Tkinter

❖ Tkinter

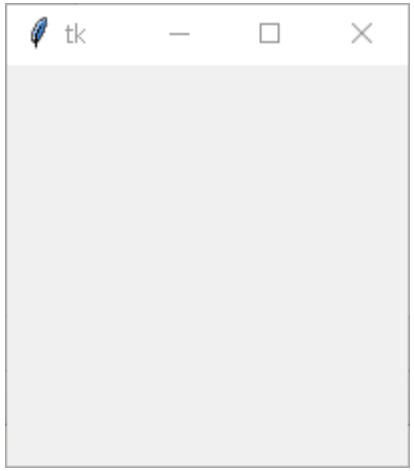
- Tcl/Tk에 대한 파이썬 Wrapper
- Tcl/Tk를 파이썬에 사용할 수 있도록 한 Lightweight GUI 모듈
- Tcl(Tool Command Language) : 일종의 프로그래밍 언어
- Tk:크로스 플랫폼에 사용되는 일종의 GUI 툴킷
- 장점
  - 파이썬의 기본 모듈
- 단점
  - 타 GUI 프레임워크나 툴킷에 비해 지원되는 위젯들이 부족
  - UI가 예쁘지 않음

- http://effbot.org/tkinterbook/

❖ Tkinter의 기본 문장

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

- mainloop()
  - 이벤트 메시지 루프
  - 다양한 이벤트로부터 오는 메시지를 받고 전달하여 처리하는 역할



❖ 위젯

- 화면을 한 구성 요소
  - 버튼, 라벨, 엔트리(텍스트 입력 위젯), 리스트 등

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

lbl = Label(root, text="이름")
lbl.pack()

txt = Entry(root)
txt.pack()

btn = Button(root, text="OK")
btn.pack()

root.mainloop()
```

❖ Geometry Manager

- 윈도우 내에서 위젯의 배치 관리자
- 배치 방법
  - **Place (혹은 absolute)**
    - 위젯을 위치를 절대 좌표로 정하는 것
    - 윈도우 크기 변경에 따라 위젯들이 변경되지 않으므로 많으므로 비권장
    - 위젯.place() 메서드로 지정
  - **Pack**
    - 위젯들을 부모 위젯에 모두 패킹하여 불필요한 공간을 없앰
    - 위젯.pack() 메서드로 지정
  - **Grid**
    - 위젯들을 테이블 레이아웃에 배치하는 것으로 지정
    - row, column에 위젯 배치
    - 위젯.grid() 메서드로 지정

❖ Geometry Manager

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

lbl = Label(root, text="이름")
lbl.grid(row=0, column=0)

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

btn = Button(root, text="OK", width=15)
btn.grid(row=1, column=1)

root.mainloop()
```

❖ 위젯

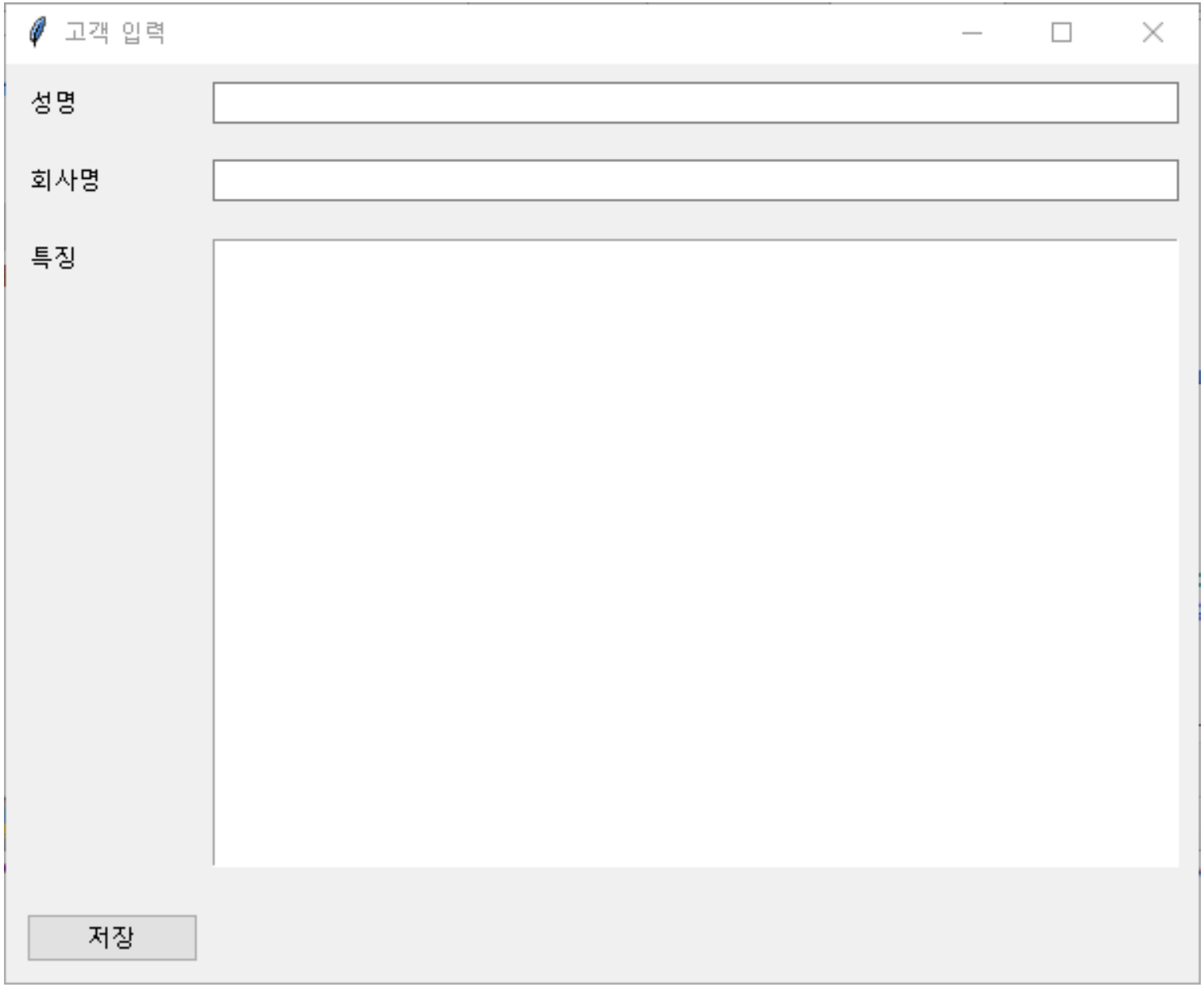
- 위젯 인스턴스화할 때 항상 부모 윈도우(또는 위젯)을 지정

위젯	설명
Button	단순한 버튼
Label	텍스트 혹은 이미지 표시
CheckButton	체크박스
Entry	단순한 한 라인 텍스트 박스
ListBox	리스트 박스
RadioButton	옵션 버튼
Message	Label과 비슷하게 텍스트 표시. Label과 달리 자동 래핑 기능이 있다.
Scale	슬라이스 바
Scrollbar	스크롤 바
Text	멀티 라인 텍스트 박스로서 일부 Rich Text 기능 제공

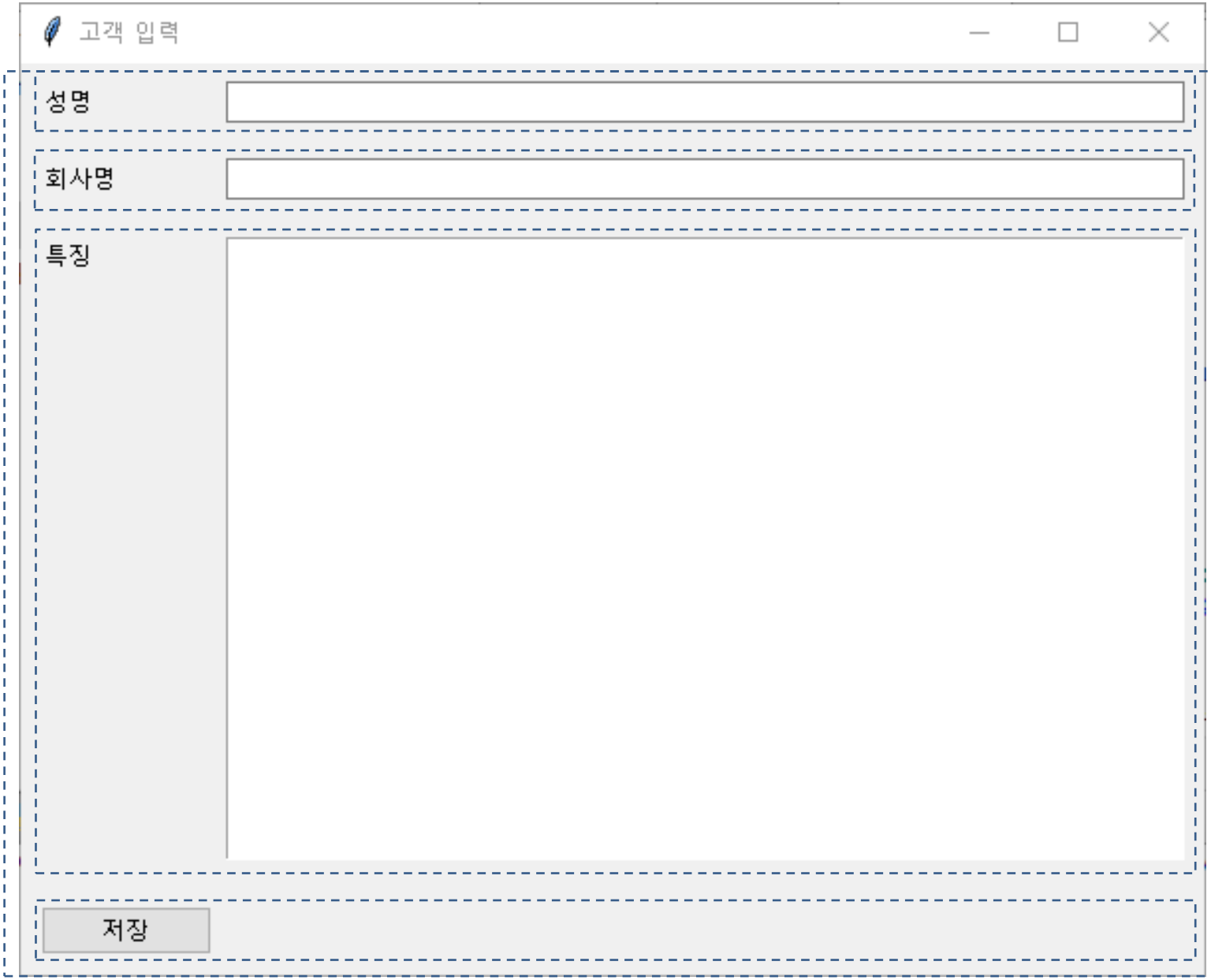
❖ 위젯

위젯	설명
Menu	메뉴 Pane
Menubutton	메뉴 버튼
Toplevel	새 윈도우를 생성할 때 사용. Tk()는 윈도우를 자동으로 생성하지만 추가로 새 윈도우 혹은 다이얼로그를 만들 경우 Toplevel를 사용한다
Frame	컨테이너 위젯. 다른 위젯들을 그룹화할 때 사용
Canvas	그래프와 점들로 그림을 그릴 수 있으며, 커스텀 위젯을 만드는데 사용될 수도 있다

Tkinter



Tkinter



❖ Geometry Manager

```
from tkinter import *
from tkinter.ttk import *

class MyFrame(Frame):     # Frame - 윈도우 관리 클래스
    def __init__(self, master):
        Frame.__init__(self, master)   # master는 부모 윈도우

        self.master = master
        self.master.title("고객 입력")
        self.pack(fill=BOTH, expand=True) # 부모 윈도우 크기에 맞게 크기 조정

def main():
    root = Tk()                          # 메인 윈도우
    root.geometry("600x300+100+100")   # 가로x세로+X위치+Y위치
    app = MyFrame(root)
    root.mainloop()

if __name__ == '__main__':
    main()
```

❖ 위젯.pack()

- anchor=
  - 위젯의 배치 위치,
  - N, NE, E, SE, S, SW, W, NW, CENTER
- expand=
  - 남은 여백이 있는 경우 확장할지 여부(false(기본), true)
- fill=
  - NONE : 원래 크기 유지
  - X, Y, BOTH : 가로, 세로, 전체로 확장
  - 숫자 : 지정한 크기로 확장
- in=
  - 지정한 위젯안에 배치
- ipadx=, ipady=
  - 내부 여백(경계선과 내용간의 간격)
- padx=, pady=
  - 외부 여백(경계선과 다른 위젯간의 간격)
- side=
  - 위젯을 포장할 때의 방향
  - TOP, BOTTOM, LEFT, RIGHT

❖ Geometry Manager

```
class MyFrame(Frame):
    def __init__(self, master):
        :

        # 성명
        frameName = Frame(self)
        frameName.pack(fill=X, expand=True)

        lblName = Label(frameName, text="성명", width=10)
        lblName.pack(side=LEFT, padx=10, pady=10)

        self.entryName = Entry(frameName)
        self.entryName.pack(fill=X, padx=10, expand=True)
```

❖ Geometry Manager

```
        # 회사
        frameCompany = Frame(self)
        frameCompany.pack(fill=X)

        lblComp = Label(frameCompany, text="회사명", width=10)
        lblComp.pack(side=LEFT, padx=10, pady=10)

        self.entryComp = Entry(frameCompany)
        self.entryComp.pack(fill=X, padx=10, expand=True)
```

❖ Geometry Manager

```
# 특징
        frameFeature = Frame(self)
        frameFeature.pack(fill=BOTH, expand=True)

        lblComment = Label(frameFeature, text="특징", width=10)
        lblComment.pack(side=LEFT, anchor=N, padx=10, pady=10)

        self.txtComment = Text(frameFeature,  height=10)
        self.txtComment.pack(fill=BOTH, pady=10, padx=10)

        # 저장
        frameSave = Frame(self)
        frameSave.pack(fill=X)

        btnSave = Button(frameSave, text="저장")
        btnSave.pack(side=LEFT, padx=10, pady=10)
```

❖ 이벤트

- 가장 기본적인 이벤트 클릭 처리
  - 위젯 생성자에 command= 에 이벤트 핸들러 지정

```
class MyFrame(Frame):
    def __init__(self, master):

        …

        self.btnSave = Button(frameSave, text="저장",
                              command=lambda : self.on_click())
        self.btnSave.pack(side=LEFT, padx=10, pady=10)

    def on_click(self):
        print('[%s]' % self.entryName.get())
        print('[%s]' % self.entryComp.get())
        print('[%s]'%self.txtComment.get("1.0"))
        # print('[%s]'%self.txtComment.get("1.0",END))
        #print('[%s]' % self.txtComment.get("1.0", "end-1c"))
```

  - get("1.0") : 첫번째 글자를 위치 0에서부터 추출
  - "end-1c" : 끝에서 1글자 앞까지 추출 - 개행 문자 제거방법

❖ 다른 이벤트 핸들러 연결하기

- 위젯.bind(이벤트명, 핸들러함수)
- 이벤트 명
  - <Button-1> 마우스 왼쪽 버튼 클릭
  - <Button-2> 마우스 중간 버튼 클릭
  - <Button-3> 마우스 오른쪽 버튼 클릭
  - <Double-Button-1> 왼쪽 버튼 더블클릭
  - <Return> Enter 키 눌려짐
  - <Key> 키가 눌려짐
- 이벤트 핸들러 매개변수 : 이벤트 객체
  - char : 키보트 이벤트에서 발생하는 문자 하나
  - keysym : 키보트 이벤트에서 발생하는 키의 심볼명
  - num : 마우스 이벤트의 버튼 번호. 왼쪽부터 1, 2, 3
  - x, y : 위젯의 죄상단으로부터의 상대적 마우스 위치
  - x_root, y_root : 화면 죄상단으로부터의 상대적 마우스 위치
  - Key : 이벤트가 발생한 위젯

❖ 이미지 출력

```
from tkinter import *


class MyFrame(Frame):
    def __init__(self, master):
        img = PhotoImage(file='light_on.png')
        lbl = Label(image=img)
        lbl.image = img  # 레퍼런스 추가
        lbl.place(x=0, y=0)


def main():
    root = Tk()
    root.title('이미지 보기')
    root.geometry('500x400+10+10')
    myframe = MyFrame(root)
    root.mainloop()

if __name__ == '__main__':
    main()
```

❖ 키 입력 이벤트 처리

```
from tkinter import *


def keyPressed(event):
    # 키보드 문자하나 출력
    print(event.char)


root = Tk()

frame = Frame(root, width=100, height=100)
# Key 이벤트 바인딩
frame.bind('<Key>', keyPressed)
frame.place(x=0, y=0)

# 키보드 포커를 갖게 한다
frame.focus_set()

root.mainloop()
```

이미지 처리

❖ Pillow 모듈

- 파이썬 이미징 라이브러리
- 여러 이미지 파일 포맷 지원
- 이미지 내부 데이타 액세스 및 다양한 이미지 처리 기능 제공
  - Thumbnail 이미지 생성
  - 다른 이미지 포맷으로 변환
  - 이미지 프린트
  - 이미지 크기 변형, 회전, Transform, 그리고 필터링 등 다양한 이미지 프로세싱

- `pip install Pillow`