From The Amazon #1
Best-Selling Author


# TKINTER WIDGET

# QUICK REFERENCE GUIDE

JOHN ELDER

**ALSO BY JOHN ELDER**

*Intro To Python Programming: Beginners Guide Series*

*Intro To Ruby Programming: Beginners Guide Series*

*Learn Ruby On Rails For Web Development*

*PHP Programming For Affiliate Marketers*

*The Smart Startup: How To Crush It*
*Without Falling Into The Venture Capital Trap*

*Adsense Niche Sites Unleashed*

*Social Media Marketing Unleashed*

*SEO Optimization: A How To SEO Guide*
*To Dominating The Search Engines*

# TKinter Widget Quick Reference Guide

John Elder

Tkinter.com
Las Vegas

# TKinter Widget
# Quick Reference Guide

By: John Elder

Published By Codemy.com, Inc.
Las Vegas, NV USA

ISBN
979-8-9859654-2-1

First Edition



https://www.Tkinter.com

FOR APRIL

# TABLE OF CONTENTS

# TKINTER WIDGET QUICK REFERENCE GUIDE

## ABOUT THE AUTHOR

John Elder is a Web Developer, Entrepreneur, and Author living in Las Vegas, NV. He created one of the earliest online advertising networks in the late nineties and sold it to publicly traded WebQuest International Inc. at the height of the first dot-com boom.

He went on to develop one of the Internet's first Search Engine Optimization tools, the Submission-Spider that was used by over three million individuals, small businesses, and governments in over forty-two countries.

These days John writes about Coding and wealth creation; produces several popular coding and wealth creation YouTube Channels with millions of video views, runs **Codemy.com** the online learning platform that teaches Coding to hundreds of thousands of students as well as **Tkinter.com** which strives to be the number one Tkinter resource on the Internet.

His YouTube Channels are:

| | |
|---|---|
| **https://www.youtube.com/c/TkinterPython** | Tkinter.com |
| **http://www.youtube.com/c/Codemycom** | Codemy.com |
| **https://www.youtube.com/c/JohnElder** | John Elder Wealth |

John graduated with honors from Washington University in St. Louis with a degree in Economics. He can be reached at john@codemy.com And his personal Website is: **JohnElder.com**

## INTRODUCTION

I love Tkinter! It may not be the most modern GUI tool out there but it's certainly the easiest and fastest to use GUI framework for Python. And hey, you can make it *"look modern"* with a few simple tweaks that I talk about at Tkinter.com.

Since Tkinter comes with Python, there's nothing to buy, nothing to download, nothing to install…it just works right out of the box!

And it's so EASY to use!

Tkinter is essentially built around **WIDGETS**. Everything in Tkinter is a Widget (more or less).

If you understand the Widgets, you understand Tkinter.

That's where this book comes in. If you look around, you'll discover that there isn't really a very good reference guide to Tkinter's widgets that shows all of their attributes in one easy to look-up place.

That's what this book is.

This book is nothing more than a list of all the Tkinter Widgets (and the add-on TTK Widgets as well) with all of their configurable attributes listed and explained.

Want to make a button bigger? Does a button have a height and width attribute or do you have to just make the font bigger? A quick glance through the Button Widget Attribute list in this guide will show you the answer.

What are the different border relief options for the Canvas widget? Just flip to the Canvas widget section, scroll down to the relief attribute and see for yourself instantly.

That's what this book does.

It's a quick reference guide to all of the attributes of every single Tkinter widget…listed alphabetically, all in one place.

That's it!

This is not a book on programming in Tkinter. I haven't written code or any tutorials of any sort in this book.

If you want that sort of thing, check out **Tkinter.com** or my Tkinter YouTube Channel.

This is just a list of all the widget attributes…

But that should be SUPER useful to you! There are hundreds, maybe thousands of attributes. I can't tell you how many times I've had to look one up and spent ten minutes surfing around trying to find a list, only to find incomplete info.

And forget about trying to dig through the documentation…

So I hope you find this short reference guide useful… now get out there and start writing some Tkinter Python code!

**FREE STUFF – FREE PDF COPY OF THIS BOOK!**

I love having an actual paperback book to flip through, but you might not always have this book with you…

So everyone who purchases this book can grab a free pdf version of it that you can download onto your computer, or phone, or tablet, or cloud account (Dropbox, whatever) so that you can reference it anytime, from anywhere.

Check the **APPENDIX B** at the end of this book for instructions on claiming your free pdf version.

There's nothing to pay, just head over to the link in the appendix and enter your email address and I'll send the pdf straight out to you.

Don't worry, I'm not going to spam your email or anything weird… I just need an address to send the book to.

Enjoy!

## CONVENTIONS USED IN THIS BOOK

Unlike most programming books, there's no code in this book; so there are no coding conventions to talk about.

But I *would* like to point out a couple of things that you may find useful when using this book as the tool that it's meant to be.

**First off**…the widgets are listed alphabetically. Sure, use the table of contents to find whatever you're looking for…but in a pinch, if you're just flipping through the book….look alphabetically.

**Second off**…the widget attributes themselves are listed alphabetically. So if you happen to know the attribute you're looking for, look for it alphabetically.

**Third off**… The TTK widgets are listed in two sections. The first section is full of TTK widgets that simply replace the exactly named Tkinter widget.

For instance there's a regular Tkinter button widget, and a TTK button widget.

Those are all listed in the first TTK section.

The second TTK section is full of TTK Widgets that don't have a corresponding Tkinter widget. I call them "new" TTK Widgets.

Those are things like the TTK.Notebook widget and the TTK.TreeView widget (amongst others).

There is no regular Tkinter Notebook or TreeView Widget, so those TTK widgets don't really replace anything. They're…new (though they've been around forever).

And really that's really all you need to know!

Ok, enough with this jibber jabber...let's get to the Widgets!

# SECTION ONE
# TKINTER WIDGETS

# CHAPTER ONE

# BUTTON WIDGET

# Button Widget

my_button = Button(root, **options)
.config()

activebackground = The background color when the widget  is active.

activeforeground = The foreground color when the widget is active.

anchor = Where in the widget the text will be located. Options are: N, NE, E, SE, S, SW, W, NW, or CENTER (default is CENTER).

background = The background color of the widget.

bg = Same as background.

bitmap = Displays a bitmap if Image isn't given.

borderwidth = The width of the widget's border.

bd = Same as borderwidth.

command = The function name that gets called when you click the button.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = What type of cursor to show when the mouse moves over the widget.

default = Set's the button as default (gives it a little extra border or something along those lines). **ACTIVE**, **DISABLED**, **NORMAL**. Default is **DISABLED**.

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines what font to use on the button.

foreground = Defines the color to use for text on the button.

fg = Same as foreground.

height = Defines the height of the button. If there's text, it's in the text unit. If an image is used, the height is pixels.

highlightbackground = Defines the color used for the highlight border when the button doesn't have focus.

highlightcolor = Defines the color used for the highlight border when the button has focus.

highlightthickness = Defines width of highlight border.

image = Defines the image used (if there's an image).

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

overrelief = Defines the relief when the mouse moves over the widget. See relief for options.

padx = Defines horizontal padding between text and widget.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **SUNKEN** when pressed. **RAISED** default. Options: **GROOVE**, **RIDGE**, **FLAT**.

repeateddelay = Designates the number of milliseconds a button or key must be held down before it begins to auto-repeat. Used, for example, on the up- and down-arrows in scrollbars.

repeatinterval = Used in conjunction with repeateddelay: once auto-repeat begins, this option determines the number of milliseconds between auto-repeats.

state = **NORMAL** (default), **ACTIVE**, **DISABLED**.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Sets the text of the button.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Defines the width of the button. If text is used, width is in font units. If image is used, width is pixels.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER TWO

# CANVAS WIDGET

# Canvas Widget

my_canvas = Canvas(root, **options)
.config()

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

closeenough = Default is 1. Specifies a floating-point value indicating how close the mouse cursor must be to an item before it is considered to be “inside” the item.

confine = Default is True. Defines if canvas cannot be scrolled outside of the scrollregion.

cursor = Defines the cursor to use when the mouse is moved over the widget.

height = Defines the canvas height.

highlightbackground = Defines the color used for the highlight border when the canvas doesn't have focus.

highlightcolor = Defines the color used for the highlight border when the canvas has focus.

highlightthickness = Defines the width of the highlight border.

insertbackground = Defines the color used for the text insertion cursor.
insertborderwidth = Defines the width of the insertion cursor border.

insertofftime = Controls cursor blinking (with insertontime). Given in milliseconds.

insertontime = Controls cursor blinking (with insertofftime). Given in milliseconds.

insertwidth = Defines the width of the insertion cursor.

offset = Default is 0,0.

relief = Defines border style. **FLAT** (default), **SUNKEN**, **RAISED**, **GROOVE**, **RIDGE**.

scrollregion = Defines the canvas scroll region.

selectbackground = Defines selection background color.

selectborderwidth = Defines selection border width.

selectforeground = Defines selection text color.

state = **NORMAL** (default), **DISABLED**, **HIDDEN**.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

width = Defines canvas width.

xscrollcommand = Connect to horizontal scrollbar.

xscrollincrement = Default is 0.

yscrollcommand = Connect to Vertical scrollbar.

yscrollincrement = Default is 0.

# CHAPTER THREE

# CHECKBUTTON WIDGET

# Checkbutton Widget

my_checkbutton = Checkbutton(root, **options)
.config()

activebackground = Defines the background color when the widget is activated.

activeforeground = Defines the foreground color when the widget is activated.

anchor = Defines where in the widget the text goes. **N, NE, E, SE, S, SW, W, NW,** or **CENTER** (default).

background = Defines the background color of the widget.

bg = Same as background.

bitmap = Defines a bitmap to display if image isn't used. Use @ sign in front of filename if loading from file (ie. @yourimage.xbm).

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

command = Defines the function name to call when widget is clicked.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the cursor to show when the mouse is moved over the widget.

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines the font to use in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Defines the height of the widget. If text, size is in text units. If image, size is in pixels.

highlightbackground = Defines the highlight background color when widget doesn't have focus.

highlightcolor = Defines the highlight color when the widget has focus.

highlightthickness = Defines the highlight thickness. Default is 1.

image = Defines the image to display, if any.

indicatoron = Defines whether or not the indicator is drawn. Default is **TRUE**. Setting it to false means relief is used as indicator.

justify = **LEFT**, **RIGHT**, **CENTER**. Defines how to align multiple lines of text.

offrelief = **RAISED** (default). **FLAT**.

offvalue = Defines the value of a non-checked button. Default is 0.

onvalue = Defines the value of a checked button. Default is 1.

overrelief = Specifies an alternative relief for the checkbutton, to be used when the mouse cursor is over the widget. **RAISED, FLAT.**

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **RAISED, SUNKEN, FLAT, RIDGE, SOLID, GROOVE**.

selectcolor = Defines color of selector.

selectimage = Defines image used for selector.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Defines the text to use.

textvariable = Associates a Tkinter Variable like StringVar.

underline = Defines which character is underlined on the button text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

variable = Associates a Tkinter Variable, when pressed toggles between **offvalue** and **onvalue**.

width = Defines the size of the widget.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER FOUR

# ENTRY WIDGET

# Entry Widget

my_entry = Entry(root, **options)
.config()

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

disabledbackground = Defines the background color used when the widget is disabled.

disabledforeground = Defines the text foreground color used when the widget is disabled.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

insertbackground = Defines the color used on the insertion cursor.

insertborderwidth = Defines with width of the insertion cursor border.

insertofftime = Defines cursor blinking in milliseconds. Used with insertontime.

insertontime = Defines cursor blinking in milliseconds. Used with insertofftime.

insertwidth = Defines the width of the insertion cursor.

invalidcommand = Sets a script to eval when validatecommand returns False. Default is nothing, recommended is to set to bell. See validate.

invcmd = Same as invalidcommand.

justify = Defines how to align text in the widget. **LEFT** (default), **RIGHT**, **CENTER**.

readonlybackground = Defines the background color when the state is set to readonly.

relief = Defines the border style of the widget. **SUNKEN** (default), **FLAT**, **RAISED**, **RIDGE**, **GROOVE**.

selectbackground = Defines the background color of selected text.

selectborderwidth = Defines the border width of selected text.

selectforeground = Defines the color of the selected text.

show = Defines how to show the contents of the widget. Good for passwords, set to "*".

state = Defines the state of the widget. **NORMAL** (default), **DISABLED**, "**readonly**" (like disabled but text can still be selected and copied).

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

textvariable = Associates a Tkinter variable like StringVar.

validate = Defines when validation is done. Sets Mode of Validation; **NONE** (default), **FOCUS**, **FOCUSIN**, **FOCUSOUT**, **KEY**, **ALL**.

validatecommand = Defines a function to call to check if contents are valid. Returns True/False. Used only if validate is not NONE.

vcmd = Same as validatecommand.

width = Defines the width of the widget in characters.

xscrollcommand = Connects to a horizontal scrollbar.

# CHAPTER FIVE

# FRAME WIDGET

# Frame Widget

my_frame = Frame(root, **options)
.config()

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

class = Specifies a class for the widget. Default is Frame.

colormap = Defines the colormap to use (especially for older monitors that only support 256 colors or less).

container = Default is False. If true, it means that this widget will be used as a container in which some other application will be embedded.

cursor = Defines the widget cursor.

height = Defines the height, default it 0.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines the border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**. To see this, change the borderwidth to at least 1.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

visual = Specifies visual information for the new window.

width = Define widget width. Default is 0.

# CHAPTER SIX

# LABEL WIDGET

# Label Widget

my_label = Label(root, **options)
.config()

activebackground = Defines the background color to use when the widget is active.

activeforeground = Defines the text color to use when the widget is active.

anchor = Defines where the text is located. **N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, **NW**, **CENTER** (default).

background = Defines the background color of the widget.

bd = Same as borderwidth.

bitmap = Defines the bitmap to be used in the widget.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Defines the height of the widget in text units if text is used. In pixels if images are used.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

image = Defines the image to display in the widget. Photoimage, BitmapImage, etc.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Defines the texts of the label.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Defines the width of the widget in text units if text is used, in pixels if images are used.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER SEVEN

# LABELFRAME WIDGET

# LabelFrame Widget

my_labelframe = LabelFrame(root, **options)
.config()

background = Defines the background color of the widget.

bd = Same as borderwidth.

bg = Same as background.

borderwidth = Defines the width of the border.  Default is 0 (no border).

class = Specifies a class for the widget.

colormap = Defines the colormap to use (especially for older monitors that only support 256 colors or less).

container = If true, this widget is a container widget. Default is false.

cursor = Defines the widget cursor.

fg = Same as foreground.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

height = Defines the widget height in pixels.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

labelanchor = Defines where to draw the text. Default is **NW** (upper left corner), **N, NE, EN, E, ES, SE, S, SW, WS, W, WN**.

labelwidget = Defines the widget to use for the label, default is **text**. The widget must exist before being used

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Defines the widget text.

visual = Specifies visual information.

width = Defines the frame width.

# CHAPTER EIGHT

# LISTBOX WIDGET

# ListBox Widget

my_listbox = Listbox(root, **options)
.config()

activestyle = Defines the style of the active element. **underline** (default), **dotbox**, **none**.

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

disabledforeground = Defines the text foreground color used when the widget is disabled.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Defines the height of the widget. Default is 10.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

listvariable = Specifies the name of a global variable. The value of the variable is a list to be displayed inside the widget.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

selectbackground = Defines the selectbackground color.

selectborderwidth = Default is 1.

selectforeground = Defines Select foreground color.

selectmode = Specifies one of several styles for manipulating the selection. **BROWSE** (default), **SINGLE**, **MULTIPLE**, **EXTENDED**.

setgrid = Specifies a boolean value that determines whether this widget controls the resizing grid for its top-level window. Default is 0.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

width = Defines widget width. Default is 20.

xscrollcommand = Connects to a horizontal scrollbar.

yscrollcommand = Connects to a vertical scrollbar.

# CHAPTER NINE

# MENU WIDGET

# Menu Widget

my_menu = Menu(root, **options)
.config()

activebackground = Defines background color when active. Default is "SystemHighlight".

activeborderwidth = Defines the active border width. Default is 0.

activeforeground = Defines active foreground color. Default is " SystemHighlightText".

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

postcommand = If this option is specified then it provides a Tcl command to execute each time the menu is posted.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

selectcolor = For menu entries that are check buttons or radio buttons, this option specifies the color to display in the indicator when the check button or radio button is selected. Default is "SystemMenuText".

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

tearoff = Defines whether or not the menu will tear off. Default Value is 1.

tearoffcommand = Invokes a command if the menu is torn off.

title = Defines the string to be used to title the window created when this menu is torn off.

type = "**normal**" (default), "**menubar**", "**tearoff**".

# CHAPTER TEN

# MENUBUTTON WIDGET

# MenuButton Widget

my_menubutton = Menubutton(root, **options)
.config()

activebackground = Defines the active background color.

activeforeground = Defines the active foreground color.

anchor = Defines where the text is located. **N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, **NW**, **CENTER** (default).

background = Defines the background color of the widget.

bg = Same as background.

bitmap = No default.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

direction = Specifies where the menu is going to be popup up. "**below**" (default), "**above**", "**left**", "**right**", "**flush**".

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Defines the height. Default is 0.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

image = No default.

indicatoron = If it is true then a small indicator rectangle will be displayed on the right side of the menubutton and the default menu bindings will treat this as an option menubutton. If false then no indicator will be displayed. Default is 0.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

menu = Specifies the path name of the menu associated with this menubutton. No Default.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Set's the text.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Set's the width. Default is 0.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER ELEVEN

# MESSAGE WIDGET

# Message Widget

my_message = Message(root, **options)
.config()

anchor = Defines where the text is located. **N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, **NW**, **CENTER** (default).

aspect = Defines the aspect ratio as width/height related by percentage. Default is 150 (message 50% wider than high). Aspect is ignored if width is set.

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Defines the message text. Line breaks added automatically.

textvariable = Associates a Tkinter variable like StringVar.

width = Defines the width of the widget in character units.

# CHAPTER TWELVE

# OPTIONMENU WIDGET

# OptionMenu Widget

my_optionmenu = OptionMenu(root, **options)
.config()

activebackground = Defines the active background color.

activeforeground = Defines the active foreground color.

anchor = Defines where the text is located. **N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, **NW**, **CENTER** (default).

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

direction = Specifies where the menu is going to be popup up. "**below**" (default), "**above**", "**left**", "**right**", "**flush**".

disabledforeground = Defines the text foreground color used when the widget is disabled.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

font = Defines the text font used in the widget.

height = Defines the height.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

image = No Default.

indicatoron = If it is true then a small indicator rectangle will be displayed on the right side of the menubutton and the default menu bindings will treat this as an option menubutton. If false then no indicator will be displayed. Default is 0.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

menu = Specifies the path name of the menu associated with this menubutton. No Default.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Sets the text.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Sets the width.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER THIRTEEN

# PANEDWINDOW WIDGET

# PanedWindow Widget

my_panedwindow = PanedWindow(root, **options)
.config()

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

handlepad = When handles are drawn, specifies the distance from the top or left end of the sash (depending on the orientation of the widget) at which to draw the handle. Default is 8.

handlesize = Specifies the side length of a sash handle. Handles are always drawn as squares. Default is 8.

height = Defines the height.

opaqueresize = Specifies whether panes should be resized as a sash is moved (true), or if resizing should be deferred until the sash is placed (false).

orient = **HORIZONTAL** (default), or **VERTICAL**.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

sashcursor = Mouse cursor to use when over a sash.

sashpad = Specifies the amount of padding to leave of each side of a sash. Default is 2.

sashrelief = Relief to use when drawing a sash. Can use any standard relief options like **FLAT, GROOVE**, **RAISED** (default), **RIDGE**, **SUNKEN**.

sashwidth = Specifies the width of each sash. Default is 2.

showhandle = Specifies whether sash handles should be shown. True, False.

width = Defines the width.

# .paneconfigure()

after = Insert after the widget.

before = Insert before the widget.

height = Defines the widget height.

minsize = Defines the minimum size; width for horizontal panes and height for vertical panes.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

sticky = Sets how to expand the child widget. **S, SE, SW, W, E, N, NE, NW**.

width = Defines the width.

# CHAPTER FOURTEEN

# RADIOBUTTON WIDGET

# RadioButton Widget

my_radiobutton = Radiobutton(root, **options)
.config()

activebackground = Defines the background color when the widget is active.

activeforeground = Defines the foreground color when the widget is active.

anchor = Where in the widget the text will be located. Options are: N, NE, E, SE, S, SW, W, NW, or CENTER (default is CENTER).

background = Defines the background color of the widget.

bg = Same as background.

bitmap = Defines the bitmap to display.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

command = Defines the function to call when the widget is clicked.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

disabledforeground = Defines the text foreground color used when the widget is disabled.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Defines the height in text units if text, in pixels if image.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

image = Defines the image used in the widget.

indicatoron = Uses the standard radio button if set to true. Uses **SUNKEN** if set to true.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

offrelief = Specifies the relief for the checkbutton when the indicator is not drawn and the checkbutton is off. Default is **RAISED**. By setting this option to **FLAT** and setting -indicatoron to false and -overrelief to **RAISED**, the effect is achieved of having a flat button that raises on mouse-over and which is depressed when activated.

overrelief = Defines an alternative relief when the mouse is moved over the radiobutton.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

selectcolor = Specifies a background color to use when the button is selected. Default value is "SystemWindow".

selectimage = Specifies an image to display (in place of the image option) when the radiobutton is selected.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

text = Defines the text to display.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

value = Defines the value associated with the radiobutton.

variable = Defines the variable associated with the radiobutton.

width = The width of the radiobutton. If text, size is in text units. If image, size is in pixels.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER FIFTEEN

# SCALE WIDGET

# Scale Widget

my_scale = Scale(root, **options)
.config()

activebackground = Defines the background color when the widget is active.

background = Defines the background color of the widget.

bg = Same as background.

bigincrement = Defines the size of the large increments. Default is 0.

borderwidth = Defines the width of the border.  Default is 0 (no border).

bd = Same as borderwidth.

command = Defines the function to call when the widget is clicked.

cursor = Defines the widget cursor.

digits = An integer specifying how many significant digits should be retained when converting the value of the scale to a string. Default is 0.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

from_ = Defines the left or top end of the scale. Default is 0.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

label = Defines the label of the scale.

length = Defines the length of the scale units. Default is 100.

orient = Defines the orientation of the widget. **VERTICAL** (default), or **HORIZONTAL**.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

repeatdelay = Defines the number of milliseconds a button or key must be held down before it begins to auto-repeatDefault is 300.

repeatinterval = Once auto-repeat begins, this option determines the number of milliseconds between auto-repeats.Default is 100.

resolution = If this value is greater than zero then the scale's value will always be rounded to an even multiple of this value, as will tick marks and the endpoints of the scale. If the value is less than zero then no rounding occurs. Default is 1.

showvalue = Specifies a boolean value indicating whether or not the current value of the scale is to be displayed. Default is 1.

sliderlength = Defines the length of the slider. Default is 30.

sliderrelief = Defines the relief of the slider. **FLAT**, **GROOVE**, **RAISED** (default), **RIDGE**, **SUNKEN**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

tickinterval = Determines the spacing between numerical tick marks displayed below or to the left of the slider. Default is 0.

to = Defines the value corresponding to the right or bottom end of the scale. Default is 100.

troughcolor = Defines the color to use for the rectangular trough areas in widgets such as scrollbars and scales.

variable = Specifies the name of a global variable to link to the scale. Whenever the value of the variable changes, the scale will update to reflect this value. Whenever the scale is manipulated interactively, the variable will be modified to reflect the scale's new value.

width = Defines the narrow dimension of the trough in screen units. Default is 15.

# CHAPTER SIXTEEN

# SCROLLBAR WIDGET

# Scrollbar Widget

my_scrollbar = Scrollbar(root, **options)
.config()

activebackground = Defines the background color when the widget is active.

activerelief = Specifies the relief to use when displaying the element that is active. **RAISED** (default).

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

command = Defines the function to call when the widget is clicked. Usually the xview or yview of the scrolled widget. If user drags the scrollbar slider, command is called as callback('moveto', offset), where offset 0.0 means that the slider is in its topmost (or leftmost) position, and offset 1.0 means that it is in its bottommost (or rightmost) position. If user clicks the arrow buttons, or clicks in the trough, command is called as callback('scroll', step, what). The second argument is either '-1' or '1' depending on the direction, and the third argument is 'units' to scroll lines (or other unit relevant for the scrolled widget), or 'pages' to scroll full pages.

cursor = Defines the widget cursor.

elementborderwidth = Specifies the width of borders drawn around the internal elements of the scrollbar (the two arrows and the slider).Default is -1.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

jump = If the value is false, updates are made continuously as the slider is dragged. If the value is true, updates are delayed until the mouse button is released to end the drag; at that point a single notification is made (the value "jumps" rather than changing smoothly). Default is 0.

orient = Defines the orientation. **VERTICAL** (default), or **HORIZONTAL**.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

repeatdelay = Specifies the number of milliseconds a button or key must be held down before it begins to auto-repeat. Default is 300.

repeatinterval = Once auto-repeat begins, this option determines the number of milliseconds between auto-repeats. Default is 100.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

troughcolor = Specifies the color to use for the rectangular trough areas in widgets such as scrollbars and scales. Ignored on Windows.

width = Defines the scrollbar width in pixels. Default is 16.

# CHAPTER SEVENTEEN

# SPINBOX WIDGET

# Spinbox Widget

my_spinbox = Spinbox(root, **options)
.config()

activebackground = Defines the background color when the widget is active.

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

buttonbackground = Defines the button background color.

buttoncursor = Defines the cursor used when the mouse is moved over the button part of the widget.

buttondownrelief = Defines the border style for the lower spin button. **RAISED** (default).

buttonuprelief = Defines the border style for the upper spin button. **RAISED** (default).

command = Defines the function to call when the widget is clicked.

cursor = Defines the widget cursor.

disabledbackground = Defines the background used when the widget is disabled.

disabledforeground = Defines the text foreground color used when the widget is disabled.

exportselection = Copies selected text to clipboard if set to True.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

font = Defines the text font used in the widget.

format = Specifies an alternate format to use when setting the string value when using the -from and -to range.

from_ = Defines the minimum value used in the spinbox range.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

increment = Defines the value adjusted when the spinbox is clicked. Default is 1.0.

insertbackground = Defines the color used for the insertion cursor.

insertborderwidth = Defines the width of the insertion cursor border.

insertofftime = Controls cursor blinking in milliseconds.

insertontime = Controls cursor blinking in milliseconds.

insertwidth = Defines the width of the insertion cursor.

invalidcommand = Specifies a script to eval when validateCommand returns 0. Set to bell.

invcmd = Same as invalidcommand.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

readonlybackground = Specifies the background color to use when the spinbox is readonly.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

repeatdelay = With repeatinterval, controls button auto-repeat.

repeatinterval = With repeatdelay, controls button auto-repeat.

selectbackground = Specifies the background color to use when displaying selected items.

selectborderwidth = Specifies a non-negative value indicating the width of the 3-D border to draw around selected items.

selectforeground = Specifies the foreground color to use when displaying selected items.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

textvariable = Associates a Tkinter variable like StringVar.

to = Defines the maximum value used in the spinbox range.

validate = Specifies the mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**.

validatecommand = Specifies a script to evaluate when you want to validate the input in the widget.

values = Defines a tuple containing valid values for the widget; overrides from/to/increment.

vcmd = Same as validatecommand.

width = Defines the width in character units. Default is 20.

wrap = Sets the up and down buttons to wrap around if set to **true**.

xscrollcommand = Used to connect the spinbox to a horizontal scrollbar.

# CHAPTER EIGHTEEN

# TEXT WIDGET

# Text Widget

my_text = Text(root, **options)
.config()

autoseparators = Specifies a boolean that says whether separators are automatically inserted in the undo stack. Only meaningful when the -undo option is true. Default is 1.

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

cursor = Defines the widget cursor.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

fg = Same as foreground.

height = Specifies the height for the widget, in units of characters in the font given by the -font option. Must be at least one. Default is 24.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

insertbackground = Specifies the color to use as background in the area covered by the insertion cursor.

insertborderwidth = Specifies a value indicating the width of the 3-D border to draw around the insertion cursor.

insertofftime = Specifies a value indicating the number of milliseconds the insertion cursor should remain 'off' in each blink cycle. Default is 300.

insertontime = Specifies a value indicating the number of milliseconds the insertion cursor should remain 'on' in each blink cycle. Default is 600

insertwidth = Specifies a value indicating the total width of the insertion cursor. Default is 2.

maxundo = Specifies the maximum number of compound undo actions on the undo stack. A zero or a negative value imply an unlimited undo stack.  Default is 0.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

selectbackground = Specifies the background color to use when displaying selected items.

selectborderwidth = Specifies a value indicating the width of the 3-D border to draw around selected items. Default is 0.

selectforeground = Specifies the foreground color to use when displaying selected items.

setgrid = Specifies a boolean value that determines whether the widget controls the resizing grid for its top-level window. Default is 0.

spacing1 = Requests additional space above each text line in the widget, using any of the standard forms for screen distances. If a line wraps, this option only applies to the first line on the display. Default is 0.

spacing2 = For lines that wrap (so that they cover more than one line on the display) this option specifies additional space to provide between the display lines that represent a single line of text. Default is 0.

spacing3 = Requests additional space below each text line in the widget, using any of the standard forms for screen distances. If a line wraps, this option only applies to the last line on the display. Default is 0.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

tabs = Specifies a set of tab stops for the window.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

undo = Specifies a boolean that says whether the undo mechanism is active or not. Default is 0.

width = Defines the width of the widget. Default is 80.

wrap = Specifies how to handle lines in the text that are too long to be displayed in a single line of the text's window. The value must be **none** or **char** or **word**. Default is **CHAR**.

xscrollcommand = Connect to horizontal scrollbar.

yscrollcommand = Connect to vertical scrollbar.

# CHAPTER NINETEEN

# TOPLEVEL WIDGET

# Toplevel Widget

my_toplevel = Toplevel( **options)
.config()

background = Defines the background color of the widget.

bg = Same as background.

borderwidth = Defines the width of the border. Default is 0 (no border).

bd = Same as borderwidth.

class = Specifies a class for the widget.

colormap = Defines the colormap to use (especially for older monitors that only support 256 colors or less).

container = If true, this widget is a container widget. Default is false.

cursor = Defines the widget cursor.

height = Defines the window height in pixels.

highlightbackground = When the widget doesn't have focus, defines the color of the focus highlight border.

highlightcolor = When the widget has focus, defines the color of the focus highlight border.

highlightthickness = Defines the width of the focus highlight border.

menu = Defines the menu to associate with the window.

padx = Defines horizontal padding between text and widget border.

pady = Defines vertical padding between text and widget border.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

screen = Specifies the screen on which to place the new window. Any valid screen name may be used, even one associated with a different display.

takefocus = Defines if a user can use the Tab key to move to this widget. Options: **TRUE**, **FALSE**.

use = Used for embedding. If the value is not an empty string, it must be the window identifier of a container window, specified as a hexadecimal string like the ones returned by the winfo id command. The toplevel widget will be created as a child of the given container instead of the root window for the screen.

visual = Specifies visual information for the new window. Often omitted to inherit from the root window.

width = Defines the window width in pixels.

# SECTION TWO
# TTK WIDGETS

# CHAPTER TWENTY

# TTK BUTTON WIDGET

# TTK Button Widget

my_ttkButton = ttk.Button(root, **options)
.config()

command = A script to evaluate when the widget is invoked.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. Center: text on top of image. Bottom, TOP, RIGHT, LEFT: puts the text at that location relative to the image. Default is NONE.

cursor = Defines the widget cursor.

default = May be set to one of **normal**, **active**, or **disabled**.

image = Specifies an image to display.

padding = Sets the padding between the text and the borders of the button.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Set's the text of the widget.

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Defines the space in character units to allocate for the text label.

# CHAPTER TWENTY ONE

# TTK CHECKBUTTON WIDGET

# TTK Checkbutton Widget

my_ttkCheckbutton = ttk.Checkbutton(root, **options)
.config()

command = Defines the function to execute when the widget is clicked.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

image = Defines an image to display.

offvalue = The value to store in the associated -variable when the widget is deselected. Defaults to 0.

onvalue = The value to store in the associated -variable when the widget is selected. Defaults to 1.

padding = Sets the padding between the text and the borders of the widget.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Designates text to be displayed inside the widget (unless overridden by textvariable).

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

variable = Designates a global variable whose value will be used in place of the text resource.

width = Sets the size in character width for the text label. If less than 0, it specifies a minimum width.

# CHAPTER TWENTY TWO

# TTK ENTRY WIDGET

# TTK Entry Widget

my_ttkEntry = ttk.Entry(root, **options)
.config()

background = Defines the background color of the widget.

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

invalidcommand = Sets a script to eval when validatecommand returns False. Default is nothing, recommended is to set to bell. See validate.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

show = designates the characters to be displayed. Useful to use for password * characters.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

textvariable = Associates a Tkinter variable like StringVar.

validate = Designates the mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**.

validatecommand = Evaluates whenever validation is triggered. If set to the empty string (default), validation is disabled. The script must return a boolean value.

width = Designates an integer indicating the width of the entry widget, in average-size characters of the widget's font.

xscrollcommand = Connect to horizontal scrollbar.

# CHAPTER TWENTY THREE

# TTK FRAME WIDGET

# TTK Frame Widget

my_ttkFrame = ttk.Frame(root, **options)
.config()

borderwidth = Defines the width of the border.  Default is 0 (no border).

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

height = Designates the height in pixels.

padding = Sets the padding between the text and the borders of the widget.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SOLID**, **SUNKEN**.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

width = Designates width in pixels.

# CHAPTER TWENTY FOUR

# TTK LABEL WIDGET

# TTK Label Widget

my_ttkLabel = ttk.Label(root, **options)
.config()

anchor = Designates how the information in the widget is positioned relative to the inner margins. **n, ne, e, se, s, sw, w, nw, and center**.

background = Defines the background color of the widget.

borderwidth = Defines the width of the border. Default is 0 (no border).

class = Specifies a class for the widget.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

image = Designates an image to display.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

padding = Sets the padding between the text and the borders of the widget.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Designates text to be displayed inside the widget (unless overridden by textvariable).

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Designates an integer indicating the width of the entry window, in characters of the widget's font.

wraplength = Defines whether or not text should be wrapped to multiple lines. Default is 0 (in screen units so try like 50 to see it in action).

# CHAPTER TWENTY FIVE

# TTK LABELFRAME WIDGET

# TTK LabelFrame Widget

my_ttkLabelFrame = ttk.LabelFrame(root, **options)
.config()

borderwidth = Defines the width of the border. Default is 0 (no border).

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

height = Sets the height in pixels.

labelanchor = Designates where to place the label. Clockwise from the top upper left corner: **nw, n, ne, en, e, es, se, s,sw, ws, w and wn**.

labelwidget = The name of a widget to use for the label. If set, overrides the text option. The labelwidget must be a child of the labelframe widget or one of the labelframe's ancestors, and must belong to the same top-level widget as the labelframe.

padding = Sets the padding between the text and the borders of the widget.

relief = Defines border decoration. **FLAT** (default), **GROOVE**, **RAISED**, **RIDGE**, **SUNKEN**.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Designates text to be displayed inside the.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Sets the width in pixels.

# CHAPTER TWENTY SIX

# TTK MENUBUTTON WIDGET

# TTK MenuButton Widget

my_ttkMenubutton = ttk.Menubutton(root, **options)
.config()

class = Specifies a class for the widget.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

direction = Designates where the menu is to be popped up relative to the menubutton. **above**, **below** (default), **left**, **right**, or **flush** (directly over the menubutton).

image = Designates an image to use.

menu = Designates the path name of the menu associated with the menubutton.

padding = Sets the padding between the text and the borders of the widget.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Designates text to be displayed inside the widget (unless overridden by textvariable).

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

width = Designates width of the label text, in characters widths. If less than 0, it specifies a minimum width.

# CHAPTER TWENTY SEVEN

# TTK PANEDWINDOW WIDGET

# TTK PanedWindow Widget

my_ttkPanedwindow = ttk.Panedwindow(root, **options)
.config()

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

height = Designates height in pixels.

orient = Designates the orientation of the widget. **vertical** (stacked top to bottom), **horizontal** (stacked left to right).

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

weight = For panes; designates the stretchability of the pane. When pane is resized, extra space is added or subtracted in proportion to it's weight.

width = Designates width in pixels.

# CHAPTER TWENTY EIGHT

# TTK RADIOBUTTON WIDGET

# TTK RadioButton Widget

my_ttkRadiobutton = ttk.Radiobutton(root, **options)
.config()

class = Specifies a class for the widget.

compound = How to combine text and image (if there's an image). By default, if there's an image it will cover the text unless you specify compound. **Center**: text on top of image. **Bottom**, **TOP**, **RIGHT**, **LEFT**: puts the text at that location relative to the image. Default is **NONE**.

cursor = Defines the widget cursor.

image = Designates an image to show.

padding = Sets the padding between the text and the borders of the widget.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

text = Designates text to be displayed inside the widget (unless overridden by textvariable).

textvariable = Associates a Tkinter variable like StringVar.

underline = Defines which character is underlined on the widget text. Default is -1 (nothing underlined). Options start at 0 and increase by 1 for each letter in the text.

value = The value to store in the associated variable when the widget is selected.

variable = The name of a global variable whose value is linked to the widget.

width = Sets the width in character widths for the text label. If less than 0, sets a minimum width.

# CHAPTER THIRTY

# TTK SCALE WIDGET

# TTK Scale Widget

my_ttkScale = ttk.Scale(root, **options)
.config()

class = Specifies a class for the widget.

command = Sets the function to call when the scale's value changes.

cursor = Defines the widget cursor.

from_ = Designates the value corresponding to the left or top end of the scale.

length = Designates the long dimension of the scale in screen unit. For vertical scales; sets the height. For horizontal scales; sets the width.

orient = Designates whether the scale is **horizontal** or **vertical**.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

to = Designates the value corresponding to the  right or bottom end of the scale. This value may be either less than or greater than the from option.

value = Designates the current floating-point value of the variable.

variable = Designates the name of a global variable to link to the scale. When the value of the variable changes, the scale will update to reflect this value.

# CHAPTER THIRTY ONE

# TTK SCROLLBAR WIDGET

# TTK Scrollbar Widget

my_ttkScrollbar = ttk.Scrollbar(root, **options)
.config()

class = Specifies a class for the widget.

command = Changes the view of the widget associated with the scrollbar. Used with xview and yview.

cursor = Defines the widget cursor.

orient = Sets the scrollbar orientation to **vertical** or **horizontal**.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

# CHAPTER THIRTY TWO

# TTK SPINBOX WIDGET

# TTK Spinbox Widget

my_ttkSpinbox = ttk.Spinbox(root, **options)
.config()

background = Defines the background color of the widget.

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

format = Designates an alternate format to use when setting the string value when using the -from and -to range.

from_ = A floating-point value designating the lowest value for the spinbox. This is used with **to** and **increment** to set a numerical range.

increment = A floating-point value specifying the change in value to be applied each time one of the widget spin buttons is pressed. The up button applies a positive increment, the down button applies a negative increment.

invalidcommand = Sets a script to eval when validatecommand returns False. Default is nothing, recommended is to set to bell. See validate.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

show = designates the characters to be displayed. Useful to use for password * characters.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

textvariable = Associates a Tkinter variable like StringVar.

to = A floating-point value specifying the highest value for the widget.

validate = Specifies the mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**.

validatecommand = A script to evaluate whenever validation is triggered. If set to the empty string (the default), validation is disabled. The script must return a boolean value.

values = If this option is set then this will override any range set using the **from**, **to** and **increment** options.

width = Designates the width.

wrap = A Boolean value that designates whether or not the spinbox will wrap around the data.

xscrollcommand = Connect to horizontal scrollbar.

# SECTION THREE
# NEW TTK WIDGETS

# CHAPTER THIRTY THREE

# TTK COMBOBOX WIDGET

# TTK Combobox Widget

my_ttkCombobox = ttk.Combobox(root, **options)
.config()

background = Defines the background color of the widget.

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

exportselection = Copies selected text to clipboard if set to True.

font = Defines the text font used in the widget.

foreground = Defines the foreground color of the widget.

height = Designates the height of the pop-down listbox, in rows.

invalidcommand = Sets a script to eval when validatecommand returns False. Default is nothing, recommended is to set to bell. See validate.

justify = **LEFT**, **RIGHT**, **CENTER** (default) to define alignment of multiple lines of text.

postcommand = Evaluates immediately before displaying the listbox. Specifies the **values** to display.

show = designates the characters to be displayed. Useful to use for password * characters.

state = **NORMAL** (default)**, ACTIVE, DISABLED.**

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

textvariable = Associates a Tkinter variable like StringVar.

validate = Specifies the mode in which validation should operate: **none** (default), **focus**, **focusin**, **focusout**, **key**, or **all**.

validatecommand = A script to evaluate whenever validation is triggered. If set to the empty string (the default), validation is disabled. The script must return a boolean value.

values = Designates the list of values to display in the drop down listbox.

width = Sets the width of the entry window in average-size characters of the widget's font.

xscrollcommand = Connect to horizontal scrollbar.

# CHAPTER THIRTY FOUR

# TTK NOTEBOOK WIDGET

# TTK Notebook Widget

my_ttkNotebook = ttk.Notebook(root, **options)
.config()

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

height = Sets the height of the paned area (not including tabs or padding).

padding = Sets the padding between the text and the borders of the widget.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

width = Sets the width of the paned area (not including tabs or padding).

# CHAPTER THIRTY FIVE

# TTK PROGRESSBAR WIDGET

# TTK Progressbar Widget

my_ttkProgressbar = ttk.Progressbar(root, **options)
.config()

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

length = Designates the length of the long axis of the progress bar (width if horizontal, height if vertical).

maximum = A floating point number specifying the maximum **value**. Default is 100.

mode = **determinate** or **indeterminate**.

orient = Specifies the orientation of the progress bar. **horizontal** or **vertical**.

phase = Read-only option. The widget periodically increments the value of phase whenever the **value** is greater than 0 and, in determinate mode, less than **maximum**. This option may be used by the current theme to provide additional animation effects.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

value = The current value of the progress bar. In determinate mode, this represents the amount of work completed. In indeterminate mode, it is interpreted modulo **maximum**; that is, the progress bar completes one "cycle" when the **value** increases by **maximum**.

variable = The name of a global variable linked to the **value**. If specified, the **value** of the progress bar is automatically set to the value of the variable whenever the latter is modified.

# CHAPTER THIRTY SIX

# TTK SEPARATOR WIDGET

# TTK Separator Widget

my_ttkSeparator = ttk.Separator(root, **options)
.config()

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

orient = Sets the orientation to **vertical** or **horizontal**.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

# CHAPTER THIRTY SEVEN

# TTK SIZEGRIP WIDGET

# TTK Sizegrip Widget

my_ttkSizegrip = ttk.Sizegrip(root, **options)
.config()

class = Specifies a class for the widget.

cursor = Defines the widget cursor.

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

# CHAPTER THIRTY EIGHT

# TTK TREEVIEW WIDGET

# TTK Treeview Widget

my_ttkTreeview = ttk.Treeview(root, **options)
.config()

class = Specifies a class for the widget.

columns = A list of column identifiers, specifying the number of columns and their names.

cursor = Defines the widget cursor.

displaycolumns = A list of column identifiers (either symbolic names or integer indices) specifying which data columns are displayed and the order in which they appear, or the string #all. If set to #all (the default), all columns are shown in the order given.

height = Specifies the number of rows that should be visible. *The requested width is determined from the sum of the column widths.

padding = Sets the padding between the text and the borders of the widget.

selectmode = Controls how the built-in class bindings manage the selection. Can be **extended** (default – multiple items can be selected), **browse** (only 1 item can be selected at a time), or **none**.

show = A list containing zero or more of the following values: **tree** (default – display tree feadings in column 0), **headings** (display the heading rows).

style = Used to specify a custom widget style.

takefocus = Determines whether the widget accepts the focus during keyboard traversal. Can be 0, 1.

xscrollcommand = Connect to horizontal scrollbar.

yscrollcommand = Connect to vertical scrollbar.

# CHAPTER THIRTY NINE

# CONCLUSION

You made it! We're done! How 'bout them apples?

What'd you think? Having a Widget quick reference guide is pretty freaking useful – right?

I hope you enjoyed the book, I really enjoyed putting it together for you!

Tkinter may not be the flashiest GUI framework out there, but there are a few new libraries out there that really transform it into a modern looking GUI framework.

I didn't discuss any of these libraries in this book, but I do over at Tkinter.com so check that out if you're interested!

**CAN I ASK YOU A HUGE FAVOR?**

Book reviews at Amazon.com are a HUGE deal for small writers like me. Just a few reviews can mean the difference between a book getting ranked well by Amazon's algorithm, and complete oblivion. Without reviews, the book won't even show up when someone searches for Tkinter at Amazon.

I'd consider it a great favor if you would head back to Amazon.com and leave a quick review of this book. It doesn't have to be long-winded, just a few words will make a big big difference as to whether my book gets out there or not.

So if you enjoyed the book, got a lot out of it...heck even if you didn't like it, please head back to Amazon and leave a review. Then shoot me an email and let me know so I can thank you properly!

Just go to **Tkinter.com/bookreview**

And that URL will re-direct to this book's Amazon page where you can leave a quick review.

I really appreciate it!!

Thanks!
*-John Elder*
**Tkinter.com**
**john@Codemy.com**

## APPENDIX A

## SPECIAL TKINTER.COM MEMBERSHIP OFFER

Learning never stops, especially for coders. There's always something new and cool to learn. I've tried to build a website that makes it super easy to learn how to build modern looking GUI apps with Tkinter …and it's called **Tkinter.com**

Each course at Tkinter.com is a series of videos that you watch over my shoulder as I build something.

In one course I build an MP3 Player with Tkinter. In another course I build a Paint program with Tkinter. In another course I build a CRM Database tool with Tkinter.

Then there's the course where I just explain all the different widgets in deatail; and another where I show you how to create Modern Looking GUI programs with Tkinter…

And much much more!

Membership at Tkinter.com regularly costs $149, and that gets you access to all my Tkinter courses and all future courses.

## AS A SPECIAL THANK YOU FOR READING THIS BOOK…

I'd love to see you over at Tkinter.com and I'd like to bribe you to join today; so I'm giving you a special coupon code (**widgetbook)** that will knock $50 off membership - (so you pay just $99 instead of $149)…

It's my gift to you! **https://www.Tkinter.com**

So you get access to ALL my best-selling courses for just **$99** instead of the regular $149.

And I offer a month-long 100% money back guarantee. Check out the site, if it isn't for you…just shoot me a message and I'll immediately refund your money, no questions asked, no hoops to jump through.

**HANDS ON HELP**

Membership doesn't just get you videos…you also get hands on help from me and other members. Any time you get stuck with something from a video, you can post a question to me directly, or post a question directly under the video itself.

It's a great resource and I hope you'll take advantage of it.

Just use coupon code **widgetbook** at checkout for the special $99 price.
**http://www.Tkinter.com**

See you on the inside!

*-John Elder*
Codemy.com

## APPENDIX B

## DOWNLOAD THE FREE PDF VERSION OF THIS BOOK

I love having an actual paperback book to flip through, but you might not always have this book with you…

So everyone who purchases this book can grab a free pdf version of it that you can download onto your computer, or phone, or tablet, or cloud account (Dropbox, whatever) so that you can reference it anytime, from anywhere.

Just head over to **https://tkinter.com/widget-book**

There's nothing to pay, just head over to that link and enter your email address and I'll send the pdf straight out to you.

Don't worry, I'm not going to spam your email or anything weird… I just need an address to send the book to.

Enjoy!

THE END

# NOTES

# NOTES

# NOTES

# NOTES

# NOTES

# NOTES