

  Label(창변수,text='텍스트',font=('글꼴',크기),fg=색상,bg=색상,width=숫자,height=숫자,anchor=값)

  Label(창변수,image=이미지경로)

레이블(Label) 위젯 생성

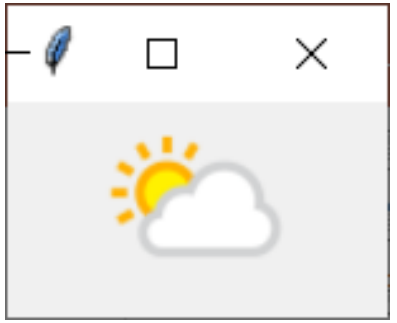

```
# GUI 모듈 import
from tkinter import *

# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
     w = Tk()

    # 창 크기 및 제목 설정하기
     w.title("GUI 프로그래밍")
    w.geometry("300x300")

    # 레이블 위젯 생성
     label = Label(w, text="Hello~")
    # 윈도우에 위젯 배치하기
     label.pack()

    # 창이 닫히지 않게 대기 상태로 만들기
     w.mainloop()
```




```
label=Label(w,text='Hi~~~~~',font=('휴먼편지체',20),fg='red')
label=Label(w,text='Hi~~~~~',font=('휴먼편지체',20),fg='red',bg='yellow',width=30,height=10
,anchor=SE)
```

레이블(Label) 위젯에 이미지 넣기


```
# GUI 모듈 import
from tkinter import *

# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
    w = Tk()

    # 창 크기 및 제목 설정하기
    w.title("GUI 프로그래밍")
    w.geometry("300x300")

    # 이미지 로드하기
    img = PhotoImage(file="wicon.png")

    # 레이블 위젯 생성
    label = Label(w, image=img)

    # 윈도우에 위젯 배치하기
    label.pack()

    # 창이 닫히지 않게 대기 상태로 만들기
    w.mainloop()
```



```
label.place(x=50,y=100)
label.grid(row=0,column=0)
```

위젯의 배치

- **pack()**
- place()
- grid()

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

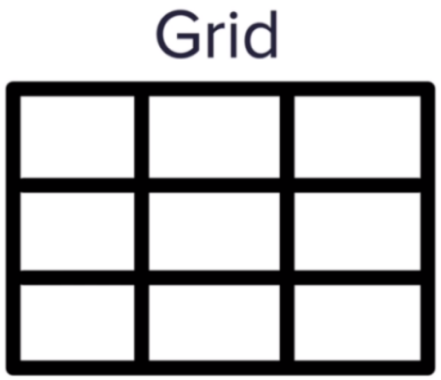
위젯의 배치

- pack()
- **place()**
- grid()

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

위젯의 배치

- pack()
- place()
- **grid()**



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

Button 위젯 생성

- Button 위젯 생성

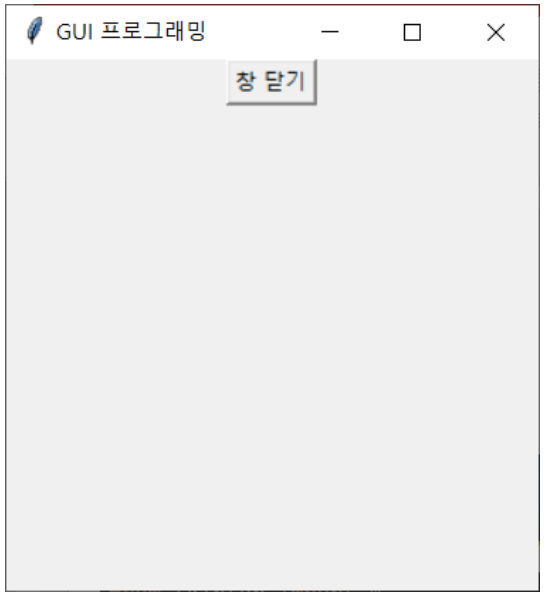
Button(창,text='텍스트',command=처리할 함수)

Button(창,image=이미지경로 ,command=처리할 함수)

이름	의미	기본값	속성
width	버튼의 너비	0	상수
height	버튼의 높이	0	상수
relief	버튼의 테두리 모양	flat	flat, groove, raised, ridge, solid, sunken
overrelief	버튼에 마우스를 올렸을 때 버튼의 테두리 모양	raised	flat, groove, raised, ridge, solid, sunken
borderwidth=bd	버튼의 테두리 두께	2	상수
background=bg	버튼의 배경 색상	SystemButtonFace	color
foreground=fg	버튼의 문자열 색상	SystemButtonFace	color

버튼(Button) 위젯 생성하기


```python
# GUI 모듈 import
from tkinter import *

# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
    w = Tk()

    # 창 크기 및 제목 설정하기
    w.title("GUI 프로그래밍")
    w.geometry("300x300")

    # 버튼 위젯 생성
    button = Button(w, text="창 닫기", command=quit)

    # 윈도우에 위젯 배치하기
    button.pack()

    # 창이 닫히지 않게 대기 상태로 만들기
    w.mainloop()
```

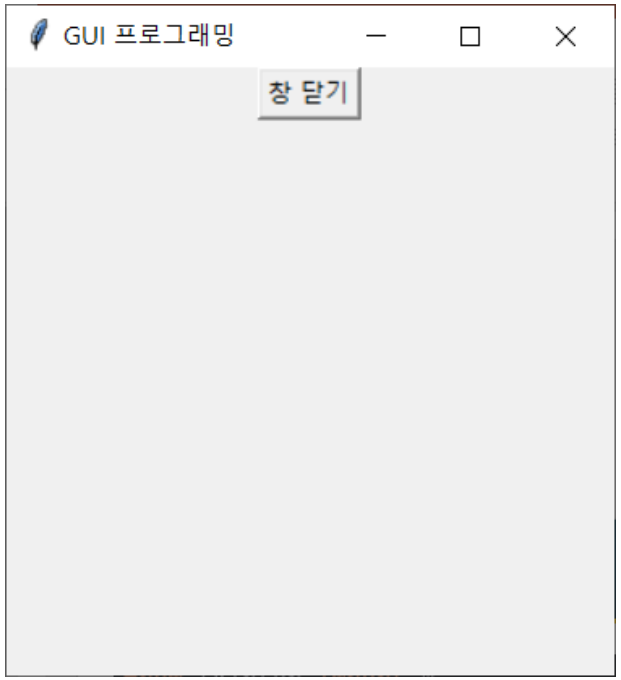


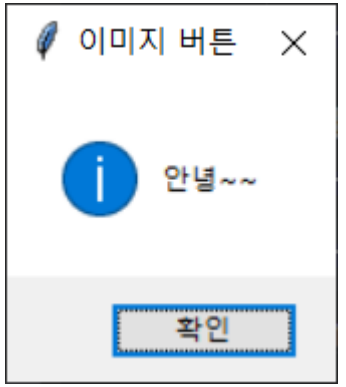

```python
button = Button(w, text='클릭', command=lambda: print("버튼이 클릭되었습니다!"))
```

버튼(Button) 클릭시 "안녕!" 메시지 표시하기

Python 2

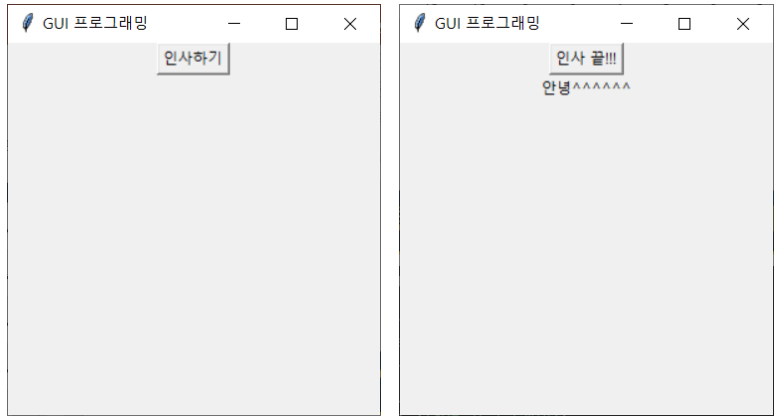
```
from tkinter import *
from tkinter import messagebox

def show_hello():
    messagebox.showinfo('이미지 버튼', '안녕~~')

w = Tk()
w.title('GUI 프로그래밍')
w.geometry('300x300')
button = Button(w,text="인사하기",command=show_hello)
button.pack()

w.mainloop()
```





버튼 클릭시 레이블에 "안녕~" 메시지 표시하기


```
from tkinter import *

def show_hello():
    label.config(text="안녕^^^^^^")
    button.config(text="인사 끝!!!")

w = Tk()
w.title('GUI 프로그래밍')
w.geometry('300x300')
button = Button(w,text="인사하기",command=show_hello)
button.pack()
label=Label(w,text='')
label.pack()

w.mainloop()
```



Label 위젯의 내용을 변경할 때 config()를 사용한다.

Text, Entry 위젯 생성

- Text 위젯, Entry 위젯 생성
  Entry(창변수,width=숫자)
  Text(창변수,width=숫자,height=숫자)

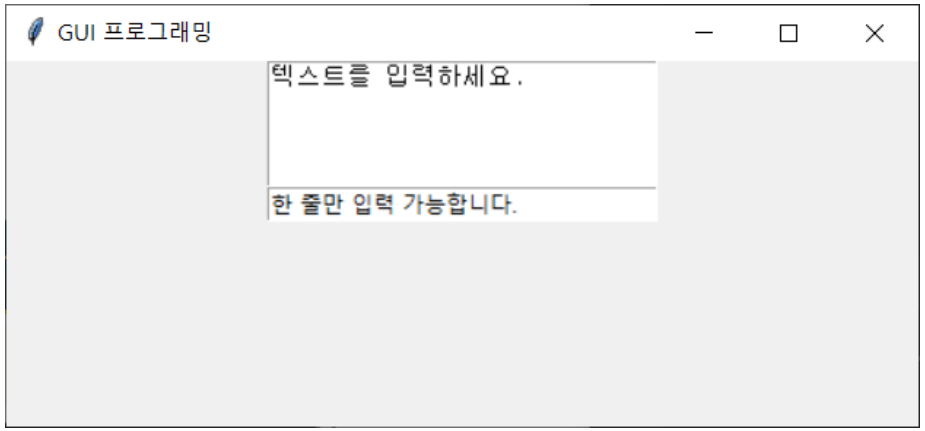
이름	의미	설명
insert(index, “문자열”)	문자열 삽입	해당 `index` 위치에 `문자열` 을 삽입
delete(start_index, end_index)	문자열 삭제	`start_index` 부터 `end_index` 까지의 문자열 삭제
get(start_index, end_index)	문자열 반환	`start_index` 부터 `end_index` 까지의 문자열 반환

Tip : `index` 는 `y.x` 를 사용, `y줄, x번째 문자` 를 의미함 `예) 첫 번째 줄, 첫 번째 문자 = 1.0`

텍스트(Text) 위젯, 엔트리(Entry) 위젯 사용

Text 위젯 : 여러 줄 입력 가능
Entry 위젯 : 한 줄만 입력 가능

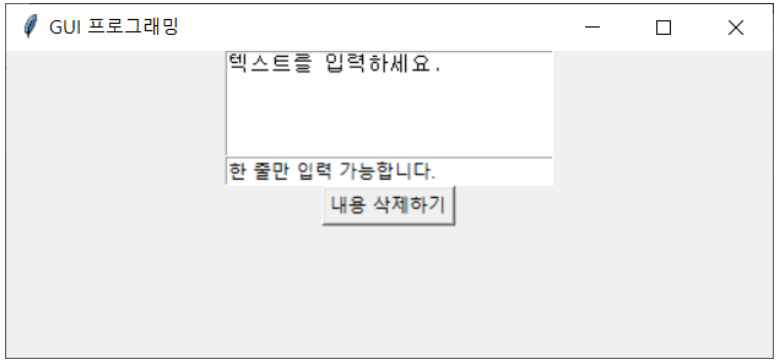
```
from tkinter import *

w=Tk()
w.title('GUI 프로그래밍')
w.geometry('500x200')
text=Text(w,width=30,height=5)
text.insert(END,'텍스트를 입력하세요.')
text.pack()

entry=Entry(w,width=30)
entry.insert(END,'한 줄만 입력 가능합니다.')
entry.pack()

w.mainloop()
```



END: 현재 텍스트의 끝을 의미. 이 위치에 텍스트를 추가한다.

버튼 클릭시 텍스트, 엔트리의 내용을 출력 또는 삭제하기 Python 2

```
from tkinter import *

w=Tk()
w.title('GUI 프로그래밍')
w.geometry('500x200')
text=Text(w,width=30,height=5)
text.insert(END,'텍스트를 입력하세요.')
text.pack()
entry=Entry(w,width=30)
entry.insert(END,'한 줄만 입력 가능합니다.')
entry.pack()
def del_text():
    print(text.get('1.0',END))
    print(entry.get())
    text.delete('1.0',END)
    entry.delete(0,END)

btn=Button(w,text='내용 삭제하기',command=del_text)
btn.pack()
w.mainloop()
```




```
text.delete('2.0',END)
entry.delete(3,END)
```

마우스 이벤트 처리

- 마우스 이벤트 처리하기

```
def 이벤트처리함수(event):
        처리할 내용
```

위젯명.bind('마우스이벤트',이벤트처리함수)

event

tkinter에서 발생한 이벤트에 관한 정보를 담고 있는 객체로서 함수를 이벤트에 바인딩할 때 이벤트 객체가 해당 함수에 전달된다.

매개변수명이므로 사용자가 임의로 지정할 수 있다.

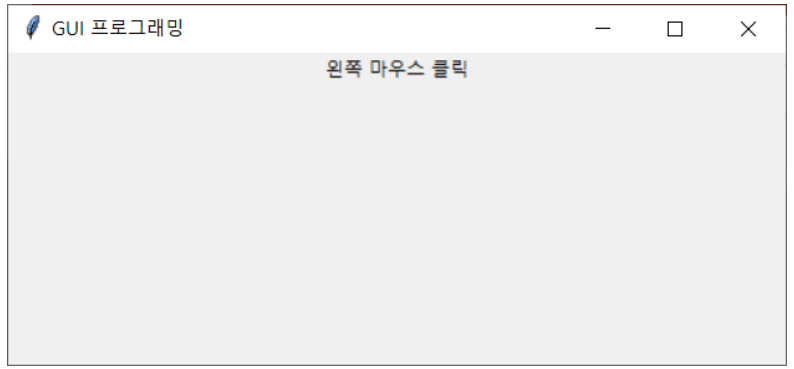
마우스 이벤트

마우스 이벤트	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>
마우스가 위젯 위에 올려질 때		<Enter>
마우스가 위젯에서 떠났을 때		<Leave>

마우스 이벤트, 레이블 클릭시 메시지 출력하기

Python 2

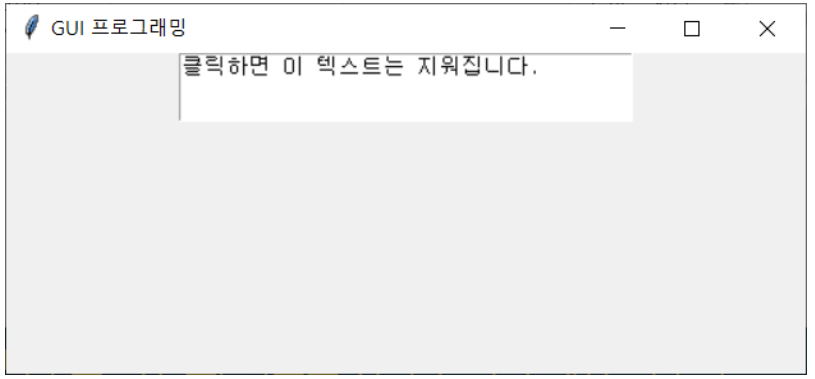
```
from tkinter import *

w=Tk()
w.title('GUI 프로그래밍')
w.geometry('500x200')
def click_left(event):
    label.config(text="왼쪽 마우스 클릭")

label=Label(w,text='hello~~~~~')
label.bind('<Button-1>',click_left)
# w.bind('<Button-1>',click_left)  # 창 아무데나 클릭시
label.pack()

w.mainloop()
```



마우스 이벤트, 텍스트 상자 클릭시 내용 지우기

Python 2

```
from tkinter import *

def clear_text(event):
    text.delete(1.0, END)

w = Tk()
w.title('GUI 프로그래밍')
w.geometry('500x200')

text = Text(w, height=3, width=40)
text.pack()
text.insert(END, "클릭하면 이 텍스트는 지워집니다.")

# 텍스트 상자를 마우스로 클릭할 때 clear_text 함수 호출
text.bind("<Button-1>", clear_text)

w.mainloop()
```



키보드 이벤트 처리

- 키보드 이벤트 처리하기

```
def 이벤트처리함수(event):
    처리할 내용
```

위젯명.bind('키보드이벤트',이벤트처리함수)

키보드 이벤트

이름	의미
<Key>	특정 키가 입력되었을 때
<Return>	Enter 키가 입력되었을 때
<Cancel>	Break 키가 입력되었을 때
<Pause>	Pause 키가 입력되었을 때
<BackSpace>	백스페이스 키가 입력되었을 때
<Caps_Lock>	캡스 락 키가 입력되었을 때
<Escape>	이스케이프 키가 입력되었을 때
<Home>	Home 키가 입력되었을 때
<End>	End 키가 입력되었을 때

이름	의미
<Print>	Print 키가 입력되었을 때
<Insert>	Insert 키가 입력되었을 때
<Delete>	Delete 키가 입력되었을 때
<Prior>	Page UP 키가 입력되었을 때
<Up>	윗쪽 방향키가 입력되었을 때
<Down>	아랫쪽 방향키가 입력되었을 때
<Right>	오른쪽 방향키가 입력되었을 때
<Left>	왼쪽 방향키가 입력되었을 때

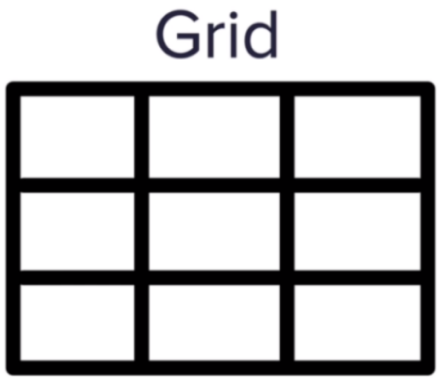
키보드 이벤트, 엔터키를 누르면 entry내용 지우기

Python 2

```
from tkinter import *

w = Tk()
w.title('GUI 프로그래밍')
w.geometry('300x120')
entry = Entry(w)
entry.insert(0,"엔터키를 누르면 내용이 삭제됩니다.")
entry.pack()

def on_enter(event):
    entry.delete(0,END)

w.bind("<Return>", on_enter)

w.mainloop()
```

위젯의 배치

- pack()
- **grid()**
- place()



grid()함수의 매개변수

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

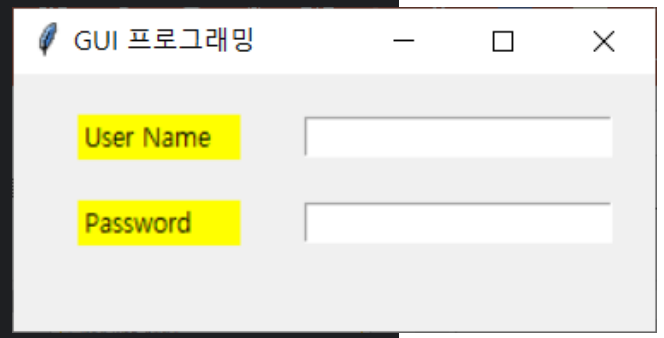
grid로 위젯 배치하기


```
# GUI 모듈 import
from tkinter import *
# 프로그램 시작
if __name__ == "__main__":
    # 창 만들기
     w = Tk()
    # 창 크기 및 제목 설정하기
     w.title("GUI 프로그래밍")
    w.geometry("300x120")
    # 위젯 생성
     lbl_name = Label(w, text="User Name", width=10, bg="yellow", anchor="w")
    lbl_pwd = Label(w, text="Password", width=10, bg="yellow", anchor="w")
    ent_name = Entry(w, justify="center")
    ent_pwd = Entry(w)
    # 윈도우에 위젯 배치하기
     lbl_name.grid(row=0, column=0, padx=30)
    lbl_pwd.grid(row=1, column=0)
    ent_name.grid(row=0, column=1, pady=20)
    ent_pwd.grid(row=1, column=1)

    # 창이 닫히지 않게 대기 상태로 만들기
     w.mainloop()
```

anchor, justify

anchor는 단일 행에 대한 정렬

justify는 다중 행에 대한 정렬



weatherAPI의 현재 날씨 정보앱 만들기


현재 날씨 정보 조회
검색할 도시(지역)를 입력하고 엔터를 누르세요.


현재 날씨 정보 조회
검색할 도시(지역)를 입력하고 엔터를 누르세요.
seoul


현재 날씨 정보 조회
검색할 도시(지역)를 입력하고 엔터를 누르세요.
South Korea, Seoul
25.0℃
Partly cloudy

```python
# 모듈 import
from tkinter import *
from tkinter import messagebox
import requests

# 프로그램 시작
if __name__ == "__main__":
    w = Tk()
    w.title("현재 날씨 조회")
    w.geometry("400x400")

    # weatherAPI에서 날씨 정보 받는 함수(get_weather 작성)

    # 위젯 생성
    # 제목이 표시될 레이블
    label_search = Label(w, padx=10, pady=20, font=("맑은 고딕", 12), text="검색할 도시(지역)을 입력하고 엔터를 누르세요.")
    # 검색어 입력 상자
    entry_city = Entry(w, width=20, font=("hy견고딕", 20), justify="center")
    # 도시명이 표시될 레이블
    label_country = Label(w, font=("hy견고딕", 20), fg="red", height=2)
    # 기온이 표시될 레이블
    label_temp = Label(w, font=("hy견고딕", 20), fg="red", height=2)
    # 날씨 상태가 표시될 레이블
    label_condition = Label(w, font=("hy견고딕", 20), fg="red", height=2)
    # 이미지가 표시될 레이블
    image_weather = Label(w)
```

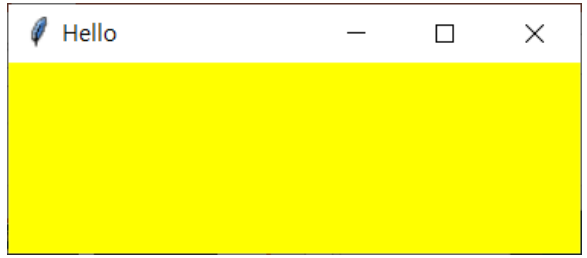
```
# 윈도우에 위젯 배치하기
label_search.pack()
entry_city.pack()
label_country.pack()
label_temp.pack()
label_condition.pack()
image_weather.pack()

# 이벤트 지정
 w.bind("<Return>", get_weather)

# 창이 닫히지 않게 대기 상태로 만들기
 w.mainloop()
```

```python
def get_weather(event):
    # 엔트리 상자의 값을 가져와 city 변수에 저장
    city = entry_city.get()
    key = "e7c52d2af12e42c781112815230710"

    # 서버에 데이터 요청
    response = requests.get(f"http://api.weatherapi.com/v1/current.json?key={key}&q={city}")
    if response.status_code == 200:
        # 응답받은 데이터를 파이썬 자료형으로 변환
        weather = response.json()
        # 날씨 이미지가 있는 주소 생성
        icon_addr = "https:" + weather['current']['condition']['icon']
        # 주소에 요청 보내고 이미지 받기
        img = requests.get(icon_addr)
        # 이미지 저장하기
        with open("current_weather.png", "wb") as f:
            f.write(img.content)

        # 윈도우에 표시할 이미지 로드
        img = PhotoImage(file="current_weather.png")
        # GUI 업데이트
        label_country.config(text=weather['location']['country'] + '\n' + weather['location']['name'])
        label_temp.config(text=str(weather['current']['temp_c']) + "℃")
        label_condition.config(text=weather['current']['condition']['text'])
        image_weather.config(image=img)
        image_weather.image = img
        entry_city.delete(0, END)
    elif response.status_code == 400:
        messagebox.showinfo("지명 오류", "입력한 지명이 없습니다.")
        entry_city.delete(0, END)
    else:
        messagebox.showinfo("접속 오류", "서버에 접속할 수 없습니다.")
        entry_city.delete(0, END)
```

시계앱 만들기

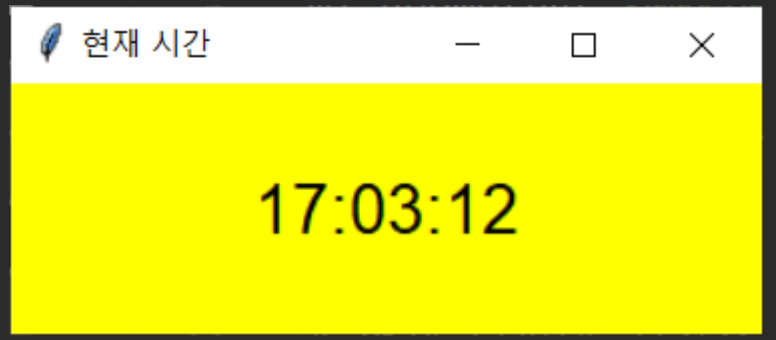
시계 만들기


```
from tkinter import *

w = Tk()
# 창의 제목 설정
w.title("Hello")
# 창의 크기 설정
w.geometry('300x100')
#배경색을 노랑으로 설정
w.config(bg="yellow")
# 창이 닫히지 않고 계속 실행되도록 설정한다.
w.mainloop()
```



시계 만들기


```
from tkinter import *
import time

w = Tk()
w.title("현재 시간")
w.geometry('300x100')

# 배경색을 노랑으로 설정
w.config(bg='yellow')
# 현재 날짜와 시간을 얻어오기
current_time = time.strftime("%H:%M:%S")
# 폰트 설정
font_style = ("Arial", 20)
# 레이블을 통해 날짜와 시간을 표시하고 가운데 정렬
label = Label(w, text=current_time, font=font_style, bg='yellow',pady=40)
label.pack()
w.mainloop()
```



1초마다 시간이 변경되는 시계 만들기


```
from tkinter import *
import time

def update_time():
    current_time = time.strftime("%%H:%M:%S")
    label.config(text=current_time)
    w.after(1000, update_time)  # 1초마다 update_time 함수를 호출하여 시간 업데이트

w = Tk()
w.title("현재 시간")
w.geometry('300x100')
# 배경색을 노랑으로 설정
w.config(bg='yellow')
# 폰트 설정
font_style = ("Arial", 40)
# 레이블을 통해 날짜와 시간을 표시하고 가운데 정렬
label = Label(w, text="", font=font_style, bg='yellow', pady=40)
label.pack()
# 최초 업데이트 실행
update_time()
w.mainloop()
```