|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_message\_box.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  win=tk.Tk()  win.title(*"Python GUI"*)  # 새로운 메시지박스 위젯  def **\_msgBox**():  msg.showinfo(*'Python Message Info Box'*, *'A Python GUI created using tkinter:\nThe year is 2017.'* )  # 메뉴 생성  menu\_bar = Menu(win)  win.config(menu=menu\_bar)  # 메뉴 아이템 생성  file\_menu = Menu(menu\_bar,tearoff=0) # 메뉴안에 메뉴를 생성=>(메뉴 안) 아이템칸 생성  file\_menu.add\_command(label=*"New"*) # 아이템 추가  file\_menu.add\_command(label=*"Exit"*,command=win.quit) # 아이템 추가 및 입력 받을 시 콜백함수 호출  file\_menu.add\_separator()  menu\_bar.add\_cascade(label=*"File"*,menu=file\_menu)  help\_menu=Menu(menu\_bar, tearoff=0)  menu\_bar.add\_cascade(label=*"Help"*,menu=help\_menu)  help\_menu.add\_command(label=*"About"*,command=\_msgBox) | | |  | | --- | |  |   메시지 박스 위젯은 말 그대로 메시지 창을 나타내는 위젯이다.  메시지 박스를 만드려면 먼저 tkinter패키지에 있는 messagebox라이브러리를 가져와야 한다. 이후 이 모듈을 쉽게 쓰기 위해서 msg라고 쓰기로 하였다.  이 메시지 박스는 help메뉴 아이템 속성을 봐도 알겠지만 “About”이라는 메뉴 아이템을 클릭하면 command속성에 의해서 \_msgbox콜백함수가 실행되면서 나오게 된다.  콜백함수의 내용은 다음과 같다.  msg모듈의 showinfo(클래스의)생성자를 호출한다. 호출할 때 첫 번째 인수는 메시지 박스의 제목 부분을 요구하는 것이고, 두 번째 인수는 내용 부분을 요구하는 것이다.  위 생성자가 결국 전달된 인수를 바탕으로 메시지 박스 위젯을 나타낸다. |
| |  | | --- | | 사용자 정의 룩앤필GUI\_independent\_msg.py | | from tkinter import messagebox as msg  import tkinter as tk  root=tk.Tk()  root.withdraw() # 룻 폼창 제거  msg.showerror(*'Python Message Error Box'*,*'A Python GUI created using tkinter:\nThe year is 2017.'* )  msg.showwarning(*'Python Message Warning Box '*,*'Python GUI created using tkinter:\nThe year is 2017.'*)  root.mainloop() | | |  | | --- | |  |     메시지 박스 위젯의 종류는 여러가지가 있다. 정보를 알려주는 메시지 창도 있겠지만 에러 창을 띄울 수도 있고 경고 창을 띄울 수도 있는 것이다. 이전 소스에서 (showinfo)생성자 같은 경우, 정보를 알려주는 메시지 창을 담당한다.  왼쪽 소스에 나오는 (showerror)은  (에러 메시지를 띄우는 메시지 박스)를 생성하는 생성자이고,  (showwarning)은 (경고 메시지를 띄우는 메시지 박스)를 생성하는 생성자이다.  이러한 생성자의 호출법은 (showinfo)와 같다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_independent\_msg2.py | | from tkinter import messagebox as msg  import tkinter as tk  root=tk.Tk()  root.withdraw() # 룻 폼창 제거  ret1 =msg.askquestion(*"Askquestion Message Box"*, *"Is it correct?"*)  print(*ret1)*    ret2 =msg.askretrycancel(*"Askretrycancel Message Box"*, *"Is it correct?"*)  print(*ret2*)  root.mainloop() | | |  | | --- | |  |     이전 소스의 메시지 박스는 ‘정보,에러,경고’를 위한 메시지 박스였다. 하지만 우리가 프로그램을 사용하면서 봤겠지만, (예,아니요) 혹은 (다시 시도, 취소) 등 직접 사용자가 상태를 클릭하는 메시지 창도 있다.  왼쪽 소스의 (askquestion)생성자는 [사용자에게 질문을 하고 (예,아니요)버튼 클릭을 요구하는 메시지 박스]를 생성한다.  만약 (예)를 클릭하면 (askquestion)생성자는 문자열’yes’를 리턴하게 되고 ret1에 저장시킨다. 그래서 출력 결과가 ‘yes’가 나오는 것이다. 반대로 (아니요)를 클릭하면 ‘no’가 출력될 것이다.  왼쪽 소스의 (askretrycancel)생성자는 [사용자가 취한 액션이 올바르지 않았을 때, (다시 시도,취소)버튼 클릭을 요구하는 메시지 박스]를 생성한다.  만약 (다시 시도)를 클릭하면 (askretrycancel)생성자는 문자열’True’를 리턴하게 되고 ret2에 저장시킨다. 그래서 출력 결과가 ‘True’가 나오는 것이다. 반대로 (아니요)를 클릭하면 ‘False’가 출력될 것이다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_independent\_msg3.py | | from tkinter import messagebox as msg  import tkinter as tk  root=tk.Tk()  root.withdraw() # 룻 폼창 제거  ret1 =msg.askyesno(*"Askyesno Message Box"*, *"Is it correct?"*)  if ret1 >0:  print(*"RETRY"*)  else:  print(*"CANCEL"*)  ret2 =msg.askokcancel(*"Askokcancel Message Box"*, *"Is it correct?"*)  if ret2 >0:  print(*"RETRY"*)  else:  print(*"CANCEL"*)  root.mainloop() | | |  | | --- | | **추가설명** askyesno생성자로부터 생성된 메시지 박스 위젯(질문 메시지 박스 위젯),  askquestion 생성자로부터 생성된 메시지 박스 위젯(질문 메시지 박스 위젯)이 위젯의 풀네임이다. 하지만 간편함을 위해서  askyesno위젯, askquestio위젯이라고 부르도록 하겠다. |     Msg2소스에서 다루는 메시지 박스가 ‘질문,다시 시도 질문’을 위한 것이였다면,  왼쪽 소스에서 다루는 메시지 박스는  ‘질문,확인 질문’을 위해 있다.  사진에도 보다 싶이 askquestion생성자와 askyesno생성자는 둘 다 ‘질문’을 위한 메시지 창을 생성한다. 하지만 왼쪽 소스에서 버튼 (예)를 클릭한다면 (askyesno) 생성자가 ‘True’를 리턴해서 ret1변수에 저장하게 된다. 컴퓨터에서 ‘True’는 참을 의미함과 동시에 1(정수)로 인식이 된다. 그래서 조건문에서 0과 크기를 비교할 수 있는 것이다.  즉 생성자로부터 리턴된 값(ret1변수 값)을 가지고 상수와 비교할 수 있느냐,없느냐가 askyesno와 askquestion 위젯의 차이가 되겠다.  (Askokcancel)는 ‘확인 질문’ 메시지 박스를 생성하기 위한 생성자이다. 이 위젯도 ‘확인’을 누르면 ret2값이 ‘True’를 갖게 된다.  ‘True’는 0보다 크므로 ‘RETRY’가 출력된 것을 볼 수가 있다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_spinbox.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  # (tab1-라벨프레임 내) 스핀박스  spin=tk.Spinbox(mighty,values=(1,2,4,42,100),width=5,bd=8)  spin.grid(column=0,row=2) | | |  | | --- | |  |     스핀박스 위젯은 콤보박스와 비슷하지만 화살표 버튼을 이용해서 값을 선택한다는 차이점을 지닌다.  이 위젯을 생성하려면 체크버튼이나 라디오버튼처럼 tk라이브러리에서 생성자를 가져와야 한다. 생성자의 이름은 (Spinbox)이다.  왼쪽 소스는 mighty(라벨 프레임 위젯)안에 값들을 (1,2,4,42,100)에서 선택할 수 있고 가로 길이가 5이면서 테두리가 8인 스핀박스 위젯을 생성하는 소스이다.  이후 grid 메소드를 이용해서 3열 1행에 위치하도록 세팅하였다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_spinbox\_small\_two\_ridge.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  def **\_spin**():  value=spin.get()  print(value)  scr.insert(tk.INSERT, value+*'\n'*)  # (tab1-라벨프레임 내) 스핀박스  spin=tk.Spinbox(mighty,values=(1,2,4,42,100),width=5,bd=8,command=\_spin)  spin.grid(column=0,row=2)  spin2=tk.Spinbox(mighty,values=(0,50,100),width=5,bd=9,relief=tk.FLAT) # relief는 스핀박스 테두리를 디자인하는 메소드  spin.grid(column=1,row=2) | | |  | | --- | |  |     \_spin함수는 spin객체의 위젯에서 화살표 버튼이 클릭되었을 때 호출되는 콜백함수이다.  함수에서는 spin객체의 값을 가져와서 value변수에 저장하고,  value값을 출력하고, 스크롤드텍스트 위젯에 value값을 삽입하는 기능을 가진다.  그래서 spin객체의 위젯에서 위쪽 화살표를 클릭 시 1에서 2가 됐고 스크롤드 텍스트에 2가 삽입된다.  같은 방식으로 scr이 참조하는 ScrolledText 클래스의 메소드인 insert를 이용하면 스크롤드텍스트 위젯에 데이터를 삽입할 수 있다.  첫 번째 인수에 (tk.INSERT)(tk모듈의 삽입 함수 호출)을 적고 두 번째 인수에 삽입하고 싶은 데이터를 작성하면 된다.  Spin2객체에서는 relief라는 새로운 속성이 나왔다. 이 속성은 스핀 박스의 테두리 디자인을 담당한다. 등호 뒤에 tk모듈의 선언된 상수를 선택해서 호출하는 방식이다.  여러 상수 중 FLAT을 이용하면 왼쪽 사진과 같이 흰색 테두리 디자인을 볼 수 있다. |
| |  | | --- | | 사용자 정의 룩앤필GUI\_spinbox\_small\_two\_ridge2.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  def **\_spin**():  value=spin.get()  print(value)  scr.insert(tk.INSERT, value+*'\n'*)  # (tab1-라벨프레임 내) 스핀박스  spin=tk.Spinbox(mighty,values=(1,2,4,42,100),width=5,bd=8,command=\_spin, relief=tk.SUNKEN)  spin.grid(column=0,row=2)  spin2=tk.Spinbox(mighty,values=(0,50,100),width=5,bd=9, relief=tk.RAISED) # relief는 스핀박스 테두리를 디자인하는 메소드  spin.grid(column=1,row=2) | | |  | | --- | | 추가설명 tk에서 선언된 상수(GROOVE,RIDGE 등)는 1,2,3등과 같은 변하지 않는 수이다. 하지만 이 상수를 relief속성의 옵션으로 사용하였을 때, 테두리를 디자인할 수 있게 된다. 예를 들어 (relief=1일 때 테두리를 하얗게 한다.)를  (relief=tk.blabla일 때 테두리를 하얗게 한다.) 라고 표현한 것이다. |     relief속성에는 tk.FLAT옵션도 있지만 tk.SUNKEN,tk.RAISED옵션도 있다.  SUNKEN을 상수로 쓰면, 기본 테두리와 똑같고  RAISED을 쓰면, 왼쪽 상위 부분은 밝은 흰색, 오른쪽 하위 부분은 어두운 회색을 담당하게 한다.  - tk.SUNKEN: 기본으로 (테두리) 디자인한다.  - tk.RAISED: 왼쪽 상위 부분은 밝은 흰색, 오른쪽 하위 부분은 어두운 회색을 담당하게 (테두리) 디자인한다.  - tk.FLAT: 흰색으로 (테두리) 디자인한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_spinbox\_small\_two\_ridge.py3.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  def **\_spin**():  value=spin.get()  print(value)  scr.insert(tk.INSERT, value+*'\n'*)  # (tab1-라벨프레임 내) 스핀박스  spin=tk.Spinbox(mighty,values=(1,2,4,42,100),width=5,bd=8,command=\_spin,relief=tk.GROOVE)  spin.grid(column=0,row=2)  spin2=tk.Spinbox(mighty,values=(0,50,100),width=5,bd=9, relief=tk.RIDGE) # relief는 스핀박스 테두리를 디자인하는 메소드  spin.grid(column=1,row=2) | | |  | | --- | |  |     relief속성에는 tk.GROOVE와 tk.RIDGE옵션도 있다.  GROOVE상수를 쓰면, 흰색 파인 테두리로 디자인한다.  RIDGE상수를 쓰면, 흰색 튀어나온 테두리로 디자인한다.  - tk.SUNKEN: 기본으로 (테두리) 디자인한다.  - tk.RAISED: 왼쪽 상위 부분은 밝은 흰색, 오른쪽 하위 부분은 어두운 회색을 담당하게 (테두리) 디자인한다.  - tk.FLAT: 흰색으로 (테두리) 디자인한다.  - tk.GROOVE: 흰색으로 파이게 (테두리) 디자인한다.  - tk.RIDGE: 흰색으로 튀어나오게 (테두리) 디자인한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 사용자 정의 룩앤필GUI\_title.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  from tkinter import messagebox as msg  win=tk.Tk()  win.title(*"Python GUI"*)  win.iconbitmap(*'pyc.ico'*)  (~)  win.mainloop() | | |  | | --- | |  |     파이썬 윈도우 폼창의 타이틀 옆에 보면 이미지가 있다. 이 이미지도 메소드를 통해 변경할 수 있다.  iconbitmap이라는 Tk클래스의 메소드를 이용하면 괄호 안에 있는 내용으로 이미지를 변경할 수가 있다.  대신 소스를 작성하기 전에, 해당 이름의 이미지 파일을 파이썬 폴더에 미리 넣을 필요성이 있다. |

**사용자 정의 룩앤필**

* **새로운 위젯 및 속성 정리**

**메시지 박스 위젯: 프로그램에서 ‘메시지 창’을 나타냄**

* **정보 메시지 박스 위젯:** 정보를 알려주는 메시지 창이다.
* **오류 메시지 박스 위젯:** 오류가 있다고 알려주는 메시지 창이다.
* **경고 메시지 박스 위젯:** 경고문을 나타내는 메시지 창이다.
* **askquestion 위젯**: 사용자에게 질문을 하는 메시지 창이다.
* **askyesno위젯:** 사용자에게 질문을 하는 메시지 창이다.
* **askokcancel 위젯:** 사용자에게 확인 질문을 하는 메시지 창이다.
* **askretrycancel 위젯:** 사용자가 상태를 클릭하는 메시지 창이다.
* **askyesnocancel 위젯:** 사용자에게 질문을 하는 메시지 창인데, ‘취소’버튼도 있다.

P.s. 모든 메시지 박스 위젯은 모달이다. 모달은 자식 윈도우 폼(ex)메시지창)이 꺼질 때까지 부모 윈도우 폼(ex)윈도우창)을 끄게 할 수 없는 것을 말한다.

**스핀박스 위젯:** 값을 선택할 수 있는 위젯으로 콤보박스와 비슷하나, 화살표 버튼 두 개로만 값을 선택할 수 있다.

**bd속성:** 스핀박스 위젯에서 테두리의 크기를 결정하는 속성이다.

**relief 속성:** 스핀박스 위젯에서 테두리의 디자인을 결정하는 속성이다. 디자인은 tk모듈에 선언된 상수를 옵션으로 사용하여 설정한다.

- tk.SUNKEN: 기본으로 (테두리) 디자인한다.

- tk.RAISED: 왼쪽 상위 부분은 밝은 흰색, 오른쪽 하위 부분은 어두운 회색을 담당하게 (테두리) 디자인한다.

- tk.FLAT: 흰색으로 (테두리) 디자인한다.

- tk.GROOVE: 흰색으로 파이게 (테두리) 디자인한다.

- tk.RIDGE: 흰색으로 튀어나오게 (테두리) 디자인한다.