Help on package tkinter:

NAME

tkinter - Wrapper functions for Tcl/Tk.

DESCRIPTION

Tkinter provides classes which allow the display, positioning and

control of widgets. Toplevel widgets are Tk and Toplevel. Other

widgets are Frame, Label, Entry, Text, Canvas, Button, Radiobutton,

Checkbutton, Scale, Listbox, Scrollbar, OptionMenu, Spinbox

LabelFrame and PanedWindow.

Properties of the widgets are specified with keyword arguments.

Keyword arguments have the same name as the corresponding resource

under Tk.

Widgets are positioned with one of the geometry managers Place, Pack

or Grid. These managers can be called with methods place, pack, grid

available in every Widget.

Actions are bound to events by resources (e.g. keyword argument

command) or with the method bind.

Example (Hello, World):

import tkinter

from tkinter.constants import \*

tk = tkinter.Tk()

frame = tkinter.Frame(tk, relief=RIDGE, borderwidth=2)

frame.pack(fill=BOTH,expand=1)

label = tkinter.Label(frame, text="Hello, World")

label.pack(fill=X, expand=1)

button = tkinter.Button(frame,text="Exit",command=tk.destroy)

button.pack(side=BOTTOM)

tk.mainloop()

PACKAGE CONTENTS

\_\_main\_\_

colorchooser

commondialog

constants

dialog

dnd

filedialog

font

messagebox

scrolledtext

simpledialog

test (package)

tix

ttk

CLASSES

builtins.object

CallWrapper

Event

Grid

Image

BitmapImage

PhotoImage

Misc

BaseWidget

Toplevel(BaseWidget, Wm)

Widget(BaseWidget, Pack, Place, Grid)

Button

Canvas(Widget, XView, YView)

Checkbutton

Entry(Widget, XView)

Frame

Label

LabelFrame

Listbox(Widget, XView, YView)

Menu

Menubutton

OptionMenu

Message

PanedWindow

Radiobutton

Scale

Scrollbar

Spinbox(Widget, XView)

Text(Widget, XView, YView)

Tk(Misc, Wm)

Pack

Place

Variable

BooleanVar

DoubleVar

IntVar

StringVar

Wm

XView

YView

builtins.str(builtins.object)

EventType(builtins.str, enum.Enum)

enum.Enum(builtins.object)

EventType(builtins.str, enum.Enum)

class BaseWidget(Misc)

| BaseWidget(master, widgetName, cnf={}, kw={}, extra=())

|

| Internal class.

|

| Method resolution order:

| BaseWidget

| Misc

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master, widgetName, cnf={}, kw={}, extra=())

| Construct a widget with the parent widget MASTER, a name WIDGETNAME

| and appropriate options.

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class BitmapImage(Image)

| BitmapImage(name=None, cnf={}, master=None, \*\*kw)

|

| Widget which can display images in XBM format.

|

| Method resolution order:

| BitmapImage

| Image

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, name=None, cnf={}, master=None, \*\*kw)

| Create a bitmap with NAME.

|

| Valid resource names: background, data, file, foreground, maskdata, maskfile.

|

| ----------------------------------------------------------------------

| Methods inherited from Image:

|

| \_\_del\_\_(self)

|

| \_\_getitem\_\_(self, key)

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return str(self).

|

| config = configure(self, \*\*kw)

|

| configure(self, \*\*kw)

| Configure the image.

|

| height(self)

| Return the height of the image.

|

| type(self)

| Return the type of the image, e.g. "photo" or "bitmap".

|

| width(self)

| Return the width of the image.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Image:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class BooleanVar(Variable)

| BooleanVar(master=None, value=None, name=None)

|

| Value holder for boolean variables.

|

| Method resolution order:

| BooleanVar

| Variable

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, value=None, name=None)

| Construct a boolean variable.

|

| MASTER can be given as master widget.

| VALUE is an optional value (defaults to False)

| NAME is an optional Tcl name (defaults to PY\_VARnum).

|

| If NAME matches an existing variable and VALUE is omitted

| then the existing value is retained.

|

| get(self)

| Return the value of the variable as a bool.

|

| initialize = set(self, value)

|

| set(self, value)

| Set the variable to VALUE.

|

| ----------------------------------------------------------------------

| Methods inherited from Variable:

|

| \_\_del\_\_(self)

| Unset the variable in Tcl.

|

| \_\_eq\_\_(self, other)

| Comparison for equality (==).

|

| Note: if the Variable's master matters to behavior

| also compare self.\_master == other.\_master

|

| \_\_str\_\_(self)

| Return the name of the variable in Tcl.

|

| trace = trace\_variable(self, mode, callback)

|

| trace\_add(self, mode, callback)

| Define a trace callback for the variable.

|

| Mode is one of "read", "write", "unset", or a list or tuple of

| such strings.

| Callback must be a function which is called when the variable is

| read, written or unset.

|

| Return the name of the callback.

|

| trace\_info(self)

| Return all trace callback information.

|

| trace\_remove(self, mode, cbname)

| Delete the trace callback for a variable.

|

| Mode is one of "read", "write", "unset" or a list or tuple of

| such strings. Must be same as were specified in trace\_add().

| cbname is the name of the callback returned from trace\_add().

|

| trace\_variable(self, mode, callback)

| Define a trace callback for the variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CALLBACK must be a function which is called when

| the variable is read, written or undefined.

|

| Return the name of the callback.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_add() instead.

|

| trace\_vdelete(self, mode, cbname)

| Delete the trace callback for a variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CBNAME is the name of the callback returned from trace\_variable or trace.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_remove() instead.

|

| trace\_vinfo(self)

| Return all trace callback information.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_info() instead.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Variable:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Data and other attributes inherited from Variable:

|

| \_\_hash\_\_ = None

class Button(Widget)

| Button(master=None, cnf={}, \*\*kw)

|

| Button widget.

|

| Method resolution order:

| Button

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a button widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| activebackground, activeforeground, anchor,

| background, bitmap, borderwidth, cursor,

| disabledforeground, font, foreground

| highlightbackground, highlightcolor,

| highlightthickness, image, justify,

| padx, pady, relief, repeatdelay,

| repeatinterval, takefocus, text,

| textvariable, underline, wraplength

|

| WIDGET-SPECIFIC OPTIONS

|

| command, compound, default, height,

| overrelief, state, width

|

| flash(self)

| Flash the button.

|

| This is accomplished by redisplaying

| the button several times, alternating between active and

| normal colors. At the end of the flash the button is left

| in the same normal/active state as when the command was

| invoked. This command is ignored if the button's state is

| disabled.

|

| invoke(self)

| Invoke the command associated with the button.

|

| The return value is the return value from the command,

| or an empty string if there is no command associated with

| the button. This command is ignored if the button's state

| is disabled.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class CallWrapper(builtins.object)

| CallWrapper(func, subst, widget)

|

| Internal class. Stores function to call when some user

| defined Tcl function is called e.g. after an event occurred.

|

| Methods defined here:

|

| \_\_call\_\_(self, \*args)

| Apply first function SUBST to arguments, than FUNC.

|

| \_\_init\_\_(self, func, subst, widget)

| Store FUNC, SUBST and WIDGET as members.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class Canvas(Widget, XView, YView)

| Canvas(master=None, cnf={}, \*\*kw)

|

| Canvas widget to display graphical elements like lines or text.

|

| Method resolution order:

| Canvas

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| XView

| YView

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a canvas widget with the parent MASTER.

|

| Valid resource names: background, bd, bg, borderwidth, closeenough,

| confine, cursor, height, highlightbackground, highlightcolor,

| highlightthickness, insertbackground, insertborderwidth,

| insertofftime, insertontime, insertwidth, offset, relief,

| scrollregion, selectbackground, selectborderwidth, selectforeground,

| state, takefocus, width, xscrollcommand, xscrollincrement,

| yscrollcommand, yscrollincrement.

|

| addtag(self, \*args)

| Internal function.

|

| addtag\_above(self, newtag, tagOrId)

| Add tag NEWTAG to all items above TAGORID.

|

| addtag\_all(self, newtag)

| Add tag NEWTAG to all items.

|

| addtag\_below(self, newtag, tagOrId)

| Add tag NEWTAG to all items below TAGORID.

|

| addtag\_closest(self, newtag, x, y, halo=None, start=None)

| Add tag NEWTAG to item which is closest to pixel at X, Y.

| If several match take the top-most.

| All items closer than HALO are considered overlapping (all are

| closests). If START is specified the next below this tag is taken.

|

| addtag\_enclosed(self, newtag, x1, y1, x2, y2)

| Add tag NEWTAG to all items in the rectangle defined

| by X1,Y1,X2,Y2.

|

| addtag\_overlapping(self, newtag, x1, y1, x2, y2)

| Add tag NEWTAG to all items which overlap the rectangle

| defined by X1,Y1,X2,Y2.

|

| addtag\_withtag(self, newtag, tagOrId)

| Add tag NEWTAG to all items with TAGORID.

|

| bbox(self, \*args)

| Return a tuple of X1,Y1,X2,Y2 coordinates for a rectangle

| which encloses all items with tags specified as arguments.

|

| canvasx(self, screenx, gridspacing=None)

| Return the canvas x coordinate of pixel position SCREENX rounded

| to nearest multiple of GRIDSPACING units.

|

| canvasy(self, screeny, gridspacing=None)

| Return the canvas y coordinate of pixel position SCREENY rounded

| to nearest multiple of GRIDSPACING units.

|

| coords(self, \*args)

| Return a list of coordinates for the item given in ARGS.

|

| create\_arc(self, \*args, \*\*kw)

| Create arc shaped region with coordinates x1,y1,x2,y2.

|

| create\_bitmap(self, \*args, \*\*kw)

| Create bitmap with coordinates x1,y1.

|

| create\_image(self, \*args, \*\*kw)

| Create image item with coordinates x1,y1.

|

| create\_line(self, \*args, \*\*kw)

| Create line with coordinates x1,y1,...,xn,yn.

|

| create\_oval(self, \*args, \*\*kw)

| Create oval with coordinates x1,y1,x2,y2.

|

| create\_polygon(self, \*args, \*\*kw)

| Create polygon with coordinates x1,y1,...,xn,yn.

|

| create\_rectangle(self, \*args, \*\*kw)

| Create rectangle with coordinates x1,y1,x2,y2.

|

| create\_text(self, \*args, \*\*kw)

| Create text with coordinates x1,y1.

|

| create\_window(self, \*args, \*\*kw)

| Create window with coordinates x1,y1,x2,y2.

|

| dchars(self, \*args)

| Delete characters of text items identified by tag or id in ARGS (possibly

| several times) from FIRST to LAST character (including).

|

| delete(self, \*args)

| Delete items identified by all tag or ids contained in ARGS.

|

| dtag(self, \*args)

| Delete tag or id given as last arguments in ARGS from items

| identified by first argument in ARGS.

|

| find(self, \*args)

| Internal function.

|

| find\_above(self, tagOrId)

| Return items above TAGORID.

|

| find\_all(self)

| Return all items.

|

| find\_below(self, tagOrId)

| Return all items below TAGORID.

|

| find\_closest(self, x, y, halo=None, start=None)

| Return item which is closest to pixel at X, Y.

| If several match take the top-most.

| All items closer than HALO are considered overlapping (all are

| closest). If START is specified the next below this tag is taken.

|

| find\_enclosed(self, x1, y1, x2, y2)

| Return all items in rectangle defined

| by X1,Y1,X2,Y2.

|

| find\_overlapping(self, x1, y1, x2, y2)

| Return all items which overlap the rectangle

| defined by X1,Y1,X2,Y2.

|

| find\_withtag(self, tagOrId)

| Return all items with TAGORID.

|

| focus(self, \*args)

| Set focus to the first item specified in ARGS.

|

| gettags(self, \*args)

| Return tags associated with the first item specified in ARGS.

|

| icursor(self, \*args)

| Set cursor at position POS in the item identified by TAGORID.

| In ARGS TAGORID must be first.

|

| index(self, \*args)

| Return position of cursor as integer in item specified in ARGS.

|

| insert(self, \*args)

| Insert TEXT in item TAGORID at position POS. ARGS must

| be TAGORID POS TEXT.

|

| itemcget(self, tagOrId, option)

| Return the resource value for an OPTION for item TAGORID.

|

| itemconfig = itemconfigure(self, tagOrId, cnf=None, \*\*kw)

|

| itemconfigure(self, tagOrId, cnf=None, \*\*kw)

| Configure resources of an item TAGORID.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method without arguments.

|

| lift = tag\_raise(self, \*args)

|

| lower = tag\_lower(self, \*args)

|

| move(self, \*args)

| Move an item TAGORID given in ARGS.

|

| postscript(self, cnf={}, \*\*kw)

| Print the contents of the canvas to a postscript

| file. Valid options: colormap, colormode, file, fontmap,

| height, pageanchor, pageheight, pagewidth, pagex, pagey,

| rotate, width, x, y.

|

| scale(self, \*args)

| Scale item TAGORID with XORIGIN, YORIGIN, XSCALE, YSCALE.

|

| scan\_dragto(self, x, y, gain=10)

| Adjust the view of the canvas to GAIN times the

| difference between X and Y and the coordinates given in

| scan\_mark.

|

| scan\_mark(self, x, y)

| Remember the current X, Y coordinates.

|

| select\_adjust(self, tagOrId, index)

| Adjust the end of the selection near the cursor of an item TAGORID to index.

|

| select\_clear(self)

| Clear the selection if it is in this widget.

|

| select\_from(self, tagOrId, index)

| Set the fixed end of a selection in item TAGORID to INDEX.

|

| select\_item(self)

| Return the item which has the selection.

|

| select\_to(self, tagOrId, index)

| Set the variable end of a selection in item TAGORID to INDEX.

|

| tag\_bind(self, tagOrId, sequence=None, func=None, add=None)

| Bind to all items with TAGORID at event SEQUENCE a call to function FUNC.

|

| An additional boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or whether it will

| replace the previous function. See bind for the return value.

|

| tag\_lower(self, \*args)

| Lower an item TAGORID given in ARGS

| (optional below another item).

|

| tag\_raise(self, \*args)

| Raise an item TAGORID given in ARGS

| (optional above another item).

|

| tag\_unbind(self, tagOrId, sequence, funcid=None)

| Unbind for all items with TAGORID for event SEQUENCE the

| function identified with FUNCID.

|

| tkraise = tag\_raise(self, \*args)

|

| type(self, tagOrId)

| Return the type of the item TAGORID.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

|

| ----------------------------------------------------------------------

| Methods inherited from XView:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

|

| ----------------------------------------------------------------------

| Methods inherited from YView:

|

| yview(self, \*args)

| Query and change the vertical position of the view.

|

| yview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total height of the canvas is off-screen to the top.

|

| yview\_scroll(self, number, what)

| Shift the y-view according to NUMBER which is measured in

| "units" or "pages" (WHAT).

class Checkbutton(Widget)

| Checkbutton(master=None, cnf={}, \*\*kw)

|

| Checkbutton widget which is either in on- or off-state.

|

| Method resolution order:

| Checkbutton

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a checkbutton widget with the parent MASTER.

|

| Valid resource names: activebackground, activeforeground, anchor,

| background, bd, bg, bitmap, borderwidth, command, cursor,

| disabledforeground, fg, font, foreground, height,

| highlightbackground, highlightcolor, highlightthickness, image,

| indicatoron, justify, offvalue, onvalue, padx, pady, relief,

| selectcolor, selectimage, state, takefocus, text, textvariable,

| underline, variable, width, wraplength.

|

| deselect(self)

| Put the button in off-state.

|

| flash(self)

| Flash the button.

|

| invoke(self)

| Toggle the button and invoke a command if given as resource.

|

| select(self)

| Put the button in on-state.

|

| toggle(self)

| Toggle the button.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class DoubleVar(Variable)

| DoubleVar(master=None, value=None, name=None)

|

| Value holder for float variables.

|

| Method resolution order:

| DoubleVar

| Variable

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, value=None, name=None)

| Construct a float variable.

|

| MASTER can be given as master widget.

| VALUE is an optional value (defaults to 0.0)

| NAME is an optional Tcl name (defaults to PY\_VARnum).

|

| If NAME matches an existing variable and VALUE is omitted

| then the existing value is retained.

|

| get(self)

| Return the value of the variable as a float.

|

| ----------------------------------------------------------------------

| Methods inherited from Variable:

|

| \_\_del\_\_(self)

| Unset the variable in Tcl.

|

| \_\_eq\_\_(self, other)

| Comparison for equality (==).

|

| Note: if the Variable's master matters to behavior

| also compare self.\_master == other.\_master

|

| \_\_str\_\_(self)

| Return the name of the variable in Tcl.

|

| initialize = set(self, value)

|

| set(self, value)

| Set the variable to VALUE.

|

| trace = trace\_variable(self, mode, callback)

|

| trace\_add(self, mode, callback)

| Define a trace callback for the variable.

|

| Mode is one of "read", "write", "unset", or a list or tuple of

| such strings.

| Callback must be a function which is called when the variable is

| read, written or unset.

|

| Return the name of the callback.

|

| trace\_info(self)

| Return all trace callback information.

|

| trace\_remove(self, mode, cbname)

| Delete the trace callback for a variable.

|

| Mode is one of "read", "write", "unset" or a list or tuple of

| such strings. Must be same as were specified in trace\_add().

| cbname is the name of the callback returned from trace\_add().

|

| trace\_variable(self, mode, callback)

| Define a trace callback for the variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CALLBACK must be a function which is called when

| the variable is read, written or undefined.

|

| Return the name of the callback.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_add() instead.

|

| trace\_vdelete(self, mode, cbname)

| Delete the trace callback for a variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CBNAME is the name of the callback returned from trace\_variable or trace.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_remove() instead.

|

| trace\_vinfo(self)

| Return all trace callback information.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_info() instead.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Variable:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Data and other attributes inherited from Variable:

|

| \_\_hash\_\_ = None

class Entry(Widget, XView)

| Entry(master=None, cnf={}, \*\*kw)

|

| Entry widget which allows displaying simple text.

|

| Method resolution order:

| Entry

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| XView

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct an entry widget with the parent MASTER.

|

| Valid resource names: background, bd, bg, borderwidth, cursor,

| exportselection, fg, font, foreground, highlightbackground,

| highlightcolor, highlightthickness, insertbackground,

| insertborderwidth, insertofftime, insertontime, insertwidth,

| invalidcommand, invcmd, justify, relief, selectbackground,

| selectborderwidth, selectforeground, show, state, takefocus,

| textvariable, validate, validatecommand, vcmd, width,

| xscrollcommand.

|

| delete(self, first, last=None)

| Delete text from FIRST to LAST (not included).

|

| get(self)

| Return the text.

|

| icursor(self, index)

| Insert cursor at INDEX.

|

| index(self, index)

| Return position of cursor.

|

| insert(self, index, string)

| Insert STRING at INDEX.

|

| scan\_dragto(self, x)

| Adjust the view of the canvas to 10 times the

| difference between X and Y and the coordinates given in

| scan\_mark.

|

| scan\_mark(self, x)

| Remember the current X, Y coordinates.

|

| select\_adjust = selection\_adjust(self, index)

|

| select\_clear = selection\_clear(self)

|

| select\_from = selection\_from(self, index)

|

| select\_present = selection\_present(self)

|

| select\_range = selection\_range(self, start, end)

|

| select\_to = selection\_to(self, index)

|

| selection\_adjust(self, index)

| Adjust the end of the selection near the cursor to INDEX.

|

| selection\_clear(self)

| Clear the selection if it is in this widget.

|

| selection\_from(self, index)

| Set the fixed end of a selection to INDEX.

|

| selection\_present(self)

| Return True if there are characters selected in the entry, False

| otherwise.

|

| selection\_range(self, start, end)

| Set the selection from START to END (not included).

|

| selection\_to(self, index)

| Set the variable end of a selection to INDEX.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

|

| ----------------------------------------------------------------------

| Methods inherited from XView:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

class Event(builtins.object)

| Container for the properties of an event.

|

| Instances of this type are generated if one of the following events occurs:

|

| KeyPress, KeyRelease - for keyboard events

| ButtonPress, ButtonRelease, Motion, Enter, Leave, MouseWheel - for mouse events

| Visibility, Unmap, Map, Expose, FocusIn, FocusOut, Circulate,

| Colormap, Gravity, Reparent, Property, Destroy, Activate,

| Deactivate - for window events.

|

| If a callback function for one of these events is registered

| using bind, bind\_all, bind\_class, or tag\_bind, the callback is

| called with an Event as first argument. It will have the

| following attributes (in braces are the event types for which

| the attribute is valid):

|

| serial - serial number of event

| num - mouse button pressed (ButtonPress, ButtonRelease)

| focus - whether the window has the focus (Enter, Leave)

| height - height of the exposed window (Configure, Expose)

| width - width of the exposed window (Configure, Expose)

| keycode - keycode of the pressed key (KeyPress, KeyRelease)

| state - state of the event as a number (ButtonPress, ButtonRelease,

| Enter, KeyPress, KeyRelease,

| Leave, Motion)

| state - state as a string (Visibility)

| time - when the event occurred

| x - x-position of the mouse

| y - y-position of the mouse

| x\_root - x-position of the mouse on the screen

| (ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion)

| y\_root - y-position of the mouse on the screen

| (ButtonPress, ButtonRelease, KeyPress, KeyRelease, Motion)

| char - pressed character (KeyPress, KeyRelease)

| send\_event - see X/Windows documentation

| keysym - keysym of the event as a string (KeyPress, KeyRelease)

| keysym\_num - keysym of the event as a number (KeyPress, KeyRelease)

| type - type of the event as a number

| widget - widget in which the event occurred

| delta - delta of wheel movement (MouseWheel)

|

| Methods defined here:

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class EventType(builtins.str, enum.Enum)

| EventType(value, names=None, \*, module=None, qualname=None, type=None, start=1)

|

| An enumeration.

|

| Method resolution order:

| EventType

| builtins.str

| enum.Enum

| builtins.object

|

| Data and other attributes defined here:

|

| Activate = <EventType.Activate: '36'>

|

| ButtonPress = <EventType.ButtonPress: '4'>

|

| ButtonRelease = <EventType.ButtonRelease: '5'>

|

| Circulate = <EventType.Circulate: '26'>

|

| CirculateRequest = <EventType.CirculateRequest: '27'>

|

| ClientMessage = <EventType.ClientMessage: '33'>

|

| Colormap = <EventType.Colormap: '32'>

|

| Configure = <EventType.Configure: '22'>

|

| ConfigureRequest = <EventType.ConfigureRequest: '23'>

|

| Create = <EventType.Create: '16'>

|

| Deactivate = <EventType.Deactivate: '37'>

|

| Destroy = <EventType.Destroy: '17'>

|

| Enter = <EventType.Enter: '7'>

|

| Expose = <EventType.Expose: '12'>

|

| FocusIn = <EventType.FocusIn: '9'>

|

| FocusOut = <EventType.FocusOut: '10'>

|

| GraphicsExpose = <EventType.GraphicsExpose: '13'>

|

| Gravity = <EventType.Gravity: '24'>

|

| KeyPress = <EventType.KeyPress: '2'>

|

| KeyRelease = <EventType.KeyRelease: '3'>

|

| Keymap = <EventType.Keymap: '11'>

|

| Leave = <EventType.Leave: '8'>

|

| Map = <EventType.Map: '19'>

|

| MapRequest = <EventType.MapRequest: '20'>

|

| Mapping = <EventType.Mapping: '34'>

|

| Motion = <EventType.Motion: '6'>

|

| MouseWheel = <EventType.MouseWheel: '38'>

|

| NoExpose = <EventType.NoExpose: '14'>

|

| Property = <EventType.Property: '28'>

|

| Reparent = <EventType.Reparent: '21'>

|

| ResizeRequest = <EventType.ResizeRequest: '25'>

|

| Selection = <EventType.Selection: '31'>

|

| SelectionClear = <EventType.SelectionClear: '29'>

|

| SelectionRequest = <EventType.SelectionRequest: '30'>

|

| Unmap = <EventType.Unmap: '18'>

|

| VirtualEvent = <EventType.VirtualEvent: '35'>

|

| Visibility = <EventType.Visibility: '15'>

|

| ----------------------------------------------------------------------

| Data descriptors inherited from enum.Enum:

|

| name

| The name of the Enum member.

|

| value

| The value of the Enum member.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from enum.EnumMeta:

|

| \_\_members\_\_

| Returns a mapping of member name->value.

|

| This mapping lists all enum members, including aliases. Note that this

| is a read-only view of the internal mapping.

class Frame(Widget)

| Frame(master=None, cnf={}, \*\*kw)

|

| Frame widget which may contain other widgets and can have a 3D border.

|

| Method resolution order:

| Frame

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a frame widget with the parent MASTER.

|

| Valid resource names: background, bd, bg, borderwidth, class,

| colormap, container, cursor, height, highlightbackground,

| highlightcolor, highlightthickness, relief, takefocus, visual, width.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Grid(builtins.object)

| Geometry manager Grid.

|

| Base class to use the methods grid\_\* in every widget.

|

| Methods defined here:

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = grid\_configure(self, cnf={}, \*\*kw)

|

| configure = grid\_configure(self, cnf={}, \*\*kw)

|

| forget = grid\_forget(self)

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| info = grid\_info(self)

|

| location = grid\_location(self, x, y)

|

| propagate = grid\_propagate(self, flag=['\_noarg\_'])

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| size = grid\_size(self)

|

| slaves = grid\_slaves(self, row=None, column=None)

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class Image(builtins.object)

| Image(imgtype, name=None, cnf={}, master=None, \*\*kw)

|

| Base class for images.

|

| Methods defined here:

|

| \_\_del\_\_(self)

|

| \_\_getitem\_\_(self, key)

|

| \_\_init\_\_(self, imgtype, name=None, cnf={}, master=None, \*\*kw)

| Initialize self. See help(type(self)) for accurate signature.

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return str(self).

|

| config = configure(self, \*\*kw)

|

| configure(self, \*\*kw)

| Configure the image.

|

| height(self)

| Return the height of the image.

|

| type(self)

| Return the type of the image, e.g. "photo" or "bitmap".

|

| width(self)

| Return the width of the image.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class IntVar(Variable)

| IntVar(master=None, value=None, name=None)

|

| Value holder for integer variables.

|

| Method resolution order:

| IntVar

| Variable

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, value=None, name=None)

| Construct an integer variable.

|

| MASTER can be given as master widget.

| VALUE is an optional value (defaults to 0)

| NAME is an optional Tcl name (defaults to PY\_VARnum).

|

| If NAME matches an existing variable and VALUE is omitted

| then the existing value is retained.

|

| get(self)

| Return the value of the variable as an integer.

|

| ----------------------------------------------------------------------

| Methods inherited from Variable:

|

| \_\_del\_\_(self)

| Unset the variable in Tcl.

|

| \_\_eq\_\_(self, other)

| Comparison for equality (==).

|

| Note: if the Variable's master matters to behavior

| also compare self.\_master == other.\_master

|

| \_\_str\_\_(self)

| Return the name of the variable in Tcl.

|

| initialize = set(self, value)

|

| set(self, value)

| Set the variable to VALUE.

|

| trace = trace\_variable(self, mode, callback)

|

| trace\_add(self, mode, callback)

| Define a trace callback for the variable.

|

| Mode is one of "read", "write", "unset", or a list or tuple of

| such strings.

| Callback must be a function which is called when the variable is

| read, written or unset.

|

| Return the name of the callback.

|

| trace\_info(self)

| Return all trace callback information.

|

| trace\_remove(self, mode, cbname)

| Delete the trace callback for a variable.

|

| Mode is one of "read", "write", "unset" or a list or tuple of

| such strings. Must be same as were specified in trace\_add().

| cbname is the name of the callback returned from trace\_add().

|

| trace\_variable(self, mode, callback)

| Define a trace callback for the variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CALLBACK must be a function which is called when

| the variable is read, written or undefined.

|

| Return the name of the callback.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_add() instead.

|

| trace\_vdelete(self, mode, cbname)

| Delete the trace callback for a variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CBNAME is the name of the callback returned from trace\_variable or trace.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_remove() instead.

|

| trace\_vinfo(self)

| Return all trace callback information.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_info() instead.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Variable:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Data and other attributes inherited from Variable:

|

| \_\_hash\_\_ = None

class Label(Widget)

| Label(master=None, cnf={}, \*\*kw)

|

| Label widget which can display text and bitmaps.

|

| Method resolution order:

| Label

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a label widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| activebackground, activeforeground, anchor,

| background, bitmap, borderwidth, cursor,

| disabledforeground, font, foreground,

| highlightbackground, highlightcolor,

| highlightthickness, image, justify,

| padx, pady, relief, takefocus, text,

| textvariable, underline, wraplength

|

| WIDGET-SPECIFIC OPTIONS

|

| height, state, width

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class LabelFrame(Widget)

| LabelFrame(master=None, cnf={}, \*\*kw)

|

| labelframe widget.

|

| Method resolution order:

| LabelFrame

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a labelframe widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| borderwidth, cursor, font, foreground,

| highlightbackground, highlightcolor,

| highlightthickness, padx, pady, relief,

| takefocus, text

|

| WIDGET-SPECIFIC OPTIONS

|

| background, class, colormap, container,

| height, labelanchor, labelwidget,

| visual, width

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Listbox(Widget, XView, YView)

| Listbox(master=None, cnf={}, \*\*kw)

|

| Listbox widget which can display a list of strings.

|

| Method resolution order:

| Listbox

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| XView

| YView

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a listbox widget with the parent MASTER.

|

| Valid resource names: background, bd, bg, borderwidth, cursor,

| exportselection, fg, font, foreground, height, highlightbackground,

| highlightcolor, highlightthickness, relief, selectbackground,

| selectborderwidth, selectforeground, selectmode, setgrid, takefocus,

| width, xscrollcommand, yscrollcommand, listvariable.

|

| activate(self, index)

| Activate item identified by INDEX.

|

| bbox(self, index)

| Return a tuple of X1,Y1,X2,Y2 coordinates for a rectangle

| which encloses the item identified by the given index.

|

| curselection(self)

| Return the indices of currently selected item.

|

| delete(self, first, last=None)

| Delete items from FIRST to LAST (included).

|

| get(self, first, last=None)

| Get list of items from FIRST to LAST (included).

|

| index(self, index)

| Return index of item identified with INDEX.

|

| insert(self, index, \*elements)

| Insert ELEMENTS at INDEX.

|

| itemcget(self, index, option)

| Return the resource value for an ITEM and an OPTION.

|

| itemconfig = itemconfigure(self, index, cnf=None, \*\*kw)

|

| itemconfigure(self, index, cnf=None, \*\*kw)

| Configure resources of an ITEM.

|

| The values for resources are specified as keyword arguments.

| To get an overview about the allowed keyword arguments

| call the method without arguments.

| Valid resource names: background, bg, foreground, fg,

| selectbackground, selectforeground.

|

| nearest(self, y)

| Get index of item which is nearest to y coordinate Y.

|

| scan\_dragto(self, x, y)

| Adjust the view of the listbox to 10 times the

| difference between X and Y and the coordinates given in

| scan\_mark.

|

| scan\_mark(self, x, y)

| Remember the current X, Y coordinates.

|

| see(self, index)

| Scroll such that INDEX is visible.

|

| select\_anchor = selection\_anchor(self, index)

|

| select\_clear = selection\_clear(self, first, last=None)

|

| select\_includes = selection\_includes(self, index)

|

| select\_set = selection\_set(self, first, last=None)

|

| selection\_anchor(self, index)

| Set the fixed end oft the selection to INDEX.

|

| selection\_clear(self, first, last=None)

| Clear the selection from FIRST to LAST (included).

|

| selection\_includes(self, index)

| Return 1 if INDEX is part of the selection.

|

| selection\_set(self, first, last=None)

| Set the selection from FIRST to LAST (included) without

| changing the currently selected elements.

|

| size(self)

| Return the number of elements in the listbox.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

|

| ----------------------------------------------------------------------

| Methods inherited from XView:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

|

| ----------------------------------------------------------------------

| Methods inherited from YView:

|

| yview(self, \*args)

| Query and change the vertical position of the view.

|

| yview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total height of the canvas is off-screen to the top.

|

| yview\_scroll(self, number, what)

| Shift the y-view according to NUMBER which is measured in

| "units" or "pages" (WHAT).

class Menu(Widget)

| Menu(master=None, cnf={}, \*\*kw)

|

| Menu widget which allows displaying menu bars, pull-down menus and pop-up menus.

|

| Method resolution order:

| Menu

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct menu widget with the parent MASTER.

|

| Valid resource names: activebackground, activeborderwidth,

| activeforeground, background, bd, bg, borderwidth, cursor,

| disabledforeground, fg, font, foreground, postcommand, relief,

| selectcolor, takefocus, tearoff, tearoffcommand, title, type.

|

| activate(self, index)

| Activate entry at INDEX.

|

| add(self, itemType, cnf={}, \*\*kw)

| Internal function.

|

| add\_cascade(self, cnf={}, \*\*kw)

| Add hierarchical menu item.

|

| add\_checkbutton(self, cnf={}, \*\*kw)

| Add checkbutton menu item.

|

| add\_command(self, cnf={}, \*\*kw)

| Add command menu item.

|

| add\_radiobutton(self, cnf={}, \*\*kw)

| Addd radio menu item.

|

| add\_separator(self, cnf={}, \*\*kw)

| Add separator.

|

| delete(self, index1, index2=None)

| Delete menu items between INDEX1 and INDEX2 (included).

|

| entrycget(self, index, option)

| Return the resource value of a menu item for OPTION at INDEX.

|

| entryconfig = entryconfigure(self, index, cnf=None, \*\*kw)

|

| entryconfigure(self, index, cnf=None, \*\*kw)

| Configure a menu item at INDEX.

|

| index(self, index)

| Return the index of a menu item identified by INDEX.

|

| insert(self, index, itemType, cnf={}, \*\*kw)

| Internal function.

|

| insert\_cascade(self, index, cnf={}, \*\*kw)

| Add hierarchical menu item at INDEX.

|

| insert\_checkbutton(self, index, cnf={}, \*\*kw)

| Add checkbutton menu item at INDEX.

|

| insert\_command(self, index, cnf={}, \*\*kw)

| Add command menu item at INDEX.

|

| insert\_radiobutton(self, index, cnf={}, \*\*kw)

| Addd radio menu item at INDEX.

|

| insert\_separator(self, index, cnf={}, \*\*kw)

| Add separator at INDEX.

|

| invoke(self, index)

| Invoke a menu item identified by INDEX and execute

| the associated command.

|

| post(self, x, y)

| Display a menu at position X,Y.

|

| tk\_popup(self, x, y, entry='')

| Post the menu at position X,Y with entry ENTRY.

|

| type(self, index)

| Return the type of the menu item at INDEX.

|

| unpost(self)

| Unmap a menu.

|

| xposition(self, index)

| Return the x-position of the leftmost pixel of the menu item

| at INDEX.

|

| yposition(self, index)

| Return the y-position of the topmost pixel of the menu item at INDEX.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Menubutton(Widget)

| Menubutton(master=None, cnf={}, \*\*kw)

|

| Menubutton widget, obsolete since Tk8.0.

|

| Method resolution order:

| Menubutton

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a widget with the parent widget MASTER, a name WIDGETNAME

| and appropriate options.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Message(Widget)

| Message(master=None, cnf={}, \*\*kw)

|

| Message widget to display multiline text. Obsolete since Label does it too.

|

| Method resolution order:

| Message

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a widget with the parent widget MASTER, a name WIDGETNAME

| and appropriate options.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Misc(builtins.object)

| Internal class.

|

| Base class which defines methods common for interior widgets.

|

| Methods defined here:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| destroy(self)

| Internal function.

|

| Delete all Tcl commands created for

| this widget in the Tcl interpreter.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class OptionMenu(Menubutton)

| OptionMenu(master, variable, value, \*values, \*\*kwargs)

|

| OptionMenu which allows the user to select a value from a menu.

|

| Method resolution order:

| OptionMenu

| Menubutton

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_getitem\_\_(self, name)

| Return the resource value for a KEY given as string.

|

| \_\_init\_\_(self, master, variable, value, \*values, \*\*kwargs)

| Construct an optionmenu widget with the parent MASTER, with

| the resource textvariable set to VARIABLE, the initially selected

| value VALUE, the other menu values VALUES and an additional

| keyword argument command.

|

| destroy(self)

| Destroy this widget and the associated menu.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Pack(builtins.object)

| Geometry manager Pack.

|

| Base class to use the methods pack\_\* in every widget.

|

| Methods defined here:

|

| config = pack\_configure(self, cnf={}, \*\*kw)

|

| configure = pack\_configure(self, cnf={}, \*\*kw)

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| slaves = pack\_slaves(self)

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class PanedWindow(Widget)

| PanedWindow(master=None, cnf={}, \*\*kw)

|

| panedwindow widget.

|

| Method resolution order:

| PanedWindow

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a panedwindow widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| background, borderwidth, cursor, height,

| orient, relief, width

|

| WIDGET-SPECIFIC OPTIONS

|

| handlepad, handlesize, opaqueresize,

| sashcursor, sashpad, sashrelief,

| sashwidth, showhandle,

|

| add(self, child, \*\*kw)

| Add a child widget to the panedwindow in a new pane.

|

| The child argument is the name of the child widget

| followed by pairs of arguments that specify how to

| manage the windows. The possible options and values

| are the ones accepted by the paneconfigure method.

|

| forget = remove(self, child)

|

| identify(self, x, y)

| Identify the panedwindow component at point x, y

|

| If the point is over a sash or a sash handle, the result

| is a two element list containing the index of the sash or

| handle, and a word indicating whether it is over a sash

| or a handle, such as {0 sash} or {2 handle}. If the point

| is over any other part of the panedwindow, the result is

| an empty list.

|

| panecget(self, child, option)

| Query a management option for window.

|

| Option may be any value allowed by the paneconfigure subcommand

|

| paneconfig = paneconfigure(self, tagOrId, cnf=None, \*\*kw)

|

| paneconfigure(self, tagOrId, cnf=None, \*\*kw)

| Query or modify the management options for window.

|

| If no option is specified, returns a list describing all

| of the available options for pathName. If option is

| specified with no value, then the command returns a list

| describing the one named option (this list will be identical

| to the corresponding sublist of the value returned if no

| option is specified). If one or more option-value pairs are

| specified, then the command modifies the given widget

| option(s) to have the given value(s); in this case the

| command returns an empty string. The following options

| are supported:

|

| after window

| Insert the window after the window specified. window

| should be the name of a window already managed by pathName.

| before window

| Insert the window before the window specified. window

| should be the name of a window already managed by pathName.

| height size

| Specify a height for the window. The height will be the

| outer dimension of the window including its border, if

| any. If size is an empty string, or if -height is not

| specified, then the height requested internally by the

| window will be used initially; the height may later be

| adjusted by the movement of sashes in the panedwindow.

| Size may be any value accepted by Tk\_GetPixels.

| minsize n

| Specifies that the size of the window cannot be made

| less than n. This constraint only affects the size of

| the widget in the paned dimension -- the x dimension

| for horizontal panedwindows, the y dimension for

| vertical panedwindows. May be any value accepted by

| Tk\_GetPixels.

| padx n

| Specifies a non-negative value indicating how much

| extra space to leave on each side of the window in

| the X-direction. The value may have any of the forms

| accepted by Tk\_GetPixels.

| pady n

| Specifies a non-negative value indicating how much

| extra space to leave on each side of the window in

| the Y-direction. The value may have any of the forms

| accepted by Tk\_GetPixels.

| sticky style

| If a window's pane is larger than the requested

| dimensions of the window, this option may be used

| to position (or stretch) the window within its pane.

| Style is a string that contains zero or more of the

| characters n, s, e or w. The string can optionally

| contains spaces or commas, but they are ignored. Each

| letter refers to a side (north, south, east, or west)

| that the window will "stick" to. If both n and s

| (or e and w) are specified, the window will be

| stretched to fill the entire height (or width) of

| its cavity.

| width size

| Specify a width for the window. The width will be

| the outer dimension of the window including its

| border, if any. If size is an empty string, or

| if -width is not specified, then the width requested

| internally by the window will be used initially; the

| width may later be adjusted by the movement of sashes

| in the panedwindow. Size may be any value accepted by

| Tk\_GetPixels.

|

| panes(self)

| Returns an ordered list of the child panes.

|

| proxy(self, \*args)

| Internal function.

|

| proxy\_coord(self)

| Return the x and y pair of the most recent proxy location

|

| proxy\_forget(self)

| Remove the proxy from the display.

|

| proxy\_place(self, x, y)

| Place the proxy at the given x and y coordinates.

|

| remove(self, child)

| Remove the pane containing child from the panedwindow

|

| All geometry management options for child will be forgotten.

|

| sash(self, \*args)

| Internal function.

|

| sash\_coord(self, index)

| Return the current x and y pair for the sash given by index.

|

| Index must be an integer between 0 and 1 less than the

| number of panes in the panedwindow. The coordinates given are

| those of the top left corner of the region containing the sash.

| pathName sash dragto index x y This command computes the

| difference between the given coordinates and the coordinates

| given to the last sash coord command for the given sash. It then

| moves that sash the computed difference. The return value is the

| empty string.

|

| sash\_mark(self, index)

| Records x and y for the sash given by index;

|

| Used in conjunction with later dragto commands to move the sash.

|

| sash\_place(self, index, x, y)

| Place the sash given by index at the given coordinates

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class PhotoImage(Image)

| PhotoImage(name=None, cnf={}, master=None, \*\*kw)

|

| Widget which can display images in PGM, PPM, GIF, PNG format.

|

| Method resolution order:

| PhotoImage

| Image

| builtins.object

|

| Methods defined here:

|

| \_\_getitem\_\_(self, key)

| # XXX config

|

| \_\_init\_\_(self, name=None, cnf={}, master=None, \*\*kw)

| Create an image with NAME.

|

| Valid resource names: data, format, file, gamma, height, palette,

| width.

|

| blank(self)

| Display a transparent image.

|

| cget(self, option)

| Return the value of OPTION.

|

| copy(self)

| Return a new PhotoImage with the same image as this widget.

|

| get(self, x, y)

| Return the color (red, green, blue) of the pixel at X,Y.

|

| put(self, data, to=None)

| Put row formatted colors to image starting from

| position TO, e.g. image.put("{red green} {blue yellow}", to=(4,6))

|

| subsample(self, x, y='')

| Return a new PhotoImage based on the same image as this widget

| but use only every Xth or Yth pixel. If y is not given, the

| default value is the same as x.

|

| write(self, filename, format=None, from\_coords=None)

| Write image to file FILENAME in FORMAT starting from

| position FROM\_COORDS.

|

| zoom(self, x, y='')

| Return a new PhotoImage with the same image as this widget

| but zoom it with a factor of x in the X direction and y in the Y

| direction. If y is not given, the default value is the same as x.

|

| ----------------------------------------------------------------------

| Methods inherited from Image:

|

| \_\_del\_\_(self)

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return str(self).

|

| config = configure(self, \*\*kw)

|

| configure(self, \*\*kw)

| Configure the image.

|

| height(self)

| Return the height of the image.

|

| type(self)

| Return the type of the image, e.g. "photo" or "bitmap".

|

| width(self)

| Return the width of the image.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Image:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class Place(builtins.object)

| Geometry manager Place.

|

| Base class to use the methods place\_\* in every widget.

|

| Methods defined here:

|

| config = place\_configure(self, cnf={}, \*\*kw)

|

| configure = place\_configure(self, cnf={}, \*\*kw)

|

| forget = place\_forget(self)

|

| info = place\_info(self)

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| slaves = place\_slaves(self)

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class Radiobutton(Widget)

| Radiobutton(master=None, cnf={}, \*\*kw)

|

| Radiobutton widget which shows only one of several buttons in on-state.

|

| Method resolution order:

| Radiobutton

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a radiobutton widget with the parent MASTER.

|

| Valid resource names: activebackground, activeforeground, anchor,

| background, bd, bg, bitmap, borderwidth, command, cursor,

| disabledforeground, fg, font, foreground, height,

| highlightbackground, highlightcolor, highlightthickness, image,

| indicatoron, justify, padx, pady, relief, selectcolor, selectimage,

| state, takefocus, text, textvariable, underline, value, variable,

| width, wraplength.

|

| deselect(self)

| Put the button in off-state.

|

| flash(self)

| Flash the button.

|

| invoke(self)

| Toggle the button and invoke a command if given as resource.

|

| select(self)

| Put the button in on-state.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Scale(Widget)

| Scale(master=None, cnf={}, \*\*kw)

|

| Scale widget which can display a numerical scale.

|

| Method resolution order:

| Scale

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a scale widget with the parent MASTER.

|

| Valid resource names: activebackground, background, bigincrement, bd,

| bg, borderwidth, command, cursor, digits, fg, font, foreground, from,

| highlightbackground, highlightcolor, highlightthickness, label,

| length, orient, relief, repeatdelay, repeatinterval, resolution,

| showvalue, sliderlength, sliderrelief, state, takefocus,

| tickinterval, to, troughcolor, variable, width.

|

| coords(self, value=None)

| Return a tuple (X,Y) of the point along the centerline of the

| trough that corresponds to VALUE or the current value if None is

| given.

|

| get(self)

| Get the current value as integer or float.

|

| identify(self, x, y)

| Return where the point X,Y lies. Valid return values are "slider",

| "though1" and "though2".

|

| set(self, value)

| Set the value to VALUE.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Scrollbar(Widget)

| Scrollbar(master=None, cnf={}, \*\*kw)

|

| Scrollbar widget which displays a slider at a certain position.

|

| Method resolution order:

| Scrollbar

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a scrollbar widget with the parent MASTER.

|

| Valid resource names: activebackground, activerelief,

| background, bd, bg, borderwidth, command, cursor,

| elementborderwidth, highlightbackground,

| highlightcolor, highlightthickness, jump, orient,

| relief, repeatdelay, repeatinterval, takefocus,

| troughcolor, width.

|

| activate(self, index=None)

| Marks the element indicated by index as active.

| The only index values understood by this method are "arrow1",

| "slider", or "arrow2". If any other value is specified then no

| element of the scrollbar will be active. If index is not specified,

| the method returns the name of the element that is currently active,

| or None if no element is active.

|

| delta(self, deltax, deltay)

| Return the fractional change of the scrollbar setting if it

| would be moved by DELTAX or DELTAY pixels.

|

| fraction(self, x, y)

| Return the fractional value which corresponds to a slider

| position of X,Y.

|

| get(self)

| Return the current fractional values (upper and lower end)

| of the slider position.

|

| identify(self, x, y)

| Return the element under position X,Y as one of

| "arrow1","slider","arrow2" or "".

|

| set(self, first, last)

| Set the fractional values of the slider position (upper and

| lower ends as value between 0 and 1).

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Spinbox(Widget, XView)

| Spinbox(master=None, cnf={}, \*\*kw)

|

| spinbox widget.

|

| Method resolution order:

| Spinbox

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| XView

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a spinbox widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| activebackground, background, borderwidth,

| cursor, exportselection, font, foreground,

| highlightbackground, highlightcolor,

| highlightthickness, insertbackground,

| insertborderwidth, insertofftime,

| insertontime, insertwidth, justify, relief,

| repeatdelay, repeatinterval,

| selectbackground, selectborderwidth

| selectforeground, takefocus, textvariable

| xscrollcommand.

|

| WIDGET-SPECIFIC OPTIONS

|

| buttonbackground, buttoncursor,

| buttondownrelief, buttonuprelief,

| command, disabledbackground,

| disabledforeground, format, from,

| invalidcommand, increment,

| readonlybackground, state, to,

| validate, validatecommand values,

| width, wrap,

|

| bbox(self, index)

| Return a tuple of X1,Y1,X2,Y2 coordinates for a

| rectangle which encloses the character given by index.

|

| The first two elements of the list give the x and y

| coordinates of the upper-left corner of the screen

| area covered by the character (in pixels relative

| to the widget) and the last two elements give the

| width and height of the character, in pixels. The

| bounding box may refer to a region outside the

| visible area of the window.

|

| delete(self, first, last=None)

| Delete one or more elements of the spinbox.

|

| First is the index of the first character to delete,

| and last is the index of the character just after

| the last one to delete. If last isn't specified it

| defaults to first+1, i.e. a single character is

| deleted. This command returns an empty string.

|

| get(self)

| Returns the spinbox's string

|

| icursor(self, index)

| Alter the position of the insertion cursor.

|

| The insertion cursor will be displayed just before

| the character given by index. Returns an empty string

|

| identify(self, x, y)

| Returns the name of the widget at position x, y

|

| Return value is one of: none, buttondown, buttonup, entry

|

| index(self, index)

| Returns the numerical index corresponding to index

|

| insert(self, index, s)

| Insert string s at index

|

| Returns an empty string.

|

| invoke(self, element)

| Causes the specified element to be invoked

|

| The element could be buttondown or buttonup

| triggering the action associated with it.

|

| scan(self, \*args)

| Internal function.

|

| scan\_dragto(self, x)

| Compute the difference between the given x argument

| and the x argument to the last scan mark command

|

| It then adjusts the view left or right by 10 times the

| difference in x-coordinates. This command is typically

| associated with mouse motion events in the widget, to

| produce the effect of dragging the spinbox at high speed

| through the window. The return value is an empty string.

|

| scan\_mark(self, x)

| Records x and the current view in the spinbox window;

|

| used in conjunction with later scan dragto commands.

| Typically this command is associated with a mouse button

| press in the widget. It returns an empty string.

|

| selection(self, \*args)

| Internal function.

|

| selection\_adjust(self, index)

| Locate the end of the selection nearest to the character

| given by index,

|

| Then adjust that end of the selection to be at index

| (i.e including but not going beyond index). The other

| end of the selection is made the anchor point for future

| select to commands. If the selection isn't currently in

| the spinbox, then a new selection is created to include

| the characters between index and the most recent selection

| anchor point, inclusive.

|

| selection\_clear(self)

| Clear the selection

|

| If the selection isn't in this widget then the

| command has no effect.

|

| selection\_element(self, element=None)

| Sets or gets the currently selected element.

|

| If a spinbutton element is specified, it will be

| displayed depressed.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

|

| ----------------------------------------------------------------------

| Methods inherited from XView:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

class StringVar(Variable)

| StringVar(master=None, value=None, name=None)

|

| Value holder for strings variables.

|

| Method resolution order:

| StringVar

| Variable

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, value=None, name=None)

| Construct a string variable.

|

| MASTER can be given as master widget.

| VALUE is an optional value (defaults to "")

| NAME is an optional Tcl name (defaults to PY\_VARnum).

|

| If NAME matches an existing variable and VALUE is omitted

| then the existing value is retained.

|

| get(self)

| Return value of variable as string.

|

| ----------------------------------------------------------------------

| Methods inherited from Variable:

|

| \_\_del\_\_(self)

| Unset the variable in Tcl.

|

| \_\_eq\_\_(self, other)

| Comparison for equality (==).

|

| Note: if the Variable's master matters to behavior

| also compare self.\_master == other.\_master

|

| \_\_str\_\_(self)

| Return the name of the variable in Tcl.

|

| initialize = set(self, value)

|

| set(self, value)

| Set the variable to VALUE.

|

| trace = trace\_variable(self, mode, callback)

|

| trace\_add(self, mode, callback)

| Define a trace callback for the variable.

|

| Mode is one of "read", "write", "unset", or a list or tuple of

| such strings.

| Callback must be a function which is called when the variable is

| read, written or unset.

|

| Return the name of the callback.

|

| trace\_info(self)

| Return all trace callback information.

|

| trace\_remove(self, mode, cbname)

| Delete the trace callback for a variable.

|

| Mode is one of "read", "write", "unset" or a list or tuple of

| such strings. Must be same as were specified in trace\_add().

| cbname is the name of the callback returned from trace\_add().

|

| trace\_variable(self, mode, callback)

| Define a trace callback for the variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CALLBACK must be a function which is called when

| the variable is read, written or undefined.

|

| Return the name of the callback.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_add() instead.

|

| trace\_vdelete(self, mode, cbname)

| Delete the trace callback for a variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CBNAME is the name of the callback returned from trace\_variable or trace.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_remove() instead.

|

| trace\_vinfo(self)

| Return all trace callback information.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_info() instead.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Variable:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Data and other attributes inherited from Variable:

|

| \_\_hash\_\_ = None

class Text(Widget, XView, YView)

| Text(master=None, cnf={}, \*\*kw)

|

| Text widget which can display text in various forms.

|

| Method resolution order:

| Text

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| XView

| YView

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a text widget with the parent MASTER.

|

| STANDARD OPTIONS

|

| background, borderwidth, cursor,

| exportselection, font, foreground,

| highlightbackground, highlightcolor,

| highlightthickness, insertbackground,

| insertborderwidth, insertofftime,

| insertontime, insertwidth, padx, pady,

| relief, selectbackground,

| selectborderwidth, selectforeground,

| setgrid, takefocus,

| xscrollcommand, yscrollcommand,

|

| WIDGET-SPECIFIC OPTIONS

|

| autoseparators, height, maxundo,

| spacing1, spacing2, spacing3,

| state, tabs, undo, width, wrap,

|

| bbox(self, index)

| Return a tuple of (x,y,width,height) which gives the bounding

| box of the visible part of the character at the given index.

|

| compare(self, index1, op, index2)

| Return whether between index INDEX1 and index INDEX2 the

| relation OP is satisfied. OP is one of <, <=, ==, >=, >, or !=.

|

| count(self, index1, index2, \*args)

| Counts the number of relevant things between the two indices.

| If index1 is after index2, the result will be a negative number

| (and this holds for each of the possible options).

|

| The actual items which are counted depends on the options given by

| args. The result is a list of integers, one for the result of each

| counting option given. Valid counting options are "chars",

| "displaychars", "displayindices", "displaylines", "indices",

| "lines", "xpixels" and "ypixels". There is an additional possible

| option "update", which if given then all subsequent options ensure

| that any possible out of date information is recalculated.

|

| debug(self, boolean=None)

| Turn on the internal consistency checks of the B-Tree inside the text

| widget according to BOOLEAN.

|

| delete(self, index1, index2=None)

| Delete the characters between INDEX1 and INDEX2 (not included).

|

| dlineinfo(self, index)

| Return tuple (x,y,width,height,baseline) giving the bounding box

| and baseline position of the visible part of the line containing

| the character at INDEX.

|

| dump(self, index1, index2=None, command=None, \*\*kw)

| Return the contents of the widget between index1 and index2.

|

| The type of contents returned in filtered based on the keyword

| parameters; if 'all', 'image', 'mark', 'tag', 'text', or 'window' are

| given and true, then the corresponding items are returned. The result

| is a list of triples of the form (key, value, index). If none of the

| keywords are true then 'all' is used by default.

|

| If the 'command' argument is given, it is called once for each element

| of the list of triples, with the values of each triple serving as the

| arguments to the function. In this case the list is not returned.

|

| edit(self, \*args)

| Internal method

|

| This method controls the undo mechanism and

| the modified flag. The exact behavior of the

| command depends on the option argument that

| follows the edit argument. The following forms

| of the command are currently supported:

|

| edit\_modified, edit\_redo, edit\_reset, edit\_separator

| and edit\_undo

|

| edit\_modified(self, arg=None)

| Get or Set the modified flag

|

| If arg is not specified, returns the modified

| flag of the widget. The insert, delete, edit undo and

| edit redo commands or the user can set or clear the

| modified flag. If boolean is specified, sets the

| modified flag of the widget to arg.

|

| edit\_redo(self)

| Redo the last undone edit

|

| When the undo option is true, reapplies the last

| undone edits provided no other edits were done since

| then. Generates an error when the redo stack is empty.

| Does nothing when the undo option is false.

|

| edit\_reset(self)

| Clears the undo and redo stacks

|

| edit\_separator(self)

| Inserts a separator (boundary) on the undo stack.

|

| Does nothing when the undo option is false

|

| edit\_undo(self)

| Undoes the last edit action

|

| If the undo option is true. An edit action is defined

| as all the insert and delete commands that are recorded

| on the undo stack in between two separators. Generates

| an error when the undo stack is empty. Does nothing

| when the undo option is false

|

| get(self, index1, index2=None)

| Return the text from INDEX1 to INDEX2 (not included).

|

| image\_cget(self, index, option)

| Return the value of OPTION of an embedded image at INDEX.

|

| image\_configure(self, index, cnf=None, \*\*kw)

| Configure an embedded image at INDEX.

|

| image\_create(self, index, cnf={}, \*\*kw)

| Create an embedded image at INDEX.

|

| image\_names(self)

| Return all names of embedded images in this widget.

|

| index(self, index)

| Return the index in the form line.char for INDEX.

|

| insert(self, index, chars, \*args)

| Insert CHARS before the characters at INDEX. An additional

| tag can be given in ARGS. Additional CHARS and tags can follow in ARGS.

|

| mark\_gravity(self, markName, direction=None)

| Change the gravity of a mark MARKNAME to DIRECTION (LEFT or RIGHT).

| Return the current value if None is given for DIRECTION.

|

| mark\_names(self)

| Return all mark names.

|

| mark\_next(self, index)

| Return the name of the next mark after INDEX.

|

| mark\_previous(self, index)

| Return the name of the previous mark before INDEX.

|

| mark\_set(self, markName, index)

| Set mark MARKNAME before the character at INDEX.

|

| mark\_unset(self, \*markNames)

| Delete all marks in MARKNAMES.

|

| peer\_create(self, newPathName, cnf={}, \*\*kw)

| Creates a peer text widget with the given newPathName, and any

| optional standard configuration options. By default the peer will

| have the same start and end line as the parent widget, but

| these can be overridden with the standard configuration options.

|

| peer\_names(self)

| Returns a list of peers of this widget (this does not include

| the widget itself).

|

| replace(self, index1, index2, chars, \*args)

| Replaces the range of characters between index1 and index2 with

| the given characters and tags specified by args.

|

| See the method insert for some more information about args, and the

| method delete for information about the indices.

|

| scan\_dragto(self, x, y)

| Adjust the view of the text to 10 times the

| difference between X and Y and the coordinates given in

| scan\_mark.

|

| scan\_mark(self, x, y)

| Remember the current X, Y coordinates.

|

| search(self, pattern, index, stopindex=None, forwards=None, backwards=None, exact=None, regexp=None, nocase=None, count=None, elide=None)

| Search PATTERN beginning from INDEX until STOPINDEX.

| Return the index of the first character of a match or an

| empty string.

|

| see(self, index)

| Scroll such that the character at INDEX is visible.

|

| tag\_add(self, tagName, index1, \*args)

| Add tag TAGNAME to all characters between INDEX1 and index2 in ARGS.

| Additional pairs of indices may follow in ARGS.

|

| tag\_bind(self, tagName, sequence, func, add=None)

| Bind to all characters with TAGNAME at event SEQUENCE a call to function FUNC.

|

| An additional boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or whether it will

| replace the previous function. See bind for the return value.

|

| tag\_cget(self, tagName, option)

| Return the value of OPTION for tag TAGNAME.

|

| tag\_config = tag\_configure(self, tagName, cnf=None, \*\*kw)

|

| tag\_configure(self, tagName, cnf=None, \*\*kw)

| Configure a tag TAGNAME.

|

| tag\_delete(self, \*tagNames)

| Delete all tags in TAGNAMES.

|

| tag\_lower(self, tagName, belowThis=None)

| Change the priority of tag TAGNAME such that it is lower

| than the priority of BELOWTHIS.

|

| tag\_names(self, index=None)

| Return a list of all tag names.

|

| tag\_nextrange(self, tagName, index1, index2=None)

| Return a list of start and end index for the first sequence of

| characters between INDEX1 and INDEX2 which all have tag TAGNAME.

| The text is searched forward from INDEX1.

|

| tag\_prevrange(self, tagName, index1, index2=None)

| Return a list of start and end index for the first sequence of

| characters between INDEX1 and INDEX2 which all have tag TAGNAME.

| The text is searched backwards from INDEX1.

|

| tag\_raise(self, tagName, aboveThis=None)

| Change the priority of tag TAGNAME such that it is higher

| than the priority of ABOVETHIS.

|

| tag\_ranges(self, tagName)

| Return a list of ranges of text which have tag TAGNAME.

|

| tag\_remove(self, tagName, index1, index2=None)

| Remove tag TAGNAME from all characters between INDEX1 and INDEX2.

|

| tag\_unbind(self, tagName, sequence, funcid=None)

| Unbind for all characters with TAGNAME for event SEQUENCE the

| function identified with FUNCID.

|

| window\_cget(self, index, option)

| Return the value of OPTION of an embedded window at INDEX.

|

| window\_config = window\_configure(self, index, cnf=None, \*\*kw)

|

| window\_configure(self, index, cnf=None, \*\*kw)

| Configure an embedded window at INDEX.

|

| window\_create(self, index, cnf={}, \*\*kw)

| Create a window at INDEX.

|

| window\_names(self)

| Return all names of embedded windows in this widget.

|

| yview\_pickplace(self, \*what)

| Obsolete function, use see.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

|

| ----------------------------------------------------------------------

| Methods inherited from XView:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

|

| ----------------------------------------------------------------------

| Methods inherited from YView:

|

| yview(self, \*args)

| Query and change the vertical position of the view.

|

| yview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total height of the canvas is off-screen to the top.

|

| yview\_scroll(self, number, what)

| Shift the y-view according to NUMBER which is measured in

| "units" or "pages" (WHAT).

class Tk(Misc, Wm)

| Tk(screenName=None, baseName=None, className='Tk', useTk=1, sync=0, use=None)

|

| Toplevel widget of Tk which represents mostly the main window

| of an application. It has an associated Tcl interpreter.

|

| Method resolution order:

| Tk

| Misc

| Wm

| builtins.object

|

| Methods defined here:

|

| \_\_getattr\_\_(self, attr)

| Delegate attribute access to the interpreter object

|

| \_\_init\_\_(self, screenName=None, baseName=None, className='Tk', useTk=1, sync=0, use=None)

| Return a new Toplevel widget on screen SCREENNAME. A new Tcl interpreter will

| be created. BASENAME will be used for the identification of the profile file (see

| readprofile).

| It is constructed from sys.argv[0] without extensions if None is given. CLASSNAME

| is the name of the widget class.

|

| destroy(self)

| Destroy this and all descendants widgets. This will

| end the application of this Tcl interpreter.

|

| loadtk(self)

|

| readprofile(self, baseName, className)

| Internal function. It reads BASENAME.tcl and CLASSNAME.tcl into

| the Tcl Interpreter and calls exec on the contents of BASENAME.py and

| CLASSNAME.py if such a file exists in the home directory.

|

| report\_callback\_exception(self, exc, val, tb)

| Report callback exception on sys.stderr.

|

| Applications may want to override this internal function, and

| should when sys.stderr is None.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Wm:

|

| aspect = wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

|

| attributes = wm\_attributes(self, \*args)

|

| client = wm\_client(self, name=None)

|

| colormapwindows = wm\_colormapwindows(self, \*wlist)

|

| command = wm\_command(self, value=None)

|

| deiconify = wm\_deiconify(self)

|

| focusmodel = wm\_focusmodel(self, model=None)

|

| forget = wm\_forget(self, window)

|

| frame = wm\_frame(self)

|

| geometry = wm\_geometry(self, newGeometry=None)

|

| grid = wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

|

| group = wm\_group(self, pathName=None)

|

| iconbitmap = wm\_iconbitmap(self, bitmap=None, default=None)

|

| iconify = wm\_iconify(self)

|

| iconmask = wm\_iconmask(self, bitmap=None)

|

| iconname = wm\_iconname(self, newName=None)

|

| iconphoto = wm\_iconphoto(self, default=False, \*args)

|

| iconposition = wm\_iconposition(self, x=None, y=None)

|

| iconwindow = wm\_iconwindow(self, pathName=None)

|

| manage = wm\_manage(self, widget)

|

| maxsize = wm\_maxsize(self, width=None, height=None)

|

| minsize = wm\_minsize(self, width=None, height=None)

|

| overrideredirect = wm\_overrideredirect(self, boolean=None)

|

| positionfrom = wm\_positionfrom(self, who=None)

|

| protocol = wm\_protocol(self, name=None, func=None)

|

| resizable = wm\_resizable(self, width=None, height=None)

|

| sizefrom = wm\_sizefrom(self, who=None)

|

| state = wm\_state(self, newstate=None)

|

| title = wm\_title(self, string=None)

|

| transient = wm\_transient(self, master=None)

|

| withdraw = wm\_withdraw(self)

|

| wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

| Instruct the window manager to set the aspect ratio (width/height)

| of this widget to be between MINNUMER/MINDENOM and MAXNUMER/MAXDENOM. Return a tuple

| of the actual values if no argument is given.

|

| wm\_attributes(self, \*args)

| This subcommand returns or sets platform specific attributes

|

| The first form returns a list of the platform specific flags and

| their values. The second form returns the value for the specific

| option. The third form sets one or more of the values. The values

| are as follows:

|

| On Windows, -disabled gets or sets whether the window is in a

| disabled state. -toolwindow gets or sets the style of the window

| to toolwindow (as defined in the MSDN). -topmost gets or sets

| whether this is a topmost window (displays above all other

| windows).

|

| On Macintosh, XXXXX

|

| On Unix, there are currently no special attribute values.

|

| wm\_client(self, name=None)

| Store NAME in WM\_CLIENT\_MACHINE property of this widget. Return

| current value.

|

| wm\_colormapwindows(self, \*wlist)

| Store list of window names (WLIST) into WM\_COLORMAPWINDOWS property

| of this widget. This list contains windows whose colormaps differ from their

| parents. Return current list of widgets if WLIST is empty.

|

| wm\_command(self, value=None)

| Store VALUE in WM\_COMMAND property. It is the command

| which shall be used to invoke the application. Return current

| command if VALUE is None.

|

| wm\_deiconify(self)

| Deiconify this widget. If it was never mapped it will not be mapped.

| On Windows it will raise this widget and give it the focus.

|

| wm\_focusmodel(self, model=None)

| Set focus model to MODEL. "active" means that this widget will claim

| the focus itself, "passive" means that the window manager shall give

| the focus. Return current focus model if MODEL is None.

|

| wm\_forget(self, window)

| The window will be unmapped from the screen and will no longer

| be managed by wm. toplevel windows will be treated like frame

| windows once they are no longer managed by wm, however, the menu

| option configuration will be remembered and the menus will return

| once the widget is managed again.

|

| wm\_frame(self)

| Return identifier for decorative frame of this widget if present.

|

| wm\_geometry(self, newGeometry=None)

| Set geometry to NEWGEOMETRY of the form =widthxheight+x+y. Return

| current value if None is given.

|

| wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

| Instruct the window manager that this widget shall only be

| resized on grid boundaries. WIDTHINC and HEIGHTINC are the width and

| height of a grid unit in pixels. BASEWIDTH and BASEHEIGHT are the

| number of grid units requested in Tk\_GeometryRequest.

|

| wm\_group(self, pathName=None)

| Set the group leader widgets for related widgets to PATHNAME. Return

| the group leader of this widget if None is given.

|

| wm\_iconbitmap(self, bitmap=None, default=None)

| Set bitmap for the iconified widget to BITMAP. Return

| the bitmap if None is given.

|

| Under Windows, the DEFAULT parameter can be used to set the icon

| for the widget and any descendents that don't have an icon set

| explicitly. DEFAULT can be the relative path to a .ico file

| (example: root.iconbitmap(default='myicon.ico') ). See Tk

| documentation for more information.

|

| wm\_iconify(self)

| Display widget as icon.

|

| wm\_iconmask(self, bitmap=None)

| Set mask for the icon bitmap of this widget. Return the

| mask if None is given.

|

| wm\_iconname(self, newName=None)

| Set the name of the icon for this widget. Return the name if

| None is given.

|

| wm\_iconphoto(self, default=False, \*args)

| Sets the titlebar icon for this window based on the named photo

| images passed through args. If default is True, this is applied to

| all future created toplevels as well.

|

| The data in the images is taken as a snapshot at the time of

| invocation. If the images are later changed, this is not reflected

| to the titlebar icons. Multiple images are accepted to allow

| different images sizes to be provided. The window manager may scale

| provided icons to an appropriate size.

|

| On Windows, the images are packed into a Windows icon structure.

| This will override an icon specified to wm\_iconbitmap, and vice

| versa.

|

| On X, the images are arranged into the \_NET\_WM\_ICON X property,

| which most modern window managers support. An icon specified by

| wm\_iconbitmap may exist simultaneously.

|

| On Macintosh, this currently does nothing.

|

| wm\_iconposition(self, x=None, y=None)

| Set the position of the icon of this widget to X and Y. Return

| a tuple of the current values of X and X if None is given.

|

| wm\_iconwindow(self, pathName=None)

| Set widget PATHNAME to be displayed instead of icon. Return the current

| value if None is given.

|

| wm\_manage(self, widget)

| The widget specified will become a stand alone top-level window.

| The window will be decorated with the window managers title bar,

| etc.

|

| wm\_maxsize(self, width=None, height=None)

| Set max WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_minsize(self, width=None, height=None)

| Set min WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_overrideredirect(self, boolean=None)

| Instruct the window manager to ignore this widget

| if BOOLEAN is given with 1. Return the current value if None

| is given.

|

| wm\_positionfrom(self, who=None)

| Instruct the window manager that the position of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_protocol(self, name=None, func=None)

| Bind function FUNC to command NAME for this widget.

| Return the function bound to NAME if None is given. NAME could be

| e.g. "WM\_SAVE\_YOURSELF" or "WM\_DELETE\_WINDOW".

|

| wm\_resizable(self, width=None, height=None)

| Instruct the window manager whether this width can be resized

| in WIDTH or HEIGHT. Both values are boolean values.

|

| wm\_sizefrom(self, who=None)

| Instruct the window manager that the size of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_state(self, newstate=None)

| Query or set the state of this widget as one of normal, icon,

| iconic (see wm\_iconwindow), withdrawn, or zoomed (Windows only).

|

| wm\_title(self, string=None)

| Set the title of this widget.

|

| wm\_transient(self, master=None)

| Instruct the window manager that this widget is transient

| with regard to widget MASTER.

|

| wm\_withdraw(self)

| Withdraw this widget from the screen such that it is unmapped

| and forgotten by the window manager. Re-draw it with wm\_deiconify.

class Toplevel(BaseWidget, Wm)

| Toplevel(master=None, cnf={}, \*\*kw)

|

| Toplevel widget, e.g. for dialogs.

|

| Method resolution order:

| Toplevel

| BaseWidget

| Misc

| Wm

| builtins.object

|

| Methods defined here:

|

| \_\_init\_\_(self, master=None, cnf={}, \*\*kw)

| Construct a toplevel widget with the parent MASTER.

|

| Valid resource names: background, bd, bg, borderwidth, class,

| colormap, container, cursor, height, highlightbackground,

| highlightcolor, highlightthickness, menu, relief, screen, takefocus,

| use, visual, width.

|

| ----------------------------------------------------------------------

| Methods inherited from BaseWidget:

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Wm:

|

| aspect = wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

|

| attributes = wm\_attributes(self, \*args)

|

| client = wm\_client(self, name=None)

|

| colormapwindows = wm\_colormapwindows(self, \*wlist)

|

| command = wm\_command(self, value=None)

|

| deiconify = wm\_deiconify(self)

|

| focusmodel = wm\_focusmodel(self, model=None)

|

| forget = wm\_forget(self, window)

|

| frame = wm\_frame(self)

|

| geometry = wm\_geometry(self, newGeometry=None)

|

| grid = wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

|

| group = wm\_group(self, pathName=None)

|

| iconbitmap = wm\_iconbitmap(self, bitmap=None, default=None)

|

| iconify = wm\_iconify(self)

|

| iconmask = wm\_iconmask(self, bitmap=None)

|

| iconname = wm\_iconname(self, newName=None)

|

| iconphoto = wm\_iconphoto(self, default=False, \*args)

|

| iconposition = wm\_iconposition(self, x=None, y=None)

|

| iconwindow = wm\_iconwindow(self, pathName=None)

|

| manage = wm\_manage(self, widget)

|

| maxsize = wm\_maxsize(self, width=None, height=None)

|

| minsize = wm\_minsize(self, width=None, height=None)

|

| overrideredirect = wm\_overrideredirect(self, boolean=None)

|

| positionfrom = wm\_positionfrom(self, who=None)

|

| protocol = wm\_protocol(self, name=None, func=None)

|

| resizable = wm\_resizable(self, width=None, height=None)

|

| sizefrom = wm\_sizefrom(self, who=None)

|

| state = wm\_state(self, newstate=None)

|

| title = wm\_title(self, string=None)

|

| transient = wm\_transient(self, master=None)

|

| withdraw = wm\_withdraw(self)

|

| wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

| Instruct the window manager to set the aspect ratio (width/height)

| of this widget to be between MINNUMER/MINDENOM and MAXNUMER/MAXDENOM. Return a tuple

| of the actual values if no argument is given.

|

| wm\_attributes(self, \*args)

| This subcommand returns or sets platform specific attributes

|

| The first form returns a list of the platform specific flags and

| their values. The second form returns the value for the specific

| option. The third form sets one or more of the values. The values

| are as follows:

|

| On Windows, -disabled gets or sets whether the window is in a

| disabled state. -toolwindow gets or sets the style of the window

| to toolwindow (as defined in the MSDN). -topmost gets or sets

| whether this is a topmost window (displays above all other

| windows).

|

| On Macintosh, XXXXX

|

| On Unix, there are currently no special attribute values.

|

| wm\_client(self, name=None)

| Store NAME in WM\_CLIENT\_MACHINE property of this widget. Return

| current value.

|

| wm\_colormapwindows(self, \*wlist)

| Store list of window names (WLIST) into WM\_COLORMAPWINDOWS property

| of this widget. This list contains windows whose colormaps differ from their

| parents. Return current list of widgets if WLIST is empty.

|

| wm\_command(self, value=None)

| Store VALUE in WM\_COMMAND property. It is the command

| which shall be used to invoke the application. Return current

| command if VALUE is None.

|

| wm\_deiconify(self)

| Deiconify this widget. If it was never mapped it will not be mapped.

| On Windows it will raise this widget and give it the focus.

|

| wm\_focusmodel(self, model=None)

| Set focus model to MODEL. "active" means that this widget will claim

| the focus itself, "passive" means that the window manager shall give

| the focus. Return current focus model if MODEL is None.

|

| wm\_forget(self, window)

| The window will be unmapped from the screen and will no longer

| be managed by wm. toplevel windows will be treated like frame

| windows once they are no longer managed by wm, however, the menu

| option configuration will be remembered and the menus will return

| once the widget is managed again.

|

| wm\_frame(self)

| Return identifier for decorative frame of this widget if present.

|

| wm\_geometry(self, newGeometry=None)

| Set geometry to NEWGEOMETRY of the form =widthxheight+x+y. Return

| current value if None is given.

|

| wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

| Instruct the window manager that this widget shall only be

| resized on grid boundaries. WIDTHINC and HEIGHTINC are the width and

| height of a grid unit in pixels. BASEWIDTH and BASEHEIGHT are the

| number of grid units requested in Tk\_GeometryRequest.

|

| wm\_group(self, pathName=None)

| Set the group leader widgets for related widgets to PATHNAME. Return

| the group leader of this widget if None is given.

|

| wm\_iconbitmap(self, bitmap=None, default=None)

| Set bitmap for the iconified widget to BITMAP. Return

| the bitmap if None is given.

|

| Under Windows, the DEFAULT parameter can be used to set the icon

| for the widget and any descendents that don't have an icon set

| explicitly. DEFAULT can be the relative path to a .ico file

| (example: root.iconbitmap(default='myicon.ico') ). See Tk

| documentation for more information.

|

| wm\_iconify(self)

| Display widget as icon.

|

| wm\_iconmask(self, bitmap=None)

| Set mask for the icon bitmap of this widget. Return the

| mask if None is given.

|

| wm\_iconname(self, newName=None)

| Set the name of the icon for this widget. Return the name if

| None is given.

|

| wm\_iconphoto(self, default=False, \*args)

| Sets the titlebar icon for this window based on the named photo

| images passed through args. If default is True, this is applied to

| all future created toplevels as well.

|

| The data in the images is taken as a snapshot at the time of

| invocation. If the images are later changed, this is not reflected

| to the titlebar icons. Multiple images are accepted to allow

| different images sizes to be provided. The window manager may scale

| provided icons to an appropriate size.

|

| On Windows, the images are packed into a Windows icon structure.

| This will override an icon specified to wm\_iconbitmap, and vice

| versa.

|

| On X, the images are arranged into the \_NET\_WM\_ICON X property,

| which most modern window managers support. An icon specified by

| wm\_iconbitmap may exist simultaneously.

|

| On Macintosh, this currently does nothing.

|

| wm\_iconposition(self, x=None, y=None)

| Set the position of the icon of this widget to X and Y. Return

| a tuple of the current values of X and X if None is given.

|

| wm\_iconwindow(self, pathName=None)

| Set widget PATHNAME to be displayed instead of icon. Return the current

| value if None is given.

|

| wm\_manage(self, widget)

| The widget specified will become a stand alone top-level window.

| The window will be decorated with the window managers title bar,

| etc.

|

| wm\_maxsize(self, width=None, height=None)

| Set max WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_minsize(self, width=None, height=None)

| Set min WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_overrideredirect(self, boolean=None)

| Instruct the window manager to ignore this widget

| if BOOLEAN is given with 1. Return the current value if None

| is given.

|

| wm\_positionfrom(self, who=None)

| Instruct the window manager that the position of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_protocol(self, name=None, func=None)

| Bind function FUNC to command NAME for this widget.

| Return the function bound to NAME if None is given. NAME could be

| e.g. "WM\_SAVE\_YOURSELF" or "WM\_DELETE\_WINDOW".

|

| wm\_resizable(self, width=None, height=None)

| Instruct the window manager whether this width can be resized

| in WIDTH or HEIGHT. Both values are boolean values.

|

| wm\_sizefrom(self, who=None)

| Instruct the window manager that the size of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_state(self, newstate=None)

| Query or set the state of this widget as one of normal, icon,

| iconic (see wm\_iconwindow), withdrawn, or zoomed (Windows only).

|

| wm\_title(self, string=None)

| Set the title of this widget.

|

| wm\_transient(self, master=None)

| Instruct the window manager that this widget is transient

| with regard to widget MASTER.

|

| wm\_withdraw(self)

| Withdraw this widget from the screen such that it is unmapped

| and forgotten by the window manager. Re-draw it with wm\_deiconify.

class Variable(builtins.object)

| Variable(master=None, value=None, name=None)

|

| Class to define value holders for e.g. buttons.

|

| Subclasses StringVar, IntVar, DoubleVar, BooleanVar are specializations

| that constrain the type of the value returned from get().

|

| Methods defined here:

|

| \_\_del\_\_(self)

| Unset the variable in Tcl.

|

| \_\_eq\_\_(self, other)

| Comparison for equality (==).

|

| Note: if the Variable's master matters to behavior

| also compare self.\_master == other.\_master

|

| \_\_init\_\_(self, master=None, value=None, name=None)

| Construct a variable

|

| MASTER can be given as master widget.

| VALUE is an optional value (defaults to "")

| NAME is an optional Tcl name (defaults to PY\_VARnum).

|

| If NAME matches an existing variable and VALUE is omitted

| then the existing value is retained.

|

| \_\_str\_\_(self)

| Return the name of the variable in Tcl.

|

| get(self)

| Return value of variable.

|

| initialize = set(self, value)

|

| set(self, value)

| Set the variable to VALUE.

|

| trace = trace\_variable(self, mode, callback)

|

| trace\_add(self, mode, callback)

| Define a trace callback for the variable.

|

| Mode is one of "read", "write", "unset", or a list or tuple of

| such strings.

| Callback must be a function which is called when the variable is

| read, written or unset.

|

| Return the name of the callback.

|

| trace\_info(self)

| Return all trace callback information.

|

| trace\_remove(self, mode, cbname)

| Delete the trace callback for a variable.

|

| Mode is one of "read", "write", "unset" or a list or tuple of

| such strings. Must be same as were specified in trace\_add().

| cbname is the name of the callback returned from trace\_add().

|

| trace\_variable(self, mode, callback)

| Define a trace callback for the variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CALLBACK must be a function which is called when

| the variable is read, written or undefined.

|

| Return the name of the callback.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_add() instead.

|

| trace\_vdelete(self, mode, cbname)

| Delete the trace callback for a variable.

|

| MODE is one of "r", "w", "u" for read, write, undefine.

| CBNAME is the name of the callback returned from trace\_variable or trace.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_remove() instead.

|

| trace\_vinfo(self)

| Return all trace callback information.

|

| This deprecated method wraps a deprecated Tcl method that will

| likely be removed in the future. Use trace\_info() instead.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Data and other attributes defined here:

|

| \_\_hash\_\_ = None

class Widget(BaseWidget, Pack, Place, Grid)

| Widget(master, widgetName, cnf={}, kw={}, extra=())

|

| Internal class.

|

| Base class for a widget which can be positioned with the geometry managers

| Pack, Place or Grid.

|

| Method resolution order:

| Widget

| BaseWidget

| Misc

| Pack

| Place

| Grid

| builtins.object

|

| Methods inherited from BaseWidget:

|

| \_\_init\_\_(self, master, widgetName, cnf={}, kw={}, extra=())

| Construct a widget with the parent widget MASTER, a name WIDGETNAME

| and appropriate options.

|

| destroy(self)

| Destroy this and all descendants widgets.

|

| ----------------------------------------------------------------------

| Methods inherited from Misc:

|

| \_\_getitem\_\_ = cget(self, key)

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, value)

|

| \_\_str\_\_(self)

| Return the window path name of this widget.

|

| after(self, ms, func=None, \*args)

| Call function once after given time.

|

| MS specifies the time in milliseconds. FUNC gives the

| function which shall be called. Additional parameters

| are given as parameters to the function call. Return

| identifier to cancel scheduling with after\_cancel.

|

| after\_cancel(self, id)

| Cancel scheduling of function identified with ID.

|

| Identifier returned by after or after\_idle must be

| given as first parameter.

|

| after\_idle(self, func, \*args)

| Call FUNC once if the Tcl main loop has no event to

| process.

|

| Return an identifier to cancel the scheduling with

| after\_cancel.

|

| anchor = grid\_anchor(self, anchor=None)

|

| bbox = grid\_bbox(self, column=None, row=None, col2=None, row2=None)

|

| bell(self, displayof=0)

| Ring a display's bell.

|

| bind(self, sequence=None, func=None, add=None)

| Bind to this widget at event SEQUENCE a call to function FUNC.

|

| SEQUENCE is a string of concatenated event

| patterns. An event pattern is of the form

| <MODIFIER-MODIFIER-TYPE-DETAIL> where MODIFIER is one

| of Control, Mod2, M2, Shift, Mod3, M3, Lock, Mod4, M4,

| Button1, B1, Mod5, M5 Button2, B2, Meta, M, Button3,

| B3, Alt, Button4, B4, Double, Button5, B5 Triple,

| Mod1, M1. TYPE is one of Activate, Enter, Map,

| ButtonPress, Button, Expose, Motion, ButtonRelease

| FocusIn, MouseWheel, Circulate, FocusOut, Property,

| Colormap, Gravity Reparent, Configure, KeyPress, Key,

| Unmap, Deactivate, KeyRelease Visibility, Destroy,

| Leave and DETAIL is the button number for ButtonPress,

| ButtonRelease and DETAIL is the Keysym for KeyPress and

| KeyRelease. Examples are

| <Control-Button-1> for pressing Control and mouse button 1 or

| <Alt-A> for pressing A and the Alt key (KeyPress can be omitted).

| An event pattern can also be a virtual event of the form

| <<AString>> where AString can be arbitrary. This

| event can be generated by event\_generate.

| If events are concatenated they must appear shortly

| after each other.

|

| FUNC will be called if the event sequence occurs with an

| instance of Event as argument. If the return value of FUNC is

| "break" no further bound function is invoked.

|

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function.

|

| Bind will return an identifier to allow deletion of the bound function with

| unbind without memory leak.

|

| If FUNC or SEQUENCE is omitted the bound function or list

| of bound events are returned.

|

| bind\_all(self, sequence=None, func=None, add=None)

| Bind to all widgets at an event SEQUENCE a call to function FUNC.

| An additional boolean parameter ADD specifies whether FUNC will

| be called additionally to the other bound function or whether

| it will replace the previous function. See bind for the return value.

|

| bind\_class(self, className, sequence=None, func=None, add=None)

| Bind to widgets with bindtag CLASSNAME at event

| SEQUENCE a call of function FUNC. An additional

| boolean parameter ADD specifies whether FUNC will be

| called additionally to the other bound function or

| whether it will replace the previous function. See bind for

| the return value.

|

| bindtags(self, tagList=None)

| Set or get the list of bindtags for this widget.

|

| With no argument return the list of all bindtags associated with

| this widget. With a list of strings as argument the bindtags are

| set to this list. The bindtags determine in which order events are

| processed (see bind).

|

| cget(self, key)

| Return the resource value for a KEY given as string.

|

| clipboard\_append(self, string, \*\*kw)

| Append STRING to the Tk clipboard.

|

| A widget specified at the optional displayof keyword

| argument specifies the target display. The clipboard

| can be retrieved with selection\_get.

|

| clipboard\_clear(self, \*\*kw)

| Clear the data in the Tk clipboard.

|

| A widget specified for the optional displayof keyword

| argument specifies the target display.

|

| clipboard\_get(self, \*\*kw)

| Retrieve data from the clipboard on window's display.

|

| The window keyword defaults to the root window of the Tkinter

| application.

|

| The type keyword specifies the form in which the data is

| to be returned and should be an atom name such as STRING

| or FILE\_NAME. Type defaults to STRING, except on X11, where the default

| is to try UTF8\_STRING and fall back to STRING.

|

| This command is equivalent to:

|

| selection\_get(CLIPBOARD)

|

| columnconfigure = grid\_columnconfigure(self, index, cnf={}, \*\*kw)

|

| config = configure(self, cnf=None, \*\*kw)

|

| configure(self, cnf=None, \*\*kw)

| Configure resources of a widget.

|

| The values for resources are specified as keyword

| arguments. To get an overview about

| the allowed keyword arguments call the method keys.

|

| deletecommand(self, name)

| Internal function.

|

| Delete the Tcl command provided in NAME.

|

| event\_add(self, virtual, \*sequences)

| Bind a virtual event VIRTUAL (of the form <<Name>>)

| to an event SEQUENCE such that the virtual event is triggered

| whenever SEQUENCE occurs.

|

| event\_delete(self, virtual, \*sequences)

| Unbind a virtual event VIRTUAL from SEQUENCE.

|

| event\_generate(self, sequence, \*\*kw)

| Generate an event SEQUENCE. Additional

| keyword arguments specify parameter of the event

| (e.g. x, y, rootx, rooty).

|

| event\_info(self, virtual=None)

| Return a list of all virtual events or the information

| about the SEQUENCE bound to the virtual event VIRTUAL.

|

| focus = focus\_set(self)

|

| focus\_displayof(self)

| Return the widget which has currently the focus on the

| display where this widget is located.

|

| Return None if the application does not have the focus.

|

| focus\_force(self)

| Direct input focus to this widget even if the

| application does not have the focus. Use with

| caution!

|

| focus\_get(self)

| Return the widget which has currently the focus in the

| application.

|

| Use focus\_displayof to allow working with several

| displays. Return None if application does not have

| the focus.

|

| focus\_lastfor(self)

| Return the widget which would have the focus if top level

| for this widget gets the focus from the window manager.

|

| focus\_set(self)

| Direct input focus to this widget.

|

| If the application currently does not have the focus

| this widget will get the focus if the application gets

| the focus through the window manager.

|

| getboolean(self, s)

| Return a boolean value for Tcl boolean values true and false given as parameter.

|

| getdouble(self, s)

|

| getint(self, s)

|

| getvar(self, name='PY\_VAR')

| Return value of Tcl variable NAME.

|

| grab\_current(self)

| Return widget which has currently the grab in this application

| or None.

|

| grab\_release(self)

| Release grab for this widget if currently set.

|

| grab\_set(self)

| Set grab for this widget.

|

| A grab directs all events to this and descendant

| widgets in the application.

|

| grab\_set\_global(self)

| Set global grab for this widget.

|

| A global grab directs all events to this and

| descendant widgets on the display. Use with caution -

| other applications do not get events anymore.

|

| grab\_status(self)

| Return None, "local" or "global" if this widget has

| no, a local or a global grab.

|

| grid\_anchor(self, anchor=None)

| The anchor value controls how to place the grid within the

| master when no row/column has any weight.

|

| The default anchor is nw.

|

| grid\_bbox(self, column=None, row=None, col2=None, row2=None)

| Return a tuple of integer coordinates for the bounding

| box of this widget controlled by the geometry manager grid.

|

| If COLUMN, ROW is given the bounding box applies from

| the cell with row and column 0 to the specified

| cell. If COL2 and ROW2 are given the bounding box

| starts at that cell.

|

| The returned integers specify the offset of the upper left

| corner in the master widget and the width and height.

|

| grid\_columnconfigure(self, index, cnf={}, \*\*kw)

| Configure column INDEX of a grid.

|

| Valid resources are minsize (minimum size of the column),

| weight (how much does additional space propagate to this column)

| and pad (how much space to let additionally).

|

| grid\_location(self, x, y)

| Return a tuple of column and row which identify the cell

| at which the pixel at position X and Y inside the master

| widget is located.

|

| grid\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given, the current setting will be returned.

|

| grid\_rowconfigure(self, index, cnf={}, \*\*kw)

| Configure row INDEX of a grid.

|

| Valid resources are minsize (minimum size of the row),

| weight (how much does additional space propagate to this row)

| and pad (how much space to let additionally).

|

| grid\_size(self)

| Return a tuple of the number of column and rows in the grid.

|

| grid\_slaves(self, row=None, column=None)

| Return a list of all slaves of this widget

| in its packing order.

|

| image\_names(self)

| Return a list of all existing image names.

|

| image\_types(self)

| Return a list of all available image types (e.g. photo bitmap).

|

| keys(self)

| Return a list of all resource names of this widget.

|

| lift = tkraise(self, aboveThis=None)

|

| lower(self, belowThis=None)

| Lower this widget in the stacking order.

|

| mainloop(self, n=0)

| Call the mainloop of Tk.

|

| nametowidget(self, name)

| Return the Tkinter instance of a widget identified by

| its Tcl name NAME.

|

| option\_add(self, pattern, value, priority=None)

| Set a VALUE (second parameter) for an option

| PATTERN (first parameter).

|

| An optional third parameter gives the numeric priority

| (defaults to 80).

|

| option\_clear(self)

| Clear the option database.

|

| It will be reloaded if option\_add is called.

|

| option\_get(self, name, className)

| Return the value for an option NAME for this widget

| with CLASSNAME.

|

| Values with higher priority override lower values.

|

| option\_readfile(self, fileName, priority=None)

| Read file FILENAME into the option database.

|

| An optional second parameter gives the numeric

| priority.

|

| pack\_propagate(self, flag=['\_noarg\_'])

| Set or get the status for propagation of geometry information.

|

| A boolean argument specifies whether the geometry information

| of the slaves will determine the size of this widget. If no argument

| is given the current setting will be returned.

|

| pack\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| place\_slaves(self)

| Return a list of all slaves of this widget

| in its packing order.

|

| propagate = pack\_propagate(self, flag=['\_noarg\_'])

|

| quit(self)

| Quit the Tcl interpreter. All widgets will be destroyed.

|

| register = \_register(self, func, subst=None, needcleanup=1)

|

| rowconfigure = grid\_rowconfigure(self, index, cnf={}, \*\*kw)

|

| selection\_clear(self, \*\*kw)

| Clear the current X selection.

|

| selection\_get(self, \*\*kw)

| Return the contents of the current X selection.

|

| A keyword parameter selection specifies the name of

| the selection and defaults to PRIMARY. A keyword

| parameter displayof specifies a widget on the display

| to use. A keyword parameter type specifies the form of data to be

| fetched, defaulting to STRING except on X11, where UTF8\_STRING is tried

| before STRING.

|

| selection\_handle(self, command, \*\*kw)

| Specify a function COMMAND to call if the X

| selection owned by this widget is queried by another

| application.

|

| This function must return the contents of the

| selection. The function will be called with the

| arguments OFFSET and LENGTH which allows the chunking

| of very long selections. The following keyword

| parameters can be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| selection\_own(self, \*\*kw)

| Become owner of X selection.

|

| A keyword parameter selection specifies the name of

| the selection (default PRIMARY).

|

| selection\_own\_get(self, \*\*kw)

| Return owner of X selection.

|

| The following keyword parameter can

| be provided:

| selection - name of the selection (default PRIMARY),

| type - type of the selection (e.g. STRING, FILE\_NAME).

|

| send(self, interp, cmd, \*args)

| Send Tcl command CMD to different interpreter INTERP to be executed.

|

| setvar(self, name='PY\_VAR', value='1')

| Set Tcl variable NAME to VALUE.

|

| size = grid\_size(self)

|

| slaves = pack\_slaves(self)

|

| tk\_bisque(self)

| Change the color scheme to light brown as used in Tk 3.6 and before.

|

| tk\_focusFollowsMouse(self)

| The widget under mouse will get automatically focus. Can not

| be disabled easily.

|

| tk\_focusNext(self)

| Return the next widget in the focus order which follows

| widget which has currently the focus.

|

| The focus order first goes to the next child, then to

| the children of the child recursively and then to the

| next sibling which is higher in the stacking order. A

| widget is omitted if it has the takefocus resource set

| to 0.

|

| tk\_focusPrev(self)

| Return previous widget in the focus order. See tk\_focusNext for details.

|

| tk\_setPalette(self, \*args, \*\*kw)

| Set a new color scheme for all widget elements.

|

| A single color as argument will cause that all colors of Tk

| widget elements are derived from this.

| Alternatively several keyword parameters and its associated

| colors can be given. The following keywords are valid:

| activeBackground, foreground, selectColor,

| activeForeground, highlightBackground, selectBackground,

| background, highlightColor, selectForeground,

| disabledForeground, insertBackground, troughColor.

|

| tk\_strictMotif(self, boolean=None)

| Set Tcl internal variable, whether the look and feel

| should adhere to Motif.

|

| A parameter of 1 means adhere to Motif (e.g. no color

| change if mouse passes over slider).

| Returns the set value.

|

| tkraise(self, aboveThis=None)

| Raise this widget in the stacking order.

|

| unbind(self, sequence, funcid=None)

| Unbind for this widget for event SEQUENCE the

| function identified with FUNCID.

|

| unbind\_all(self, sequence)

| Unbind for all widgets for event SEQUENCE all functions.

|

| unbind\_class(self, className, sequence)

| Unbind for all widgets with bindtag CLASSNAME for event SEQUENCE

| all functions.

|

| update(self)

| Enter event loop until all pending events have been processed by Tcl.

|

| update\_idletasks(self)

| Enter event loop until all idle callbacks have been called. This

| will update the display of windows but not process events caused by

| the user.

|

| wait\_variable(self, name='PY\_VAR')

| Wait until the variable is modified.

|

| A parameter of type IntVar, StringVar, DoubleVar or

| BooleanVar must be given.

|

| wait\_visibility(self, window=None)

| Wait until the visibility of a WIDGET changes

| (e.g. it appears).

|

| If no parameter is given self is used.

|

| wait\_window(self, window=None)

| Wait until a WIDGET is destroyed.

|

| If no parameter is given self is used.

|

| waitvar = wait\_variable(self, name='PY\_VAR')

|

| winfo\_atom(self, name, displayof=0)

| Return integer which represents atom NAME.

|

| winfo\_atomname(self, id, displayof=0)

| Return name of atom with identifier ID.

|

| winfo\_cells(self)

| Return number of cells in the colormap for this widget.

|

| winfo\_children(self)

| Return a list of all widgets which are children of this widget.

|

| winfo\_class(self)

| Return window class name of this widget.

|

| winfo\_colormapfull(self)

| Return true if at the last color request the colormap was full.

|

| winfo\_containing(self, rootX, rootY, displayof=0)

| Return the widget which is at the root coordinates ROOTX, ROOTY.

|

| winfo\_depth(self)

| Return the number of bits per pixel.

|

| winfo\_exists(self)

| Return true if this widget exists.

|

| winfo\_fpixels(self, number)

| Return the number of pixels for the given distance NUMBER

| (e.g. "3c") as float.

|

| winfo\_geometry(self)

| Return geometry string for this widget in the form "widthxheight+X+Y".

|

| winfo\_height(self)

| Return height of this widget.

|

| winfo\_id(self)

| Return identifier ID for this widget.

|

| winfo\_interps(self, displayof=0)

| Return the name of all Tcl interpreters for this display.

|

| winfo\_ismapped(self)

| Return true if this widget is mapped.

|

| winfo\_manager(self)

| Return the window manager name for this widget.

|

| winfo\_name(self)

| Return the name of this widget.

|

| winfo\_parent(self)

| Return the name of the parent of this widget.

|

| winfo\_pathname(self, id, displayof=0)

| Return the pathname of the widget given by ID.

|

| winfo\_pixels(self, number)

| Rounded integer value of winfo\_fpixels.

|

| winfo\_pointerx(self)

| Return the x coordinate of the pointer on the root window.

|

| winfo\_pointerxy(self)

| Return a tuple of x and y coordinates of the pointer on the root window.

|

| winfo\_pointery(self)

| Return the y coordinate of the pointer on the root window.

|

| winfo\_reqheight(self)

| Return requested height of this widget.

|

| winfo\_reqwidth(self)

| Return requested width of this widget.

|

| winfo\_rgb(self, color)

| Return tuple of decimal values for red, green, blue for

| COLOR in this widget.

|

| winfo\_rootx(self)

| Return x coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_rooty(self)

| Return y coordinate of upper left corner of this widget on the

| root window.

|

| winfo\_screen(self)

| Return the screen name of this widget.

|

| winfo\_screencells(self)

| Return the number of the cells in the colormap of the screen

| of this widget.

|

| winfo\_screendepth(self)

| Return the number of bits per pixel of the root window of the

| screen of this widget.

|

| winfo\_screenheight(self)

| Return the number of pixels of the height of the screen of this widget

| in pixel.

|

| winfo\_screenmmheight(self)

| Return the number of pixels of the height of the screen of

| this widget in mm.

|

| winfo\_screenmmwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in mm.

|

| winfo\_screenvisual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the default

| colormodel of this screen.

|

| winfo\_screenwidth(self)

| Return the number of pixels of the width of the screen of

| this widget in pixel.

|

| winfo\_server(self)

| Return information of the X-Server of the screen of this widget in

| the form "XmajorRminor vendor vendorVersion".

|

| winfo\_toplevel(self)

| Return the toplevel widget of this widget.

|

| winfo\_viewable(self)

| Return true if the widget and all its higher ancestors are mapped.

|

| winfo\_visual(self)

| Return one of the strings directcolor, grayscale, pseudocolor,

| staticcolor, staticgray, or truecolor for the

| colormodel of this widget.

|

| winfo\_visualid(self)

| Return the X identifier for the visual for this widget.

|

| winfo\_visualsavailable(self, includeids=False)

| Return a list of all visuals available for the screen

| of this widget.

|

| Each item in the list consists of a visual name (see winfo\_visual), a

| depth and if includeids is true is given also the X identifier.

|

| winfo\_vrootheight(self)

| Return the height of the virtual root window associated with this

| widget in pixels. If there is no virtual root window return the

| height of the screen.

|

| winfo\_vrootwidth(self)

| Return the width of the virtual root window associated with this

| widget in pixel. If there is no virtual root window return the

| width of the screen.

|

| winfo\_vrootx(self)

| Return the x offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_vrooty(self)

| Return the y offset of the virtual root relative to the root

| window of the screen of this widget.

|

| winfo\_width(self)

| Return the width of this widget.

|

| winfo\_x(self)

| Return the x coordinate of the upper left corner of this widget

| in the parent.

|

| winfo\_y(self)

| Return the y coordinate of the upper left corner of this widget

| in the parent.

|

| ----------------------------------------------------------------------

| Data descriptors inherited from Misc:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| ----------------------------------------------------------------------

| Methods inherited from Pack:

|

| forget = pack\_forget(self)

|

| info = pack\_info(self)

|

| pack = pack\_configure(self, cnf={}, \*\*kw)

|

| pack\_configure(self, cnf={}, \*\*kw)

| Pack a widget in the parent widget. Use as options:

| after=widget - pack it after you have packed widget

| anchor=NSEW (or subset) - position widget according to

| given direction

| before=widget - pack it before you will pack widget

| expand=bool - expand widget if parent size grows

| fill=NONE or X or Y or BOTH - fill widget if widget grows

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| side=TOP or BOTTOM or LEFT or RIGHT - where to add this widget.

|

| pack\_forget(self)

| Unmap this widget and do not use it for the packing order.

|

| pack\_info(self)

| Return information about the packing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Place:

|

| place = place\_configure(self, cnf={}, \*\*kw)

|

| place\_configure(self, cnf={}, \*\*kw)

| Place a widget in the parent widget. Use as options:

| in=master - master relative to which the widget is placed

| in\_=master - see 'in' option description

| x=amount - locate anchor of this widget at position x of master

| y=amount - locate anchor of this widget at position y of master

| relx=amount - locate anchor of this widget between 0.0 and 1.0

| relative to width of master (1.0 is right edge)

| rely=amount - locate anchor of this widget between 0.0 and 1.0

| relative to height of master (1.0 is bottom edge)

| anchor=NSEW (or subset) - position anchor according to given direction

| width=amount - width of this widget in pixel

| height=amount - height of this widget in pixel

| relwidth=amount - width of this widget between 0.0 and 1.0

| relative to width of master (1.0 is the same width

| as the master)

| relheight=amount - height of this widget between 0.0 and 1.0

| relative to height of master (1.0 is the same

| height as the master)

| bordermode="inside" or "outside" - whether to take border width of

| master widget into account

|

| place\_forget(self)

| Unmap this widget.

|

| place\_info(self)

| Return information about the placing options

| for this widget.

|

| ----------------------------------------------------------------------

| Methods inherited from Grid:

|

| grid = grid\_configure(self, cnf={}, \*\*kw)

|

| grid\_configure(self, cnf={}, \*\*kw)

| Position a widget in the parent widget in a grid. Use as options:

| column=number - use cell identified with given column (starting with 0)

| columnspan=number - this widget will span several columns

| in=master - use master to contain this widget

| in\_=master - see 'in' option description

| ipadx=amount - add internal padding in x direction

| ipady=amount - add internal padding in y direction

| padx=amount - add padding in x direction

| pady=amount - add padding in y direction

| row=number - use cell identified with given row (starting with 0)

| rowspan=number - this widget will span several rows

| sticky=NSEW - if cell is larger on which sides will this

| widget stick to the cell boundary

|

| grid\_forget(self)

| Unmap this widget.

|

| grid\_info(self)

| Return information about the options

| for positioning this widget in a grid.

|

| grid\_remove(self)

| Unmap this widget but remember the grid options.

|

| location = grid\_location(self, x, y)

class Wm(builtins.object)

| Provides functions for the communication with the window manager.

|

| Methods defined here:

|

| aspect = wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

|

| attributes = wm\_attributes(self, \*args)

|

| client = wm\_client(self, name=None)

|

| colormapwindows = wm\_colormapwindows(self, \*wlist)

|

| command = wm\_command(self, value=None)

|

| deiconify = wm\_deiconify(self)

|

| focusmodel = wm\_focusmodel(self, model=None)

|

| forget = wm\_forget(self, window)

|

| frame = wm\_frame(self)

|

| geometry = wm\_geometry(self, newGeometry=None)

|

| grid = wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

|

| group = wm\_group(self, pathName=None)

|

| iconbitmap = wm\_iconbitmap(self, bitmap=None, default=None)

|

| iconify = wm\_iconify(self)

|

| iconmask = wm\_iconmask(self, bitmap=None)

|

| iconname = wm\_iconname(self, newName=None)

|

| iconphoto = wm\_iconphoto(self, default=False, \*args)

|

| iconposition = wm\_iconposition(self, x=None, y=None)

|

| iconwindow = wm\_iconwindow(self, pathName=None)

|

| manage = wm\_manage(self, widget)

|

| maxsize = wm\_maxsize(self, width=None, height=None)

|

| minsize = wm\_minsize(self, width=None, height=None)

|

| overrideredirect = wm\_overrideredirect(self, boolean=None)

|

| positionfrom = wm\_positionfrom(self, who=None)

|

| protocol = wm\_protocol(self, name=None, func=None)

|

| resizable = wm\_resizable(self, width=None, height=None)

|

| sizefrom = wm\_sizefrom(self, who=None)

|

| state = wm\_state(self, newstate=None)

|

| title = wm\_title(self, string=None)

|

| transient = wm\_transient(self, master=None)

|

| withdraw = wm\_withdraw(self)

|

| wm\_aspect(self, minNumer=None, minDenom=None, maxNumer=None, maxDenom=None)

| Instruct the window manager to set the aspect ratio (width/height)

| of this widget to be between MINNUMER/MINDENOM and MAXNUMER/MAXDENOM. Return a tuple

| of the actual values if no argument is given.

|

| wm\_attributes(self, \*args)

| This subcommand returns or sets platform specific attributes

|

| The first form returns a list of the platform specific flags and

| their values. The second form returns the value for the specific

| option. The third form sets one or more of the values. The values

| are as follows:

|

| On Windows, -disabled gets or sets whether the window is in a

| disabled state. -toolwindow gets or sets the style of the window

| to toolwindow (as defined in the MSDN). -topmost gets or sets

| whether this is a topmost window (displays above all other

| windows).

|

| On Macintosh, XXXXX

|

| On Unix, there are currently no special attribute values.

|

| wm\_client(self, name=None)

| Store NAME in WM\_CLIENT\_MACHINE property of this widget. Return

| current value.

|

| wm\_colormapwindows(self, \*wlist)

| Store list of window names (WLIST) into WM\_COLORMAPWINDOWS property

| of this widget. This list contains windows whose colormaps differ from their

| parents. Return current list of widgets if WLIST is empty.

|

| wm\_command(self, value=None)

| Store VALUE in WM\_COMMAND property. It is the command

| which shall be used to invoke the application. Return current

| command if VALUE is None.

|

| wm\_deiconify(self)

| Deiconify this widget. If it was never mapped it will not be mapped.

| On Windows it will raise this widget and give it the focus.

|

| wm\_focusmodel(self, model=None)

| Set focus model to MODEL. "active" means that this widget will claim

| the focus itself, "passive" means that the window manager shall give

| the focus. Return current focus model if MODEL is None.

|

| wm\_forget(self, window)

| The window will be unmapped from the screen and will no longer

| be managed by wm. toplevel windows will be treated like frame

| windows once they are no longer managed by wm, however, the menu

| option configuration will be remembered and the menus will return

| once the widget is managed again.

|

| wm\_frame(self)

| Return identifier for decorative frame of this widget if present.

|

| wm\_geometry(self, newGeometry=None)

| Set geometry to NEWGEOMETRY of the form =widthxheight+x+y. Return

| current value if None is given.

|

| wm\_grid(self, baseWidth=None, baseHeight=None, widthInc=None, heightInc=None)

| Instruct the window manager that this widget shall only be

| resized on grid boundaries. WIDTHINC and HEIGHTINC are the width and

| height of a grid unit in pixels. BASEWIDTH and BASEHEIGHT are the

| number of grid units requested in Tk\_GeometryRequest.

|

| wm\_group(self, pathName=None)

| Set the group leader widgets for related widgets to PATHNAME. Return

| the group leader of this widget if None is given.

|

| wm\_iconbitmap(self, bitmap=None, default=None)

| Set bitmap for the iconified widget to BITMAP. Return

| the bitmap if None is given.

|

| Under Windows, the DEFAULT parameter can be used to set the icon

| for the widget and any descendents that don't have an icon set

| explicitly. DEFAULT can be the relative path to a .ico file

| (example: root.iconbitmap(default='myicon.ico') ). See Tk

| documentation for more information.

|

| wm\_iconify(self)

| Display widget as icon.

|

| wm\_iconmask(self, bitmap=None)

| Set mask for the icon bitmap of this widget. Return the

| mask if None is given.

|

| wm\_iconname(self, newName=None)

| Set the name of the icon for this widget. Return the name if

| None is given.

|

| wm\_iconphoto(self, default=False, \*args)

| Sets the titlebar icon for this window based on the named photo

| images passed through args. If default is True, this is applied to

| all future created toplevels as well.

|

| The data in the images is taken as a snapshot at the time of

| invocation. If the images are later changed, this is not reflected

| to the titlebar icons. Multiple images are accepted to allow

| different images sizes to be provided. The window manager may scale

| provided icons to an appropriate size.

|

| On Windows, the images are packed into a Windows icon structure.

| This will override an icon specified to wm\_iconbitmap, and vice

| versa.

|

| On X, the images are arranged into the \_NET\_WM\_ICON X property,

| which most modern window managers support. An icon specified by

| wm\_iconbitmap may exist simultaneously.

|

| On Macintosh, this currently does nothing.

|

| wm\_iconposition(self, x=None, y=None)

| Set the position of the icon of this widget to X and Y. Return

| a tuple of the current values of X and X if None is given.

|

| wm\_iconwindow(self, pathName=None)

| Set widget PATHNAME to be displayed instead of icon. Return the current

| value if None is given.

|

| wm\_manage(self, widget)

| The widget specified will become a stand alone top-level window.

| The window will be decorated with the window managers title bar,

| etc.

|

| wm\_maxsize(self, width=None, height=None)

| Set max WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_minsize(self, width=None, height=None)

| Set min WIDTH and HEIGHT for this widget. If the window is gridded

| the values are given in grid units. Return the current values if None

| is given.

|

| wm\_overrideredirect(self, boolean=None)

| Instruct the window manager to ignore this widget

| if BOOLEAN is given with 1. Return the current value if None

| is given.

|

| wm\_positionfrom(self, who=None)

| Instruct the window manager that the position of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_protocol(self, name=None, func=None)

| Bind function FUNC to command NAME for this widget.

| Return the function bound to NAME if None is given. NAME could be

| e.g. "WM\_SAVE\_YOURSELF" or "WM\_DELETE\_WINDOW".

|

| wm\_resizable(self, width=None, height=None)

| Instruct the window manager whether this width can be resized

| in WIDTH or HEIGHT. Both values are boolean values.

|

| wm\_sizefrom(self, who=None)

| Instruct the window manager that the size of this widget shall

| be defined by the user if WHO is "user", and by its own policy if WHO is

| "program".

|

| wm\_state(self, newstate=None)

| Query or set the state of this widget as one of normal, icon,

| iconic (see wm\_iconwindow), withdrawn, or zoomed (Windows only).

|

| wm\_title(self, string=None)

| Set the title of this widget.

|

| wm\_transient(self, master=None)

| Instruct the window manager that this widget is transient

| with regard to widget MASTER.

|

| wm\_withdraw(self)

| Withdraw this widget from the screen such that it is unmapped

| and forgotten by the window manager. Re-draw it with wm\_deiconify.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class XView(builtins.object)

| Mix-in class for querying and changing the horizontal position

| of a widget's window.

|

| Methods defined here:

|

| xview(self, \*args)

| Query and change the horizontal position of the view.

|

| xview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total width of the canvas is off-screen to the left.

|

| xview\_scroll(self, number, what)

| Shift the x-view according to NUMBER which is measured in "units"

| or "pages" (WHAT).

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

class YView(builtins.object)

| Mix-in class for querying and changing the vertical position

| of a widget's window.

|

| Methods defined here:

|

| yview(self, \*args)

| Query and change the vertical position of the view.

|

| yview\_moveto(self, fraction)

| Adjusts the view in the window so that FRACTION of the

| total height of the canvas is off-screen to the top.

|

| yview\_scroll(self, number, what)

| Shift the y-view according to NUMBER which is measured in

| "units" or "pages" (WHAT).

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

FUNCTIONS

NoDefaultRoot()

Inhibit setting of default root window.

Call this function to inhibit that the first instance of

Tk is used for windows without an explicit parent window.

Tcl(screenName=None, baseName=None, className='Tk', useTk=0)

getboolean(s)

Convert true and false to integer values 1 and 0.

image\_names()

image\_types()

mainloop(n=0)

Run the main loop of Tcl.

DATA

ACTIVE = 'active'

ALL = 'all'

ANCHOR = 'anchor'

ARC = 'arc'

BASELINE = 'baseline'

BEVEL = 'bevel'

BOTH = 'both'

BOTTOM = 'bottom'

BROWSE = 'browse'

BUTT = 'butt'

CASCADE = 'cascade'

CENTER = 'center'

CHAR = 'char'

CHECKBUTTON = 'checkbutton'

CHORD = 'chord'

COMMAND = 'command'

CURRENT = 'current'

DISABLED = 'disabled'

DOTBOX = 'dotbox'

E = 'e'

END = 'end'

EW = 'ew'

EXCEPTION = 8

EXTENDED = 'extended'

FALSE = 0

FIRST = 'first'

FLAT = 'flat'

GROOVE = 'groove'

HIDDEN = 'hidden'

HORIZONTAL = 'horizontal'

INSERT = 'insert'

INSIDE = 'inside'

LAST = 'last'

LEFT = 'left'

MITER = 'miter'

MOVETO = 'moveto'

MULTIPLE = 'multiple'

N = 'n'

NE = 'ne'

NO = 0

NONE = 'none'

NORMAL = 'normal'

NS = 'ns'

NSEW = 'nsew'

NUMERIC = 'numeric'

NW = 'nw'

OFF = 0

ON = 1

OUTSIDE = 'outside'

PAGES = 'pages'

PIESLICE = 'pieslice'

PROJECTING = 'projecting'

RADIOBUTTON = 'radiobutton'

RAISED = 'raised'

READABLE = 2

RIDGE = 'ridge'

RIGHT = 'right'

ROUND = 'round'

S = 's'

SCROLL = 'scroll'

SE = 'se'

SEL = 'sel'

SEL\_FIRST = 'sel.first'

SEL\_LAST = 'sel.last'

SEPARATOR = 'separator'

SINGLE = 'single'

SOLID = 'solid'

SUNKEN = 'sunken'

SW = 'sw'

TOP = 'top'

TRUE = 1

TclVersion = 8.6

TkVersion = 8.6

UNDERLINE = 'underline'

UNITS = 'units'

VERTICAL = 'vertical'

W = 'w'

WORD = 'word'

WRITABLE = 4

X = 'x'

Y = 'y'

YES = 1

wantobjects = 1

FILE

c:\users\m parth\appdata\local\programs\python\python37\lib\tkinter\\_\_init\_\_.py

Help on package matplotlib:

NAME

matplotlib - This is an object-oriented plotting library.

DESCRIPTION

A procedural interface is provided by the companion pyplot module,

which may be imported directly, e.g.::

import matplotlib.pyplot as plt

or using ipython::

ipython

at your terminal, followed by::

In [1]: %matplotlib

In [2]: import matplotlib.pyplot as plt

at the ipython shell prompt.

For the most part, direct use of the object-oriented library is

encouraged when programming; pyplot is primarily for working

interactively. The

exceptions are the pyplot commands :func:`~matplotlib.pyplot.figure`,

:func:`~matplotlib.pyplot.subplot`,

:func:`~matplotlib.pyplot.subplots`, and

:func:`~pyplot.savefig`, which can greatly simplify scripting.

Modules include:

:mod:`matplotlib.axes`

defines the :class:`~matplotlib.axes.Axes` class. Most pyplot

commands are wrappers for :class:`~matplotlib.axes.Axes`

methods. The axes module is the highest level of OO access to

the library.

:mod:`matplotlib.figure`

defines the :class:`~matplotlib.figure.Figure` class.

:mod:`matplotlib.artist`

defines the :class:`~matplotlib.artist.Artist` base class for

all classes that draw things.

:mod:`matplotlib.lines`

defines the :class:`~matplotlib.lines.Line2D` class for

drawing lines and markers

:mod:`matplotlib.patches`

defines classes for drawing polygons

:mod:`matplotlib.text`

defines the :class:`~matplotlib.text.Text`,

:class:`~matplotlib.text.TextWithDash`, and

:class:`~matplotlib.text.Annotate` classes

:mod:`matplotlib.image`

defines the :class:`~matplotlib.image.AxesImage` and

:class:`~matplotlib.image.FigureImage` classes

:mod:`matplotlib.collections`

classes for efficient drawing of groups of lines or polygons

:mod:`matplotlib.colors`

classes for interpreting color specifications and for making

colormaps

:mod:`matplotlib.cm`

colormaps and the :class:`~matplotlib.image.ScalarMappable`

mixin class for providing color mapping functionality to other

classes

:mod:`matplotlib.ticker`

classes for calculating tick mark locations and for formatting

tick labels

:mod:`matplotlib.backends`

a subpackage with modules for various gui libraries and output

formats

The base matplotlib namespace includes:

:data:`~matplotlib.rcParams`

a global dictionary of default configuration settings. It is

initialized by code which may be overridden by a matplotlibrc

file.

:func:`~matplotlib.rc`

a function for setting groups of rcParams values

:func:`~matplotlib.use`

a function for setting the matplotlib backend. If used, this

function must be called immediately after importing matplotlib

for the first time. In particular, it must be called

\*\*before\*\* importing pyplot (if pyplot is imported).

matplotlib was initially written by John D. Hunter (1968-2012) and is now

developed and maintained by a host of others.

Occasionally the internal documentation (python docstrings) will refer

to MATLAB&reg;, a registered trademark of The MathWorks, Inc.

PACKAGE CONTENTS

\_animation\_data

\_cm

\_cm\_listed

\_color\_data

\_constrained\_layout

\_contour

\_image

\_layoutbox

\_mathtext\_data

\_path

\_png

\_pylab\_helpers

\_qhull

\_tri

\_version

afm

animation

artist

axes (package)

axis

backend\_bases

backend\_managers

backend\_tools

backends (package)

bezier

blocking\_input

category

cbook (package)

cm

collections

colorbar

colors

compat (package)

container

contour

dates

docstring

dviread

figure

font\_manager

fontconfig\_pattern

ft2font

gridspec

hatch

image

legend

legend\_handler

lines

markers

mathtext

mlab

offsetbox

patches

path

patheffects

projections (package)

pylab

pyplot

quiver

rcsetup

sankey

scale

sphinxext (package)

spines

stackplot

streamplot

style (package)

table

testing (package)

texmanager

text

textpath

ticker

tight\_bbox

tight\_layout

transforms

tri (package)

ttconv

type1font

units

widgets

CLASSES

builtins.dict(builtins.object)

RcParams(collections.abc.MutableMapping, builtins.dict)

builtins.object

rc\_context

collections.abc.MutableMapping(collections.abc.Mapping)

RcParams(collections.abc.MutableMapping, builtins.dict)

class RcParams(collections.abc.MutableMapping, builtins.dict)

| RcParams(\*args, \*\*kwargs)

|

| A dictionary object including validation

|

| validating functions are defined and associated with rc parameters in

| :mod:`matplotlib.rcsetup`

|

| Method resolution order:

| RcParams

| collections.abc.MutableMapping

| collections.abc.Mapping

| collections.abc.Collection

| collections.abc.Sized

| collections.abc.Iterable

| collections.abc.Container

| builtins.dict

| builtins.object

|

| Methods defined here:

|

| \_\_getitem\_\_(self, key)

| x.\_\_getitem\_\_(y) <==> x[y]

|

| \_\_init\_\_(self, \*args, \*\*kwargs)

| Initialize self. See help(type(self)) for accurate signature.

|

| \_\_iter\_\_(self)

| Yield sorted list of keys.

|

| \_\_len\_\_(self)

| Return len(self).

|

| \_\_repr\_\_(self)

| Return repr(self).

|

| \_\_setitem\_\_(self, key, val)

| Set self[key] to value.

|

| \_\_str\_\_(self)

| Return str(self).

|

| copy(self)

| D.copy() -> a shallow copy of D

|

| find\_all(self, pattern)

| Return the subset of this RcParams dictionary whose keys match,

| using :func:`re.search`, the given ``pattern``.

|

| .. note::

|

| Changes to the returned dictionary are \*not\* propagated to

| the parent RcParams dictionary.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

|

| msg\_backend\_obsolete

|

| msg\_depr

|

| msg\_depr\_ignore

|

| msg\_depr\_set

|

| msg\_obsolete

|

| ----------------------------------------------------------------------

| Data and other attributes defined here:

|

| \_\_abstractmethods\_\_ = frozenset({'\_\_delitem\_\_'})

|

| validate = {'\_internal.classic\_mode': <function validate\_bool>, 'agg.p...

|

| ----------------------------------------------------------------------

| Methods inherited from collections.abc.MutableMapping:

|

| \_\_delitem\_\_(self, key)

|

| clear(self)

| D.clear() -> None. Remove all items from D.

|

| pop(self, key, default=<object object at 0x0000003221EB7140>)

| D.pop(k[,d]) -> v, remove specified key and return the corresponding value.

| If key is not found, d is returned if given, otherwise KeyError is raised.

|

| popitem(self)

| D.popitem() -> (k, v), remove and return some (key, value) pair

| as a 2-tuple; but raise KeyError if D is empty.

|

| setdefault(self, key, default=None)

| D.setdefault(k[,d]) -> D.get(k,d), also set D[k]=d if k not in D

|

| update(\*args, \*\*kwds)

| D.update([E, ]\*\*F) -> None. Update D from mapping/iterable E and F.

| If E present and has a .keys() method, does: for k in E: D[k] = E[k]

| If E present and lacks .keys() method, does: for (k, v) in E: D[k] = v

| In either case, this is followed by: for k, v in F.items(): D[k] = v

|

| ----------------------------------------------------------------------

| Methods inherited from collections.abc.Mapping:

|

| \_\_contains\_\_(self, key)

|

| \_\_eq\_\_(self, other)

| Return self==value.

|

| get(self, key, default=None)

| D.get(k[,d]) -> D[k] if k in D, else d. d defaults to None.

|

| items(self)

| D.items() -> a set-like object providing a view on D's items

|

| keys(self)

| D.keys() -> a set-like object providing a view on D's keys

|

| values(self)

| D.values() -> an object providing a view on D's values

|

| ----------------------------------------------------------------------

| Data and other attributes inherited from collections.abc.Mapping:

|

| \_\_hash\_\_ = None

|

| \_\_reversed\_\_ = None

|

| ----------------------------------------------------------------------

| Class methods inherited from collections.abc.Collection:

|

| \_\_subclasshook\_\_(C) from abc.ABCMeta

| Abstract classes can override this to customize issubclass().

|

| This is invoked early on by abc.ABCMeta.\_\_subclasscheck\_\_().

| It should return True, False or NotImplemented. If it returns

| NotImplemented, the normal algorithm is used. Otherwise, it

| overrides the normal algorithm (and the outcome is cached).

|

| ----------------------------------------------------------------------

| Methods inherited from builtins.dict:

|

| \_\_ge\_\_(self, value, /)

| Return self>=value.

|

| \_\_getattribute\_\_(self, name, /)

| Return getattr(self, name).

|

| \_\_gt\_\_(self, value, /)

| Return self>value.

|

| \_\_le\_\_(self, value, /)

| Return self<=value.

|

| \_\_lt\_\_(self, value, /)

| Return self<value.

|

| \_\_ne\_\_(self, value, /)

| Return self!=value.

|

| \_\_sizeof\_\_(...)

| D.\_\_sizeof\_\_() -> size of D in memory, in bytes

|

| ----------------------------------------------------------------------

| Class methods inherited from builtins.dict:

|

| fromkeys(iterable, value=None, /) from abc.ABCMeta

| Create a new dictionary with keys from iterable and values set to value.

|

| ----------------------------------------------------------------------

| Static methods inherited from builtins.dict:

|

| \_\_new\_\_(\*args, \*\*kwargs) from builtins.type

| Create and return a new object. See help(type) for accurate signature.

class rc\_context(builtins.object)

| rc\_context(rc=None, fname=None)

|

| Return a context manager for managing rc settings.

|

| This allows one to do::

|

| with mpl.rc\_context(fname='screen.rc'):

| plt.plot(x, a)

| with mpl.rc\_context(fname='print.rc'):

| plt.plot(x, b)

| plt.plot(x, c)

|

| The 'a' vs 'x' and 'c' vs 'x' plots would have settings from

| 'screen.rc', while the 'b' vs 'x' plot would have settings from

| 'print.rc'.

|

| A dictionary can also be passed to the context manager::

|

| with mpl.rc\_context(rc={'text.usetex': True}, fname='screen.rc'):

| plt.plot(x, a)

|

| The 'rc' dictionary takes precedence over the settings loaded from

| 'fname'. Passing a dictionary only is also valid. For example a

| common usage is::

|

| with mpl.rc\_context(rc={'interactive': False}):

| fig, ax = plt.subplots()

| ax.plot(range(3), range(3))

| fig.savefig('A.png', format='png')

| plt.close(fig)

|

| Methods defined here:

|

| \_\_enter\_\_(self)

|

| \_\_exit\_\_(self, exc\_type, exc\_value, exc\_tb)

|

| \_\_init\_\_(self, rc=None, fname=None)

| Initialize self. See help(type(self)) for accurate signature.

|

| ----------------------------------------------------------------------

| Data descriptors defined here:

|

| \_\_dict\_\_

| dictionary for instance variables (if defined)

|

| \_\_weakref\_\_

| list of weak references to the object (if defined)

FUNCTIONS

checkdep\_dvipng()

[\*Deprecated\*]

Notes

-----

.. deprecated:: 3.1

\

checkdep\_ghostscript()

[\*Deprecated\*]

Notes

-----

.. deprecated:: 3.1

\

checkdep\_inkscape()

[\*Deprecated\*]

Notes

-----

.. deprecated:: 3.1

\

checkdep\_pdftops()

[\*Deprecated\*]

Notes

-----

.. deprecated:: 3.1

\

checkdep\_ps\_distiller(s)

checkdep\_usetex(s)

compare\_versions(a, b)

Return whether version \*a\* is greater than or equal to version \*b\*.

get\_backend()

Return the name of the current backend.

See Also

--------

matplotlib.use

get\_cachedir()

Return the location of the cache directory.

The procedure used to find the directory is the same as for

\_get\_config\_dir, except using `$XDG\_CACHE\_HOME`/`~/.cache` instead.

get\_configdir()

Return the string representing the configuration directory.

The directory is chosen as follows:

1. If the MPLCONFIGDIR environment variable is supplied, choose that.

2a. On Linux, follow the XDG specification and look first in

`$XDG\_CONFIG\_HOME`, if defined, or `$HOME/.config`.

2b. On other platforms, choose `$HOME/.matplotlib`.

3. If the chosen directory exists and is writable, use that as the

configuration directory.

4. If possible, create a temporary directory, and use it as the

configuration directory.

5. A writable directory could not be found or created; return None.

get\_data\_path()

get\_home()

Return the user's home directory.

If the user's home directory cannot be found, return None.

get\_py2exe\_datafiles()

[\*Deprecated\*]

Notes

-----

.. deprecated:: 3.1

\

interactive(b)

Set interactive mode to boolean b.

If b is True, then draw after every plotting command, e.g., after xlabel

is\_interactive()

Return true if plot mode is interactive

is\_url(filename)

Return True if string is an http, ftp, or file URL path.

matplotlib\_fname()

Get the location of the config file.

The file location is determined in the following order

- ``$PWD/matplotlibrc``

- ``$MATPLOTLIBRC`` if it is not a directory

- ``$MATPLOTLIBRC/matplotlibrc``

- ``$MPLCONFIGDIR/matplotlibrc``

- On Linux,

- ``$XDG\_CONFIG\_HOME/matplotlib/matplotlibrc`` (if ``$XDG\_CONFIG\_HOME``

is defined)

- or ``$HOME/.config/matplotlib/matplotlibrc`` (if ``$XDG\_CONFIG\_HOME``

is not defined)

- On other platforms,

- ``$HOME/.matplotlib/matplotlibrc`` if ``$HOME`` is defined

- Lastly, it looks in ``$MATPLOTLIBDATA/matplotlibrc``, which should always

exist.

rc(group, \*\*kwargs)

Set the current rc params. \*group\* is the grouping for the rc, e.g.,

for ``lines.linewidth`` the group is ``lines``, for

``axes.facecolor``, the group is ``axes``, and so on. Group may

also be a list or tuple of group names, e.g., (\*xtick\*, \*ytick\*).

\*kwargs\* is a dictionary attribute name/value pairs, e.g.,::

rc('lines', linewidth=2, color='r')

sets the current rc params and is equivalent to::

rcParams['lines.linewidth'] = 2

rcParams['lines.color'] = 'r'

The following aliases are available to save typing for interactive

users:

===== =================

Alias Property

===== =================

'lw' 'linewidth'

'ls' 'linestyle'

'c' 'color'

'fc' 'facecolor'

'ec' 'edgecolor'

'mew' 'markeredgewidth'

'aa' 'antialiased'

===== =================

Thus you could abbreviate the above rc command as::

rc('lines', lw=2, c='r')

Note you can use python's kwargs dictionary facility to store

dictionaries of default parameters. e.g., you can customize the

font rc as follows::

font = {'family' : 'monospace',

'weight' : 'bold',

'size' : 'larger'}

rc('font', \*\*font) # pass in the font dict as kwargs

This enables you to easily switch between several configurations. Use

``matplotlib.style.use('default')`` or :func:`~matplotlib.rcdefaults` to

restore the default rc params after changes.

rc\_file(fname, \*, use\_default\_template=True)

Update rc params from file.

Style-blacklisted rc params (defined in

`matplotlib.style.core.STYLE\_BLACKLIST`) are not updated.

Parameters

----------

fname : str

Name of file parsed for matplotlib settings.

use\_default\_template : bool

If True, initialize with default parameters before updating with those

in the given file. If False, the current configuration persists

and only the parameters specified in the file are updated.

rc\_file\_defaults()

Restore the rc params from the original rc file loaded by Matplotlib.

Style-blacklisted rc params (defined in

`matplotlib.style.core.STYLE\_BLACKLIST`) are not updated.

rc\_params(fail\_on\_error=False)

Return a :class:`matplotlib.RcParams` instance from the

default matplotlib rc file.

rc\_params\_from\_file(fname, fail\_on\_error=False, use\_default\_template=True)

Return :class:`matplotlib.RcParams` from the contents of the given file.

Parameters

----------

fname : str

Name of file parsed for matplotlib settings.

fail\_on\_error : bool

If True, raise an error when the parser fails to convert a parameter.

use\_default\_template : bool

If True, initialize with default parameters before updating with those

in the given file. If False, the configuration class only contains the

parameters specified in the file. (Useful for updating dicts.)

rcdefaults()

Restore the rc params from Matplotlib's internal default style.

Style-blacklisted rc params (defined in

`matplotlib.style.core.STYLE\_BLACKLIST`) are not updated.

See Also

--------

rc\_file\_defaults

Restore the rc params from the rc file originally loaded by Matplotlib.

matplotlib.style.use :

Use a specific style file. Call ``style.use('default')`` to restore

the default style.

set\_loglevel(level)

Sets the Matplotlib's root logger and root logger handler level, creating

the handler if it does not exist yet.

Typically, one should call ``set\_loglevel("info")`` or

``set\_loglevel("debug")`` to get additional debugging information.

Parameters

----------

level : {"notset", "debug", "info", "warning", "error", "critical"}

The log level of the handler.

Notes

-----

The first time this function is called, an additional handler is attached

to Matplotlib's root handler; this handler is reused every time and this

function simply manipulates the logger and handler's level.

test(verbosity=None, coverage=False, switch\_backend\_warn=True, recursionlimit=0, \*\*kwargs)

Run the matplotlib test suite.

tk\_window\_focus()

[\*Deprecated\*] Return true if focus maintenance under TkAgg on win32 is on.

This currently works only for python.exe and IPython.exe.

Both IDLE and Pythonwin.exe fail badly when tk\_window\_focus is on.

Notes

-----

.. deprecated:: 3.1

use(backend, warn=False, force=True)

Select the backend used for rendering and GUI integration.

Parameters

----------

backend : str

The backend to switch to. This can either be one of the standard

backend names, which are case-insensitive:

- interactive backends:

GTK3Agg, GTK3Cairo, MacOSX, nbAgg,

Qt4Agg, Qt4Cairo, Qt5Agg, Qt5Cairo,

TkAgg, TkCairo, WebAgg, WX, WXAgg, WXCairo

- non-interactive backends:

agg, cairo, pdf, pgf, ps, svg, template

or a string of the form: ``module://my.module.name``.

warn : bool, optional, default: False

If True and not \*force\*, warn that the call will have no effect if

this is called after pyplot has been imported and a backend is set up.

force : bool, optional, default: True

If True, attempt to switch the backend. An ImportError is raised if

an interactive backend is selected, but another interactive

backend has already started.

See Also

--------

:ref:`backends`

matplotlib.get\_backend

DATA

URL\_REGEX = re.compile('http://|https://|ftp://|file://|file:\\\\')

\_\_bibtex\_\_ = '@Article{Hunter:2007,\n Author = {Hunter, J. ...ishe...

defaultParams = {'\_internal.classic\_mode': [False, <function validate\_...

default\_test\_modules = ['matplotlib.tests', 'matplotlib.sphinxext.test...

rcParams = RcParams({'\_internal.classic\_mode': False,

...nor.widt...

rcParamsDefault = RcParams({'\_internal.classic\_mode': False,

...n...

rcParamsOrig = RcParams({'\_internal.classic\_mode': False,

...nor....

VERSION

3.1.2

FILE

c:\users\m parth\appdata\local\programs\python\python37\lib\site-packages\matplotlib\\_\_init\_\_.py