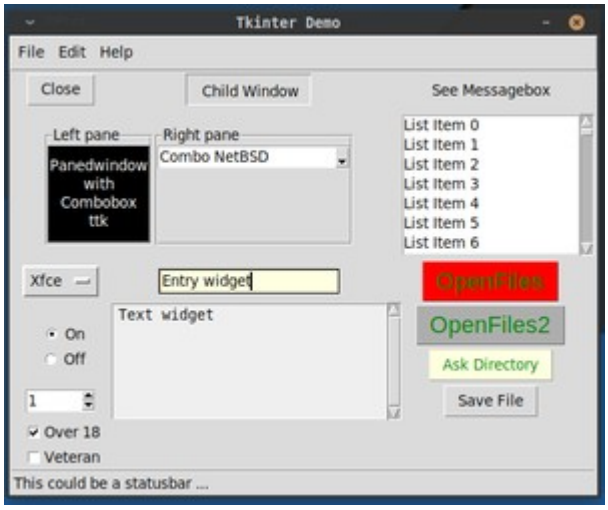
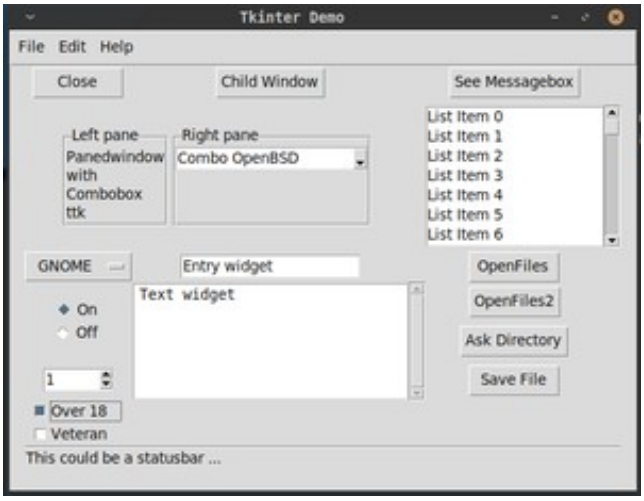
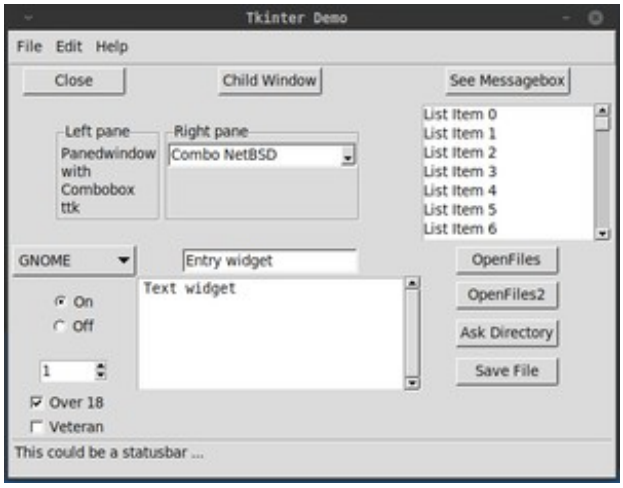


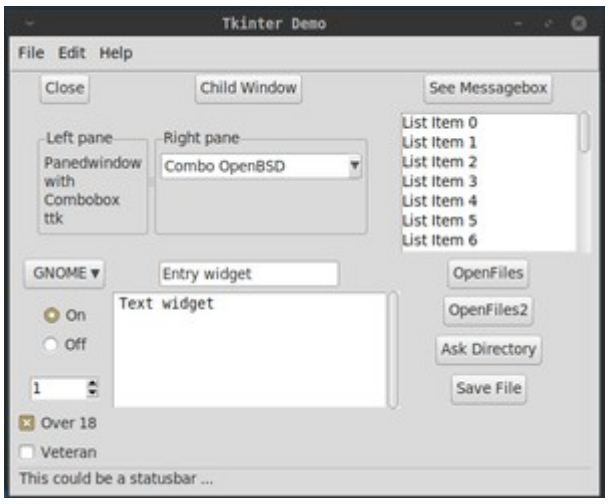
Tkinter GUI render with **no Theme support**



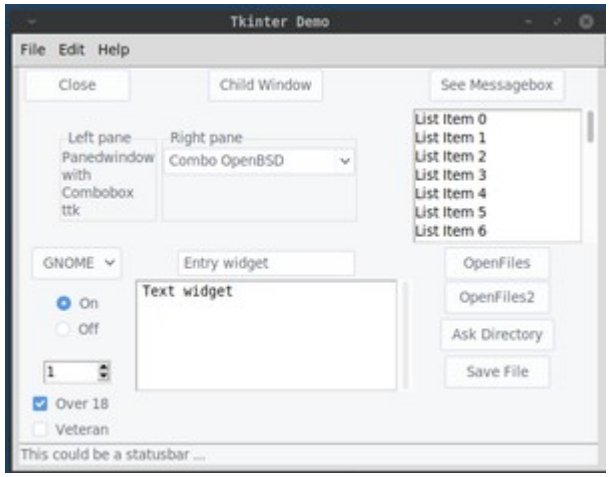
Tkinter ttk with Theme: "**default**"



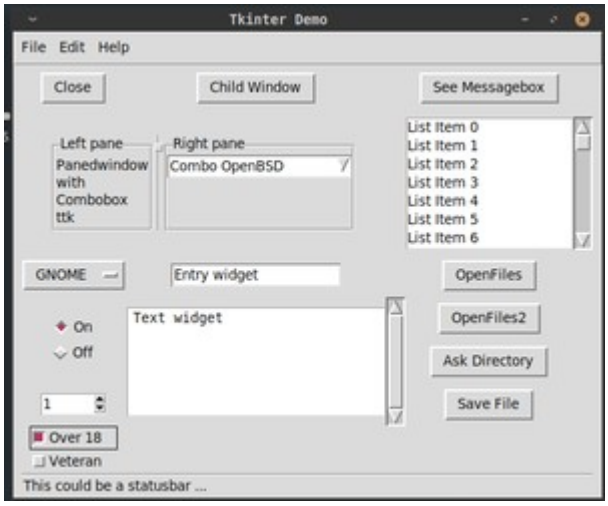
Tkinter ttk with Theme: "**alt**"



Tkinter ttk with Theme: "**scidsand**"

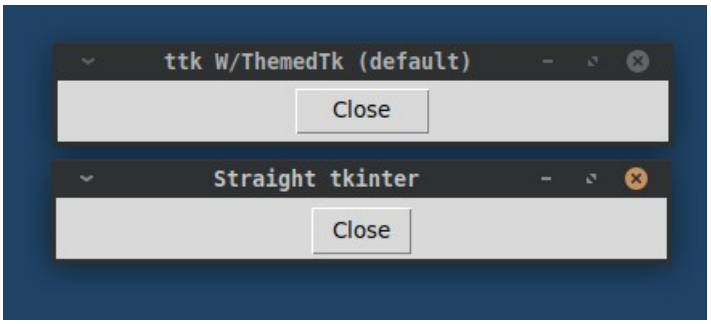


Tkinter ttk with Theme: "**arc**"



Tkinter ttk with There: "**classic**"

Tkinter ttk with Theme: "**clam**"



Python and tkinter "one button" GUI

The simple "one button" GUI on the right shows the output of the code comparisons below. Specifically it shows how easy it is to add "ttkthemes" to an existing tkinter project.

```
1   # tkeasy.py
2
3   from tkinter import *
4
5   class Application(Frame):
6       ''' use oop format for GUI program '''
7       def __init__(self, master=None):
8           super().__init__(master)
9           self.pack()
10          self.create_widgets()
11
12      def create_widgets(self):
13          ''' define widgets and show '''
14          btn1 = Button(self, text="Close", command=exit)
15          btn1.grid(row=0,column=0, padx=5, pady=5)
16
17  root = Tk()
18  root.geometry("380x38") # WxH+left+top
19  root.title("Straight tkinter")
20  app = Application(master=root)
21  app.mainloop()
22
```

First above is the code for a straight tkinter 'one button' GUI in Python.

Here you can see the code for the same program but with a theme added.

```
1   # tkeasyx.py
2
3   from tkinter import *
4   from tkinter.ttk import *  # defaults all widgets as ttk
5   from ttkthemes import ThemedTk  # module applied to all widgets
6                                   # pip install ttkthemes
7
8   class Application(Frame):
9       ''' use oop format for GUI program '''
10      def __init__(self, master=None):
11          super().__init__(master)
12          self.pack()
13          self.create_widgets()
14
15      def create_widgets(self):
16          ''' define widgets and show '''
17          btn1 = Button(self, text="Close", command=exit)
18          btn1.grid(row=0,column=0, padx=5, pady=5)
19
20  # 'alt', 'scidsand', 'classic', 'scidblue',
21  # 'scidmint', 'scidgreen', 'default', 'scidpink',
22  # 'arc', 'scidgrey', 'scidpurple', 'clam'
23  root = ThemedTk(theme="default")
24  # root = Tk()
25  root.geometry("380x38") # WxH+left+top
26  root.title("ttk W/ThemedTk (default)")
27  app = Application(master=root)
28  app.mainloop()
29
```

Notice that two more "`import`" statements have been added (lines 4-5) and the "`root = ...`" on line 23 has been modified. That is all that is necessary to convert a tkinter GUI into a themed tkinter.ttk GUI.

The advantage of using ttkthemes is not so much eye candy as achieving a more standarized look and feel among all of the GUI widgets. Also using a theme may eliminate the need for many additional attributes for each widget in order to achieve a unified *look and feel*.

Next I'll show how you can tweek individual widget classes even after using a theme.

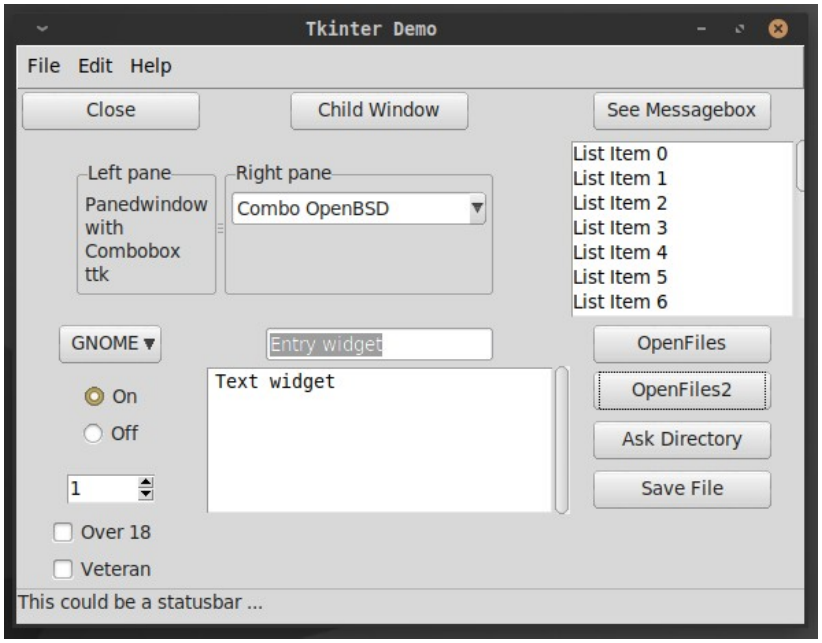
You may notice that the "`clam`" sample theme provides a unified width for Buttons. However, the "`scidsand`" theme example does not standarize the Button width. I really like the "`scidsand`" theme however I also like the uniform Button width as seen in the "`clam`" theme.

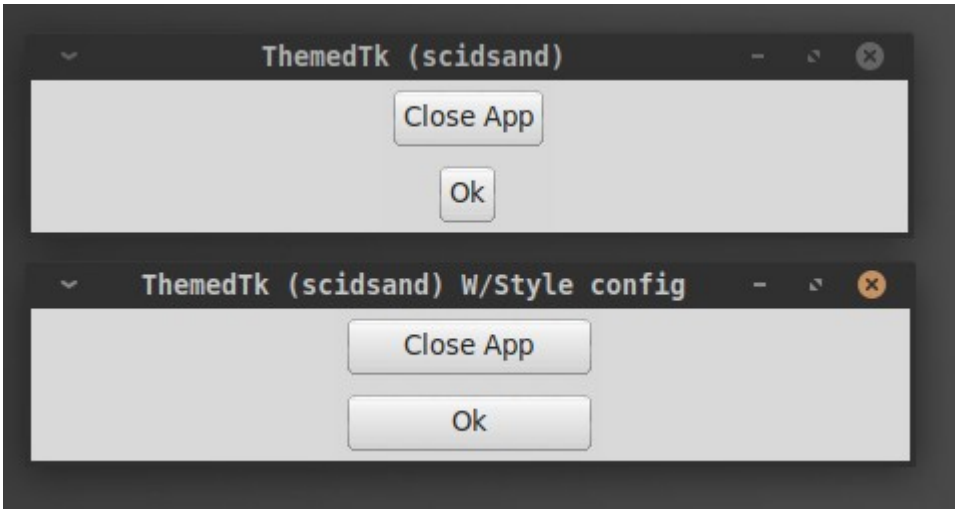
There is an easy way to take care of the Button width problem so that I can still use the "`scidsand`" theme.

```
28          style = Style()
29          style.configure("TButton", width=15)
30
```

By using the `configure` method on the `Style` class you can alter the GUI of an entire class of widgets (like Button widget.) This is possible because we're using ttk. The only purpose of the ttk and ttkthemes modules is for enhanced styling. So basically using ttkthemes provides "styling" with less work. Personally I like the way themes standardize the look and feel more than the actual styles.

So, here is the "`scidsand`" theme sample from above with the `style.configure..` code added to standardize the width of the Buttons.





Remember the style.configure method can use any of the allowed attributes for any of the ttk widget classes (which is all of the tkinter widgets and then some.) In my example above '`TButton`' Style denotes the `Button` class. Below is a list of all the Style Names.

Widget class	Style name
Button	**TButton**
Checkbutton	**TCheckbutton**
Combobox	**TCombobox**
Entry	**TEntry**
Frame	**TFrame**
Label	**TLabel**
LabelFrame	**TLabelFrame**
Menubutton	**TMenubutton**
Notebook	**TNotebook**
PanedWindow	**Tpanedwindow**
Progressbar	**Horizontal.Tprogressbar** or **Vertical.TProgressbar**, depending on the orient option.
Radiobutton	**TRadiobutton**
Scale	**Horizontal.Tscale** or **Vertical.TScale**, depending on the orient option.
Scrollbar	**Horizontal.Tscrollbar** or **Vertical.TScrollbar**, depending on the orient option.
Separator	**TSeparator**
Sizegrip	**TSizegrip**
Treeview	**Treeview**