PAGE 7.5 Widget Demo README file

Normally, for a simple demo, I wouldn’t do much more than document the program using comments in the code file, but I’ve done so many “strange” things in this demo, I figured I would need to add a README file, to help PAGE users understand some of the “stranger” things that I’ve done.

Why would I write such a thing?

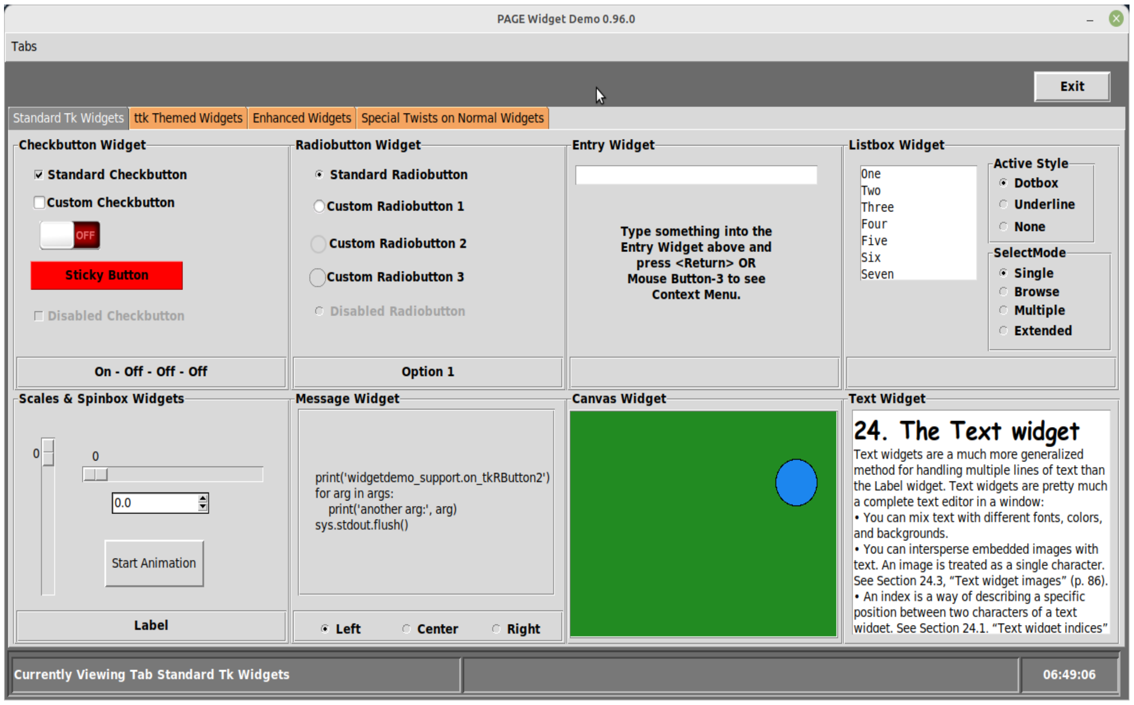
Over the past 5 years, I’ve been helping Don Rozenberg test upcoming releases of PAGE, trying to break each one. For the most part, I think I’ve done a pretty good job, but sometimes I miss some things. Over the years, I’ve changed the way I test and came up with creating a widget demo, to test the “normal” way that a user would use PAGE. I’ve found that this, at least for me, makes more sense than the “slap and tickle” way I used to test. Once I got to the latest version of PAGE, I couldn’t help myself and added a bunch of other things to show off some of the things that can be done with PAGE. In many ways, this is way “over the top”, but I couldn’t help myself.

Copying and Moving The Code

If you plan to copy, modify and/or move the code, feel free. However, you will need to make sure that the images folder and the file ScrolledCheckedListBox.py are both included. The ScrolledCheckedListBox.py file is required to support the special custom widget I created, and the images folder is required for various special widgets and other portions of the demo.

Screenshots

Let’s start with a series of screenshots of the demo program.



Type something into the Entry Widget above and press <Return> OR Mouse Button-3 to see Context Menu.

```
print('widgetdemo_support.on_tkRButton2')
for arg in args:
    print('another arg:', arg)
sys.stdout.flush()
```

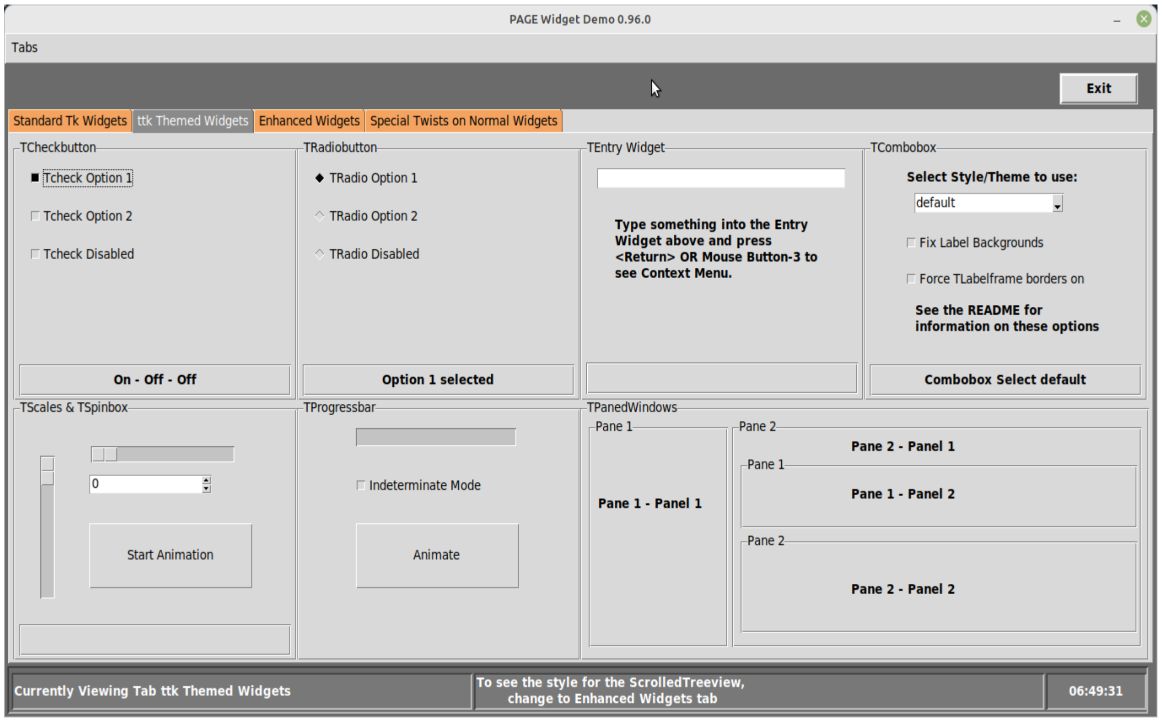
24. The Text widget

Text widgets are a much more generalized method for handling multiple lines of text than the Label widget. Text widgets are pretty much a complete text editor in a window:
• You can mix text with different fonts, colors, and backgrounds.
• You can intersperse embedded images with text. An image is treated as a single character. See Section 24.3, "Text widget images" (p. 86).
• An index is a way of describing a specific position between two characters of a text widget. See Section 24.1, "Text widget indices"

Currently Viewing Tab Standard Tk Widgets

06:49:06

This is the way the program will start up. I decided to use a ttk TNotebook to show the various types of widgets. The first tab shows all of the "standard" Tk widgets.



Type something into the Entry Widget above and press <Return> OR Mouse Button-3 to see Context Menu.

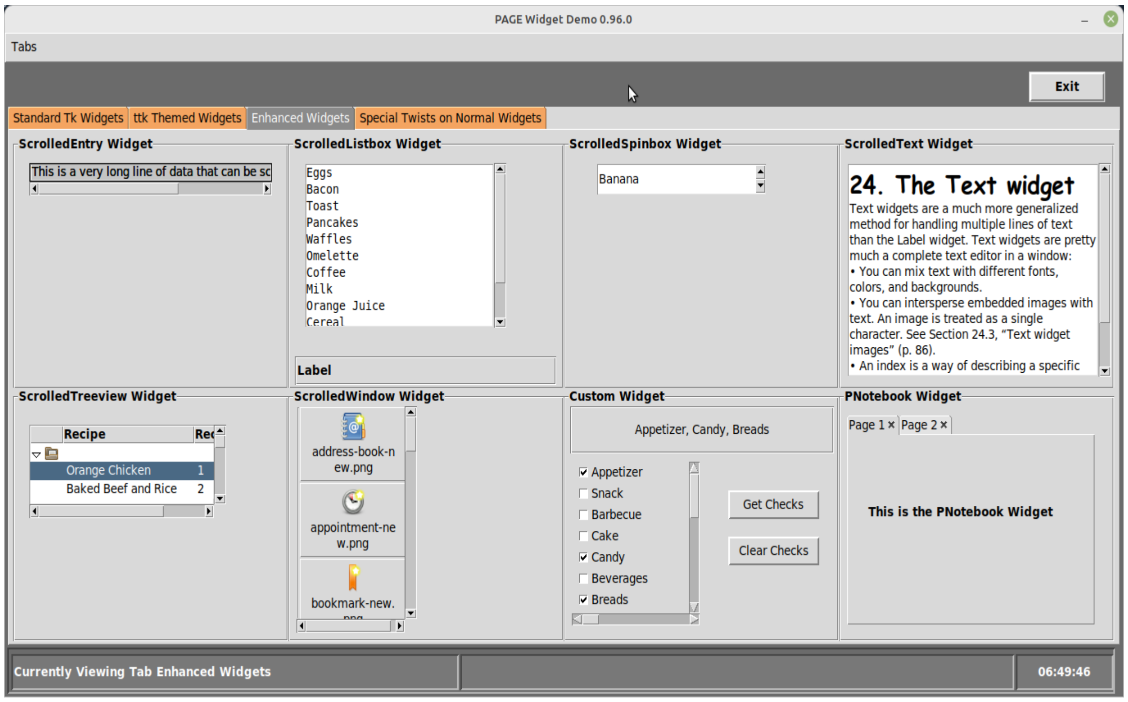
See the README for information on these options

Currently Viewing Tab ttk Themed Widgets

To see the style for the ScrolledTreeview, change to Enhanced Widgets tab

06:49:31

The second tab shows all of the ttk Themed widgets. In addition you can see how the widgets react to different themes. To get the most out of the demo, you should consider installing the **ttkthemes** library which adds a number of pretty nice themes. You can use pip it install it. You can go to https://ttkthemes.readthedocs.io/en/latest/installation.html for the instructions and documentation.



24. The Text widget

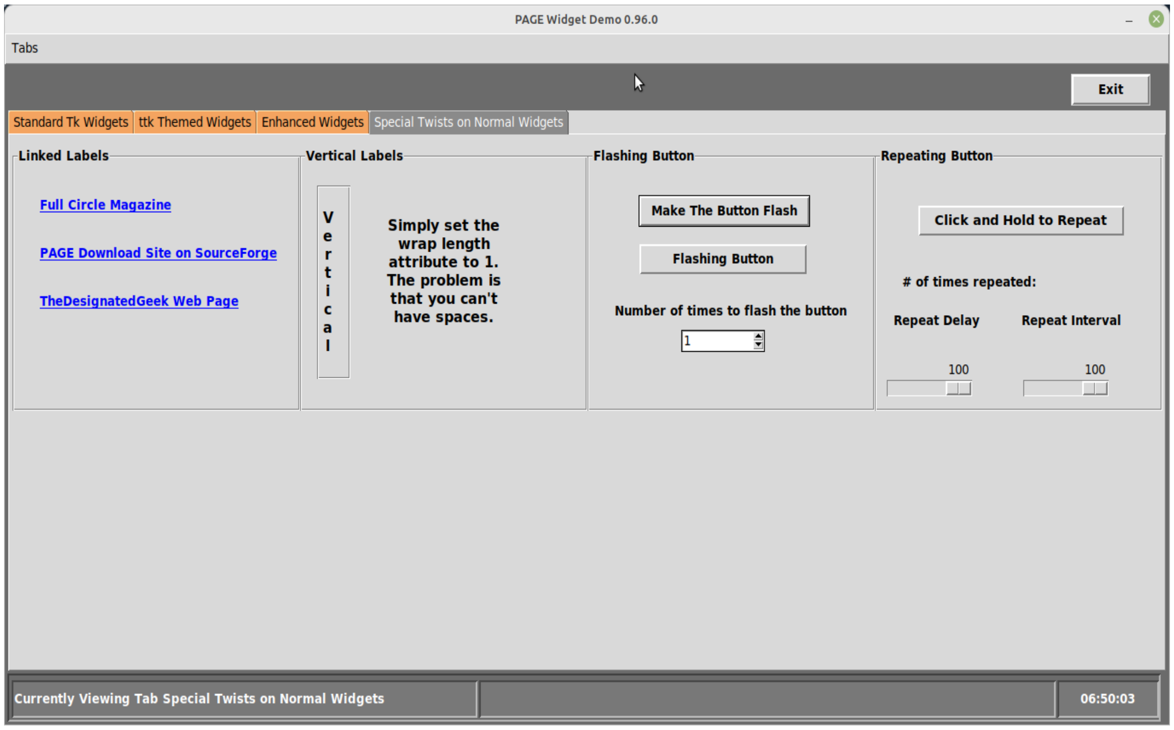
Text widgets are a much more generalized method for handling multiple lines of text than the Label widget. Text widgets are pretty much a complete text editor in a window:

• You can mix text with different fonts, colors, and backgrounds.

• You can intersperse embedded images with text. An image is treated as a single character. See Section 24.3, "Text widget images" (p. 86).

• An index is a way of describing a specific

The third tab contains the Enhanced widgets including the Custom Widget that allows you to add third party (or widgets you have created). Most of the Enhanced widgets are based on existing widgets but add and enhance them by using scrollbars. A prime example is the ScrolledListbox. I use this all the time instead of the standard Tk Listbox widget, since I always have more items in my Listbox than will show on the single screen. Yes, you can use the scroll wheel or up/down arrows to move through the list of items in a "normal" Listbox, but I like the ability to have scrollbars.



Simply set the wrap length attribute to 1. The problem is that you can't have spaces.

Currently Viewing Tab Special Twists on Normal Widgets

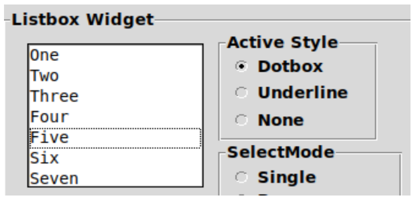
Finally the fourth tab shows some neat things that you can do with “normal” Tk widgets that many users don’t know about. Things like creating a vertical label or Hyperlink Labels from plain Tk Labels.

The real reason behind this document is to provide a bit of extra information that I wanted to provide where things are not quite clear, and didn’t want to force the user of the demo program to dig through over 1200 lines of code to find just one thing. That brings us to the initial reason for a README, which is about the styles/themes on the ttk Themed widget page (second tab).

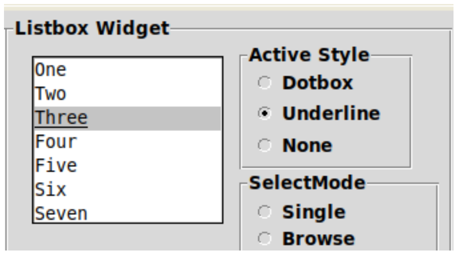
Tk Listbox

One of the things that I spent a good amount of time on is enhancing the Tk Listbox demo. I’ve added two additional sets of controls to see what happens when you use the ActiveStyle attribute and the SelectMode attribute. Many PAGE users overlook these attributes that can extend the usability of the Tk Listbox.

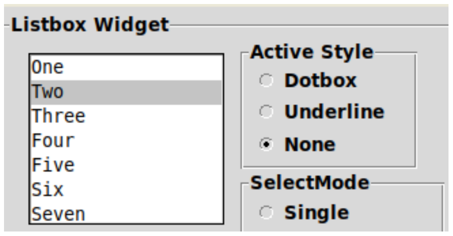
The ActiveStyle attribute allows you to change the way the Selected item in the Listbox. There are three options available. They are Dotbox, Underline and None. Let’s take a look at each.



The Dotbox style shows the selection with a box created out of dots. This is the default.



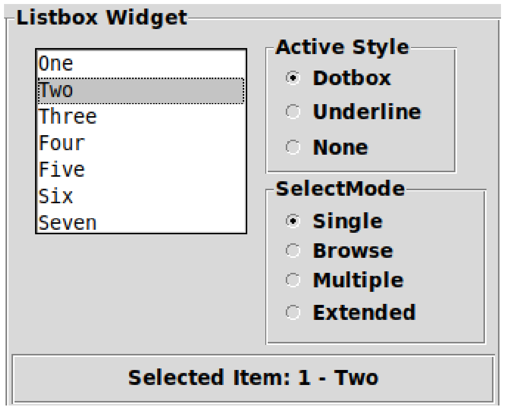
The Underline style shows the selected item with the text underlined.



The None mode simply shows a gray bar assuming the Listbox has focus.

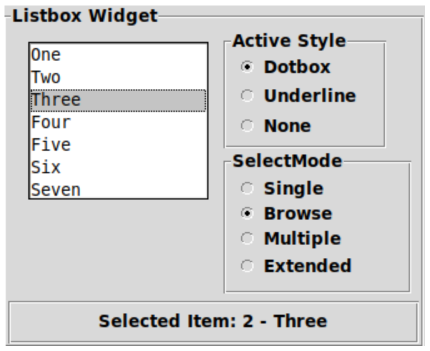
The SelectMode is another overlooked attribute of the Tk Listbox. This attribute set allows you to select only a single item or multiple items at a time. The options are Single, Browse, Multiple and Extended with Browse being the default.

In the Single mode, you can only select one item in the listbox at a time and you can’t drag the mouse to make selections.



Selected Item: 1 - Two

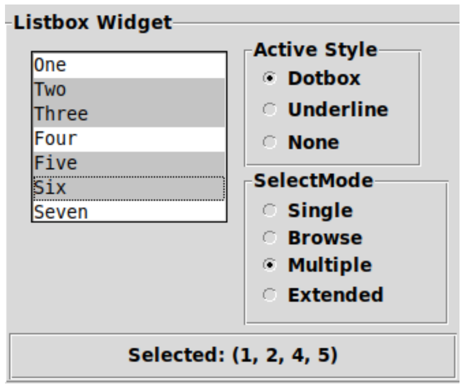
In the Browse mode, you can only select one item at a time, but you can select a single item and drag mouse to select a different item.



Active Style: Dotbox [✓]
SelectMode: Browse [✓]

Selected Item: 2 - Three

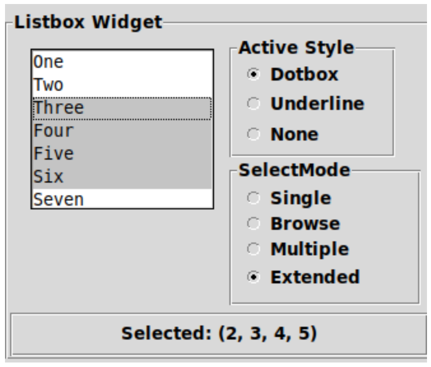
In the Multiple mode, you can select multiple items by clicking on any item you wish. If an item is already selected and you click on it, it will deselect it without "disturbing any of the other selected items.



Active Style: Dotbox [✓]
SelectMode: Multiple [✓]

Selected: (1, 2, 4, 5)

In the Extended mode, you can select a group of items that are next to each other. You start the selection by clicking on the first item you want and then drag the mouse (with the mouse button still clicked to select the group you want. Once you let up of the mouse button, the selection process stops.



Active Style: Dotbox [✓]
SelectMode: Extended [✓]

Selected: (2, 3, 4, 5)

Anytime you want to get the selected item(s), Python returns the information as a tuple.

To obtain the selected item you would use…

```
indx = _w1.Listbox1.curselection()
```

So if you are in the Single or Browse mode, the returned information would look like this…

```
 (1,)
```

If you are in the Multiple or Extended mode. The returned information would look like this…

```
(1, 2, 5)
```

If you want to also get the text of the item, you would use the following code, but only if you are in the Single or Browse modes.

```
        print(f"Selected Item: {indx[0]} - {itm}")
        print(f'Index: {indx}')
```

Which returns:

```
Selected Item: 1 - Two
Index: (1,)
```

Remember that all the returned index numbers are zero based, so item 1 is actually the second item in the listbox.

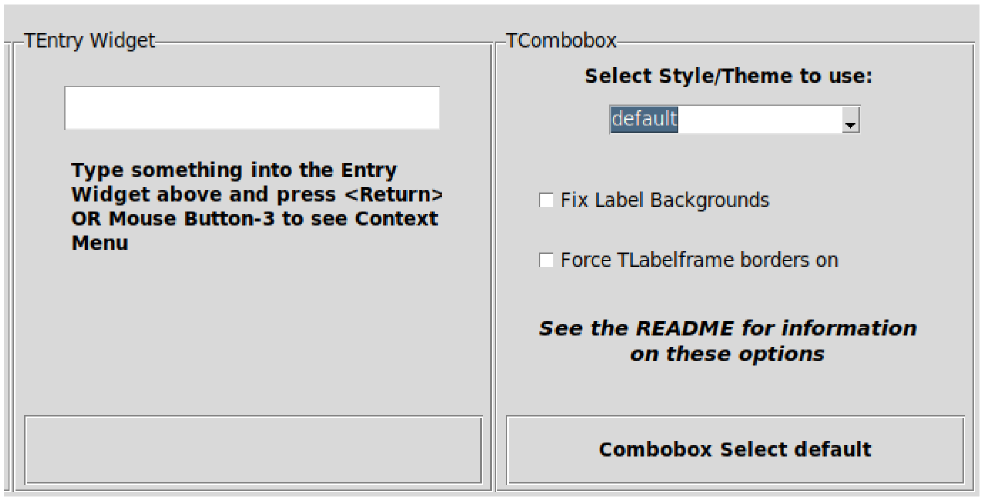
Styles and Themes issues

Most users of Tkinter (or Tcl/Tk for that matter) either love or hate Styles and Themes. Styles change the look of a single ttk widget. Themes contain multiple (usually) Styles. The reasons that users would dislike them are pretty clear.

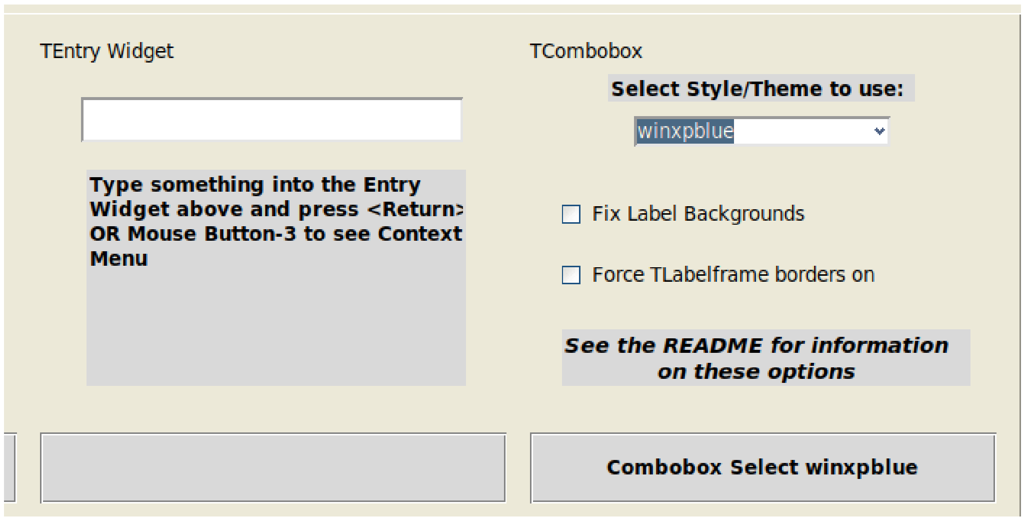
- They are pretty much undocumented.
- They are inconsistent, many times using Tk widgets as the base widget.
- Much of the documentation is just plain wrong if it exists at all.
- Many of them are flat out ugly.

I fall into a middle ground when it comes to Styles and Themes. They are pretty much a necessary evil in that there are some widgets, like the combo box, just isn’t available in the standard Tk widget set. And the biggest complaint about Tkinter is that is looks so old school, like something written for Windows 95. That seems to be the reason for Themes to begin with.

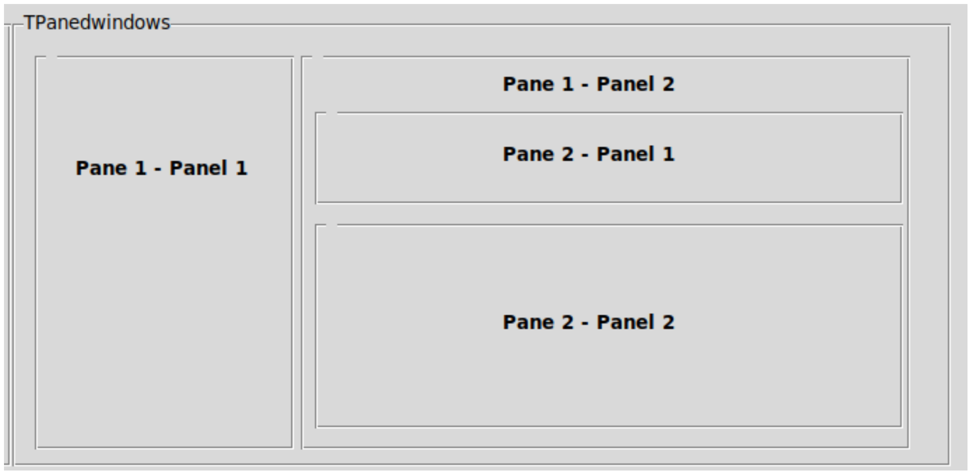
Anyway, many of the Themes, don’t bother to deal with the background for the TLabel widgets. Others don’t set the borders for TLabelframes. This makes things look really strange. For Linux there are four “standard” Themes which are, alt, default, clam and classic. Windows adds a couple of extra themes that aren’t available on other Operating systems and I’m not sure what is available for Mac, since I don’t have one. Of the four that are available for Linux, they all handle things nicely and include support for all the widgets. However, when you expand your Theme set is when you get into trouble. Here is an example.


TEntry Widget
Type something into the Entry Widget above and press <Return> OR Mouse Button-3 to see Context Menu
TCombobox
Select Style/Theme to use:
default
Fix Label Backgrounds
Force TLabelframe borders on
See the README for information on these options
Combobox Select default

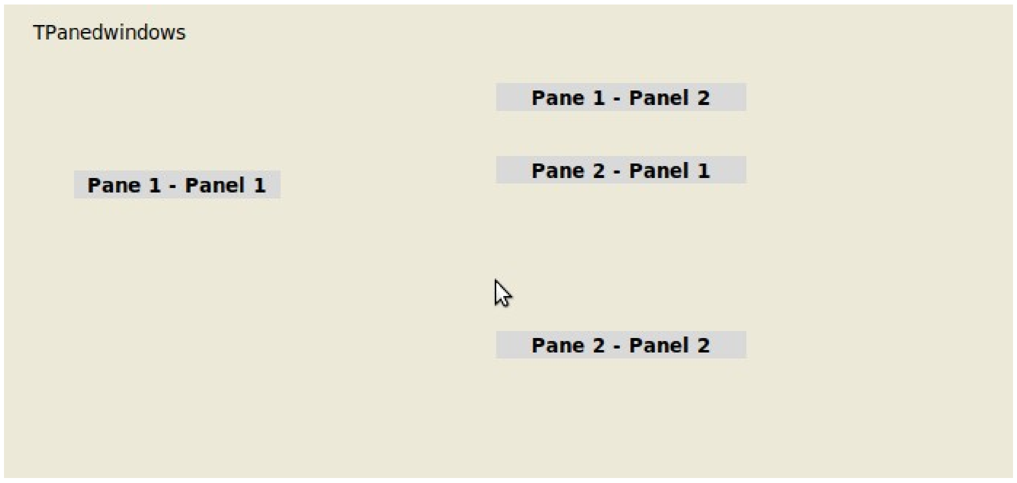
This is the way I designed the Second tab (which uses the Default Theme). You can see that each of the Widget types are within a TLabelframe and it’s very obvious which widgets belong to which group. Now look at one of the nicer Themes available from the **TTKTheme** package called WinXPBlue.


TEntry Widget
Type something into the Entry Widget above and press <Return> OR Mouse Button-3 to see Context Menu
TCombobox
Select Style/Theme to use:
winxpblue
Fix Label Backgrounds
Force TLabelframe borders on
See the README for information on these options
Combobox Select winxpblue

Not only is it very clear that the TLabel widgets have the “default” background color, but the borders of the TLabelframes are non-existent. That makes things (in my mind) pretty ugly and not really worth using the ttk widgets at all. To make matters worse, take a look at the next two images. The first is a set of Tpanedwindows as a demonstration using the “default” Theme.

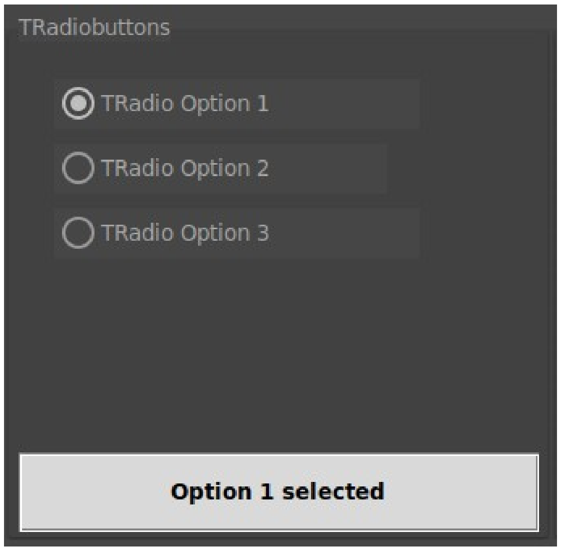


You can see that the TLabel backgrounds are taken care of and the borders around the panes are also the way you would expect them. Now back to the WinXPBlue Theme.



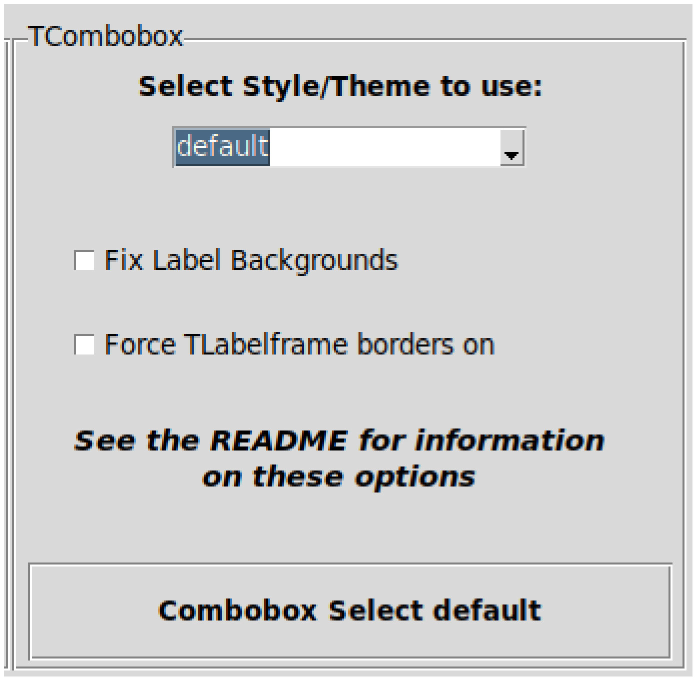
No borders at all and the TLabels again don't get set to the default background for the Theme.

As an extreme example of what could be a very nice dark Theme that wasn't well thought out is the Equilux Theme. Here's a screenshot of the TRadiobutton TLabelframe using this Theme.

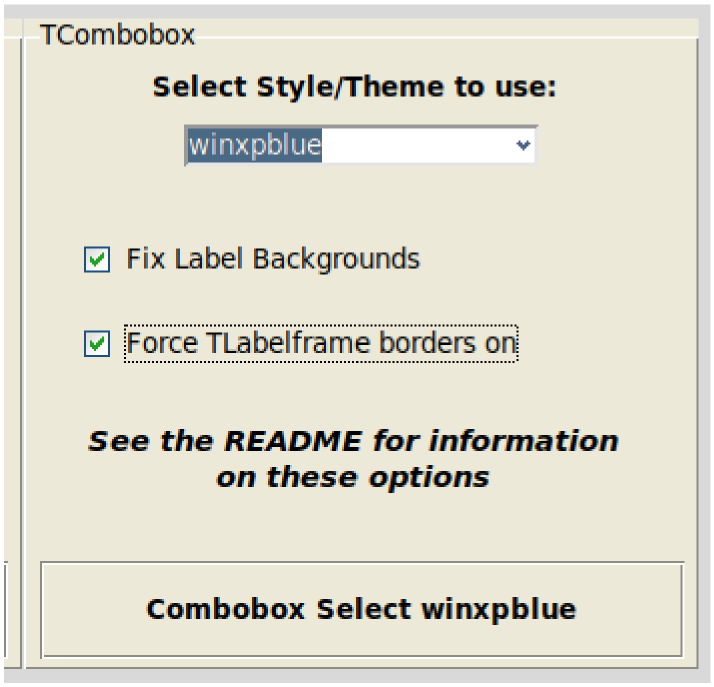


One would think that if you are going to set backgrounds for the various widgets, you would set it to be consistent with the container (TLabelframe). Also notice that the TLabel widget didn't get the background set either. While I could have made the widths of the TRadiobuttons consistent, I did this to point out the fact that the background attribute probably should be the same as the background of the container widget.

So, in order to combat this, I spent a great deal of time coming up with a work around to fix both the background of TLabels and separately a work around to fix the Border issue.


TCombobox
Select Style/Theme to use:
default
Fix Label Backgrounds
Force TLabelframe borders on
See the README for information on these options
Combobox Select default

In the TCombobox TLabelframe, I added two TCheckbuttons, one to “fix” the background issues and one to “fix” the Borders on the TLabelframes.


TCombobox

Select Style/Theme to use:

winxpblue

✓ Fix Label Backgrounds

✓ Force TLabelframe borders on

See the README for information on these options

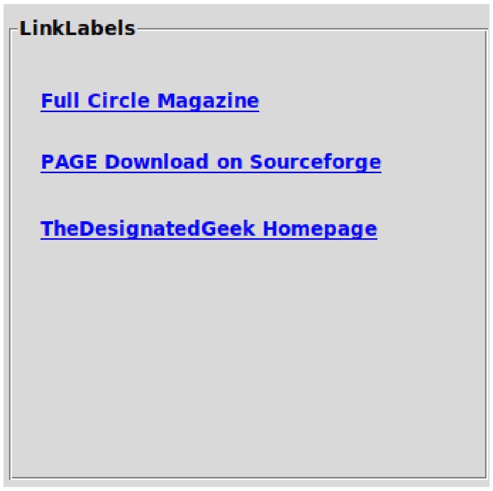
Combobox Select winxpblue

You can see that when they are checked, the workaround goes into effect.

Special Twists on Normal Widgets

The demos on this tab are things that I felt most PAGE users didn’t know about, so I threw together a few examples.

First, we have the LinkLabel Widgets, which really are standard Tk Label Widgets.



To create one is simply to place the Label widget where you want it and set the color and font the way you want it. Typically a hyperlink is done in blue and is sometimes underlined, so that's what I tried to emulate. Remembering the Alias that you set for the label, in your Support module, you only need to do a couple of things. First is to import the webbrowser library. I usually place this with all the other imports at the top of the file. Then in your startup function, bind the <Button-1> mouse button to that label and assign a callback.

```
 _w1.Label10.bind("<Button-1>", lambda e: LabelLinkClick(1))
```

I have three of these lines, one for each link. The parameters are set to 1, 2 and 3.

Finally, you create your callback (in this case **LabelLinkClick**) and use the following code

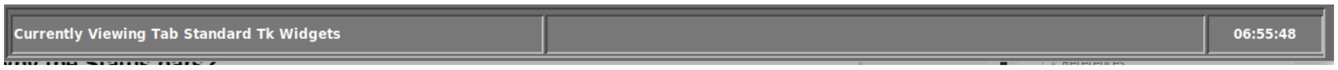
```
def LabelLinkClick(*args):
    print('Into LabelLinkClick')
    for arg in args:
        print(arg)
    which = args[0]
    print(which)
    if which == 1:
        url = "https://fullcirclemagazine.org"
    elif which == 2:
        url = "https://sourceforge.net/projects/page/"
    elif which == 3:
        url = "https://thedesignatedgeek.xyz"
    webbrowser.open_new_tab(url)
```

The important thing here is to assign a url and send it as a parameter to the **`webbrowser.open_new_tab()`** or **`webbrowser.open_new()`** method.

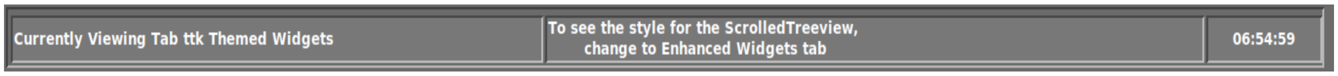
That’s it. No need to go looking for a special hyperlink supporting widget.

Why the Status bars?

Status bars can provide the user valuable information about the status of your program at any time. It can show what the program is doing and/or simple information such as the time of day.


Currently Viewing Tab Standard Tk Widgets
06:55:48

In the first Status bar image you see that the left status box shows which TNotebook tab the user is currently viewing.


Currently Viewing Tab ttk Themed Widgets
To see the style for the ScrolledTreeview,
change to Enhanced Widgets tab
06:54:59

In the second Status bar image, the center status box shows some extra information.

Status bars are easy to create, simple place standard Tk Label widgets within a Tk Frame. Set the Text Var attribute of each Label to a variable that you control to display whatever information you need to pass on at the time.