BIG DADDY'S PYTHON TOOLBOX For 3.5

tkinter Journeyman Reference

excludes ttk except noted p10 **color codes: *explanatory*, option, example, note, syntax,** ↳ = "results in"/"yields"

Import: from tkinter import * (or **import tkinter as tk** - requires using "tk." prefix to commands)

Establish root: *root=Tk()* **Define root or toplevel Geometry:** root.Geometry(str(width)+"x"+ str(height)+"+"+str(xoffset)+"+"+str(yoffset)) ; find screen dimensions *with* ***=root.info_screenwidth()*** *and* ***=root.infoscreenheight();*** *in TCL8.4:* **root.attributes("-fullscreen", True)** Observe syntax carefully. To position root or toplevel windows use **window.lift()** or **.lower()**, or call on the window manager: **top1.attributes("-topmost", True)** to hold it in place.

18 Widgets (w/o ttk)

- Button
- Canvas
- Checkbutton
- Entry
- Frame
- Label
- Labelframe
- Listbox
- Menubutton
- Message
- Panedwindow
- Radiobutton
- Scale
- Scrollbar
- Spinbox
- Text
- Toplevel
- Messagebox

Create Widgets: widget_name = widget_type(attributes (a.k.a options), and **command callbacks)**; use any callable object as a callback: **functions** (most common), bound methods, lambda expressions, commands. No event information provided. Syntax:

Your widget name option (ex: background color) function to bind when **auto event** occurs

b1=Button(parent, bg= 'light blue', text= 'Push me!', command= myCallbackFunction)

widget type parent window name option (ex: label text) *button click binding is* ***automatic***

Bindings - events can bind at 4 levels: Application, Toplevel, Class, **Instance** (most common) ex: **label1.bind**("Button-1", callback function name) **Manual Bindings** - To respond to an event **not** auto bound to a widget, use the **.bind** method to connect a callback. **widget.bind (event, callback)** This binding delivers event objects to callback if a function explicitly accepts it: def callback(**event**)

Event Objects passed to **callback**: **widget** - tkinter instance; ***mouse button events:*** **num** - button number ; **x, y** - mouse position; **x_root, y_root** - mouse position relative to toplevel; ***keyboard events*****: char** - character code (as a string); **keysym** - key symbol ; **keycode** - key code; **other events: width, height** - new widget size (pixels); **type** - event type

Common Manual Bindings

Bound event : **sequence** : Comment

Button events:

<Button-1> : leftmost button, *<1> is alias*
<Button-2> : middle button if available
<Button-3> : rightmost mouse button
<ButtonRelease1> : button up
<Leave> : mouse pointer left widget
<B1-Motion>: movement with button down
<DoubleButton1> : double click
<Enter> : mouse pointer entered widget

Keyboard events:

<FocusIn> : keyboard focus moved to w
<FocusOut> : keyboard focus moved away
<Return> : the keyboard enter key
<Key> : w.bind("<Key>", cback) any keypress "**x**" : a letter ex: frame.bind("x", **callback**)

Options, values, processes, and constants

Compass points: 'n', 'ne', 'e', 'se', 's', 'sw', 'w', 'nw', 'center'.
Bitmaps: 'error', 'gray75', 'gray50', 'gray25', 'gray12', 'hourglass', 'info', 'questhead', 'question', 'warning'

Colors: can be given as the names of colors in the rgb.txt file (downloaded with tkinter); also hex definitions #rgb, #rrggbb, or #rrrgggbbb
Distance: Pixels → numeric; absolute distances → strings with a trailing character denoting units: c- centimeters, i-inches, m-millimeters, p- printer's points - these vary with font used.
Fonts: Ex: font=("Verdana 10 bold"). Font sizes with positive numbers measured in **points**; negative numbered sizes are measured in **pixels**.
Justify: "left", "center", "right", "fill" *include quotes*
Region: 4 space-delimited distances "3i 2i 4.5i 2i"
Relief: "raised", "sunken", "flat", "groove", "ridge"

RAISED SUNKEN FLAT GROOVE RIDGE

Wrap: "none", "char", "word"
Cursors: many available such as: "arrow" "circle" "clock" "cross" "dotbox" "exchange" "fleur" "heart" "man" "mouse" "pirate" "plus" "shuttle" "sizing" "spider" "spraycan" "star" "trek" "watch"
Images: B&W: id constructor: myBWpic = tk.BitmapImage(file=myimagefile.xbm); **Color:** myphotoimage = PhotoImage(file=myimagefile.{.gif, .pgm, .ppm formats}) see pg 6&7
Control Variables - shared by multiple widgets, updates all users automatically: constructors are =**DoubleVar()**, =**IntVar()**, =**BooleanVar()** =**StringVar**, ; use: v.get(value) and v.set(value)

Type of Events Listed above are keyboard and mouse or pointer events. Other types of events (not detailed here) are **crossing, focus, exposure, configuration, and colormap** more on pg. 6

Special key bindings Cancel *(the Break key)*, BackSpace, Tab, Return (*the Enter key)*, Shift_L *(any Shift key)*, Control_L *(any Control key)*, Alt_L *(any Alt key)*, Pause, Caps_Lock, Escape, Prior *(Page Up)*, Next *(Page Down)*, End, Home, Left, Up, Right, Down, Print, Insert, Delete, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, Num_Lock, and Scroll_Lock; *each encased in "< >"; ex: "<Tab>"*

Other bindings: **Class:** .bind_class()
ex: self. Bind_class(w_type, "<event>", function)
Application: .bind_all('<some event>', function)
ex: self. bind_all('<Key-Print>', self.__printScreen)

Using lambda to pass simple values in a binding Ordinarily no variables are passed in an automatic binding and only event data in manual bindings. A lambda statement can be used to overcome this limitation with simple data - for example regular or control variables. Auto binding example:
command=lambda: button1callback(myvar) or
lambda: button1callback(cvar.get()), the callback function must specify a receiving variable container. For manual bind:
w.bind("<event>", lambda event, arg=variable: callback (event,arg)) *callback has 2 variables such as (event, myvar)*

comments and suggestions appreciated:
oakey.john@yahoo.com WWW.WIKIPYTHON.COM

BIG DADDY'S PYTHON TOOLBOX For 3.5

tkinter Journeyman Reference

Event sequence tkinter term for an event name. It is a **string** containing one or more **event patterns**, as follows:

quote and angle bracket enclosure

" < [optional modifier(s)]… type [optional detail] >"

Alt, Control, Double, Lock, Shift, or Triple — like Button or Return — like 1, 2, 3

Second Bindings if an automatic binding is already set to a callback (using the command= callback option), another binding can be manually set by including *args in the **callback** parens: callback(*args). callback will now work for both but only the event you bind sends back event data and using it causes a built-in event error.

3 Geometries: Pack, Place, Grid - widget installation and formatting tools **Options and Attributes:**

Geometry Options	Pack	Grid	Place	Default or Requirement	Options	Note
after	●			other window		name of another window
anchor	●		●	center	compass points	default is NW
before *(other)*	●			other window		name of another window
bordermode			●	inside	outside / ignore	border influence on slave placement
column		●		0	integer	column number; columns start with 0
columnspan		●		1	integer	number of columns spanned
expand	●			FALSE	true/false: 0/1	assign more space, distribute among widgets
fill	●			"none"	"x", "y", "both"	take up entire space, may need expand=
height			●	size	distance	outer dimension of window plus border
in_= *(target)*	●	●	●	n/a	widget name	pack inside the target widget
ipadx	●	●		0	distance	internal padding pixels/distance; vertical
ipady	●	●		0	distance	internal padding pixels/distance; horizontal
padx	●	●		0	distance	external padding pixels/distance; vertical
pady	●	●		0	distance	external padding pixels/distance; horizontal
relheight			●	fp	fp 0 to 1.0	height relative to master; modified by - height
relwidth			●	fp	fp 0 to 1.0	width relative to master; modified by - width
relx			●	fp % * 100	0.0(left)-1.0(right)	anchor point x coordinates in master window
rely			●	fp % * 100	0.0(left)-1.0(right)	anchor point y coordinates in master window
row		●		first empty	row number	row number; rows start with 0
rowspan		●		1	integer	number of rows spanned
side	●			top /	left/ right/ bottom	to pack against
sticky (glue widget to cell border)		●		centered	compass points	W+E stretch horiz; W+E+N+S - fill all; use a **string="wens"** or **constants** =W+E+N+S
width			●	size	distance	outer dimension of window **plus** border
x			●	0	distance	anchor point x coordinate in master window
y			●	0	distance	anchor point y coordinate in master window

Methods: Universal Geometry Methods

(**x**=widget name.geometry name) **B1.pack_forget()**

x_forget() remove from manager but do not destroy, can reuse **ex: lab1.grid_forget()**, retrieve by repeating the original grid command

x_info() ↳ a dictionary of options **w.grid_info()**

x_slaves() returns list of sub widgets as tkinter widget references

x_configure(options) *same as .pack()*

Geometry Specific Methods

place: *has no other Methods.*

pack and grid:

x_propagate(flag) ; True/False; enables resizing of child widgets if too small

grid:

w.grid_bbox(column=None, row=None, col2=None, row2=None)

w.grid_size() tuple with number of columns and rows

w.grid_location(x,y) ↳ tuple with indexes

w.grid_remove() removes widget from manager; but the widget is available for reuse

To change the following, you must call these on a widget's **parent**:

grid_**columnconfigure**(index, **options)**

grid_**rowconfigure**(index, **options)**

index = column number

options: minsize=, pad=, weight=, uniform=

www.wikipython.com

BIG DADDY'S

JR2 p3

PYTHON TOOLBOX

tkinter Journeyman Reference

For 3.5

tkinter Widget Methods

	Button	Canvas	Checkbutton	Entry	Frame	Label	Labelframe	Listbox	Menu	Menubutton	Message	Panedwindow	Radiobutton	Scale	Scrollbar	Spinbox	Text	Toplevel	Value Type	Default Value	Note
activebackground	●		●			●				●			●	●	●	●			color	system	mouse over or b1 press
activeborderwidth									●										distance value		menu; only available if displaying > 1 element
activeforeground	●		●			●				●			●						color	system	
activerelief															●				relief	raised	displayed when element active
activestyle								●											dotbox, none,	dotbox-	listbox;
anchor	●		●			●				●	●		●						compass points or "center"	usually center	
aspect											●								positive integer aspect ratio	150	message;
autoseparators																	●		boolean		text; applies only if undo is true
background or -bg	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	color	system	
bigincrement														●					fp	.1	large movement amoount
bitmap	●		●			●				●			●						"" or filename	(see predefined list)	see native available: "question"
blockcursor																	●		boolean	false	text;
borderwidth or -bd	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	distance value	2 pixels	
buttonbackground																●			color		spinbox;
buttoncursor																●			cusor name	default cursor	spinbox;
buttondownrelief																●			relief		spinbox;
buttonuprelief																●			relief		spinbox;
class_ (note underscore)					●		●											●	class for window - determines options		toplevel, frame, labelframe;
closeenough		●																	fp value	1.0	how close is inside
colormap					●		●											●	'new' or another window	screen default	toplevel, frame, labelframe;
command	●		●										●	●	●	●			callback function name	none	
compound	●		●			●				●			●						bottom, top, left, right, center	none	controls the display of text and images at same time
confine		●																	boolean	True	canvas; T confines view to scroll region
container					●													●	boolean	false	toplevel, frame, labelframe; to embed an app
cursor	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	cursor name	system	see cursor options
default	●																		normal, active, disabled		button; disabled may draw smaller button
digits														●					integer		scale; string resolution, 0 yeilds all positions
direction										●									above, below, left, right, flush		menubutton; where menu pops up
disabledbackground				●												●			color	system	spinbox, entry;
disabledforeground	●		●	●		●		●		●			●			●			color		
elementborderwidth															●				pixels	=borderwidth?	around arrows and slider
endline																	●		next-after-last line index integer		text only;
exportselection				●				●								●	●		1 or 0	1	
font	●		●	●		●	●	●		●	●		●	●		●	●		font 3 tuple		name, size, weight
foreground or -fg	●		●	●		●	●	●		●	●		●	●		●	●		color		
format																●			fp format to string	%<pad>.<pad>f	spinbox;
from														●		●			fp - lowest value		scale, spinbox; used with to and increment
handlepad												●							distance value		paned window; sash to handle distance
handlesize												●							distance value		paned window; side length of sash handle
height	●	●	●		●	●	●	●		●		●	●				●	●	text-lines, image-distance value	1	centered
highlightbackground	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	color	system	
highlightcolor	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	color	system	
highlightthickness	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	distance value	1 pixel	width of highlight if widget has input focus
image	●		●			●				●			●						bit flie: gif, pgm, ppm		image created with image_create command
inactiveselectbackground																	●		color		text?; for selection when window does not have focus
increment																●			fp(can be + or -)		spinbox; widget adj amount

BIG DADDY'S

JR2 p4

PYTHON TOOLBOX

tkinter Journeyman Reference

For 3.5

tkinter Widget Methods	Button	Canvas	Checkbutton	Entry	Frame	Label	Labelframe	Listbox	Menu	Menubutton	Message	Panedwindow	Radiobutton	Scale	Scrollbar	Spinbox	Text	Toplevel	Value Type	Default Value	Note
indicatoron			●							●			●						boolean	false	checkbutton, radiobutton, menubutton; raised if true
insertbackground		●		●												●	●		color	black	
insertborderwidth		●		●												●	●		distance value	0	
insertofftime		●		●												●	●		milliseconds	300	
insertontime		●		●												●	●		milliseconds	600	
insertunfocussed																	●		none,(no cursor), hollow, solid	none	text; new in tcl8l.6
insertwidth		●		●												●	●		distance value	2 pixels	width of insertion cursor
invalidcommand or - invcmd				●												●			script to evaluate		spinbox, entry; best use is "bell"
jump															●				boolean F-smooth T-update at release		drag slider value change amt
justify	●		●	●		●				●	●		●			●			see justify choices	center	
label														●					string		scale; label for the scale
labelanchor							●												compass points	"nw"	lableframe; text option can not be empty
labelwidget							●												widget		lableframe; widget to use as label
length														●					distance value		scale; long dimension of the scale
listvariable								●											name of a global variable		listbox; a list to be display in the widget
maxundo																	●		integer		text; 0 or neg yield unlimited undo stack
menu										●								●	name of associated menu		toplevel, menubutton; <-not exactly equivalent
offrelief			●										●						relief		checkbutton, radiobutton; when button off
offvalue			●																value	0	checkbutton; variable when not selected
onvalue			●																value	1	checkbutton; variable when selected
opaqueresize												●							boolean		panedwindow; true-resize as sash moved
orient												●		●	●				'horizontal', 'vertical'		panedwindow, scale, scrollbar;
overrelief	●		●										●						relief	"" (empty string)	button, checkbutton, radiobutton; mouse over
padx	●		●		●	●	●			●	●		●				●	●	distance value	1 pixel	
pady	●		●		●	●	●			●	●		●				●	●	distance value	1 pixel	
proxybackground												●							color	background color	
proxyborderwidth												●							pixels		
proxyrelief												●							relief	sashrelief	
readonlybackground				●												●			color	normal background	spinbox, entry; if read only
relief	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	see relief choices	"sunken"	
repeatdelay	●													●	●	●			milliseconds before engage		
repeatinterval	●													●	●	●			milliseconds between execution		
resolution														●					real value	1 (integral)	scale; resolution of the scale
sashcursor												●							mouse cursor	horiz: sb_h_double_ arrow, vert: sb-v-double_arrow	panedwindow; cursor when mouse over
sashpad												●							distance value		panedwindow; pad on each side of sash
sashrelief												●							relief		panedwindow; sash relief
sashwidth												●							distance value		panedwindow; width of sash
screen																		●	screen name for new window		toplevel;can not modify with config
scrollregion		●																	coords: fp		rectangle: params or list
selectbackground		●		●				●								●	●		color		
selectborderwidth		●		●				●								●	●		distance value		
selectcolor			●										●						color		checkbutton, radiobutton; use when selected
selectforeground		●		●				●								●	●		text color		
selectimage			●										●						image	ignored unless image option	checkbutton, radiobutton; when button selected
selectmode								●											single, browse, multiple, extended	browse	listbox; styles for manipulatin the selection

tkinter Widget Methods	Button	Canvas	Checkbutton	Entry	Frame	Label	Labelframe	Listbox	Menu	Menubutton	Message	Panedwindow	Radiobutton	Scale	Scrollbar	Spinbox	Text	Toplevel	Value Type	Default Value	Note
setgrid								●									●		boolean		listbox, text; widget controls toplevel grid size
show				●															string - 1 character		entry; replaces entry - like "*" for password
showhandle												●							boolean		panedwindow; sash handles shown or not
showvalue														●					boolean		scale; show current value
sliderlength														●					distance value		scale; sze of slider
sliderrelief														●					relief		scale;
spacing1																	●		distance value		text; + space above text lines
spacing2																	●		distance value		text; + space above single wrap lines
spacing3																	●		distance value		text; + space above last wrap line
startline																	●		integer index or ""		text; indicate line to start, "" is first line
state	●	●	●	●		●		●		●			●	●		●	●		Normal or Disabled	normal	
tabs																	●		list of distances for	8 characters	text; stops can preceed left, right, center,
tabstyle																	●		'tabular' or 'wordprocessor'	'tabular'	text;
takefocus	●	●	●	●	●	●	●	●		●	●		●	●	●	●	●	●	0 or 1	1	
text	●		●			●	●			●	●		●						a string		
textvariable	●		●	●		●				●	●		●			●			a string		
tickinterval														●					real number		scale; tick spacing - if 0 no tick marks
to														●		●			real number		scale, entry; right or bottom end
tristateimage			●										●						button)		option unspecified
tristatevalue			●										●						value (that causes display)	empty string ("")	checkbutton, raidiobutton;
troughcolor														●	●				color (not on Windows)		scale, scrollbar;
underline	●		●			●				●			●						integer		index of char to underline
undo																	●		boolean		text; undo mechanism is active or not
use																		●	hex string window identifier		toplevel; get from winfo id
validate				●												●			none, focus, focusin, focusout, key, or all	none	spinbox, entry; specifies validation mode
validatecommand or - vcmd				●												●			script to eval, {} disables	{}	spinbox, entry; must return boolean value
value													●						proper list		over from to
values																●			proper list		spinbox; content control
variable			●										●	●					name of global variable	SelectedButton value	checkbox, radiobutton, scale;
visual					●		●											●	see TK_GetVisual for info/complex	parent values	toplevel, frame, lableframe;
width	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	characters		
wrap																●	●		boolean		spinbox, text; wrap around values of data
wraplength	●		●			●				●			●						distance value	nowrap	max line length; 0-none
xscrollcommand		●		●				●								●	●		=scrollbar.set		scrollbar.config(command=w.xview)
xscrollincrement		●																	distance		canvas; increment or horiz scroll
yscrollcommand		●						●									●		widget name + set		canvas, text, listbox;
yscrollincrement		●																	distance		canvas; increment or vert scroll

tkinter Commands and Keywords

- the following are commands OTHER than those which create and define widgets, implement their attributes and options, bind actions, provide information and manage window activity. (**Commands** is the web folder name used by the TCL/TK manual pages.) In the following, {Tk / Tcl} notes mean "see that specific command page" on the Tcl/Tk manual website.

after {Tcl} (1) **after(delay in milliseconds)** *nothing executes during the delay (2) **after(delay in milliseconds, command to evaluate)**
ex: e1.after(1000, top1.bell()) <- wait a second then execute a command to play a tone

bell() {Tk} - causes a system standard sound to play: ex: widget.bell() , or in a class function, self.bell()

bind {Tk} – see pages 1 and 2

bindtags {Tk} – produces a list of associations (a window/widget, class name, or "all" keyword) called binding tags that determines how events are processed. Ex: bindlist=[widget.bindtags()] might yield : [('.!toplevel.!mylabel', 'Label', '.!toplevel', 'all')]

*Before reviewing Photo or Bitmap see note on images at the end of this Command and Keyword section. (pg 7)

tkinter Commands and Keywords cont'd

bitmap {Tk} – **(1)** do not confuse the 10 "built-in" bitmaps that can be called by name for placement on buttons (see page 1 for a list with pictures), with other (i.e., user defined) bitmap images. Example: **b1=Button(top1, bitmap='questhead')** will yield, in your GUI, something like at right:
(2) BitmapImage Class images are used to display 2 color 'xbm' images in text widgets, canvases, buttons, or labels; they can be read from strings or text files. They are concomitantly constructed and assigned to a variable with the syntax: **name = BitmapImage(file = "some path and file name" [,bg=color][,fg=color])**. There is no "easy" way to resize or convert xbm images in tkinter – it must be done manually outside the program. The bitmap image variable name (holding it's ID) is then used to display the image anywhere. For example if an image variable is created named "initials" that is 300x86 pixels: l2= Label(top1, image=initials, height=100, width=300) . Eliminate the height and width, and the label will be auto sized to fit the image if the label is too small.

clipboard {Tk} – tkinter has its own clipboard. Multiple comments on the web give conflicting reports about its efficacy exchanging data with various operating systems, but the Tk/Tcl docs do not make the claim that it does that. There are three clipboard commands for **clear()**, **append()** and **get()**. You must clear it before calling append. Examples:

```
top1.clipboard_clear()  # test clipboard - start by clearing it
top1.clipboard_append("e1 is now: " + e1.get()) # put string data on the clipboard from and entry widget named "e1"
l4.configure(text=top1.clipboard_get())  # retrieve the clipboard
```

colors {Tk} – you can find a lot about colors on www.wikipyton.com. You can also find an "official" list of shortcut color names on the Tcl/Tk web site at: https://www.tcl.tk/man/tcl8.6/TkCmd/colors.htm

console {Tk} - See: https://www.tcl.tk/man/tcl8.6/TkCmd/console.htm - beyond this doc's scope.

cursors {Tk} -define inside configure, i.e., e1.config(cursor="plus"); see the whole list at https://www.tcl.tk/man/tcl8.6/TkCmd/cursors.htm ; our favorite: "coffee_mug"

destroy {Tk} – deletes window and all children ex: root.destroy() – if all destroyed, normal exit occurs.

event {Tk} – a series of commands to manage virtual and synthesized events
w.event_add("<< your new virtual event>>", sequences)
w.event_delete("<< your new virtual event>>", sequences)
…more… see https://www.tcl.tk/man/tcl8.6/TkCmd/event.htm

focus {Tk} – 5 Python variations
w.focus_set() – moves focus to widget w
w.focus_displayof() – returns w with focus assuming it is on current display, else "None"
w.focus_force() – override window manager and force focus to w – not considered good programming: "impolite"
w.focus_get() – if your application is active, returns w with focus, else "None"
w.focus_lastfor() – returns the widget that last had the input focus in the top-level window that contains it

font {Tk} – in modern Python it is no longer necessary to load a font submodule (fkFont) to change major font characterists; 12 of 18 widgets allow setting the font it uses with a 1 to 6 item tuple in a string; usually with spaces for delimiters, for example, w.configure(font=("Arial 8 bold italic underline overstrike")). If we want a font name with multiple words, a different format/ syntax is necessary, for example: w.configure(font=("Courier New", 12, "bold")). Note that if you drastically increase the font size of a widget that already holds text, the widget size may increase to fit the text displayed.

grab {Tk} – **w.grab_set()** for example: **self.top.grab_set()** – where top is a toplevel window the grab command limits all activity to the top window;

Information Methods (winfo._x())

atom(name (a string), displayof=0) : unique integer mapped to string
atomname(id, displayof=0) : mapped to id
cells() : # of cells in colormap
children() : list of children
class() : widget class
colormapfull() :
containing(rootx, rooty, displayof=0) : widget @ this position
depth() : bit depth
exists() : true if widget exists
fpixels(distance,) : fp-# of screen pixels
geometry() : geometry sting
height() : widget height in pixels
id() : window identifier
interps(displayof=0) : TCL interpreter mem list
ismapped() : boolean; check if window created
manager() :geo mgr: grid,pack,place,canvas, text
name() : widget name
parent() : widget parents full name
pathname(displayof=0) : full window name of id window
pixels(distance,) : convert to pixels
pointerx() : x coord of pointer on the root
pointerxy() : xy coords of pointer on the root
pointery() : y coord of pointer on the root
reqheight() : min size to display widget
reqwidth() : min size needed to display widget
rgb(color,) : an RGB 3 tuple - 0 to 65535
rootx() : left edge cood rel to screen
rooty() : left edge cood rel to screen
screen() : screen name as dec integer; always ":0.0" in Windows
screencells() : # of color cells
screendepth() : deffault bit depth
screenheight() : height of widget screen
screenmmheight() : screen height in mm
screenmmwidth() : screen width in mm
screenvisual() : "psuedocolor" or "truecolor"
screenwidth() : width of screen in pixels
server() : widget screen xserver window info
toplevel() : widget root
viewable() : True is widget chain is mapped
visual() : directcolor, grayscale, pseudocolor, staticcolor, staticgray, or truecolor
visualid() : X identifier for visual this widget
visualsavailable() : list of all visuals available for widget screen
vrootheight() : height of the virtual root window for widget
vrootwidth() : width of the virtual root window for widget
vrootx() : x offset of virtual root relative to root window of widget screen
vrooty() : yoffset of virtual root relative to root window of widget screen
width() : widget width, pixels (update_idletasks)
x() : upper corner coord
y() : upper corner coord
ex: print(top1.winfo_children())

More Event Types

syntax: "<type>"

Key
KeyRelease
Button-x, *B1-B5*
ButtonRelease
Motion *<B1-Motion>*
Enter
Leave
FocusIn
FocusOut
Expose
Visibility
Destroy
Unmap
Map
Reparent
Configure
Gravity
Circulate
Property
Colormap
Activate
Deactivate
MouseWheel

tkinter Commands and Keywords cont'd

variations:
w.grab_current() - if grab is in force returns identifier, else "None"
w.grab_release() – end any grab in force
w.grab_set_global() – considered bad if not evil programming, grabs all events for everything
w.grab_status() – returns 'local' or 'global' or 'None'

grid {Tk} – along with **pack** and **place** these are the geometry managers and are just commands like the rest of this list.
image {Tk} – both bitmap and photo images are created with the constructor **w.create_image(coords, options)** for options see below.
keysyms {Tk} – TK recognized list of names recognized for bindings (ex: 'space', 'exclaim', 'quotedbl', 'percent') can be found at: https://www.tcl.tk/man/tcl8.6/TkCmd/keysyms.htm
lift - in the Python implementations, **lift(aboveThis=None)** replaces Tk's "raise(aboveThis=none)" since raise is a keyword in Python.
loadTk {Tk} – "loads binary code from a file into the application's address space and calls an initialization procedure in the package to incorporate it into an interpreter"-far beyond the scope of this doc or the knowledge of its author.
lower(belowThis=None) {Tk} – change a window's position in the stack. "All toplevel windows may be restacked with respect to each other, whatever their relative path names, *but the window manager is not obligated to strictly honor requests to restack.*"
option {Tk} – there are several variations of the option command – one of which is dubious
w.option_add(pattern,value,priority) explaination and examples by Fredrik Lundh himself in 1998 (see: http://effbot.org/zone/tkinter-option-database.htm) , one of his examples: root.option_add("*Font", "courier"). The pattern format is explained at https://www.tcl.tk/man/tcl8.6/TkCmd/option.htm#M9 and priority values are explained by Shipman at http://infohost.nmt.edu/~shipman/soft/tkinter/web/universal.html; briefly priority (default=80) are 20 – global, 40 – applications, 60 – user files, 80-option set up after app initialization.
w.option_clear() – self descriptive
w.option_readfile(filename, priority**)** – not tested
w.opton_get(name-instance key, className – class key) – our team includes folks with high SAT's, advanced degrees, and nobody who graduated below magna and after 3 days we could not find or discover a single example of this commmand that worked. Good luck. Since all widgets support **cget()**, it would be superfluous anyway. Comments if we are wrong appreciated.
options {Tk} – see standard options list: https://www.tcl.tk/man/tcl8.6/TkCmd/options.htm
photo {Tk} – pic2=pic.subsample(6,6) A full color image; currently the **PhotoImage** class reads only **GIF, PGM, and PPM**. Use **Pillow** (Imaging Library) to work with other image types. "Pillow" replaces the "Python Image Library" in **modern** Python. PhotoImage objects are simultaneously created and assigned a variable name with a command in the form **v=PhotoImage(file="file location and name")**, ex: myphoto=PhotoImage(file="C:\BigSelfie.gif") or PhotoImage can read encoded GIF files from strings. Keep a reference to your image stored in a global variable. Images are displayed with the **create_image** command. There are useful methods associated with photo object instances scattered across the web – the most concise collecton is Lundh's 1998 book in pdf form, page 112, which can be found at https://users. tricity.wsu.edu/~bobl/cpts481/an-introduction-to-tkinter.pdf. Included are **configure(), cget(), width(), height(), type(), get(x,y), put(data), put(data,bbox), read(), write(), blank(), copy(), zoom (xscale,yscale), zoom(scale), subsample(xscale, yscale), and subsample (scale).**
raise {Tk} – RAISE DOES NOT EXIST in the Python implementation; the substituted command is **"lift(aboveThis=None)"** – see **lower()**
selection {Tk} – implements the full selection functionality described in the X Inter-Client Communication Conventions Manual (ICCCM). This includes **selection_clear(), _get(), _handle(command), _own(), _own_get()**.
send {Tk} – execute a command in a different application – see https://www.tcl.tk/man/tcl8.6/TkCmd/send.htm
update() – force process of all pending events, can be used to force response to user inputs.
update_idletasks() – processes only display and window calculations – does not allow new events

Note on images: (see **bitmap** and **photo**) tkinter natively handles two kinds of images, they are a (1) PhotoImage – full color - (GIF, PGM, PPM) or a (2) BitmapImage -monochrome X11 files - with the extension ".xbm". The command to display images is: **widgetname.create_image(position given by x, y; **other options)**. The **other options** are: activeimage=, anchor=, disabledimage=, **image= (the object),** state= (normal, disabled, hidden), or tags=. The command returns the item id.

Some Handy Color Names

white	snow	linen	blanched almond	aliceblue
pink1	red4	red2	indian red	DeepPink2
saddle brown	coral1	Dark Orange3	orange2	DarkGoldenrod4
DarkGoldenrod2	gold2	yellow	LightGoldenrod2	green yellow
dark green	sea green	PaleGreen2	navy	Dodger Blue3
blue2	SkyBlue1	SteelBlue4	VioletRed4	magenta
orchid3	purple	Dark Orchid4	thistle1	lavender blush
misty rose	grey10	grey25	grey50	grey75

Running Essential Color and Font Utilities:

Color Chooser:
```
from tkinter import colorchooser
colorchooser.askcolor(initialcolor='#ff0000')
```

FONT CHOOSER: **(requires 8.6)**
```
from tkinter import *
root=Tk()
root.call('tk','fontchooser','show')
```

Please send comments/suggestions to oakey.john@yahoo.com

BIG DADDY'S PYTHON TOOLBOX For 3.5

tkinter Journeyman Reference

Widget Methods or Commands

Six widgets have only two methods or commands:

get (option) - retrieve opt value
configure (opts/values) - set options/values

Frame	**Label**	**Labelframe**
Menubotton	**Message**	**Toplevel**

BUTTON

cget	(option)
configure	(options/values)
flash	
invoke	*callback*

MESSAGEBOX

NONE

CHECKBUTTON

cget	(option)
configure	(options/values)
deselect	
flash	
invoke	
select	
toggle	

PANEDWINDOW

add	(child, option(s))
cget	(option)
configure	(options/values)
forget	(child)
identify	(xy coord)
proxy	(arg[s])
coord	Äproxy xy
forget	
place	place @ xy
sash	
coord	index
dragto	index x y
mark	index x y
place	index x y
panecget	(window, option)
paneconfigure	(child, opt/vals)
after	(child)
before	(child)
height	size
hide	boolean
minsize	x y
padx	distance
pady	distance
sticky	*nesw*
stretch	when - one of:
	always, first, last, middle, never
width	size

SCROLLBAR

scrollbars are created and attached to other widgets

Scrollbar methods

activate	element
cget	(option)
configure	(options/values)
delta	Δx Δy
fraction	x y
get	
identify	x y
set	first, last

Scrolling methods

prefix moveto	
	fraction 0=start
prefix scroll	
	# units (chr/lns)
prefix scroll	
	# pages

TEXT

bbox	index x,y,width,height
cget	option
compare	indx1 operator indx2
configure	options/values
count	option, index1/2
chars	abbrv as 'c'
displaychars	(visbl chars)
displayindices	^+wins,imgs
displaylines	lines i1 to i2
indices	all objects
lines	abbrv as "l"
xpixels	hz pix to i2
ypixels	vt pix to i2
debug	*boolean; internal ck*
delete	index1/2
dlineinfo	ind› *5 ele area info*
dump	*default is all*
all	*all info*
command	key, value, index
image	+ in dump reslt
mark	+ in dump reslt
tag	+ in dump reslt
text	+ in dump reslt
window	+ in dump reslt
edit	option
canredo	true if possilbe
canundo	true if possilbe
modified	boolean wid flag
redo	
reset	clears stacks
separator	
undo	undo last edit
get	[displaychars]
	index1, index2
image	option, *see embeded image block for values*
cget	index option val
configure	index option val
create	index option val
image	names
index	index
insert	index
mark	option, args
gravity	Name, direction
names	currently set
next	index
previous	index

TEXT -continued

set	markName, index
unset	ma markName...
peer	options args
create	newPathname, options
names	↳ peer list
pendingsync	↳ 1 if not up to date, 0 if ok
replace	index1, index2 *(w/ chars/tags)*
scan	option, args
mark	x y
dragto	x y
search switches	
these are the switches:	
	forwards
	backwards
	exact
	regexp
	nolinestop
	nocase
	count varName
	all
	overlap
	strictlimits
	elide *(hidden)*
	-- *(terminates list)*
see	index (forces visible)
sync	(forces lines calcs)
command	*exe aft ln sync*
tag	options, args **tagname +**
add	indexes
bind	seq,script
cget	option/val
configure	option/val
delete	tagName
lower	belowThis
names	index
nextrange	indexes
prevrange	indexes
raise	aboveThis
ranges	tagName
remove	indexes
window	option, args
cget	index, option
configure	index, opt/val
create	index, opt/val

TEXT -continued

names	
xview	option, args
xview	
moveto	fraction
scroll	number,what
yview	args
yview	↳ pix after last
moveto	fraction
scroll	number,what
pickplace	index
number	*obsolete*

Embedded images

align	where
image	Tk image name
name	image name
padx	pixels
pady	pixels

SPINBOX

see indices refrence block

bbox	index
cget	(option)
configure	(options/values)
delete	first, last
get	
icursor	index
identify	x y
index	index
insert	index string
invoke	element
scan	option args
mark	x y
dragto	x y
selection	*to adj selection*
adjust sel end	index
clear	
element	element
from	index
present	
range	start, end
to	index
set	string
validate	↳ *0 or 1*
xview	args chg hz pos
index	
moveto	fraction
scroll	number, what

www.wikipython.com

BIG DADDY'S PYTHON TOOLBOX For 3.5

tkinter Journeyman Reference

LISTBOX

activate	*index*
bbox	*index*
cget	option
configure	(opts/values)
curselection	
delete	first, last
get	first, last
index	index
insert	index, element
itemcget	index, option
itemconfigur	(opts/values)
	background
	foreground
	selectbackground
	selectforeground
nearest	y
nearest	
mark	x y
dragto	x y
see	index
selection	option
anchor	index
clear	first, last
includes	index
set	first, last
size	str no of elemns
xview	
2 real fractions	visible span
index	
moveto	fraction
scroll	number, what
yview	
2 real fractions	visible span
index	
moveto	fraction
scroll	number, what

CANVAS

addtag	
above	tagOrId
all	
below	tagOrId
closest	x,y,halo,start
enclosed	x1 y1 x2 y2
overlapping	x1 y1 x2 y2
withtag	tagOrId
bbox	tagOrId...
bind	tagOrId, seq, commnd
canvasx	screenx, gridspacing
canvasy	screenx, gridspacing
cget	(option)
configure	(options/values)
coords	tagOrId, x, y
coords	tagOrId, coordList
create	type xy, xy, opt/val
create	type xy, xy, opt/val

CANVAS - continued

dchars	tagOrId, first, last
delete	tagOrId, tagOrId...
dtag	tagOrId.tagToDelete
find	searchCommand, arg..
focus	tagOrId
gettags	tagOrId
icursor	tagOrId, index
imove	tagOrId, index x y
index	tagOrId, index
insert	tagOrId beforeThis
itemcget	tagOrId, option
itemconfi	tagOrId, opt/val...
lower	tagOrId
move	tagOrId
moveto	tagOrId
postscript	option/value...
channel	channelName
colormap	varName
colormode	mode
file	fileName
fontmap	varName
height	size
pageanchor	anchor
pageheight	size
pagewidth	size
pagex	position
pagey	position
rotate	boolean
width	size
x	position
y	position
raise	tagOrId aboveThis
rchars	tagOrId first last string
scale	tagOrId xOrigin yOrigin
	xScale yScale
scan	option, args
mark	x y
dragto	x y, gain
select	
adjust	tagOrId index
clear	
from	tagOrId index
item	
to	tagOrId index
type	tagOrId
xview	args
moveto	fraction
scroll	number, what
yview	
moveto	fraction
scroll	number, what

ENTRY

bbox	index
cget	option
configure	option(s)/value(s)

ENTRY - continued

delete	first, last
get	
icursor	index
index	index
insert	index string
scan	
mark	x
dragto	x
selection	
adjust	index
clear	
from	index
present	
range	start,end
index	
validate	
xview	
2 real fractions	visible span
index	to left edge
moveto	fraction
scroll	number, what
what=units or pages	

Entry INDICES:

number	character -0 1st
anchor	point
end	char after string
insert	next after cursor
sel.first	1st char
sel.last	char after select
@number	x coord

SCALE

cget	(option)
configure	(options/values)
coords	X Y list @ val point
get	x y
identify	x y
set	value

RADIOBUTTON

cget	(option)
configure	(options/values)
deselect	
flash	
invoke	
select	

WINDOW MANAGER METHODS

attributes ex: root.attributes("-fullscreen, True)

-alpha	transparency .0-1.0	-toolwindow	only close butn
-topmost	window on top	-fullscreen	NEW in 8.6
-disabled	disables window	-transparent	window trans
-type	for xII, see tcl8.6	-titlepath	

-transparentcolor transparent color index of toplevel

aspect w/h ratio **client** gets/stores WM_CLIENT_MACHINE

colormapwindows store a list of windown names

command (value=); **deiconify** normal, set focus, raise

focusmodel active|passive (default); active claims focus

forget window unmappped; can be remanaged

frame returns window identifier

geometry width x height + xoffset + yoffset

grid use setGrid instead **group** return/set widget leader

iconbitmap (bitmap=); sets window icon

iconify iconify widget | **resizable** *toplevel only*

iconmask (bitmap=); used with iconbitmap; 0-off 1-on

iconname (newName=) displayed inside widget icon

iconphoto ([default=None], [image]...) sets titlebar icon

iconposition (x,y); set icon position

iconwindow (widget); display name instead of icon

manage (widget); widget becomes standalone top level

maxsize . minsize (self, width=None,height=None) set size

overrideredirect boolean but reported to be undependable

positionfrom (who= program or user) too complex for practice

protocol (name=name of atom for protocol)(command=)

sizefrom (who= program or user) too complex for practice

stackorder ↳ toplevel list in order

state ↳ one or normal, iconic, withdrawn, icon, zoomed

title sets toplevel window title

transient (master=); if spec'ed, window is pulll-down menu

withdraw unmap widget; remap with deiconify

Scrolled Text Widget a separate tkinter.scrolledtext module is constructed like a Text class widget but has a text widget and a vertical scroll bar packed in a frame. Special Attributes: ScrolledText.frame, and Scrolledtext.vbar

To import:
from tkinter import *
import tkinter.scrolledtext
(test header on www.wikipyton.com will be useful for trying this)

To use: (example assumes a butt-load of text in a variable called "testtext")
st= tkinter.scrolledtext.ScrolledText(top1, width=50, wrap=WORD, height=8)
st.grid(row=1, column=1)
st.insert(INSERT, testtext)
See the example and get useful Latin text filler for testing on www.wikipython.com

adding ttk

ttk was a major optional addition to tkinter. It reduces the options available per widget by employing STYLES and THEMES. A widget's style is either the value of its style configuration option or the default for the widget defined by it's class. Usually the style is "T" + the class, i.e., TButton. The exception is "Treeview", not "TTreeview". A widget is made up of elements. A **style** defines a widget's elements and how they are laid out. Each element may have multiple options. Options can be different in every theme. Every widget class has a **Style()**. **THEMES** are a collection of similar styles. Stock ttk **themes** are: **winnative, clam, alt, default, classic, Vista, xpnative.** ttk replaces some tkinter widgets, has no effect on another group of tkinter widgets, and adds six brand new widgets of it own, see table: ➡

Replaced	Added
Button	Combobox
Checkbutton	Notebook
Entry	Progressbar
Radiobutton	Sizegrip
Menubutton	Seperator
Scale	Treeview
Label	**Unaffected**
LabelFrame	Text
Frame	Spinbox
PanedWindow	Listbox
Scrollbar	Message
	Toplevel
	Canvas
	Messagebox

One purpose of ttk is to simplify code by separating methods and attributes and "automating" attributes with **styles** emulating existing platforms, or at least to provide a consistent appearance. Python has a fair, if sometimes misleading, ttk summary at:
https://docs.python.org/3/library/tkinter.ttk.html
and **most** of the following is summarized from there.

To import ttk:
from tkinter import *
from tkinter.ttk import*
root=Tk()

To Use a Theme: (winnative, calm, alt, default, classic, Vista, xpnative)
style=Style()
style.theme_use("themename")

fg & bg not supported

To Configure a style:
style.configure("myStyleName", foreground="red", background="blue")

Universal Standard ttk Widget Options:
class (read only for option values), cursor (default is inherited), takefocus (in traversal: 0,1, or ""), style (to specify custom style)

Scrollable Widget Options: xscrollcommand, yscrollcommand – *use Scrollbar.set()*

Label Options: text, textvariable, underline, image, compound, width

Universal Standard ttk Widget Methods: Tkinter.Widget.cget(), tkinter.Widget.configure(), class tkinter.ttk.Widget: identify(x,y), instate (*statespec*, callback=None, *args, **kw), state(*statespec=None*)

Widge States:
active, disabled, focus, pressed, selected, background, readonly, alternate, invalid

Style: is a class used to manipulate the style database and supports the following methods:

configure(style, query_opt= None, **kw)	**map**(style, query_opt=None, **kw)	**lookup** (style, option, state=None, default=None)
theme_create (themename, parent= None, settings=None)	**theme_names()** **theme_use** (themename=None) **theme_settings** (*themename*, settings)	**element_options** (elementname) **element_names()**
element_create (elementname, etype="image", or "from", *args, **kw= border, height, padding, sticky, and/or width)		
layout (*style*, *layoutspec=None*) (None, side:whichside, sticky:nsew, unit: 0 or 1, children:[sublayout…]		

what about tix? *Deprecated since version 3.6:*

This Tk extension is unmaintained and should not be used in new code. Use tkinter.ttk instead.