|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_menubar.py | | from tkinter import Menu  # 메뉴 생성  menu\_bar = Menu(win)  win.config(menu=menu\_bar)  file\_menu = Menu(menu\_bar) # 메뉴안에 메뉴를 생성=>기존 메뉴는 메뉴 틀이 됨!  menu\_bar.add\_cascade(label=*"File"*,menu=file\_menu)  file\_menu.add\_command(label=*"New"*) # 아이템 추가  # 이벤트순환문 실행  win.mainloop() | | |  | | --- | | 추가 설명 cascade메소드를 이용해서 상위메뉴와 하위메뉴를 연결하지 않으면, 메뉴의 틀만 있고 실질적으로 메뉴는 없게 된다.  즉 gui폼 창에 나타나지 않으므로 꼭 메소드를 호출해야만 한다!! |   메뉴 위젯을 생성하려면 새로운 클래스가 필요하다. 그래서 Menu클래스를 먼저 가져왔다.  이후 Menu생성자를 이용해 win안에 메뉴를 만들었다. 이것을 munu\_bar라는 변수에 저장하였다.  Config는 환경 설정 메소드로써, menu=menu\_bar를 인수로 전달하면  (메뉴를 win폼 창에 있는 메뉴로 인지한다)  즉 (메뉴를 GUI용 메뉴로 설정하겠다!)라는 의미이다.  앞서 만든 메뉴안에 같은 방식으로 메뉴를 만들었다. 이럴 경우, 기존 메뉴는 메뉴를  담는 틀이 되고 틀 안에 메뉴가 생기게 된다. 새로 만든 메뉴는 file\_menu라는 변수에 저장하였다.  add\_cascade메소드는 상위 메뉴(틀)와 하위 메뉴(새로운 메뉴)를 연결해주는 기능을 한다. 첫 번째 인수는 gui폼에 나타나는 하위 메뉴 이름이고 두 번째 인수는 연결하려는 하위 메뉴를 의미한다.  add\_command메소드는 메뉴에 아이템(항목)을 추가하는 기능을 한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_mennubar\_exit.py | | # 메뉴 생성  menu\_bar = Menu(win)  win.config(menu=menu\_bar)  # 메뉴 아이템 생성  file\_menu = Menu(menu\_bar,tearoff=0) # 메뉴안에 메뉴를 생성=>기존 메뉴는 메뉴 틀이 됨!  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"*)  # 이벤트순환문 실행  win.mainloop() | | |  | | --- | |  |   노란색 강조된 부분 중 첫 번째 소스를 보면 괄호 안에 command속성이 있는 것을 알 수가 있다.  command속성은 액션 시 콜백함수를 호출하는 기능이 있다. 이 속성으로 quit메소드를 불러와서 win(윈도우 폼 창)을 꺼지게 할 수 있다.  두번째 소스는 add\_seperator이라고 하는 메소드로써 메뉴 안에 선을 긋게 하는 기능을 한다.  네번째부터 여섯번째 소스는 새로운 메뉴를 생성한 뒤 상위 메뉴(틀)과 연결한 후,  새 메뉴에 아이템을 생성한다는 것이다.  새 메뉴는 help\_menu에 저장되었다. |

|  |  |
| --- | --- |
|  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_tabbed\_two.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  from tkinter import Menu  win=tk.Tk()  win.title(*"Python GUI"*)  # 탭 생성에 필요한 탭 컨트롤 생성  tabControl=ttk.Notebook(win)  # 새로운 탭(탭1) 추가  tab1=ttk.Frame(tabControl)  tabControl.add(tab1, text=*"Tab1"*)  # 새로운 탭(탭2) 추가  tab2=ttk.Frame(tabControl)  tabControl.add(tab2, text=*"Tab2"*)  # grid속성과 비슷, fill은 제공된 공간을 점유하도록 지정, expand는 여분의 공간을 채우기 위해 확장할건지  tabControl.pack(expand=1,fill=*"both"*)  win.mainloop() | | |  | | --- | | 추가 설명 pack메소드를 써서 크기를 조정하는 tab위젯은 pack이 없으면 크기가 조정이 불가능하니 폼 창에 나타나지 않게 된다. |   탭은 창을 담당한다. 탭이 여러 개 있다는 것은 여러 창이 있다고 생각하면 된다.  먼저 notebook생성자를 이용해서 win(윈도우창)에 탬 컨트롤을 생성하였다. 이는 쉽게 생각해서 탭들을 탐는 틀이라고 할 수 있다.  이 탭 틀은 tabcontrol변수에 저장되었다.  이후 Frame생성자를 이용해서  위젯을 담을 수 있는 틀을 생성하였다.  이 틀은 add메소드를 이용해서 탭에 담기고 탭 틀에 추가되게 된다. add메소드의 text속성을 이용하여 윈도우 창에 나타나는 탭의 이름도 설정할 수 있는데  왼쪽 프로그램에서는 ‘Tab1’이라는 이름으로 설정하였다.  pack메소드는 패키징한다는 것인데  위젯을 열과 행으로 관리하는 기능을 하며,  탭의 크기는 이 메소드로 설정이 된다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_tabbed\_all\_widgets.py | | # tab1에 라벨프레임 생성  mighty= ttk.Labelframe(tab1,text=*"Mighty python"*)  mighty.grid(column=0, row=0,padx=8,pady=4)  # (tab1-라벨프레임 내) 라벨 생성  a\_label=ttk.Label(mighty,text=*"Enter a name:"*)  a\_label.grid(column=0, row=0,sticky=*'W'*)  # (tab1-라벨프레임 내) 콤보박스 위젯  number=tk.StringVar()  number\_chosen=ttk.Combobox(mighty,width=12,textvariable=number, state= *'readonly'*)  number\_chosen[*'values'*]=(1,2,4,42,100)  number\_chosen.grid(column=1,row=1)  number\_chosen.current(0)  (…) | | |  | | --- | |  |   라벨프레임 생성자,콤보 박스 생성자 등을 호출할 때 첫 번째 인수는 위젯이 생성되는 장소를 의미한다.  라벨프레임 같은 경우 tab1이라고 수정되었으니, ‘탭1’이름의 탭에 존재할테고  콤보박스 같은 경우 mighty(라벨 프레임 객체)이라고 작성되었으니 라벨 프레임 내에 존재한다.  즉 여러 위젯들의 위치가 탭1 내 mighty객체의 (mighty변수가 담고 있는)라벨 프레임 안에 있다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_tabbed\_all\_widgets\_both\_tabs.py | | # tab2에 라벨프레임 생성  mighty2= ttk.LabelFrame(tab2,text=*"The Snake"*)  mighty2.grid(column=0, row=0,padx=8,pady=4)    # (tab2-라벨프레임 내) 체크버튼 위젯  chVarDis=tk.IntVar()  check1 = tk.Checkbutton(mighty2, text=*"Disabled"*,variable=chVarDis,state=*"disabled"*)  check1.select()  check1.grid(column=0, row=0, sticky=tk.W)  chVarUn=tk.IntVar()  check2 = tk.Checkbutton(mighty2, text=*"UnChecked"*,variable=chVarUn)  check2.deselect()  check2.grid(column=1, row=0, sticky=tk.W)  chVarEn=tk.IntVar()  check3 = tk.Checkbutton(mighty2, text=*"Enabled"*,variable=chVarEn)  check3.select()  check3.grid(column=2, row=0, sticky=tk.W)  (…) | | |  | | --- | |  |   위와 같은 방식인데 이번에는 체크 버튼과 라디오 버튼,라벨 프레임이 mighty2객체의 라벨 프레임에 존재한다.  Mighty2객체는 tab2에 생성된 라벨 프레임 위젯을 담고 있는 변수이다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | 레이아웃 관리GUI\_tabbed\_all\_widgets\_both\_tabs\_radio.py | | # (tab2-라벨프레임 내) 라디오버튼 위젯  colors=[*"Blue"*,*"Gold"*,*"Red"*]  def **radCall**():  redsel=radVar.get()  mighty2.configure(text=colors[redsel]) # 라디오버튼 클릭시 라벨프레임 텍스트 변경  radVar = tk.IntVar()  radVar.set(99)  for col in range(3):  curRad=tk.Radiobutton(mighty2,text=colors[col], variable =radVar,value=col, command=radCall)  curRad.grid(column=col, row=1, sticky=tk.W)  (…) | | |  | | --- | |  |   Configure은 속성 변경 메소드로서 왼쪽 소스에서는 mighty2의 text속성을 변경하는 의미를 가진다.  즉 라디오버튼이 클릭될 때,  Radcall 콜백함수가 호출되고  1. 라디오버튼 위젯에 바인딩된 radvar의 값을 불러와서 radsel변수에 저장한다.  2.redsel 변수 값에 따라 mighty2객체의 라벨프레임 텍스트를 변경시킨다.  결론적으로,  Blue라디오버튼 클릭 시 라벨 프레임 텍스트는 Blue  Gold 라디오 버튼 클릭 시 라벨 프레임 텍스트는 Gold  Red라디오 버튼 클릭 시 라벨 프레임 텍스트는 Red로 바뀐다. |

=위젯 및 속성 정리=

Tab: ‘창,화면’을 담당하는 위젯. 탭이 많을수록 한 프로그램에서 여러 창을 담고 있다!!

* state-disabled: tab을 표시할 수 없다
* state-hidden: tab이 표시되지 않는다.
* text: 탭에 표시되는 텍스트
* image: 탭에 표시할 이미지
* underline: 텍스트 문자열에 밑줄을 표시하기 위한 문자의 인덱스를 지정

Menu: 프로그램의 추가 기능을 담당하는 위젯, 예를 들어 ‘프로그램 종료, 옵션 gui폼 생성 등’ 을 담고 있는 위젯이다.

* activebackground: 마우스가 있을 때, 선택을 표현하는 배경색
* activeforeground: 마우스가 있을 때, 선택을 표현하는 글자색
* background: 마우스가 위에 없을 때 배경색
* bd: 모든 선택을 표시하는 테두리 폭
* fg/foreground: 마우스가 위에 없을 때 글자색
* image:menubutton위에 표시되는 이미지
* selectcolor:선택되었을 때 체크버튼이나 라디오 버튼에 표시되는 색상
* tearoff:점선을 표시 유무

pack: 위젯의 열과 행을 관리(크기를 관리)하는 속성이다.

* fill: 제공된 공간(남은 공간)을 점유하도록 할 것인지 지정한다.
* expand: 여분의 공간을 채우기 위해 확장할 것인지 지정한다
* side: 위젯을 위치시킬 측명을 지정한다.