|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기First\_GUI.py | | import tkinter as tk  win=tk.Tk()  win.title("Python GUI")  win.mainloop() | | |  | | --- | | 설명란Tk는 tkinter속 클래스이다. 이를 인스턴스화시킬 시 생성자가 호출되어 폼 창을 생성시키는 것이다. 하지만 매번 이렇게 생각하면 어려우니, Tk(클래스의)생성자를 호출하여 폼 창이 나왔다고 생각하자!! |   파이썬 프로그램은 tkinter모듈을 내장하고 있다. tkinter모듈은 GUI를 가능하게 하는 파이썬 파일이다.  즉 사용자는 이 모듈로부터 파이썬 프로그램을 그래픽으로 사용할 수 있다는 것이다. 이는 CLI같이 텍스트를 쳐서 프로그램을 다루는게 아니다. 버튼을 클릭하거나 스크롤을 내릴 수 있는 등  좀 더 획기적으로 프로그램을 다룰 수 있다.  첫 행은 tkinter라는 모듈을 불러오고 이 모듈을 tk라고 간단히 부르겠다는 의미이다.  이 후 win변수를 레퍼런스 변수로 두고 tk모듈에 있는Tk클래스를 인스턴스화시켰다. 이후 mainloop 메소드를 호출하면 실행결과창이 나오게 되는데 이 창을 ‘윈도우’라고 부른다.  Tk인스턴스는 윈도우를 생성하고 mainloop메소드는 윈도우를 유지시킨다고 생각하면 된다. |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기First\_GUI\_2.py | | from tkinter import \*  win=Tk()  win.title("Python GUI")  win.mainloop() | | |  | | --- | | 설명란Win.title()은 사실상 레퍼런스 변수 win이 참조하는 Tk인스턴스 안에 있는 메소드를 불러오는 것이다. 하지만 매번 이렇게 생각하면 어려우니 (win객체(윈도우창)에 title메소드를 이용하여 기능을 한다.)라고 생각하자!! |   앞 결과를 보면 알겠지만 title메소드는 윈도우에서 ‘상단바’의 내용을 결정한다. 그리고 전달받은 인수에 맞추어서 상단바의 내용을 작성한다.  First\_GUI 파일과 다른 점은 tkinter에서 모든 함수나 클래스 등을 가져온다는 점인데 이 덕분에 Tk클래스가 인스턴스화되는 과정에서 ‘tk.’이 생략될 수 있다.  그리고 내용적 의미는 거의 유사하니 같은 실행결과가 나오게 된다.  또한 mainloop메소드는 이벤트순환문을 실행시키는 함수인데 이는 무한 루프를 돌면서 발생될 이벤트를 계속 모니터링한다. 이벤트가 발생하면 해당 이벤트에 해당하는 함수를 호출하기도 한다. 콘솔 프로그램과 달리 사용자가 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_not\_resizable.py | | import tkinter as tk  win=tk.Tk()  win.title("Python GUI")  win.resizable(False, False)  win.mainloop() | | |  | | --- | | 설명란 |   어떤 행위를 할지 모르기 때문에 이런식으로 디자인이 되었다.  ----------------------------------------------  노란색으로 강조된 소스는 윈도우의 크기를 제어하는 소스이다.  resizable이라는 메소드를 통해서 제어가 가능한데 첫 번째 인수는 윈도우의 x축,두 번째 인수는 윈도우의 y축을 결정한다.  다만 이 소스에서는 둘 다 false로 작성되어 있으니 x축,y축 모두 크기를 제어할 수가 없다.  실행결과를 보면 ‘확대’버튼이 막혀 있는 것을 알 수가 있다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_add\_lable.py | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  ttk.Label(win,text="A Label") .grid(column=0,row=0)  win.mainloop() | | |  | | --- | | 설명란Label생성자도 역시 **Label(클래스의)생성자라고 하는 것이 사실상 맞다.** |   tkinter패키지에 있는 ttk모듈을 불러왔다.  ttk모듈은 gui의 룩앤필을 향상시킨다.  즉 이 모듈이 있어서 GUI를 더 꾸밀 수 있고 멋지게 만들 수 있다.  첫 번째 예시가 바로 라벨이다.  ttk모듈의 Label생성자는 인스턴스를 전달받고 text소스를 전달받았다.  결과적으로 보면 윈도우에 텍스트가 삽입된 것을 알 수가 있는데 모듈의 정확한 소스는 알 수가 없다. 하지만 win변수를 다르게 해석하면 (mainloop메소드에 의해서 나타나는) 윈도우를 담고 있는 변수이니 윈도우나 다름이 없고 뒤에 텍스트 소스가 있으니 (Label생성자를 이용해서 윈도우에 텍스트가 생기게 한다.) 라고 이해할 수 있다. 그리고 grid 괄호 안에 있는 값은 텍스트의 위치, 즉 열과 행을 결정한다! |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_create\_button\_change\_property.py | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  a\_label=ttk.Label(win,text="A Label")  a\_label.grid(column=0,row=0)  def click\_me():  action.configure(text="\*\*I have beeen clicked \*\*")  a\_label.configure(foreground='red')  a\_label.configure(text="A Red Label")  action=ttk.Button(win,text="Click Me!", command=click\_me)  action.grid(column=1, row=0)    win.mainloop() | | |  | | --- | | 설명란 |   노란색 강조라인의 7번째 줄은 action객체가 정의되어 있다. (ttk모듈의 버튼 생성자를 이용하여 윈도우에 Click Me!라고 작성된 버튼을 생성한다. 이 때 버튼이 눌려지면 click\_me콜백함수를 호출한다.)라는 의미로 action객체는 윈도우의 버튼이 저장된 변수임을 알 수 있다. grid메소드 괄호안에 열이 1이라고 되있는건 버튼의 위치를 결정해준다.  configure메소드는 위젯(버튼이나 스크롤,체크박스 따위)을 설정하는 기능을 하며 텍스트 문자열이나 글자 색 등을 조정할 수 있다.  만약 윈도우에서 버튼을 누르면 버튼의 텍스트 문자열은 “I have beeen clicked”이 될 것이고 윈도우 안의 텍스트는 빨간 색으로 되면서 “A Red Label”로 글씨가 바뀔 것이다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_textbox\_widget.py | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  a\_label=ttk.Label(win,text="A Label")  a\_label.grid(column=0,row=0)  def click\_me():  action.configure(text="Hello"+name.get())  ttk.Label(win,text="Enter a name:").grid(column=0, row=0)  name= tk.StringVar()  name\_entered = ttk.Entry(win, width=12, textvariable=name)  name\_entered.grid(column=0, row=1)  action=ttk.Button(win, text="Click Me!", command=click\_me)  action.grid(column=1, row=1)  win.mainloop() | | |  | | --- | | 설명란 |     name= tk.StringVar()은 변수 선언이다. (name이라는 변수에 문자열 데이터만 저장할 수 있게 생성한다.)라는 의미가 된다.  ttk모듈의 Entry생성자 도움을 받아 윈도우 창에 너비12의 입력창이 생기고 name변수는 entry위젯에 바인딩한다.  버튼이 클릭되면 click\_me함수가 호출되는데, 버튼의 설정 중 텍스트 문자열을 수정한다.  name.get() => name에 바인딩된 엔트리 위젯의 값을 가져온다.  실행결과는 입력창(엔트리 위젯)에 아무 기입도 안하고 버튼을 누르면 버튼의 텍스트가 Hello로 바뀐다. |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_set\_focus.py | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  a\_label=ttk.Label(win,text="A Label")  a\_label.grid(column=0,row=0)  def click\_me():  action.configure(text="Hello"+name.get())  ttk.Label(win,text="Enter a name:").grid(column=0, row=0)  name= tk.StringVar()  name\_entered = ttk.Entry(win, width=12, textvariable=name)  name\_entered.grid(column=0, row=1)  action=ttk.Button(win, text="Click Me!", command=click\_me)  action.grid(column=1, row=1)  action.configure(state=’disabled’)  name\_entered.focus()  win.mainloop() | | |  | | --- | | 설명란위젯에 포커스 설정하고 비활성화하기 |   Name\_entered변수는 entry위젯을 나타낸다. 이를 포커스하면 entry위젯에 입력하라는 표시가 보이게 된다.  action변수는 button위젯을 나타낸다. 만약 configure로 설정을 변경하는데 상태, 즉 state를 disabled로 만들면 버튼은 누를 수 없게 된다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_combobox\_widget.py | | name= tk.StringVar()  name\_entered = ttk.Entry(win, width=12, textvariable=name)  name\_entered.grid(column=0, row=1)  action=ttk.Button(win, text="Click Me!", command=click\_me)  action.grid(column=2, row=1)  ttk.Label(win,text="Choose a number").grid(column=1,row=0)  number=tk.StringVar()  number\_chosen=ttk.Combobox(win,width=12,textvariable=number)  number\_chosen['values']=(1,2,4,42,100)  number\_chosen.grid(column=1,row=1)  number\_chosen.current(0)  name\_entered.focus()  win.mainloop() | | |  | | --- | | 설명란 |   number변수는 문자열 데이터를 저장하는 변수이다. ttk모듈의 콤보박스 생성자로부터 너비12의 콤보박스를 만들고 number를 콤보박스 위젯에 바인딩한다.  콤보박스의 값들은 1,2,4,42,100으로 정한다.  콤보박스의 위치는 열이 1이고 행도 1인 곳으로 설정한다,  current메소드를 통해 콤보박스의 현재 값을 0으로 설정한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_combobox\_widget.py | | def click\_me():  action.configure(text="Hello"+name.get())  print(number.get())  print(type(number.get())) | | |  | | --- | | 설명란 (콤보박스의 반환값이 스트링으로 자동타입되는지 확인하기) |   버튼이 클릭되고 click\_me 콜백함수가 실행되면 버튼의 텍스트를 Hello+엔트리값으로 바꾼다.  파이썬 쉘 창에는 넘버 변수에 바인딩된 콤보박스의 값을 출력하고 그 때의 데이터 타입을 출력한다. |

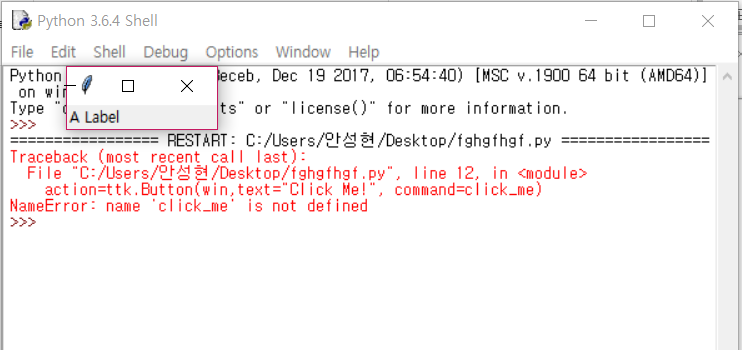
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_combobox\_widget\_randonly\_plus\_display\_number.py | | def click\_me():  action.configure(text="Hello"+name.get()+''+number\_chosen.get())  ttk.Label(win,text="Enter a name:").grid(column=0, row=0)  name= tk.StringVar()  name\_entered = ttk.Entry(win, width=12, textvariable=name)  name\_entered.grid(column=0, row=1)  action=ttk.Button(win, text="Click Me!", command=click\_me)  action.grid(column=2, row=1)  ttk.Label(win,text="Choose a number").grid(column=1,row=0)  number=tk.StringVar()  number\_chosen=ttk.Combobox(win,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) | | |  | | --- | | 설명란 |     콤보 박스에서 선택한 값을 버튼에 나타나도록 설정한다.  state상태는 readonly로 맞추어서 생성자에게 전달한다.  그러면 콤보 박스의 값들을 새로 입력하여  추가할 수 없게 된다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_checkbutton\_widget.py | | chVarDis=tk.IntVar()  check1 = tk.Checkbutton(win, text="Disabled",variable=chVarDis,state="disabled")  check1.select()  check1.grid(column=0, row=4, sticky=tk.W)  chVarUn=tk.IntVar()  check2 = tk.Checkbutton(win, text="Disabled",variable=chVarUn)  check2.deselect()  check2.grid(column=1, row=4, sticky=tk.W)  chVarEn=tk.IntVar()  check3 = tk.Checkbutton(win, text="Disabled",variable=chVarEn)  check3.select()  check3.grid(column=2, row=4, sticky=tk.W)  name\_entered.focus()  win.mainloop() | | |  | | --- | | 설명란 |   chvardis변수를 정수형 데이터를 저장하는 변수로 선언한다. tk모듈의 checkbutton생성자로부터 윈도우에 비활성화된 체크버튼을 생성한다.  체크버튼을 나타내는 변수는 세 개나 되는데 그 이유는 체크버튼 각각이 다른 의미를 지니고 있기 때문이다.  tk.W라는 것은 정렬을 서쪽기준으로 해야 한다는 의미이다.  chvardis변수는 체크박스 위젯에 바인딩하였다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_radiobutton\_widget.py | | COLOR1 = "Blue"  COLOR2 = "Gold"  COLOR3 = "Red"  def radCall():  redsel=radVar.get()  if redsel == 1:win.configure(background=COLOR1)  elif redsel == 2:win.configure(background=COLOR2)  elif redsel == 3:win.configure(background=COLOR3)  radVar = tk.IntVar()  rad1 = tk.Radiobutton(win, text=COLOR1, variable=radVar,value=1, command=radCall)  rad1.grid(column=0,row=5,sticky=tk.W)  rad2 = tk.Radiobutton(win, text=COLOR2, variable=radVar,value=2, command=radCall)  rad2.grid(column=1,row=5,sticky=tk.W)  rad3 = tk.Radiobutton(win, text=COLOR3, variable=radVar,value=3, command=radCall)  rad3.grid(column=2,row=5,sticky=tk.W)  name\_entered.focus()  win.mainloop() | | |  | | --- | | 설명란 |   Color1을 파랑, color2를 골드, color3을 빨강으로 정의한다.  tk모듈의 radiobutton 생성자를 이용해 윈도우에  라디오버튼을 생성하게 한다.  라디오버튼 또한 각각의 의미가 있어 지정한 변수(rad1,2,3)이 다른 것을 알 수가 있다.  라디오 버튼을 누르면 콜백함수 radcall이 호출되는데 만약 첫 번째 라디오버튼을 클릭하면 배경색이 color1이 저장된 파란색이 된다.  마찬가지로 두 번째 라디오버튼과 세 번째 라디오 버튼을 클릭하면 배경색이 각각 골드색과 빨간 색으로 변하게 된다. |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기GUI\_scrolledtext\_widget.py | | import tkinter as tk  from tkinter import ttk  from tkinter import scrolledtext  scrol\_w = 30  scrol\_h =3  scr = scrolledtext.ScrolledText(win, width=scrol\_w, height=scrol\_h, wrap=tk.WORD)  scr.grid(column=0, columnspan=3)  name\_entered.focus()  win.mainloop() | | |  | | --- | | 설명란 |   tkinter모듈 패키지에는 scrolledtext라는 라이브러리가 있다. 이 모듈의 기능은 말 그대로 스크롤이 가능하게 .하는 것이다.  Scroll\_w변수에 30을 저장하고 h변수에는 3을 저장한다.  scrolledText생성자를 이용해 윈도우 창에 큰 입력창을 만드는데 너비가 30, 높이가 3,단어별로 줄 바꿈이 되도록 만든다.  grid메소드로부터 위치를 결정하는데  1열부터 존재하며 3칸의 열을 차지하도록 설정한다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기<<추가실습.1>> | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  def click\_me():  action.configure(text="\*\*I have beeen clicked \*\*")  a\_label.configure(foreground='red')  a\_label.configure(text="A Red Label")  a\_label=ttk.Label(win,text="A Label")  a\_label.grid(column=0,row=0)  action=ttk.Button(win,text="Click Me!", command=click\_me)  action.grid(column=1, row=0)    win.mainloop()  (실행결과: GUI\_create\_button\_change\_property.py와 같음!!) | | |  | | --- | | 설명란Click\_me콜백함수를 라벨 선언 이전으로 옮겨도 되는가? |   옮겨도 결과는 같게 나온다.  함수는 소스에 작성되어 있다고 바로 실행되는 것이 아니라 호출이 되어야 실행이 되는 구조이다. Click\_me함수는 버튼이 눌렸을 때 이벤트로 실행이 되는데  그 호출문이 있기 전에action과 label에 대한 정의가 모두 되어 있어 오류가 걸릴 이유가 없다. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | GUI폼 만들기 및 위젯 추가하기<<추가실습.2>> | | import tkinter as tk  from tkinter import ttk  win=tk.Tk()  win.title("Python GUI")  a\_label=ttk.Label(win,text="A Label")  a\_label.grid(column=0,row=0)  action=ttk.Button(win,text="Click Me!", command=click\_me)  action.grid(column=1, row=0)    def click\_me():  action.configure(text="\*\*I have beeen clicked \*\*")  a\_label.configure(foreground='red')  a\_label.configure(text="A Red Label")  win.mainloop()  (실행결과: 다음 페이지) | | |  | | --- | | 설명란Click\_me콜백함수를 버튼 선언 이후로 옮겨도 되는가? |   옮기면 결과는 같게 나올 수 없다.  함수가 정의되기 이전에 함수 호출이 일어날 수도 있기 때문에 오류가 생긴다.  기본적으로 함수는 호출문 위에 있는 것이 올바르다. |



* 콜백함수란 무엇인가?

콜백함수는 액션을 취했을 때 호출되는 함수이다. 예를 들어서 버튼을 누르면 텍스트가 화면에

나오거나 체크박스를 선택하면 버튼의 텍스트가 바뀌는 것 등이 있다.

* 이벤트 핸들링이란 무엇인가?

버튼이 클릭되거나 체크버튼이 클릭되는 일 등을 이벤트라고 하는데 이벤트가 일어나면 특정 코드를 실행하는 경우가 있다. 이러한 경우를 이벤트 핸들링이라고 한다.

< 위젯 정리 & 속성 정리 >

1. 라벨 위젯 6. 라디오버튼 위젯
2. 버튼 위젯 7. 스크롤드텍스트 위젯
3. 엔트리 위젯(한 줄 빈칸) (스크롤 있는 여러 줄 빈칸)
4. 콤보박스 위젯
5. 체크버튼 위젯
6. 라벨 위젯 속성

=>text: 위젯에 문자열을 작성하는 속성

1. 버튼 위젯 속성

=>text: 위젯에 문자열을 작성하는 속성

Command: 입력을 하면 함수를 호출하는 속성

1. 엔트리 위젯 속성

=>width: 위젯의 크기 중 너비를 결정하는 속성

Variable: 위젯이 (등호 뒤에 있는) 변수로 바인딩되게 하는 속성

1. 콤보박스 위젯 속성

=>width: 위젯의 크기 중 너비를 결정하는 속성

Variable: 위젯이 (등호 뒤에 있는) 변수로 바인딩되게 하는 속성

1. 체크버튼 위젯 속성

=>text: 위젯에 문자열을 작성하는 속성

Variable: 위젯이 (등호 뒤에 있는) 변수로 바인딩되게 하는 속성

1. 라디오버튼 위젯 속성

=>text: 위젯에 문자열을 작성하는 속성

Variable: 위젯이 (등호 뒤에 있는) 변수로 바인딩되게 하는 속성

Command: 입력을 하면 함수를 호출하는 속성

1. 스크롤드텍스트 위젯 속성

=>width: 위젯의 크기 중 너비를 결정하는 속성

Height: 위젯의 크기 중 높이를 결정하는 속성

Wrap: 압력창에서 줄 바꿈을 결정하는 속성