Button Element

Button Element - Defines all possible buttons. The shortcuts such **as** Submit, FileBrowse, ... **each** create a Button

**Button**(button\_text="",

button\_type=7,

target=(**None**, **None**),

tooltip=**None**,

file\_types=(('ALL Files', '\*.\*'),),

initial\_folder=**None**,

disabled=**False**,

change\_submits=**False**,

enable\_events=**False**,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | (str) Text to be displayed on the button |
| button\_type | (int) You should NOT be setting this directly. ONLY the shortcut functions set this |
| target | Union[str, Tuple[int, int]] key or (row,col) target for the button. Note that -1 for column means 1 element to the left of this one. The constant ThisRow is used to indicate the current row. The Button itself is a valid target for some types of button |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| file\_types | Tuple[Tuple[str, str], ...] the filetypes that will be used to match files. To indicate all files: (("ALL Files", "*.*"),). Note - NOT SUPPORTED ON MAC |
| initial\_folder | (str) starting path for folders and files |
| disabled | (bool) If True button will be created disabled |
| click\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. If this button is a target, should it generate an event when filled in |
| image\_filename | (str) image filename if there is a button image. GIFs and PNGs only. |
| image\_data | Union[bytes, str] Raw or Base64 representation of the image to put on button. Choose either filename or data |
| image\_size | Tuple[int, int] Size of the image in pixels (width, height) |
| image\_subsample | (int) amount to reduce the size of the image. Divides the size by this number. 2=1/2, 3=1/3, 4=1/4, etc |
| border\_width | (int) width of border around button in pixels |
| size | Tuple[int, int] (width, height) of the button in characters wide, rows high |
| auto\_size\_button | (bool) if True the button size is sized to fit the text |
| button\_color | Tuple[str, str] (text color, background color) of button. Easy to remember which is which if you say "ON" between colors. "red" on "green". Note - Does not always work on Macs |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| bind\_return\_key | (bool) If True the return key will cause this button to be pressed |
| focus | (bool) if True, initial focus will be put on this button |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonCallBack

Not user callable! Called by tkinter when a button is clicked. This is where all the fun begins!

ButtonCallBack()

ButtonPressCallBack

Not a user callable method. Callback called by tkinter when a "realtime" button is pressed

**ButtonPressCallBack**(parm)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| parm | Event info passed in by tkinter |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

ButtonReleaseCallBack

Not a user callable function. Called by tkinter when a "realtime" button is released

**ButtonReleaseCallBack**(parm)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| parm | the event info from tkinter |

Click

Generates a click of the button as if the user clicked the button Calls the tkinter invoke method for the button

Click()

GetText

Returns the current text shown on a button

GetText()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) The text currently displayed on the button |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Button Element. Must call Window.Read or Window.Finalize prior

**Update**(text=**None**,

button\_color=(**None**, **None**),

disabled=**None**,

image\_data=**None**,

image\_filename=**None**,

visible=**None**,

image\_subsample=**None**,

image\_size=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) sets button text |
| button\_color | Tuple[str, str] (text color, background color) of button. Easy to remember which is which if you say "ON" between colors. "red" on "green" |
| disabled | (bool) disable or enable state of the element |
| image\_data | Union[bytes, str] Raw or Base64 representation of the image to put on button. Choose either filename or data |
| image\_filename | (str) image filename if there is a button image. GIFs and PNGs only. |
| visible | (bool) control visibility of element |
| image\_subsample | (int) amount to reduce the size of the image. Divides the size by this number. 2=1/2, 3=1/3, 4=1/4, etc |
| image\_size | Tuple[int, int] Size of the image in pixels (width, height) |

click

Generates a click of the button as if the user clicked the button Calls the tkinter invoke method for the button

click()

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Button Element. Must call Window.Read or Window.Finalize prior

update(text=**None**,

button\_color=(**None**, **None**),

disabled=**None**,

image\_data=**None**,

image\_filename=**None**,

visible=**None**,

image\_subsample=**None**,

image\_size=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) sets button text |
| button\_color | Tuple[str, str] (text color, background color) of button. Easy to remember which is which if you say "ON" between colors. "red" on "green" |
| disabled | (bool) disable or enable state of the element |
| image\_data | Union[bytes, str] Raw or Base64 representation of the image to put on button. Choose either filename or data |
| image\_filename | (str) image filename if there is a button image. GIFs and PNGs only. |
| visible | (bool) control visibility of element |
| image\_subsample | (int) amount to reduce the size of the image. Divides the size by this number. 2=1/2, 3=1/3, 4=1/4, etc |
| image\_size | Tuple[int, int] Size of the image in pixels (width, height) |

ButtonMenu Element

The Button Menu Element. Creates a button that when clicked will show a menu similar to right click menu

ButtonMenu(button\_text,

menu\_def,

tooltip=**None**,

disabled=**False**,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

pad=**None**,

key=**None**,

tearoff=**False**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | (str) Text to be displayed on the button |
| menu\_def | List[List[str]] A list of lists of Menu items to show when this element is clicked. See docs for format as they are the same for all menu types |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| disabled | (bool) If True button will be created disabled |
| image\_filename | (str) image filename if there is a button image. GIFs and PNGs only. |
| image\_data | Union[bytes, str] Raw or Base64 representation of the image to put on button. Choose either filename or data |
| image\_size | Tuple[int, int] Size of the image in pixels (width, height) |
| image\_subsample | (int) amount to reduce the size of the image. Divides the size by this number. 2=1/2, 3=1/3, 4=1/4, etc |
| border\_width | (int) width of border around button in pixels |
| size | Tuple[int, int] (width, height) of the button in characters wide, rows high |
| auto\_size\_button | (bool) if True the button size is sized to fit the text |
| button\_color | Tuple[str, str] (text color, background color) of button. Easy to remember which is which if you say "ON" between colors. "red" on "green" |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| tearoff | (bool) Determines if menus should allow them to be torn off |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Click

Generates a click of the button as if the user clicked the button Calls the tkinter invoke method for the button

Click()

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the ButtonMenu Element. Must call Window.Read or Window.Finalize prior

**Update**(menu\_definition, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| menu\_definition | (List[List]) New menu definition (in menu definition format) |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the ButtonMenu Element. Must call Window.Read or Window.Finalize prior

**update**(menu\_definition, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| menu\_definition | (List[List]) New menu definition (in menu definition format) |
| visible | (bool) control visibility of element |

Canvas Element

Canvas(canvas=**None**,

background\_color=**None**,

size=(**None**, **None**),

**pad**=**None**,

key=**None**,

tooltip=**None**,

right\_click\_menu=**None**,

visible=True,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| canvas | (tk.Canvas) Your own tk.Canvas if you already created it. Leave blank to create a Canvas |
| background\_color | (str) color of background |
| size | Tuple[int,int] (width in char, height in rows) size in pixels to make canvas |
| pad | Amount of padding to put around element |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

TKCanvas

property: TKCanvas

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

Checkbox Element

Checkbox Element - Displays a checkbox and **text** next to **it**

**Checkbox**(text,

default=**False**,

size=(**None**, **None**),

auto\_size\_text=**None**,

font=**None**,

background\_color=**None**,

text\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

disabled=**False**,

key=**None**,

pad=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) Text to display next to checkbox |
| default | (bool). Set to True if you want this checkbox initially checked |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) if True will size the element to match the length of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Checkbox events happen when an item changes |
| disabled | (bool) set disable state |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Return the current state of this checkbox

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (bool) Current state of checkbox |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Checkbox Element. Must call Window.Read or Window.Finalize prior. Note that changing visibility may cause element to change locations when made visible after invisible

Update(**value**=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (bool) if True checks the checkbox, False clears it |
| disabled | (bool) disable or enable element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Return the current state of this checkbox

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (bool) Current state of checkbox |

update

Changes some of the settings for the Checkbox Element. Must call Window.Read or Window.Finalize prior. Note that changing visibility may cause element to change locations when made visible after invisible

update(**value**=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (bool) if True checks the checkbox, False clears it |
| disabled | (bool) disable or enable element |
| visible | (bool) control visibility of element |

Column Element

A container element that **is** used **to** create a layout within your window's layout

Column(layout,

background\_color=None,

size=(None, None),

pad=None,

scrollable=False,

vertical\_scroll\_only=False,

right\_click\_menu=None,

key=None,

visible=True,

justification="left",

element\_justification="left",

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| layout | List[List[Element]] Layout that will be shown in the Column container |
| background\_color | (str) color of background of entire Column |
| size | Tuple[int, int] (width, height) size in pixels (doesn't work quite right, sometimes only 1 dimension is set by tkinter |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| scrollable | (bool) if True then scrollbars will be added to the column |
| vertical\_scroll\_only | (bool) if Truen then no horizontal scrollbar will be shown |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| visible | (bool) set visibility state of the element |
| justification | (str) set justification for the Column itself. Note entire row containing the Column will be affected |
| element\_justification | (str) All elements inside the Column will have this justification 'left', 'right', 'center' are valid values |
| metadata | (Any) User metadata that can be set to ANYTHING |

AddRow

Not recommended user call. Used to add rows of Elements to the Column Element.

**AddRow**(args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | List[Element] The list of elements for this row |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**Layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Column) Used for chaining |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Column Element. Must call Window.Read or Window.Finalize prior

**Update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Column) Used for chaining |

update

Changes some of the settings for the Column Element. Must call Window.Read or Window.Finalize prior

**update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

Combo Element

ComboBox Element - A combination of a single-line input and a drop-down menu. User can type **in** their own value or choose from list.

Combo(values,

default\_value=**None**,

size=(**None**, **None**),

auto\_size\_text=**None**,

background\_color=**None**,

text\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

disabled=**False**,

key=**None**,

pad=**None**,

tooltip=**None**,

readonly=**False**,

font=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] values to choose. While displayed as text, the items returned are what the caller supplied, not text |
| default\_value | (Any) Choice to be displayed as initial value. Must match one of values variable contents |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) True if element should be the same size as the contents |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| change\_submits | (bool) DEPRICATED DO NOT USE. Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Combo event is when a choice is made |
| disabled | (bool) set disable state for element |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text that will appear when mouse hovers over this element |
| readonly | (bool) make element readonly (user can't change). True means user cannot change |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Returns the current (right now) value of the Combo. DO NOT USE THIS AS THE NORMAL WAY OF READING A COMBO! You should be using values from your call to window.Read instead. Know what you're doing if you use it.

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Any, None] Returns the value of what is currently chosen |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Combo Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

values=**None**,

set\_to\_index=**None**,

disabled=**None**,

readonly=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) change which value is current selected hased on new list of previous list of choices |
| values | List[Any] change list of choices |
| set\_to\_index | (int) change selection to a particular choice starting with index = 0 |
| disabled | (bool) disable or enable state of the element |
| readonly | (bool) if True make element readonly (user cannot change any choices) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Returns the current (right now) value of the Combo. DO NOT USE THIS AS THE NORMAL WAY OF READING A COMBO! You should be using values from your call to window.Read instead. Know what you're doing if you use it.

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Any, None] Returns the value of what is currently chosen |

update

Changes some of the settings for the Combo Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

values=**None**,

set\_to\_index=**None**,

disabled=**None**,

readonly=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) change which value is current selected hased on new list of previous list of choices |
| values | List[Any] change list of choices |
| set\_to\_index | (int) change selection to a particular choice starting with index = 0 |
| disabled | (bool) disable or enable state of the element |
| readonly | (bool) if True make element readonly (user cannot change any choices) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) control visibility of element |

ErrorElement Element

A "dummy Element" that **is** returned **when** there are **error** conditions, **like** trying **to** find an element that*'s invalid*

**ErrorElement**(key=None, metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | Used with window.FindElement and with return values to uniquely identify this element |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

One of the method names found in other Elements. Used here to return an error string in case it's called

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) A warning text string. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Update method for the Error Element, an element that should not be directly used by developer

**Update**(silent\_on\_error=True,

args,

kwargs)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| silent\_on\_error | (bool) if False, then a Popup window will be shown |
| \*args | (Any) meant to "soak up" any normal parameters passed in |
| \*\*kwargs | (Any) meant to "soak up" any keyword parameters that were passed in |
|  |  |
| **return** | (ErrorElement) returns 'self' so call can be chained |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

One of the method names found in other Elements. Used here to return an error string in case it's called

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) A warning text string. |

update

Update method for the Error Element, an element that should not be directly used by developer

**update**(silent\_on\_error=True,

args,

kwargs)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| silent\_on\_error | (bool) if False, then a Popup window will be shown |
| \*args | (Any) meant to "soak up" any normal parameters passed in |
| \*\*kwargs | (Any) meant to "soak up" any keyword parameters that were passed in |
|  |  |
| **return** | (ErrorElement) returns 'self' so call can be chained |

Frame Element

A Frame Element that contains other Elements. Encloses with a line around elements and a **text** label.

Frame(title,

layout,

title\_color=None,

background\_color=None,

title\_location=None,

relief="groove",

size=(None, None),

font=None,

pad=None,

border\_width=None,

key=None,

tooltip=None,

right\_click\_menu=None,

visible=True,

element\_justification="left",

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| title | (str) text that is displayed as the Frame's "label" or title |
| layout | List[List[Elements]] The layout to put inside the Frame |
| title\_color | (str) color of the title text |
| background\_color | (str) background color of the Frame |
| title\_location | (enum) location to place the text title. Choices include: TITLE\_LOCATION\_TOP TITLE\_LOCATION\_BOTTOM TITLE\_LOCATION\_LEFT TITLE\_LOCATION\_RIGHT TITLE\_LOCATION\_TOP\_LEFT TITLE\_LOCATION\_TOP\_RIGHT TITLE\_LOCATION\_BOTTOM\_LEFT TITLE\_LOCATION\_BOTTOM\_RIGHT |
| relief | (enum) relief style. Values are same as other elements with reliefs. Choices include RELIEF\_RAISED RELIEF\_SUNKEN RELIEF\_FLAT RELIEF\_RIDGE RELIEF\_GROOVE RELIEF\_SOLID |
| size | Tuple[int, int] (width in characters, height in rows) (note this parameter may not always work) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| border\_width | (int) width of border around element in pixels |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| element\_justification | (str) All elements inside the Frame will have this justification 'left', 'right', 'center' are valid values |
| metadata | (Any) User metadata that can be set to ANYTHING |

AddRow

Not recommended user call. Used to add rows of Elements to the Frame Element.

**AddRow**(args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | List[Element] The list of elements for this row |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**Layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Frame) Used for chaining |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Frame Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**, visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) New text value to show on frame |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Frame) Used for chaining |

update

Changes some of the settings for the Frame Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**, visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) New text value to show on frame |
| visible | (bool) control visibility of element |

Graph Element

Creates an area **for** you **to** draw **on**. The MAGICAL **property** this Element has **is** **that** you interact **with** **the** element using your own coordinate system. This **is** an important point!! YOU define **where** **the** location **is** **for** (0,0). Want (0,0) **to** be **in** **the** **middle** **of** **the** graph like a math 4-quadrant graph? No problem! Set your lower left corner **to** be (-100,-100) **and** your upper right **to** be (100,100) **and** you've got yourself a graph **with** (0,0) **at** **the** center.

One **of** THE coolest **of** **the** Elements.

You can also use float values. To do so, be sure **and** **set** **the** float\_values parameter.

Mouse click **and** drag events are possible **and** return **the** (x,y) coordinates **of** **the** mouse

Drawing primitives return an "id" **that** **is** referenced when you want **to** operation **on** **that** item (e.g. **to** erase **it**)

**Graph**(canvas\_size,

graph\_bottom\_left,

graph\_top\_right,

background\_color=**None**,

pad=**None**,

change\_submits=**False**,

drag\_submits=**False**,

enable\_events=**False**,

key=**None**,

tooltip=**None**,

right\_click\_menu=**None**,

visible=**True**,

float\_values=**False**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| canvas\_size | Tuple[int, int] (width, height) size of the canvas area in pixels |
| graph\_bottom\_left | Tuple[int, int] (x,y) The bottoms left corner of your coordinate system |
| graph\_top\_right | Tuple[int, int] (x,y) The top right corner of your coordinate system |
| background\_color | (str) background color of the drawing area |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| change\_submits | (bool) \* DEPRICATED DO NOT USE! Same as enable\_events |
| drag\_submits | (bool) if True and Events are enabled for the Graph, will report Events any time the mouse moves while button down |
| enable\_events | (bool) If True then clicks on the Graph are immediately reported as an event. Use this instead of change\_submits |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element (Default = True) |
| float\_values | (bool) If True x,y coordinates are returned as floats, not ints |
| metadata | (Any) User metadata that can be set to ANYTHING |

BringFigureToFront

Changes Z-order of figures on the Graph. Brings the indicated figure to the front of all other drawn figures

**BringFigureToFront**(figure)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| figure | (int) value returned by tkinter when creating the figure / drawing |

ButtonPressCallBack

Not a user callable method. Used to get Graph click events. Called by tkinter when button is released

**ButtonPressCallBack**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (event) event info from tkinter. Contains the x and y coordinates of a click |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

ButtonReleaseCallBack

Not a user callable method. Used to get Graph click events. Called by tkinter when button is released

**ButtonReleaseCallBack**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (event) event info from tkinter. Note not used in this method |

DeleteFigure

Remove from the Graph the figure represented by id. The id is given to you anytime you call a drawing primitive

**DeleteFigure**(id)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| id | (int) the id returned to you when calling one of the drawing methods |

DrawArc

Draws different types of arcs. Uses a "bounding box" to define location

DrawArc(top\_left,

bottom\_right,

extent,

start\_angle,

style=None,

arc\_color="black")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| top\_left | Union[Tuple[int, int], Tuple[float, float]] the top left point of bounding rectangle |
| bottom\_right | Union[Tuple[int, int], Tuple[float, float]] the bottom right point of bounding rectangle |
| extent | (float) Andle to end drawing. Used in conjunction with start\_angle |
| start\_angle | (float) Angle to begin drawing. Used in conjunction with extent |
| style | (str) Valid choices are One of these Style strings- 'pieslice', 'chord', 'arc', 'first', 'last', 'butt', 'projecting', 'round', 'bevel', 'miter' |
| arc\_color | (str) color to draw arc with |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the arc |

DrawCircle

Draws a circle, cenetered at the location provided. Can set the fill and outline colors

DrawCircle(center\_location,

radius,

fill\_color=None,

line\_color="black")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| center\_location | Union [Tuple[int, int], Tuple[float, float]] Center location using USER'S coordinate system |
| radius | Union[int, float] Radius in user's coordinate values. |
| fill\_color | (str) color of the point to draw |
| line\_color | (str) color of the outer line that goes around the circle (sorry, can't set thickness) |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the circle |

DrawImage

Places an image onto your canvas. It's a really important method for this element as it enables so much

DrawImage(filename=None,

data=None,

location=(None, None),

color="black",

font=None,

angle=0)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) if image is in a file, path and filename for the image. (GIF and PNG only!) |
| data | Union[str, bytes] if image is in Base64 format or raw? format then use instead of filename |
| location | Union[Tuple[int, int], Tuple[float, float]] the (x,y) location to place image's top left corner |
| color | (str) text color |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| angle | (float) Angle 0 to 360 to draw the text. Zero represents horizontal text |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the image |

DrawLine

Draws a line from one point to another point using USER'S coordinates. Can set the color and width of line

DrawLine(point\_from,

point\_to,

color="black",

width=1)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| point\_from | Union[Tuple[int, int], Tuple[float, float]] Starting point for line |
| point\_to | Union[Tuple[int, int], Tuple[float, float]] Ending point for line |
| color | (str) Color of the line |
| width | (int) width of line in pixels |
|  |  |
| **return** | Union[int, None] id returned from tktiner or None if user closed the window. id is used when you  want to manipulate the line |

DrawOval

Draws an oval based on coordinates in user coordinate system. Provide the location of a "bounding rectangle"

DrawOval(top\_left,

bottom\_right,

fill\_color=**None**,

line\_color=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| top\_left | Union[Tuple[int, int], Tuple[float, float]] the top left point of bounding rectangle |
| bottom\_right | Union[Tuple[int, int], Tuple[float, float]] the bottom right point of bounding rectangle |
| fill\_color | (str) color of the interrior |
| line\_color | (str) color of outline of oval |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the oval |

DrawPoint

Draws a "dot" at the point you specify using the USER'S coordinate system

DrawPoint(point,

size=2,

color="black")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| point | Union [Tuple[int, int], Tuple[float, float]] Center location using USER'S coordinate system |
| size | Union[int, float] Radius? (Or is it the diameter?) in user's coordinate values. |
| color | (str) color of the point to draw |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the point |

DrawRectangle

Draw a rectangle given 2 points. Can control the line and fill colors

DrawRectangle(top\_left,

bottom\_right,

fill\_color=**None**,

line\_color=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| top\_left | Union[Tuple[int, int], Tuple[float, float]] the top left point of rectangle |
| bottom\_right | Union[Tuple[int, int], Tuple[float, float]] the bottom right point of rectangle |
| fill\_color | (str) color of the interior |
| line\_color | (str) color of outline |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the rectangle |

DrawText

Draw some text on your graph. This is how you label graph number lines for example

DrawText(text,

location,

color="black",

font=None,

angle=0,

text\_location="center")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) text to display |
| location | Union[Tuple[int, int], Tuple[float, float]] location to place first letter |
| color | (str) text color |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| angle | (float) Angle 0 to 360 to draw the text. Zero represents horizontal text |
| text\_location | (enum) "anchor" location for the text. Values start with TEXT\_LOCATION\_ |
|  |  |
| **return** | Union[int, None] id returned from tkinter that you'll need if you want to manipulate the text |

Erase

Erase the Graph - Removes all figures previously "drawn" using the Graph methods (e.g. DrawText)

Erase()

MotionCallBack

Not a user callable method. Used to get Graph mouse motion events. Called by tkinter when mouse moved

**MotionCallBack**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (event) event info from tkinter. Contains the x and y coordinates of a mouse |

Move

Moves the entire drawing area (the canvas) by some delta from the current position. Units are indicated in your coordinate system indicated number of ticks in your coordinate system

**Move**(x\_direction, y\_direction)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| x\_direction | Union[int, float] how far to move in the "X" direction in your coordinates |
| y\_direction | Union[int, float] how far to move in the "Y" direction in your coordinates |

MoveFigure

Moves a previously drawn figure using a "delta" from current position

MoveFigure(figure,

x\_direction,

y\_direction)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| figure | (id) Previously obtained figure-id. These are returned from all Draw methods |
| x\_direction | Union[int, float] delta to apply to position in the X direction |
| y\_direction | Union[int, float] delta to apply to position in the Y direction |

RelocateFigure

Move a previously made figure to an arbitrary (x,y) location. This differs from the Move methods because it uses absolute coordinates versus relative for Move

RelocateFigure(figure,

x,

y)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| figure | (id) Previously obtained figure-id. These are returned from all Draw methods |
| x | Union[int, float] location on X axis (in user coords) to move the upper left corner of the figure |
| y | Union[int, float] location on Y axis (in user coords) to move the upper left corner of the figure |

SendFigureToBack

Changes Z-order of figures on the Graph. Sends the indicated figure to the back of all other drawn figures

**SendFigureToBack**(figure)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| figure | (int) value returned by tkinter when creating the figure / drawing |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

TKCanvas

property: TKCanvas

Update

Changes some of the settings for the Graph Element. Must call Window.Read or Window.Finalize prior

**Update**(background\_color=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| background\_color | color of background |
| visible | (bool) control visibility of element |

erase

Erase the Graph - Removes all figures previously "drawn" using the Graph methods (e.g. DrawText)

erase()

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

move

Moves the entire drawing area (the canvas) by some delta from the current position. Units are indicated in your coordinate system indicated number of ticks in your coordinate system

**move**(x\_direction, y\_direction)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| x\_direction | Union[int, float] how far to move in the "X" direction in your coordinates |
| y\_direction | Union[int, float] how far to move in the "Y" direction in your coordinates |

update

Changes some of the settings for the Graph Element. Must call Window.Read or Window.Finalize prior

**update**(background\_color=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| background\_color | color of background |
| visible | (bool) control visibility of element |

Image Element

Image Element - show an image in the window. Should be a GIF or a PNG only

Image(filename=**None**,

**data**=**None**,

background\_color=**None**,

size=(**None**, **None**),

**pad**=**None**,

key=**None**,

tooltip=**None**,

right\_click\_menu=**None**,

visible=True,

enable\_events=False,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) image filename if there is a button image. GIFs and PNGs only. |
| data | Union[bytes, str] Raw or Base64 representation of the image to put on button. Choose either filename or data |
| background\_color | color of background |
| size | Tuple[int, int] (width, height) size of image in pixels |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| enable\_events | (bool) Turns on the element specific events. For an Image element, the event is "image clicked" |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Image Element. Must call Window.Read or Window.Finalize prior

Update(filename=**None**,

**data**=**None**,

size=(**None**, **None**),

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) filename to the new image to display. |
| data | (str) Base64 encoded string |
| size | Tuple[int,int] size of a image (w,h) w=characters-wide, h=rows-high |
| visible | (bool) control visibility of element |

UpdateAnimation

Show an Animated GIF. Call the function as often as you like. The function will determine when to show the next frame and will automatically advance to the next frame at the right time. NOTE - does NOT perform a sleep call to delay

**UpdateAnimation**(source, time\_between\_frames=0)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| source | Union[str,bytes] Filename or Base64 encoded string containing Animated GIF |
| time\_between\_frames | (int) Number of milliseconds to wait between showing frames |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Image Element. Must call Window.Read or Window.Finalize prior

update(filename=**None**,

**data**=**None**,

size=(**None**, **None**),

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) filename to the new image to display. |
| data | (str) Base64 encoded string |
| size | Tuple[int,int] size of a image (w,h) w=characters-wide, h=rows-high |
| visible | (bool) control visibility of element |

InputText Element

Display a single text input field. Based on the tkinter Widget `Entry`

InputText(default\_text="",

size=(None, None),

disabled=False,

password\_char="",

justification=None,

background\_color=None,

text\_color=None,

font=None,

tooltip=None,

change\_submits=False,

enable\_events=False,

do\_not\_clear=True,

key=None,

focus=False,

pad=None,

right\_click\_menu=None,

visible=True,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| default\_text | (str) Text initially shown in the input box as a default value(Default value = '') |
| size | Tuple[int, int] (width, height) w=characters-wide, h=rows-high |
| disabled | (bool) set disable state for element (Default = False) |
| password\_char | (char) Password character if this is a password field (Default value = '') |
| justification | (str) justification for data display. Valid choices - left, right, center |
| background\_color | (str) color of background in one of the color formats |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| change\_submits | (bool) \* DEPRICATED DO NOT USE! Same as enable\_events |
| enable\_events | (bool) If True then changes to this element are immediately reported as an event. Use this instead of change\_submits (Default = False) |
| do\_not\_clear | (bool) If False then the field will be set to blank after ANY event (button, any event) (Default = True) |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| focus | (bool) Determines if initial focus should go to this element. |
| pad | (int, int) or ((int, int), (int, int)) Tuple(s). Amount of padding to put around element. Normally (horizontal pixels, vertical pixels) but can be split apart further into ((horizontal left, horizontal right), (vertical above, vertical below)) |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element (Default = True) |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Read and return the current value of the input element. Must call Window.Read or Window.Finalize prior

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) current value of Input field or '' if error encountered |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Input Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

disabled=**None**,

**select**=**None**,

visible=**None**,

text\_color=**None**,

background\_color=**None**,

move\_cursor\_to="end")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to display as default text in Input field |
| disabled | (bool) disable or enable state of the element (sets Entry Widget to readonly or normal) |
| select | (bool) if True, then the text will be selected |
| visible | (bool) change visibility of element |
| text\_color | (str) change color of text being typed |
| background\_color | (str) change color of the background |
| move\_cursor\_to | Union[int, str] Moves the cursor to a particular offset. Defaults to 'end' |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Read and return the current value of the input element. Must call Window.Read or Window.Finalize prior

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) current value of Input field or '' if error encountered |

update

Changes some of the settings for the Input Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

disabled=**None**,

**select**=**None**,

visible=**None**,

text\_color=**None**,

background\_color=**None**,

move\_cursor\_to="end")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to display as default text in Input field |
| disabled | (bool) disable or enable state of the element (sets Entry Widget to readonly or normal) |
| select | (bool) if True, then the text will be selected |
| visible | (bool) change visibility of element |
| text\_color | (str) change color of text being typed |
| background\_color | (str) change color of the background |
| move\_cursor\_to | Union[int, str] Moves the cursor to a particular offset. Defaults to 'end' |

Listbox Element

A List Box. Provide a list of values **for** the user to choose one or more of. Returns a list of selected rows

when a window.Read() is executed.

Listbox(values,

default\_values=**None**,

select\_mode=**None**,

change\_submits=**False**,

enable\_events=**False**,

bind\_return\_key=**False**,

size=(**None**, **None**),

disabled=**False**,

auto\_size\_text=**None**,

font=**None**,

no\_scrollbar=**False**,

background\_color=**None**,

text\_color=**None**,

key=**None**,

pad=**None**,

tooltip=**None**,

right\_click\_menu=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] list of values to display. Can be any type including mixed types as long as they have **str** method |
| default\_values | List[Any] which values should be initially selected |
| select\_mode | [enum] Select modes are used to determine if only 1 item can be selected or multiple and how they can be selected. Valid choices begin with "LISTBOX\_SELECT\_MODE\_" and include: LISTBOX\_SELECT\_MODE\_SINGLE LISTBOX\_SELECT\_MODE\_MULTIPLE LISTBOX\_SELECT\_MODE\_BROWSE LISTBOX\_SELECT\_MODE\_EXTENDED |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Listbox generates events when an item is clicked |
| bind\_return\_key | (bool) If True, then the return key will cause a the Listbox to generate an event |
| size | Tuple(int, int) (width, height) width = characters-wide, height = rows-high |
| disabled | (bool) set disable state for element |
| auto\_size\_text | (bool) True if element should be the same size as the contents |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

GetIndexes

Returns the items currently selected as a list of indexes

GetIndexes()

| **Name** | **Meaning** |
| --- | --- |
| **return** | List[int] A list of offsets into values that is currently selected |

GetListValues

Returns list of Values provided by the user in the user's format

GetListValues()

| **Name** | **Meaning** |
| --- | --- |
| **return** | List[Any]. List of values. Can be any / mixed types -> [] |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

SetValue

Set listbox highlighted choices

**SetValue**(values)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] new values to choose based on previously set values |

Update

Changes some of the settings for the Listbox Element. Must call Window.Read or Window.Finalize prior

**Update**(values=**None**,

disabled=**None**,

set\_to\_index=**None**,

scroll\_to\_index=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] new list of choices to be shown to user |
| disabled | (bool) disable or enable state of the element |
| set\_to\_index | Union[int, list, tuple] highlights the item(s) indicated. If parm is an int one entry will be set. If is a list, then each entry in list is highlighted |
| scroll\_to\_index | (int) scroll the listbox so that this index is the first shown |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Listbox Element. Must call Window.Read or Window.Finalize prior

update(values=**None**,

disabled=**None**,

set\_to\_index=**None**,

scroll\_to\_index=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] new list of choices to be shown to user |
| disabled | (bool) disable or enable state of the element |
| set\_to\_index | Union[int, list, tuple] highlights the item(s) indicated. If parm is an int one entry will be set. If is a list, then each entry in list is highlighted |
| scroll\_to\_index | (int) scroll the listbox so that this index is the first shown |
| visible | (bool) control visibility of element |

Menu Element

Menu Element is the Element that provides a Menu Bar that goes across the top of the window, just below titlebar.

Here is an example layout. The "&" are shortcut keys ALT+key.

Is a List of - "Item String" + List

Where Item String is what will be displayed **on** **the** **Menubar** **itself**.

The List that follows the **item** represents the **items** that are shown **then** Menu **item** is clicked

Notice how an "entry" in a mennu can be a list which means **it** branches out and shows another menu, etc. (resursive)

menu\_def = [['&File', ['!&Open', '&Save::savekey', '---', '&Properties', 'E&xit']],

['!&Edit', ['!&Paste', ['Special', 'Normal', ], 'Undo'], ],

['&Debugger', ['Popout', 'Launch Debugger']],

['&Toolbar', ['Command &1', 'Command &2', 'Command &3', 'Command &4']],

['&Help', '&About...'], ]

Finally, "keys" can be added to entries so make them unique. The "Save" entry has a key associated with **it**. You

can see **it** has a "::" which signifies the beginning of a key. The user will not see the key portion when the

menu is shown. The key portion is returned **as** part of the event.

**Menu**(menu\_definition,

background\_color=**None**,

size=(**None**, **None**),

tearoff=**False**,

pad=**None**,

key=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| menu\_definition | List[List[Tuple[str, List[str]]] |
| background\_color | (str) color of the background |
| size | Tuple[int, int] Not used in the tkinter port |
| tearoff | (bool) if True, then can tear the menu off from the window ans use as a floating window. Very cool effect |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Update a menubar - can change the menu definition and visibility. The entire menu has to be specified

**Update**(menu\_definition=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| menu\_definition | List[List[Tuple[str, List[str]]] |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Update a menubar - can change the menu definition and visibility. The entire menu has to be specified

**update**(menu\_definition=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| menu\_definition | List[List[Tuple[str, List[str]]] |
| visible | (bool) control visibility of element |

Multiline Element

Multiline Element - Display and/or read multiple **lines** of **text**. This is both an input and output **element**.

Other PySimpleGUI ports have a separate MultilineInput and MultilineOutput elements. May want to split this

one up in the future too.

Multiline(default\_text="",

enter\_submits=**False**,

disabled=**False**,

autoscroll=**False**,

border\_width=**None**,

size=(**None**, **None**),

auto\_size\_text=**None**,

background\_color=**None**,

text\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

do\_not\_clear=**True**,

key=**None**,

focus=**False**,

font=**None**,

pad=**None**,

tooltip=**None**,

right\_click\_menu=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| default\_text | (str) Initial text to show |
| enter\_submits | (bool) if True, the Window.Read call will return is enter key is pressed in this element |
| disabled | (bool) set disable state |
| autoscroll | (bool) If True the contents of the element will automatically scroll as more data added to the end |
| border\_width | (int) width of border around element in pixels |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) if True will size the element to match the length of the text |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Spin events happen when an item changes |
| do\_not\_clear | if False the element will be cleared any time the Window.Read call returns |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| focus | (bool) if True initial focus will go to this element |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Return current contents of the Multiline Element

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) current contents of the Multiline Element (used as an input type of Multiline |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Multiline Element. Must call Window.Read or Window.Finalize prior

**Update**(value=**None**,

disabled=**None**,

append=**False**,

font=**None**,

text\_color=**None**,

background\_color=**None**,

visible=**None**,

autoscroll=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to display |
| disabled | (bool) disable or enable state of the element |
| append | (bool) if True then new value will be added onto the end of the current value. if False then contents will be replaced. |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| visible | (bool) set visibility state of the element |
| autoscroll | (bool) if True then contents of element are scrolled down when new text is added to the end |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Return current contents of the Multiline Element

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) current contents of the Multiline Element (used as an input type of Multiline |

update

Changes some of the settings for the Multiline Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

disabled=**None**,

append=False,

font=**None**,

text\_color=**None**,

background\_color=**None**,

visible=**None**,

autoscroll=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to display |
| disabled | (bool) disable or enable state of the element |
| append | (bool) if True then new value will be added onto the end of the current value. if False then contents will be replaced. |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| visible | (bool) set visibility state of the element |
| autoscroll | (bool) if True then contents of element are scrolled down when new text is added to the end |

OptionMenu Element

Option Menu **is** an Element available ONLY **on** **the** tkinter port **of** PySimpleGUI. It's **is** a widget **that** **is** unique **to** tkinter. However, **it** looks much like a ComboBox. Instead **of** an arrow **to** click **to** pull down **the** **list** **of** choices, another little graphic **is** shown **on** **the** widget **to** indicate **where** you click. After clicking **to** activate, **it** looks like a Combo Box **that** you scroll **to** select a choice.

OptionMenu(values,

default\_value=**None**,

size=(**None**, **None**),

disabled=False,

auto\_size\_text=**None**,

background\_color=**None**,

text\_color=**None**,

key=**None**,

**pad**=**None**,

tooltip=**None**,

visible=True,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] Values to be displayed |
| default\_value | (Any) the value to choose by default |
| size | Tuple[int, int] (width, height) size in characters (wide) and rows (high) |
| disabled | (bool) control enabled / disabled |
| auto\_size\_text | (bool) True if size of Element should match the contents of the items |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text that will appear when mouse hovers over this element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the OptionMenu Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

values=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) the value to choose by default |
| values | List[Any] Values to be displayed |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the OptionMenu Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

values=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) the value to choose by default |
| values | List[Any] Values to be displayed |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

Output Element

Output Element - a multi-lined **text** area where **stdout** and **stderr** are re-routed to.

Output(size=(**None**, **None**),

background\_color=**None**,

text\_color=**None**,

**pad**=**None**,

font=**None**,

tooltip=**None**,

key=**None**,

right\_click\_menu=**None**,

visible=True,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Returns the current contents of the output. Similar to Get method other Elements

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (str) the current value of the output |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Output Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**, visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) string that will replace current contents of the output area |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Output Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**, visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) string that will replace current contents of the output area |
| visible | (bool) control visibility of element |

Pane Element

A sliding Pane that **is** unique **to** tkinter. **Uses** Columns **to** **create** individual panes

Pane(pane\_list,

background\_color=None,

size=(None, None),

pad=None,

orientation="vertical",

show\_handle=True,

relief="raised",

handle\_size=None,

border\_width=None,

key=None,

visible=True,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| pane\_list | List[Column] Must be a list of Column Elements. Each Column supplied becomes one pane that's shown |
| background\_color | (str) color of background |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high How much room to reserve for the Pane |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| orientation | (str) 'horizontal' or 'vertical' or ('h' or 'v'). Direction the Pane should slide |
| show\_handle | (bool) if True, the handle is drawn that makes it easier to grab and slide |
| relief | (enum) relief style. Values are same as other elements that use relief values. RELIEF\_RAISED RELIEF\_SUNKEN RELIEF\_FLAT RELIEF\_RIDGE RELIEF\_GROOVE RELIEF\_SOLID |
| handle\_size | (int) Size of the handle in pixels |
| border\_width | (int) width of border around element in pixels |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Pane Element. Must call Window.Read or Window.Finalize prior

**Update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Pane Element. Must call Window.Read or Window.Finalize prior

**update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

ProgressBar Element

Progress Bar Element - Displays a colored bar **that** **is** shaded **as** progress **of** **some** operation **is** made

ProgressBar(max\_value,

orientation=**None**,

size=(**None**, **None**),

auto\_size\_text=**None**,

bar\_color=(**None**, **None**),

style=**None**,

border\_width=**None**,

relief=**None**,

key=**None**,

**pad**=**None**,

visible=True,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| max\_value | (int) max value of progressbar |
| orientation | (str) 'horizontal' or 'vertical' |
| size | Tuple[int, int] Size of the bar. If horizontal (chars wide, pixels high), vert (pixels wide, rows high) |
| auto\_size\_text | (bool) Not sure why this is here |
| bar\_color | Tuple[str, str] The 2 colors that make up a progress bar. One is the background, the other is the bar |
| style | (str) Progress bar style defined as one of these 'default', 'winnative', 'clam', 'alt', 'classic', 'vista', 'xpnative' |
| border\_width | (int) The amount of pixels that go around the outside of the bar |
| relief | (str) relief style. Values are same as progress meter relief values. Can be a constant or a string: RELIEF\_RAISED RELIEF\_SUNKEN RELIEF\_FLAT RELIEF\_RIDGE RELIEF\_GROOVE RELIEF\_SOLID (Default value = DEFAULT\_PROGRESS\_BAR\_RELIEF) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the ProgressBar Element. Must call Window.Read or Window.Finalize prior

**Update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

UpdateBar

Change what the bar shows by changing the current count and optionally the max count

**UpdateBar**(current\_count, max=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| current\_count | (int) sets the current value |
| max | (int) changes the max value |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the ProgressBar Element. Must call Window.Read or Window.Finalize prior

**update**(visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| visible | (bool) control visibility of element |

Radio Element

Radio Button Element - Used **in** a **group** **of** other Radio Elements **to** provide user **with** ability **to** **select** only

1 choice **in** a list **of** choices.

Radio(text,

group\_id,

default=**False**,

disabled=**False**,

size=(**None**, **None**),

auto\_size\_text=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

key=**None**,

pad=**None**,

tooltip=**None**,

change\_submits=**False**,

enable\_events=**False**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) Text to display next to button |
| group\_id | (Any) Groups together multiple Radio Buttons. Any type works |
| default | (bool). Set to True for the one element of the group you want initially selected |
| disabled | (bool) set disable state |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) if True will size the element to match the length of the text |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Radio Button events happen when an item is selected |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

A snapshot of the value of Radio Button -> (bool)

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (bool) True if this radio button is selected |

ResetGroup

Sets all Radio Buttons in the group to not selected

ResetGroup()

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Radio Button Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (bool) if True change to selected and set others in group to unselected |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

A snapshot of the value of Radio Button -> (bool)

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (bool) True if this radio button is selected |

update

Changes some of the settings for the Radio Button Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (bool) if True change to selected and set others in group to unselected |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

Slider Element

A slider, horizontal or **vertical**

**Slider**(range=(**None**, **None**),

default\_value=**None**,

resolution=**None**,

tick\_interval=**None**,

orientation=**None**,

disable\_number\_display=**False**,

border\_width=**None**,

relief=**None**,

change\_submits=**False**,

enable\_events=**False**,

disabled=**False**,

size=(**None**, **None**),

font=**None**,

background\_color=**None**,

text\_color=**None**,

key=**None**,

pad=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| range | Union[Tuple[int, int], Tuple[float, float]] slider's range (min value, max value) |
| default\_value | Union[int, float] starting value for the slider |
| resolution | Union[int, float] the smallest amount the slider can be moved |
| tick\_interval | Union[int, float] how often a visible tick should be shown next to slider |
| orientation | (str) 'horizontal' or 'vertical' ('h' or 'v' also work) |
| disable\_number\_display | (bool) if True no number will be displayed by the Slider Element |
| border\_width | (int) width of border around element in pixels |
| relief | (enum) relief style. RELIEF\_RAISED RELIEF\_SUNKEN RELIEF\_FLAT RELIEF\_RIDGE RELIEF\_GROOVE RELIEF\_SOLID |
| change\_submits | (bool) \* DEPRICATED DO NOT USE! Same as enable\_events |
| enable\_events | (bool) If True then moving the slider will generate an Event |
| disabled | (bool) set disable state for element |
| size | Tuple[int, int] (width in characters, height in rows) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| background\_color | (str) color of slider's background |
| text\_color | (str) color of the slider's text |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Slider Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

range=(**None**, **None**),

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | Union[int, float] sets current slider value |
| range | Union[Tuple[int, int], Tuple[float, float] Sets a new range for slider |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Slider Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

range=(**None**, **None**),

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | Union[int, float] sets current slider value |
| range | Union[Tuple[int, int], Tuple[float, float] Sets a new range for slider |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

Spin Element

A spinner with up/down buttons and a single line of **text**. Choose 1 values from list

Spin(values,

initial\_value=**None**,

disabled=**False**,

change\_submits=**False**,

enable\_events=**False**,

size=(**None**, **None**),

auto\_size\_text=**None**,

font=**None**,

background\_color=**None**,

text\_color=**None**,

key=**None**,

pad=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[Any] List of valid values |
| initial\_value | (Any) Initial item to show in window. Choose from list of values supplied |
| disabled | (bool) set disable state |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Spin events happen when an item changes |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) if True will size the element to match the length of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Return the current chosen value showing in spinbox. This value will be the same as what was provided as list of choices. If list items are ints, then the item returned will be an int (not a string)

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Any) The currently visible entry |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Spin Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

values=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) set the current value from list of choices |
| values | List[Any] set available choices |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Return the current chosen value showing in spinbox. This value will be the same as what was provided as list of choices. If list items are ints, then the item returned will be an int (not a string)

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Any) The currently visible entry |

update

Changes some of the settings for the Spin Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

values=**None**,

disabled=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (Any) set the current value from list of choices |
| values | List[Any] set available choices |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

StatusBar Element

A StatusBar Element creates the sunken **text**-filled strip **at** the bottom. Many Windows programs have this line

StatusBar(text,

size=(**None**, **None**),

auto\_size\_text=**None**,

click\_submits=**None**,

enable\_events=**False**,

relief="sunken",

font=**None**,

text\_color=**None**,

background\_color=**None**,

justification=**None**,

pad=**None**,

key=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) Text that is to be displayed in the widget |
| size | Tuple[(int), (int)] (w,h) w=characters-wide, h=rows-high |
| auto\_size\_text | (bool) True if size should fit the text length |
| click\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. StatusBar events occur when the bar is clicked |
| relief | (enum) relief style. Values are same as progress meter relief values. Can be a constant or a string: RELIEF\_RAISED RELIEF\_SUNKEN RELIEF\_FLAT RELIEF\_RIDGE RELIEF\_GROOVE RELIEF\_SOLID |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| justification | (str) how string should be aligned within space provided by size. Valid choices = left, right, center |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Status Bar Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to show |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) set visibility state of the element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Status Bar Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to show |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) set visibility state of the element |

Tab Element

Tab Element is another "Container" **element** that holds a layout and displays a tab with **text**. Used with TabGroup only

Tabs are never placed directly **into** a layout. They are always "Contained" in a TabGroup layout

Tab(title,

layout,

title\_color=None,

background\_color=None,

font=None,

pad=None,

disabled=False,

border\_width=None,

key=None,

tooltip=None,

right\_click\_menu=None,

visible=True,

element\_justification="left",

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| title | (str) text to show on the tab |
| layout | List[List[Element]] The element layout that will be shown in the tab |
| title\_color | (str) color of the tab text (note not currently working on tkinter) |
| background\_color | (str) color of background of the entire layout |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| disabled | (bool) If True button will be created disabled |
| border\_width | (int) width of border around element in pixels |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| element\_justification | (str) All elements inside the Tab will have this justification 'left', 'right', 'center' are valid values |
| metadata | (Any) User metadata that can be set to ANYTHING |

AddRow

Not recommended use call. Used to add rows of Elements to the Frame Element.

**AddRow**(args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | List[Element] The list of elements for this row |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Layout

Not user callable. Use layout parameter instead. Creates the layout using the supplied rows of Elements

**Layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The list of rows |
|  |  |
| **return** | (Tab) used for chaining |

Select

Create a tkinter event that mimics user clicking on a tab. Must have called window.Finalize / Read first!

Select()

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Tab Element. Must call Window.Read or Window.Finalize prior

**Update**(disabled=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

layout

Not user callable. Use layout parameter instead. Creates the layout using the supplied rows of Elements

**layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The list of rows |
|  |  |
| **return** | (Tab) used for chaining |

select

Create a tkinter event that mimics user clicking on a tab. Must have called window.Finalize / Read first!

select()

update

Changes some of the settings for the Tab Element. Must call Window.Read or Window.Finalize prior

**update**(disabled=None, visible=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| disabled | (bool) disable or enable state of the element |
| visible | (bool) control visibility of element |

TabGroup Element

TabGroup Element groups together your tabs **into** the **group** **of** tabs you see displayed **in** your window

TabGroup(layout,

tab\_location=**None**,

title\_color=**None**,

selected\_title\_color=**None**,

background\_color=**None**,

font=**None**,

change\_submits=**False**,

enable\_events=**False**,

pad=**None**,

border\_width=**None**,

theme=**None**,

key=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| layout | List[List[Tab]] Layout of Tabs. Different than normal layouts. ALL Tabs should be on first row |
| tab\_location | (str) location that tabs will be displayed. Choices are left, right, top, bottom, lefttop, leftbottom, righttop, rightbottom, bottomleft, bottomright, topleft, topright |
| title\_color | (str) color of text on tabs |
| selected\_title\_color | (str) color of tab when it is selected |
| background\_color | (str) color of background of tabs |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| change\_submits | (bool) \* DEPRICATED DO NOT USE! Same as enable\_events |
| enable\_events | (bool) If True then switching tabs will generate an Event |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| border\_width | (int) width of border around element in pixels |
| theme | (enum) tabs can be 'themed'. These are the choices (some may not work on your OS): THEME\_DEFAULT THEME\_WINNATIVE THEME\_CLAM THEME\_ALT THEME\_CLASSIC THEME\_VISTA THEME\_XPNATIVE |
| key | (any) Value that uniquely identifies this element from all other elements. Used when Finding an element or in return values. Must be unique to the window |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

AddRow

Not recommended user call. Used to add rows of Elements to the Frame Element.

**AddRow**(args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | List[Element] The list of elements for this row |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

FindKeyFromTabName

Searches through the layout to find the key that matches the text on the tab. Implies names should be unique

**FindKeyFromTabName**(tab\_name)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tab\_name |  |
|  |  |
| **return** | Union[key, None] Returns the key or None if no key found |

Get

Returns the current value for the Tab Group, which will be the currently selected tab's KEY or the text on the tab if no key is defined. Returns None if an error occurs. Note that this is exactly the same data that would be returned from a call to Window.Read. Are you sure you are using this method correctly?

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Any, None] The key of the currently selected tab or the tab's text if it has no key |

Layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**Layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Frame) Used for chaining |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Returns the current value for the Tab Group, which will be the currently selected tab's KEY or the text on the tab if no key is defined. Returns None if an error occurs. Note that this is exactly the same data that would be returned from a call to Window.Read. Are you sure you are using this method correctly?

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Any, None] The key of the currently selected tab or the tab's text if it has no key |

layout

Can use like the Window.Layout method, but it's better to use the layout parameter when creating

**layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Element]] The rows of Elements |
|  |  |
| **return** | (Frame) Used for chaining |

Table Element

Table(values,

headings=None,

visible\_column\_map=None,

col\_widths=None,

def\_col\_width=10,

auto\_size\_columns=True,

max\_col\_width=20,

select\_mode=None,

display\_row\_numbers=False,

num\_rows=None,

row\_height=None,

font=None,

justification="right",

text\_color=None,

background\_color=None,

alternating\_row\_color=None,

row\_colors=None,

vertical\_scroll\_only=True,

hide\_vertical\_scroll=False,

size=(None, None),

change\_submits=False,

enable\_events=False,

bind\_return\_key=False,

pad=None,

key=None,

tooltip=None,

right\_click\_menu=None,

visible=True,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[List[Union[str, int, float]]] |
| headings | List[str] The headings to show on the top line |
| visible\_column\_map | List[bool] One entry for each column. False indicates the column is not shown |
| col\_widths | List[int] Number of characters that each column will occupy |
| def\_col\_width | (int) Default column width in characters |
| auto\_size\_columns | (bool) if True columns will be sized automatically |
| max\_col\_width | (int) Maximum width for all columns in characters |
| select\_mode | (enum) Select Mode. Valid values start with "TABLE\_SELECT\_MODE\_". Valid values are: TABLE\_SELECT\_MODE\_NONE TABLE\_SELECT\_MODE\_BROWSE TABLE\_SELECT\_MODE\_EXTENDED |
| display\_row\_numbers | (bool) if True, the first column of the table will be the row # |
| num\_rows | (int) The number of rows of the table to display at a time |
| row\_height | (int) height of a single row in pixels |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| justification | (str) 'left', 'right', 'center' are valid choices |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| alternating\_row\_color | (str) if set then every other row will have this color in the background. |
| row\_colors |  |
| vertical\_scroll\_only | (bool) if True only the vertical scrollbar will be visible |
| hide\_vertical\_scroll | (bool) if True vertical scrollbar will be hidden |
| size | Tuple[int, int] DO NOT USE! Use num\_rows instead |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Table events happen when row is clicked |
| bind\_return\_key | (bool) if True, pressing return key will cause event coming from Table, ALSO a left button double click will generate an event if this parameter is True |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

Get

Dummy function for tkinter port. In the Qt port you can read back the values in the table in case they were edited. Don't know yet how to enable editing of a Tree in tkinter so just returning the values provided by user when Table was created or Updated.

Get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | List[List[Any]] the current table values (for now what was originally provided up updated) |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Table Element. Must call Window.Read or Window.Finalize prior

**Update**(values=**None**,

num\_rows=**None**,

visible=**None**,

select\_rows=**None**,

alternating\_row\_color=**None**,

row\_colors=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[List[Union[str, int, float]]] A new 2-dimensional table to show |
| num\_rows | (int) How many rows to display at a time |
| visible | (bool) if True then will be visible |
| select\_rows | List[int] List of rows to select as if user did |
| alternating\_row\_color | (str) the color to make every other row |
| row\_colors | List[Union[Tuple[int, str], Tuple[Int, str, str]] list of tuples of (row, background color) OR (row, foreground color, background color). Changes the colors of listed rows to the color(s) provided (note the optional foreground color) |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

get

Dummy function for tkinter port. In the Qt port you can read back the values in the table in case they were edited. Don't know yet how to enable editing of a Tree in tkinter so just returning the values provided by user when Table was created or Updated.

get()

| **Name** | **Meaning** |
| --- | --- |
| **return** | List[List[Any]] the current table values (for now what was originally provided up updated) |

update

Changes some of the settings for the Table Element. Must call Window.Read or Window.Finalize prior

update(values=**None**,

num\_rows=**None**,

visible=**None**,

select\_rows=**None**,

alternating\_row\_color=**None**,

row\_colors=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | List[List[Union[str, int, float]]] A new 2-dimensional table to show |
| num\_rows | (int) How many rows to display at a time |
| visible | (bool) if True then will be visible |
| select\_rows | List[int] List of rows to select as if user did |
| alternating\_row\_color | (str) the color to make every other row |
| row\_colors | List[Union[Tuple[int, str], Tuple[Int, str, str]] list of tuples of (row, background color) OR (row, foreground color, background color). Changes the colors of listed rows to the color(s) provided (note the optional foreground color) |

Text Element

Text - Display some **text** in the window. Usually this means a single line of **text**. However, the **text** can also be multiple **lines**. If multi-lined there are no scroll bars.

**Text**(text="",

size=(**None**, **None**),

auto\_size\_text=**None**,

click\_submits=**False**,

enable\_events=**False**,

relief=**None**,

font=**None**,

text\_color=**None**,

background\_color=**None**,

border\_width=**None**,

justification=**None**,

pad=**None**,

key=**None**,

right\_click\_menu=**None**,

tooltip=**None**,

visible=**True**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| text | (str) The text to display. Can include /n to achieve multiple lines |
| size | Tuple[int, int] (width, height) width = characters-wide, height = rows-high |
| auto\_size\_text | (bool) if True size of the Text Element will be sized to fit the string provided in 'text' parm |
| click\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Text events happen when the text is clicked |
| relief | (str/enum) relief style around the text. Values are same as progress meter relief values. Should be a constant that is defined at starting with "RELIEF\_" - RELIEF\_RAISED, RELIEF\_SUNKEN, RELIEF\_FLAT, RELIEF\_RIDGE, RELIEF\_GROOVE, RELIEF\_SOLID |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| border\_width | (int) number of pixels for the border (if using a relief) |
| justification | (str) how string should be aligned within space provided by size. Valid choices = left, right, center |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Text Element. Must call Window.Read or Window.Finalize prior

Update(**value**=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to show |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) set visibility state of the element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Text Element. Must call Window.Read or Window.Finalize prior

update(**value**=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| value | (str) new text to show |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| visible | (bool) set visibility state of the element |

Tree Element

Tree Element - Presents data **in** a tree-**like** manner, much **like** a file/folder browser. Uses the TreeData **class to** hold the user*'s data and pass to the element for display.*

Tree(data=None,

headings=None,

visible\_column\_map=None,

col\_widths=None,

col0\_width=10,

def\_col\_width=10,

auto\_size\_columns=True,

max\_col\_width=20,

select\_mode=None,

show\_expanded=False,

change\_submits=False,

enable\_events=False,

font=None,

justification="right",

text\_color=None,

background\_color=None,

num\_rows=None,

row\_height=None,

pad=None,

key=None,

tooltip=None,

right\_click\_menu=None,

visible=True,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| data | (TreeData) The data represented using a PySimpleGUI provided TreeData class |
| headings | List[str] List of individual headings for each column |
| visible\_column\_map | List[bool] Determines if a column should be visible. If left empty, all columns will be shown |
| col\_widths | List[int] List of column widths so that individual column widths can be controlled |
| col0\_width | (int) Size of Column 0 which is where the row numbers will be optionally shown |
| def\_col\_width | (int) default column width |
| auto\_size\_columns | (bool) if True, the size of a column is determined using the contents of the column |
| max\_col\_width | (int) the maximum size a column can be |
| select\_mode | (enum) Use same values as found on Table Element. Valid values include: TABLE\_SELECT\_MODE\_NONE TABLE\_SELECT\_MODE\_BROWSE TABLE\_SELECT\_MODE\_EXTENDED |
| show\_expanded | (bool) if True then the tree will be initially shown with all nodes completely expanded |
| change\_submits | (bool) DO NOT USE. Only listed for backwards compat - Use enable\_events instead |
| enable\_events | (bool) Turns on the element specific events. Tree events happen when row is clicked |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| justification | (str) 'left', 'right', 'center' are valid choices |
| text\_color | (str) color of the text |
| background\_color | (str) color of background |
| num\_rows | (int) The number of rows of the table to display at a time |
| row\_height | (int) height of a single row in pixels |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element to uniquely identify this element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| visible | (bool) set visibility state of the element |
| metadata | (Any) User metadata that can be set to ANYTHING |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

Update

Changes some of the settings for the Tree Element. Must call Window.Read or Window.Finalize prior

Update(values=**None**,

key=**None**,

**value**=**None**,

text=**None**,

icon=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | (TreeData) Representation of the tree |
| key | (Any) identifies a particular item in tree to update |
| value | (Any) sets the node identified by key to a particular value |
| text | (str) sets the node identified by ket to this string |
| icon | Union[bytes, str] can be either a base64 icon or a filename for the icon |
| visible | (bool) control visibility of element |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

update

Changes some of the settings for the Tree Element. Must call Window.Read or Window.Finalize prior

update(values=**None**,

key=**None**,

**value**=**None**,

text=**None**,

icon=**None**,

visible=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values | (TreeData) Representation of the tree |
| key | (Any) identifies a particular item in tree to update |
| value | (Any) sets the node identified by key to a particular value |
| text | (str) sets the node identified by ket to this string |
| icon | Union[bytes, str] can be either a base64 icon or a filename for the icon |
| visible | (bool) control visibility of element |

TreeData Element

**Class** that user fills **in** **to** represent their tree data. It*'s a very simple tree representation with a root "Node"* **with** possibly one **or** more children "Nodes". **Each** Node contains a **key**, **text** **to** display, list **of** values **to** display **and** an icon. The entire tree **is** built **using** a single method, Insert. Nothing **else** **is** required **to** make the tree.

Instantiate the object, initializes the Tree Data, creates a root node for you

TreeData()

Insert

Inserts a node into the tree. This is how user builds their tree, by Inserting Nodes This is the ONLY user callable method in the TreeData class

**Insert**(parent,

**key**,

text,

**values**,

icon=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| parent | (Node) the parent Node |
| key | (Any) Used to uniquely identify this node |
| text | (str) The text that is displayed at this node's location |
| values | List[Any] The list of values that are displayed at this node |
| icon | Union[str, bytes] |

VerticalSeparator Element

Vertical Separator Element draws a vertical line at the given location. It will span 1 "row". Usually paired with Column Element **if** extra height is needed

**VerticalSeparator**(pad=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| pad | (int, int) or ((int, int),(int,int)) Amount of padding to put around element (left/right, top/bottom) or ((left, right), (top, bottom)) |

ButtonReboundCallback

Used in combination with tkinter's widget.bind function. If you wish to have a double-click for a button to call back the button's normal callback routine, then you should target your call to tkinter's bind method to point to this function which will in turn call the button callback function that is normally called.

**ButtonReboundCallback**(event)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| event | (unknown) Not used in this function. |

SetFocus

Sets the current focus to be on this element

**SetFocus**(force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| force | (bool) if True will call focus\_force otherwise calls focus\_set |

SetTooltip

Called by application to change the tooltip text for an Element. Normally invoked using the Element Object such as: window.Element('key').SetToolTip('New tip').

**SetTooltip**(tooltip\_text)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| tooltip\_text | (str) the text to show in tooltip. |

expand

Causes the Element to expand to fill available space in the X and Y directions. Can specify which or both directions

**expand**(expand\_x=False, expand\_y=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| expand\_x | (Bool) If True Element will expand in the Horizontal directions |
| expand\_y | (Bool) If True Element will expand in the Vertical directions |

Window

Represents a single Window

Window(title,

layout=**None**,

default\_element\_size=(45, 1),

default\_button\_element\_size=(**None**, **None**),

auto\_size\_text=**None**,

auto\_size\_buttons=**None**,

location=(**None**, **None**),

size=(**None**, **None**),

element\_padding=**None**,

margins=(**None**, **None**),

button\_color=**None**,

font=**None**,

progress\_bar\_color=(**None**, **None**),

background\_color=**None**,

border\_depth=**None**,

auto\_close=**False**,

auto\_close\_duration=3,

icon=**None**,

force\_toplevel=**False**,

alpha\_channel=1,

return\_keyboard\_events=**False**,

use\_default\_focus=**True**,

text\_justification=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

resizable=**False**,

disable\_close=**False**,

disable\_minimize=**False**,

right\_click\_menu=**None**,

transparent\_color=**None**,

debugger\_enabled=**True**,

finalize=**False**,

element\_justification="left",

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| title | (str) The title that will be displayed in the Titlebar and on the Taskbar |
| layout | List[List[Elements]] The layout for the window. Can also be specified in the Layout method |
| default\_element\_size | Tuple[int, int] (width, height) size in characters (wide) and rows (high) for all elements in this window |
| default\_button\_element\_size | Tuple[int, int] (width, height) size in characters (wide) and rows (high) for all Button elements in this window |
| auto\_size\_text | (bool) True if Elements in Window should be sized to exactly fir the length of text |
| auto\_size\_buttons | (bool) True if Buttons in this Window should be sized to exactly fit the text on this. |
| location | Tuple[int, int] (x,y) location, in pixels, to locate the upper left corner of the window on the screen. Default is to center on screen. |
| size | Tuple[int, int] (width, height) size in pixels for this window. Normally the window is autosized to fit contents, not set to an absolute size by the user |
| element\_padding | Tuple[int, int] or ((int, int),(int,int)) Default amount of padding to put around elements in window (left/right, top/bottom) or ((left, right), (top, bottom)) |
| margins | Tuple[int, int] (left/right, top/bottom) Amount of pixels to leave inside the window's frame around the edges before your elements are shown. |
| button\_color | Tuple[str, str] (text color, button color) Default button colors for all buttons in the window |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| progress\_bar\_color | Tuple[str, str] (bar color, background color) Sets the default colors for all progress bars in the window |
| background\_color | (str) color of background |
| border\_depth | (int) Default border depth (width) for all elements in the window |
| auto\_close | (bool) If True, the window will automatically close itself |
| auto\_close\_duration | (int) Number of seconds to wait before closing the window |
| icon | Union[str, str] Can be either a filename or Base64 value. |
| force\_toplevel | (bool) If True will cause this window to skip the normal use of a hidden master window |
| alpha\_channel | (float) Sets the opacity of the window. 0 = invisible 1 = completely visible. Values bewteen 0 & 1 will produce semi-transparent windows in SOME environments (The Raspberry Pi always has this value at 1 and cannot change. |
| return\_keyboard\_events | (bool) if True key presses on the keyboard will be returned as Events from Read calls |
| use\_default\_focus | (bool) If True will use the default focus algorithm to set the focus to the "Correct" element |
| text\_justification | (str) Union ['left', 'right', 'center'] Default text justification for all Text Elements in window |
| no\_titlebar | (bool) If true, no titlebar nor frame will be shown on window. This means you cannot minimize the window and it will not show up on the taskbar |
| grab\_anywhere | (bool) If True can use mouse to click and drag to move the window. Almost every location of the window will work except input fields on some systems |
| keep\_on\_top | (bool) If True, window will be created on top of all other windows on screen. It can be bumped down if another window created with this parm |
| resizable | (bool) If True, allows the user to resize the window. Note the not all Elements will change size or location when resizing. |
| disable\_close | (bool) If True, the X button in the top right corner of the window will no work. Use with caution and always give a way out toyour users |
| disable\_minimize | (bool) if True the user won't be able to minimize window. Good for taking over entire screen and staying that way. |
| right\_click\_menu | List[List[Union[List[str],str]]] A list of lists of Menu items to show when this element is right clicked. See user docs for exact format. |
| transparent\_color | (str) Any portion of the window that has this color will be completely transparent. You can even click through these spots to the window under this window. |
| debugger\_enabled | (bool) If True then the internal debugger will be enabled |
| finalize | (bool) If True then the Finalize method will be called. Use this rather than chaining .Finalize for cleaner code |
| element\_justification | (str) All elements in the Window itself will have this justification 'left', 'right', 'center' are valid values |
| metadata | (Any) User metadata that can be set to ANYTHING |

AddRow

Adds a single row of elements to a window's self.Rows variables. Generally speaking this is NOT how users should be building Window layouts. Users, create a single layout (a list of lists) and pass as a parameter to Window object, or call Window.Layout(layout)

**AddRow**(args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | List[Elements] |

AddRows

Loops through a list of lists of elements and adds each row, list, to the layout. This is NOT the best way to go about creating a window. Sending the entire layout at one time and passing it as a parameter to the Window call is better.

**AddRows**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Elements]] A list of a list of elements |

AlphaChannel

property: AlphaChannel

A property that changes the current alpha channel value (internal value)

| **Name** | **Meaning** |
| --- | --- |
| **return** | (float) the current alpha channel setting according to self, not read directly from tkinter |

BringToFront

Brings this window to the top of all other windows (perhaps may not be brought before a window made to "stay on top")

BringToFront()

Close

Closes window. Users can safely call even if window has been destroyed. Should always call when done with a window so that resources are properly freed up within your thread.

Close()

CurrentLocation

Get the current location of the window's top left corner

CurrentLocation()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Tuple[(int), (int)] The x and y location in tuple form (x,y) |

Disable

Disables window from taking any input from the user

Disable()

DisableDebugger

Disable the internal debugger. By default the debugger is ENABLED

DisableDebugger()

Disappear

Causes a window to "disappear" from the screen, but remain on the taskbar. It does this by turning the alpha channel to 0. NOTE that on some platforms alpha is not supported. The window will remain showing on these platforms. The Raspberry Pi for example does not have an alpha setting

Disappear()

Elem

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**Elem**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

Element

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**Element**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

Enable

Re-enables window to take user input after having it be Disabled previously

Enable()

EnableDebugger

Enables the internal debugger. By default, the debugger IS enabled

EnableDebugger()

Fill

Fill in elements that are input fields with data based on a 'values dictionary'

**Fill**(values\_dict)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values\_dict | (Dict[Any:Any]) {Element key : value} pairs |
|  |  |
| **return** | (Window) returns self so can be chained with other methods |

Finalize

Use this method to cause your layout to built into a real tkinter window. In reality this method is like Read(timeout=0). It doesn't block and uses your layout to create tkinter widgets to represent the elements. Lots of action!

Finalize()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Window) Returns 'self' so that method "Chaining" can happen (read up about it as it's very cool!) |

Find

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**Find**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

FindElement

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**FindElement**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

FindElementWithFocus

Returns the Element that currently has focus as reported by tkinter. If no element is found None is returned!

FindElementWithFocus()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Element, None] An Element if one has been found with focus or None if no element found |

GetScreenDimensions

Get the screen dimensions. NOTE - you must have a window already open for this to work (blame tkinter not me)

GetScreenDimensions()

| **Name** | **Meaning** |
| --- | --- |
| **return** | Union[Tuple[None, None], Tuple[width, height]] Tuple containing width and height of screen in pixels |

GrabAnyWhereOff

Turns off Grab Anywhere functionality AFTER a window has been created. Don't try on a window that's not yet been Finalized or Read.

GrabAnyWhereOff()

GrabAnyWhereOn

Turns on Grab Anywhere functionality AFTER a window has been created. Don't try on a window that's not yet been Finalized or Read.

GrabAnyWhereOn()

Hide

Hides the window from the screen and the task bar

Hide()

Layout

Second of two preferred ways of telling a Window what its layout is. The other way is to pass the layout as a parameter to Window object. The parameter method is the currently preferred method. This call to Layout has been removed from examples contained in documents and in the Demo Programs. Trying to remove this call from history and replace with sending as a parameter to Window.

**Layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Elements]] Your entire layout |
|  |  |
| **return** | (Window} self so that you can chain method calls |

LoadFromDisk

Restore values from a previous call to SaveToDisk which saves the returned values dictionary in Pickle format

**LoadFromDisk**(filename)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) Pickle Filename to load |

Maximize

Maximize the window. This is done differently on a windows system versus a linux or mac one. For non-Windows the root attribute '-fullscreen' is set to True. For Windows the "root" state is changed to "zoomed" The reason for the difference is the title bar is removed in some cases when using fullscreen option

Maximize()

Minimize

Minimize this window to the task bar

Minimize()

Move

Move the upper left corner of this window to the x,y coordinates provided

**Move**(x, y)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| x | (int) x coordinate in pixels |
| y | (int) y coordinate in pixels |

Normal

Restore a window to a non-maximized state. Does different things depending on platform. See Maximize for more.

Normal()

Read

THE biggest deal method in the Window class! This is how you get all of your data from your Window. Pass in a timeout (in milliseconds) to wait for a maximum of timeout milliseconds. Will return timeout\_key if no other GUI events happen first.

**Read**(timeout=None, timeout\_key="\_\_TIMEOUT\_\_")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| timeout | (int) Milliseconds to wait until the Read will return IF no other GUI events happen first |
| timeout\_key | (Any) The value that will be returned from the call if the timer expired |
|  |  |
| **return** | Tuple[(Any), Union[Dict[Any:Any]], List[Any], None] (event, values)  (event or timeout\_key or None, Dictionary of values or List of values from all elements in the Window) |

Reappear

Causes a window previously made to "Disappear" (using that method). Does this by restoring the alpha channel

Reappear()

Refresh

Refreshes the window by calling tkroot.update(). Can sometimes get away with a refresh instead of a Read. Use this call when you want something to appear in your Window immediately (as soon as this function is called). Without this call your changes to a Window will not be visible to the user until the next Read call

Refresh()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Window) self so that method calls can be easily "chained" |

SaveToDisk

Saves the values contained in each of the input areas of the form. Basically saves what would be returned from a call to Read. It takes these results and saves them to disk using pickle

**SaveToDisk**(filename)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| filename | (str) Filename to save the values to in pickled form |

SetAlpha

Sets the Alpha Channel for a window. Values are between 0 and 1 where 0 is completely transparent

**SetAlpha**(alpha)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| alpha | (float) 0 to 1. 0 is completely transparent. 1 is completely visible and solid (can't see through) |

SetIcon

Sets the icon that is shown on the title bar and on the task bar. Can pass in: \* a filename which must be a .ICO icon file for windows \* a bytes object \* a BASE64 encoded file held in a variable

**SetIcon**(icon=None, pngbase64=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| icon | (str) Filename or bytes object |
| pngbase64 | (str) Base64 encoded GIF or PNG file |

SetTransparentColor

Set the color that will be transparent in your window. Areas with this color will be SEE THROUGH.

**SetTransparentColor**(color)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| color | (str) Color string that defines the transparent color |

Size

Note the Window.Size can be used for both reading and writing

property: Size

Return the current size of the window in pixels

| **Name** | **Meaning** |
| --- | --- |
| **return** | Tuple[(int), (int)] the (width, height) of the window |

UnHide

Used to bring back a window that was previously hidden using the Hide method

UnHide()

VisibilityChanged

This is a completely dummy method that does nothing. It is here so that PySimpleGUIQt programs can make this call and then have that same source run on plain PySimpleGUI.

VisibilityChanged()

| **Name** | **Meaning** |
| --- | --- |
| **return** |  |

close

Closes window. Users can safely call even if window has been destroyed. Should always call when done with a window so that resources are properly freed up within your thread.

close()

disable

Disables window from taking any input from the user

disable()

disappear

Causes a window to "disappear" from the screen, but remain on the taskbar. It does this by turning the alpha channel to 0. NOTE that on some platforms alpha is not supported. The window will remain showing on these platforms. The Raspberry Pi for example does not have an alpha setting

disappear()

elem

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**elem**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

element

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**element**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

enable

Re-enables window to take user input after having it be Disabled previously

enable()

fill

Fill in elements that are input fields with data based on a 'values dictionary'

**fill**(values\_dict)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| values\_dict | (Dict[Any:Any]) {Element key : value} pairs |
|  |  |
| **return** | (Window) returns self so can be chained with other methods |

finalize

Use this method to cause your layout to built into a real tkinter window. In reality this method is like Read(timeout=0). It doesn't block and uses your layout to create tkinter widgets to represent the elements. Lots of action!

finalize()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Window) Returns 'self' so that method "Chaining" can happen (read up about it as it's very cool!) |

find

Find element object associated with the provided key. THIS METHOD IS NO LONGER NEEDED to be called by the user

You can perform the same operation by writing this statement: element = window[key]

You can drop the entire "FindElement" function name and use [ ] instead.

Typically used in combination with a call to element's Update method (or any other element method!): window[key].Update(new\_value)

Versus the "old way" window.FindElement(key).Update(new\_value)

This call can be abbreviated to any of these: FindElement == Element == Find Rememeber that this call will return None if no match is found which may cause your code to crash if not checked for.

**find**(key, silent\_on\_error=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | (Any) Used with window.FindElement and with return values to uniquely identify this element |
| silent\_on\_error | (bool) If True do not display popup nor print warning of key errors |
|  |  |
| **return** | Union[Element, Error Element, None] Return value can be:  \* the Element that matches the supplied key if found  \* an Error Element if silent\_on\_error is False  \* None if silent\_on\_error True |

hide

Hides the window from the screen and the task bar

hide()

layout

Second of two preferred ways of telling a Window what its layout is. The other way is to pass the layout as a parameter to Window object. The parameter method is the currently preferred method. This call to Layout has been removed from examples contained in documents and in the Demo Programs. Trying to remove this call from history and replace with sending as a parameter to Window.

**layout**(rows)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| rows | List[List[Elements]] Your entire layout |
|  |  |
| **return** | (Window} self so that you can chain method calls |

maximize

Maximize the window. This is done differently on a windows system versus a linux or mac one. For non-Windows the root attribute '-fullscreen' is set to True. For Windows the "root" state is changed to "zoomed" The reason for the difference is the title bar is removed in some cases when using fullscreen option

maximize()

minimize

Minimize this window to the task bar

minimize()

move

Move the upper left corner of this window to the x,y coordinates provided

**move**(x, y)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| x | (int) x coordinate in pixels |
| y | (int) y coordinate in pixels |

normal

Restore a window to a non-maximized state. Does different things depending on platform. See Maximize for more.

normal()

read

THE biggest deal method in the Window class! This is how you get all of your data from your Window. Pass in a timeout (in milliseconds) to wait for a maximum of timeout milliseconds. Will return timeout\_key if no other GUI events happen first.

**read**(timeout=None, timeout\_key="\_\_TIMEOUT\_\_")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| timeout | (int) Milliseconds to wait until the Read will return IF no other GUI events happen first |
| timeout\_key | (Any) The value that will be returned from the call if the timer expired |
|  |  |
| **return** | Tuple[(Any), Union[Dict[Any:Any]], List[Any], None] (event, values)  (event or timeout\_key or None, Dictionary of values or List of values from all elements in the Window) |

reappear

Causes a window previously made to "Disappear" (using that method). Does this by restoring the alpha channel

reappear()

refresh

Refreshes the window by calling tkroot.update(). Can sometimes get away with a refresh instead of a Read. Use this call when you want something to appear in your Window immediately (as soon as this function is called). Without this call your changes to a Window will not be visible to the user until the next Read call

refresh()

| **Name** | **Meaning** |
| --- | --- |
| **return** | (Window) self so that method calls can be easily "chained" |

size

property: size

Return the current size of the window in pixels

| **Name** | **Meaning** |
| --- | --- |
| **return** | Tuple[(int), (int)] the (width, height) of the window |

CButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

bind\_return\_key=**False**,

disabled=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| disabled | set disable state for element (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

CalendarButton(button\_text,

target=(None, None),

close\_when\_date\_chosen=True,

default\_date\_m\_d\_y=(None, None, None),

image\_filename=None,

image\_data=None,

image\_size=(None, None),

image\_subsample=None,

tooltip=None,

border\_width=None,

size=(None, None),

auto\_size\_button=None,

button\_color=None,

disabled=False,

font=None,

bind\_return\_key=False,

focus=False,

pad=None,

key=None,

locale=None,

format=None,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| target |  |
| close\_when\_date\_chosen | (Default = True) |
| default\_date\_m\_d\_y | (Default = (None)) |
| None |  |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | (Default = (None)) |
| image\_subsample | amount to reduce the size of the image |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| border\_width | width of border around element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
| locale |  |
| format |  |
|  |  |
| **return** | (Button) |

**Cancel**(button\_text="Cancel",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Cancel') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

Change the "color scheme" of all future PySimpleGUI Windows. The scheme are string names that specify a group of colors. Background colors, text colors, button colors. There are 13 different color settings that are changed at one time using a single call to ChangeLookAndFeel The look and feel table itself has these indexe into the dictionary LOOK\_AND\_FEEL\_TABLE SystemDefault Material1 Material2 Reddit Topanga GreenTan Dark LightGreen Dark2 Black Tan TanBlue DarkTanBlue DarkAmber DarkBlue Reds Green BluePurple Purple BlueMono GreenMono BrownBlue BrightColors NeutralBlue Kayak SandyBeach TealMono

**ChangeLookAndFeel**(index, force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| index | (str) the name of the index into the Look and Feel table |
| force | (bool) if True allows Macs to use the look and feel feature. Otherwise Macs are blocked due to problems with button colors |

CloseButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

bind\_return\_key=**False**,

disabled=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| disabled | set disable state for element (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

ColorChooserButton(button\_text,

**target**=(**None**, **None**),

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

tooltip=**None**,

border\_width=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=False,

font=**None**,

bind\_return\_key=False,

focus=False,

**pad**=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| target |  |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | (Default = (None)) |
| image\_subsample | amount to reduce the size of the image |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| border\_width | width of border around element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

**Debug**(button\_text="",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

font=**None**,

tooltip=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = '') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| font | specifies the font family, size, etc |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

DummyButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

disabled=**False**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

EasyPrint(args,

size=(**None**, **None**),

end=**None**,

sep=**None**,

location=(**None**, **None**),

font=**None**,

no\_titlebar=**False**,

no\_button=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

do\_not\_reroute\_stdout=**True**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| end |  |
| sep |  |
| location | Location on screen to display |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| no\_button | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| do\_not\_reroute\_stdout | (Default = True) |

**EasyPrintClose**()

**Exit**(button\_text="Exit",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Exit') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

FileBrowse(button\_text="Browse",

target=(555666777, -1),

file\_types=(('ALL Files', '\*.\*'),),

initial\_folder=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

font=**None**,

disabled=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Browse') |
| target | key or (row,col) target for the button (Default value = (ThisRow, -1)) |
| file\_types | (Default value = (("ALL Files", "*.*"))) |
| initial\_folder | starting path for folders and files |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| change\_submits | If True, pressing Enter key submits window (Default = False) |
| enable\_events | Turns on the element specific events.(Default = False) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| disabled | set disable state for element (Default = False) |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

FileSaveAs(button\_text="Save As...",

target=(555666777, -1),

file\_types=(('ALL Files', '\*.\*'),),

initial\_folder=**None**,

disabled=**False**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

font=**None**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Save As...') |
| target | key or (row,col) target for the button (Default value = (ThisRow, -1)) |
| file\_types | (Default value = (("ALL Files", "*.*"))) |
| initial\_folder | starting path for folders and files |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| change\_submits | If True, pressing Enter key submits window (Default = False) |
| enable\_events | Turns on the element specific events.(Default = False) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

FilesBrowse(button\_text="Browse",

target=(555666777, -1),

file\_types=(('ALL Files', '\*.\*'),),

disabled=**False**,

initial\_folder=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

font=**None**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Browse') |
| target | key or (row,col) target for the button (Default value = (ThisRow, -1)) |
| file\_types | (Default value = (("ALL Files", "*.*"))) |
| disabled | set disable state for element (Default = False) |
| initial\_folder | starting path for folders and files |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| change\_submits | If True, pressing Enter key submits window (Default = False) |
| enable\_events | Turns on the element specific events.(Default = False) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Fills a window with values provided in a values dictionary { element\_key : new\_value }

**FillFormWithValues**(window, values\_dict)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| window | (Window) The window object to fill |
| values\_dict | (Dict[Any:Any]) A dictionary with element keys as key and value is values parm for Update call |

FolderBrowse(button\_text="Browse",

target=(555666777, -1),

initial\_folder=None,

tooltip=None,

size=(None, None),

auto\_size\_button=None,

button\_color=None,

disabled=False,

change\_submits=False,

enable\_events=False,

font=None,

pad=None,

key=None,

metadata=None)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Browse') |
| target | key or (row,col) target for the button (Default value = (ThisRow, -1)) |
| initial\_folder | starting path for folders and files |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| change\_submits | If True, pressing Enter key submits window (Default = False) |
| enable\_events | Turns on the element specific events.(Default = False) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Help(button\_text="Help",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

font=**None**,

tooltip=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Help') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| font | specifies the font family, size, etc |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

**ListOfLookAndFeelValues**()

No(button\_text="No",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'No') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

OK(button\_text="OK",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

bind\_return\_key=**True**,

tooltip=**None**,

font=**None**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'OK') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = True) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

**ObjToString**(obj, extra=" ")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| obj |  |
| extra | (Default value = ' ') |

**ObjToStringSingleObj**(obj)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| obj |  |

Ok(button\_text="Ok",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

bind\_return\_key=**True**,

tooltip=**None**,

font=**None**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Ok') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = True) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

OneLineProgressMeter(title,

current\_value,

max\_value,

key,

args,

orientation="v",

bar\_color=(None, None),

button\_color=None,

size=(20, 20),

border\_width=None,

grab\_anywhere=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| title | text to display |
| current\_value | current progressbar value |
| max\_value | max value of progressbar |
| key | Used with window.FindElement and with return values to uniquely identify this element |
| \*args | stuff to output. |
| orientation | 'horizontal' or 'vertical' ('h' or 'v' work) (Default value = 'vertical')(Default value = 'v') |
| bar\_color |  |
| button\_color | button color (foreground, background) |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high (Default value = DEFAULT\_PROGRESS\_BAR\_SIZE) |
| border\_width | width of border around element |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |

**OneLineProgressMeterCancel**(key)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | Used with window.FindElement and with return values to uniquely identify this element |

**Open**(button\_text="Open",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

bind\_return\_key=**True**,

tooltip=**None**,

font=**None**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Open') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = True) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

Show Popup box that doesn't block and closes itself

PopupQuick(args,

title=None,

button\_type=0,

button\_color=None,

background\_color=None,

text\_color=None,

auto\_close=True,

auto\_close\_duration=2,

non\_blocking=True,

icon=None,

line\_width=None,

font=None,

no\_titlebar=False,

grab\_anywhere=False,

keep\_on\_top=False,

location=(None, None))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration | (Default value = 2) |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show Popup window with no titlebar, doesn't block, and auto closes itself.

PopupQuickMessage(args,

title=None,

button\_type=5,

button\_color=None,

background\_color=None,

text\_color=None,

auto\_close=True,

auto\_close\_duration=2,

non\_blocking=True,

icon=None,

line\_width=None,

font=None,

no\_titlebar=True,

grab\_anywhere=False,

keep\_on\_top=False,

location=(None, None))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_NO\_BUTTONS) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration | (Default value = 2) |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = True) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show a scrolled Popup window containing the user's text that was supplied. Use with as many items to print as you want, just like a print statement.

PopupScrolled(args,

title=**None**,

button\_color=**None**,

yes\_no=**False**,

auto\_close=**False**,

auto\_close\_duration=**None**,

size=(**None**, **None**),

location=(**None**, **None**),

non\_blocking=**False**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | (Any) Variable number of items to display |
| title | (str) Title to display in the window. |
| button\_color | Tuple[str, str] button color (foreground, background) |
| yes\_no | (bool) If True, displays Yes and No buttons instead of Ok |
| auto\_close | (bool) if True window will close itself |
| auto\_close\_duration | Union[int, float] Older versions only accept int. Time in seconds until window will close |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| location | Tuple[int, int] Location on the screen to place the upper left corner of the window |
| non\_blocking | (bool) if True the call will immediately return rather than waiting on user input |
|  |  |
| **return** | Union[str, None, TIMEOUT\_KEY] Returns text of the button that was pressed. None will be returned if user closed window with X |

Popup that closes itself after some time period

PopupTimed(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**True**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Display Popup with Yes and No buttons

PopupYesNo(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location | Location on screen to display |
|  |  |
| **return** | Union["Yes", "No", None] |

**Print**(args,

size=(**None**, **None**),

end=**None**,

sep=**None**,

location=(**None**, **None**),

font=**None**,

no\_titlebar=**False**,

no\_button=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

do\_not\_reroute\_stdout=**True**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| end |  |
| sep |  |
| location | Location on screen to display |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| no\_button | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| do\_not\_reroute\_stdout | (Default = True) |

**PrintClose**()

**Quit**(button\_text="Quit",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Quit') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

RButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

bind\_return\_key=**False**,

disabled=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| disabled | set disable state for element (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

ReadButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

bind\_return\_key=**False**,

disabled=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| disabled | set disable state for element (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

RealtimeButton(button\_text,

image\_filename=**None**,

image\_data=**None**,

image\_size=(**None**, **None**),

image\_subsample=**None**,

border\_width=**None**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

font=**None**,

disabled=**False**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button |
| image\_filename | image filename if there is a button image |
| image\_data | in-RAM image to be displayed on button |
| image\_size | size of button image in pixels |
| image\_subsample | amount to reduce the size of the image |
| border\_width | width of border around element |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high (Default = (None)) |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| font | specifies the font family, size, etc |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |

**Save**(button\_text="Save",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

bind\_return\_key=**True**,

disabled=**False**,

tooltip=**None**,

font=**None**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Save') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| bind\_return\_key | (Default = True) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

SaveAs(button\_text="Save As...",

target=(555666777, -1),

file\_types=(('ALL Files', '\*.\*'),),

initial\_folder=**None**,

disabled=**False**,

tooltip=**None**,

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

change\_submits=**False**,

enable\_events=**False**,

font=**None**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Save As...') |
| target | key or (row,col) target for the button (Default value = (ThisRow, -1)) |
| file\_types | (Default value = (("ALL Files", "*.*"))) |
| initial\_folder | starting path for folders and files |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| change\_submits | If True, pressing Enter key submits window (Default = False) |
| enable\_events | Turns on the element specific events.(Default = False) |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Show a scrolled Popup window containing the user's text that was supplied. Use with as many items to print as you want, just like a print statement.

ScrolledTextBox(args,

title=**None**,

button\_color=**None**,

yes\_no=**False**,

auto\_close=**False**,

auto\_close\_duration=**None**,

size=(**None**, **None**),

location=(**None**, **None**),

non\_blocking=**False**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | (Any) Variable number of items to display |
| title | (str) Title to display in the window. |
| button\_color | Tuple[str, str] button color (foreground, background) |
| yes\_no | (bool) If True, displays Yes and No buttons instead of Ok |
| auto\_close | (bool) if True window will close itself |
| auto\_close\_duration | Union[int, float] Older versions only accept int. Time in seconds until window will close |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| location | Tuple[int, int] Location on the screen to place the upper left corner of the window |
| non\_blocking | (bool) if True the call will immediately return rather than waiting on user input |
|  |  |
| **return** | Union[str, None, TIMEOUT\_KEY] Returns text of the button that was pressed. None will be returned if user closed window with X |

Sets the icon which will be used any time a window is created if an icon is not provided when the window is created.

**SetGlobalIcon**(icon)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| icon | Union[bytes, str] Either a Base64 byte string or a filename |

**SetOptions**(icon=**None**,

button\_color=**None**,

element\_size=(**None**, **None**),

button\_element\_size=(**None**, **None**),

margins=(**None**, **None**),

element\_padding=(**None**, **None**),

auto\_size\_text=**None**,

auto\_size\_buttons=**None**,

font=**None**,

border\_width=**None**,

slider\_border\_width=**None**,

slider\_relief=**None**,

slider\_orientation=**None**,

autoclose\_time=**None**,

message\_box\_line\_width=**None**,

progress\_meter\_border\_depth=**None**,

progress\_meter\_style=**None**,

progress\_meter\_relief=**None**,

progress\_meter\_color=**None**,

progress\_meter\_size=**None**,

text\_justification=**None**,

background\_color=**None**,

element\_background\_color=**None**,

text\_element\_background\_color=**None**,

input\_elements\_background\_color=**None**,

input\_text\_color=**None**,

scrollbar\_color=**None**,

text\_color=**None**,

element\_text\_color=**None**,

debug\_win\_size=(**None**, **None**),

window\_location=(**None**, **None**),

error\_button\_color=(**None**, **None**),

tooltip\_time=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| icon | filename of icon used for taskbar and title bar |
| button\_color | button color (foreground, background) |
| element\_size | Tuple[int, int] element size (width, height) in characters |
| button\_element\_size | Tuple[int, int] |
| margins | tkinter margins around outsize (Default = (None)) |
| element\_padding | (Default = (None)) |
| auto\_size\_text | True if size should fit the text length |
| auto\_size\_buttons |  |
| font | specifies the font family, size, etc |
| border\_width | width of border around element |
| slider\_border\_width |  |
| slider\_relief |  |
| slider\_orientation |  |
| autoclose\_time |  |
| message\_box\_line\_width |  |
| progress\_meter\_border\_depth |  |
| progress\_meter\_style |  |
| progress\_meter\_relief |  |
| progress\_meter\_color |  |
| progress\_meter\_size | Tuple[int, int] |
| text\_justification |  |
| background\_color | color of background |
| element\_background\_color |  |
| text\_element\_background\_color |  |
| input\_elements\_background\_color |  |
| input\_text\_color |  |
| scrollbar\_color |  |
| text\_color | color of the text |
| element\_text\_color |  |
| debug\_win\_size | Tuple[int, int] (Default = (None)) |
| window\_location | (Default = (None)) |
| error\_button\_color | (Default = (None)) |
| tooltip\_time | time in milliseconds to wait before showing a tooltip. Default is 400ms |

Submit(button\_text="Submit",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

bind\_return\_key=**True**,

tooltip=**None**,

font=**None**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Submit') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| bind\_return\_key | (Default = True) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Time your code easily.... start the timer.

**TimerStart**()

Time your code easily.... stop the timer and print the number of ms since the timer start

**TimerStop**()

Yes(button\_text="Yes",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**True**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Yes') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = True) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Change the "color scheme" of all future PySimpleGUI Windows. The scheme are string names that specify a group of colors. Background colors, text colors, button colors. There are 13 different color settings that are changed at one time using a single call to ChangeLookAndFeel The look and feel table itself has these indexe into the dictionary LOOK\_AND\_FEEL\_TABLE SystemDefault Material1 Material2 Reddit Topanga GreenTan Dark LightGreen Dark2 Black Tan TanBlue DarkTanBlue DarkAmber DarkBlue Reds Green BluePurple Purple BlueMono GreenMono BrownBlue BrightColors NeutralBlue Kayak SandyBeach TealMono

**change\_look\_and\_feel**(index, force=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| index | (str) the name of the index into the Look and Feel table |
| force | (bool) if True allows Macs to use the look and feel feature. Otherwise Macs are blocked due to problems with button colors |

easy\_print(args,

size=(**None**, **None**),

end=**None**,

sep=**None**,

location=(**None**, **None**),

font=**None**,

no\_titlebar=**False**,

no\_button=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

do\_not\_reroute\_stdout=**True**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| end |  |
| sep |  |
| location | Location on screen to display |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| no\_button | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| do\_not\_reroute\_stdout | (Default = True) |

**easy\_print\_close**()

eprint(args,

size=(**None**, **None**),

end=**None**,

sep=**None**,

location=(**None**, **None**),

font=**None**,

no\_titlebar=**False**,

no\_button=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

do\_not\_reroute\_stdout=**True**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| end |  |
| sep |  |
| location | Location on screen to display |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| no\_button | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| do\_not\_reroute\_stdout | (Default = True) |

Fills a window with values provided in a values dictionary { element\_key : new\_value }

**fill\_form\_with\_values**(window, values\_dict)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| window | (Window) The window object to fill |
| values\_dict | (Dict[Any:Any]) A dictionary with element keys as key and value is values parm for Update call |

**list\_of\_look\_and\_feel\_values**()

The PySimpleGUI "Test Harness". This is meant to be a super-quick test of the Elements.

**main**()

**obj\_to\_string**(obj, extra=" ")

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| obj |  |
| extra | (Default value = ' ') |

**obj\_to\_string\_single\_obj**(obj)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| obj |  |

one\_line\_progress\_meter(title,

current\_value,

max\_value,

key,

args,

orientation="v",

bar\_color=(None, None),

button\_color=None,

size=(20, 20),

border\_width=None,

grab\_anywhere=False)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| title | text to display |
| current\_value | current progressbar value |
| max\_value | max value of progressbar |
| key | Used with window.FindElement and with return values to uniquely identify this element |
| \*args | stuff to output. |
| orientation | 'horizontal' or 'vertical' ('h' or 'v' work) (Default value = 'vertical')(Default value = 'v') |
| bar\_color |  |
| button\_color | button color (foreground, background) |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high (Default value = DEFAULT\_PROGRESS\_BAR\_SIZE) |
| border\_width | width of border around element |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |

**one\_line\_progress\_meter\_cancel**(key)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| key | Used with window.FindElement and with return values to uniquely identify this element |

Popup - Display a popup Window with as many parms as you wish to include. This is the GUI equivalent of the "print" statement. It's also great for "pausing" your program's flow until the user can read some error messages.

popup(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

button\_type=0,

auto\_close=**False**,

auto\_close\_duration=**None**,

custom\_text=(**None**, **None**),

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | (Any) Variable number of your arguments. Load up the call with stuff to see! |
| title | (str) Optional title for the window. If none provided, the first arg will be used instead. |
| button\_color | Tuple[str, str] Color of the buttons shown (text color, button color) |
| background\_color | (str) Window's background color |
| text\_color | (str) text color |
| button\_type | (enum) NOT USER SET! Determines which pre-defined buttons will be shown (Default value = POPUP\_BUTTONS\_OK). There are many Popup functions and they call Popup, changing this parameter to get the desired effect. |
| auto\_close | (bool) If True the window will automatically close |
| auto\_close\_duration | (int) time in seconds to keep window open before closing it automatically |
| custom\_text | Union[Tuple[str, str], str] A string or pair of strings that contain the text to display on the buttons |
| non\_blocking | (bool) If True then will immediately return from the function without waiting for the user's input. |
| icon | Union[str, bytes] icon to display on the window. Same format as a Window call |
| line\_width | (int) Width of lines in characters. Defaults to MESSAGE\_BOX\_LINE\_WIDTH |
| font | Union[str, tuple(font name, size, modifiers) specifies the font family, size, etc |
| no\_titlebar | (bool) If True will not show the frame around the window and the titlebar across the top |
| grab\_anywhere | (bool) If True can grab anywhere to move the window. If no\_titlebar is True, grab\_anywhere should likely be enabled too |
| location | Tuple[int, int] Location on screen to display the top left corner of window. Defaults to window centered on screen |
|  |  |
| **return** | Union[str, None] Returns text of the button that was pressed. None will be returned if user closed window with X |

Show animation one frame at a time. This function has its own internal clocking meaning you can call it at any frequency and the rate the frames of video is shown remains constant. Maybe your frames update every 30 ms but your event loop is running every 10 ms. You don't have to worry about delaying, just call it every time through the loop.

popup\_animated(image\_source,

message=**None**,

background\_color=**None**,

text\_color=**None**,

font=**None**,

no\_titlebar=**True**,

grab\_anywhere=**True**,

keep\_on\_top=**True**,

location=(**None**, **None**),

alpha\_channel=**None**,

time\_between\_frames=0,

transparent\_color=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| image\_source | Union[str, bytes] Either a filename or a base64 string. |
| message | (str) An optional message to be shown with the animation |
| background\_color | (str) color of background |
| text\_color | (str) color of the text |
| font | Union[str, tuple) specifies the font family, size, etc |
| no\_titlebar | (bool) If True then the titlebar and window frame will not be shown |
| grab\_anywhere | (bool) If True then you can move the window just clicking anywhere on window, hold and drag |
| keep\_on\_top | (bool) If True then Window will remain on top of all other windows currently shownn |
| location | (int, int) (x,y) location on the screen to place the top left corner of your window. Default is to center on screen |
| alpha\_channel | (float) Window transparency 0 = invisible 1 = completely visible. Values between are see through |
| time\_between\_frames | (int) Amount of time in milliseconds between each frame |
| transparent\_color | (str) This color will be completely see-through in your window. Can even click through |

Display a Popup without a titlebar. Enables grab anywhere so you can move it

popup\_annoying(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

grab\_anywhere=**True**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| grab\_anywhere | (Default = True) |
| location |  |

Popup that closes itself after some time period

popup\_auto\_close(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**True**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Display Popup with "cancelled" button text

popup\_cancel(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Popup with colored button and 'Error' as button text

popup\_error(args,

title=**None**,

button\_color=(**None**, **None**),

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location | (Default = (None)) |

Display popup window with text entry field and browse button so that a file can be chosen by user.

popup\_get\_file(message,

title=**None**,

default\_path="",

default\_extension="",

save\_as=**False**,

multiple\_files=**False**,

file\_types=(('ALL Files', '\*.\*'),),

no\_window=**False**,

size=(**None**, **None**),

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

icon=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**),

initial\_folder=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| message | (str) message displayed to user |
| title | (str) Window title |
| default\_path | (str) path to display to user as starting point (filled into the input field) |
| default\_extension | (str) If no extension entered by user, add this to filename (only used in saveas dialogs) |
| save\_as | (bool) if True, the "save as" dialog is shown which will verify before overwriting |
| multiple\_files | (bool) if True, then allows multiple files to be selected that are returned with ';' between each filename |
| file\_types | Tuple[Tuple[str,str]] List of extensions to show using wildcards. All files (the default) = (("ALL Files", "*.*"),) |
| no\_window | (bool) if True, no PySimpleGUI window will be shown. Instead just the tkinter dialog is shown |
| size | Tuple[int, int] (width, height) of the InputText Element |
| button\_color | Tuple[str, str] Color of the button (text, background) |
| background\_color | (str) background color of the entire window |
| text\_color | (str) color of the message text |
| icon | Union[bytes, str] filename or base64 string to be used for the window's icon |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| no\_titlebar | (bool) If True no titlebar will be shown |
| grab\_anywhere | (bool) If True can click and drag anywhere in the window to move the window |
| keep\_on\_top | (bool) If True the window will remain above all current windows |
| location | Tuyple[int, int] (x,y) Location on screen to display the upper left corner of window |
| initial\_folder | (str) location in filesystem to begin browsing |
|  |  |
| **return** | Union[str, None] string representing the file(s) chosen, None if cancelled or window closed with X |

Display popup with text entry field and browse button so that a folder can be chosen.

popup\_get\_folder(message,

title=**None**,

default\_path="",

no\_window=**False**,

size=(**None**, **None**),

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

icon=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**),

initial\_folder=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| message | (str) message displayed to user |
| title | (str) Window title |
| default\_path | (str) path to display to user as starting point (filled into the input field) |
| no\_window | (bool) if True, no PySimpleGUI window will be shown. Instead just the tkinter dialog is shown |
| size | Tuple[int, int] (width, height) of the InputText Element |
| button\_color | Tuple[str, str] Color of the button (text, background) |
| background\_color | (str) background color of the entire window |
| text\_color | (str) color of the message text |
| icon | Union[bytes, str] filename or base64 string to be used for the window's icon |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| no\_titlebar | (bool) If True no titlebar will be shown |
| grab\_anywhere | (bool) If True can click and drag anywhere in the window to move the window |
| keep\_on\_top | (bool) If True the window will remain above all current windows |
| location | Tuyple[int, int] (x,y) Location on screen to display the upper left corner of window |
| initial\_folder | (str) location in filesystem to begin browsing |
|  |  |
| **return** | Union[str, None] string representing the path chosen, None if cancelled or window closed with X |

Display Popup with text entry field. Returns the text entered or None if closed / cancelled

popup\_get\_text(message,

title=**None**,

default\_text="",

password\_char="",

size=(**None**, **None**),

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

icon=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| message | (str) message displayed to user |
| title | (str) Window title |
| default\_text | (str) default value to put into input area |
| password\_char | (str) character to be shown instead of actually typed characters |
| size | Tuple[int, int] (width, height) of the InputText Element |
| button\_color | Tuple[str, str] Color of the button (text, background) |
| background\_color | (str) background color of the entire window |
| text\_color | (str) color of the message text |
| icon | Union[bytes, str] filename or base64 string to be used for the window's icon |
| font | Union[str, Tuple[str, int]] specifies the font family, size, etc |
| no\_titlebar | (bool) If True no titlebar will be shown |
| grab\_anywhere | (bool) If True can click and drag anywhere in the window to move the window |
| keep\_on\_top | (bool) If True the window will remain above all current windows |
| location | Tuyple[int, int] (x,y) Location on screen to display the upper left corner of window |
|  |  |
| **return** | Union[str, None] Text entered or None if window was closed or cancel button clicked |

Display a Popup without a titlebar. Enables grab anywhere so you can move it

popup\_no\_border(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

grab\_anywhere=**True**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| grab\_anywhere | (Default = True) |
| location |  |

Show a Popup but without any buttons

popup\_no\_buttons(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Display a Popup without a titlebar. Enables grab anywhere so you can move it

popup\_no\_frame(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

grab\_anywhere=**True**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| grab\_anywhere | (Default = True) |
| location |  |

Display a Popup without a titlebar. Enables grab anywhere so you can move it

popup\_no\_titlebar(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

grab\_anywhere=**True**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| grab\_anywhere | (Default = True) |
| location |  |

Show Popup window and immediately return (does not block)

popup\_no\_wait(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**True**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show Popup window and immediately return (does not block)

popup\_non\_blocking(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**True**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Display Popup with OK button only

popup\_ok(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show Popup box that doesn't block and closes itself

popup\_quick(args,

title=None,

button\_type=0,

button\_color=None,

background\_color=None,

text\_color=None,

auto\_close=True,

auto\_close\_duration=2,

non\_blocking=True,

icon=None,

line\_width=None,

font=None,

no\_titlebar=False,

grab\_anywhere=False,

keep\_on\_top=False,

location=(None, None))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration | (Default value = 2) |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show Popup window with no titlebar, doesn't block, and auto closes itself.

popup\_quick\_message(args,

title=None,

button\_type=5,

button\_color=None,

background\_color=None,

text\_color=None,

auto\_close=True,

auto\_close\_duration=2,

non\_blocking=True,

icon=None,

line\_width=None,

font=None,

no\_titlebar=True,

grab\_anywhere=False,

keep\_on\_top=False,

location=(None, None))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_NO\_BUTTONS) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration | (Default value = 2) |
| non\_blocking | (Default = True) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = True) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Show a scrolled Popup window containing the user's text that was supplied. Use with as many items to print as you want, just like a print statement.

popup\_scrolled(args,

title=**None**,

button\_color=**None**,

yes\_no=**False**,

auto\_close=**False**,

auto\_close\_duration=**None**,

size=(**None**, **None**),

location=(**None**, **None**),

non\_blocking=**False**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | (Any) Variable number of items to display |
| title | (str) Title to display in the window. |
| button\_color | Tuple[str, str] button color (foreground, background) |
| yes\_no | (bool) If True, displays Yes and No buttons instead of Ok |
| auto\_close | (bool) if True window will close itself |
| auto\_close\_duration | Union[int, float] Older versions only accept int. Time in seconds until window will close |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| location | Tuple[int, int] Location on the screen to place the upper left corner of the window |
| non\_blocking | (bool) if True the call will immediately return rather than waiting on user input |
|  |  |
| **return** | Union[str, None, TIMEOUT\_KEY] Returns text of the button that was pressed. None will be returned if user closed window with X |

Popup that closes itself after some time period

popup\_timed(args,

title=**None**,

button\_type=0,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**True**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_type | (Default value = POPUP\_BUTTONS\_OK) |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = True) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location |  |

Display Popup with Yes and No buttons

popup\_yes\_no(args,

title=**None**,

button\_color=**None**,

background\_color=**None**,

text\_color=**None**,

auto\_close=**False**,

auto\_close\_duration=**None**,

non\_blocking=**False**,

icon=**None**,

line\_width=**None**,

font=**None**,

no\_titlebar=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

location=(**None**, **None**))

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| title |  |
| button\_color | button color (foreground, background) |
| background\_color | color of background |
| text\_color | color of the text |
| auto\_close | (Default = False) |
| auto\_close\_duration |  |
| non\_blocking | (Default = False) |
| icon | Icon to display |
| line\_width | Width of lines in characters |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| location | Location on screen to display |
|  |  |
| **return** | Union["Yes", "No", None] |

quit(button\_text="Quit",

size=(**None**, **None**),

auto\_size\_button=**None**,

button\_color=**None**,

disabled=**False**,

tooltip=**None**,

font=**None**,

bind\_return\_key=**False**,

focus=**False**,

pad=**None**,

key=**None**,

metadata=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| button\_text | text in the button (Default value = 'Quit') |
| size | (w,h) w=characters-wide, h=rows-high |
| auto\_size\_button | True if button size is determined by button text |
| button\_color | button color (foreground, background) |
| disabled | set disable state for element (Default = False) |
| tooltip | (str) text, that will appear when mouse hovers over the element |
| font | specifies the font family, size, etc |
| bind\_return\_key | (Default = False) |
| focus | if focus should be set to this |
| pad | Amount of padding to put around element |
| key | Used with window.FindElement and with return values to uniquely identify this element |
|  |  |
| **return** | (Button) |

Sets the icon which will be used any time a window is created if an icon is not provided when the window is created.

**set\_global\_icon**(icon)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| icon | Union[bytes, str] Either a Base64 byte string or a filename |

set\_options(icon=**None**,

button\_color=**None**,

element\_size=(**None**, **None**),

button\_element\_size=(**None**, **None**),

margins=(**None**, **None**),

element\_padding=(**None**, **None**),

auto\_size\_text=**None**,

auto\_size\_buttons=**None**,

font=**None**,

border\_width=**None**,

slider\_border\_width=**None**,

slider\_relief=**None**,

slider\_orientation=**None**,

autoclose\_time=**None**,

message\_box\_line\_width=**None**,

progress\_meter\_border\_depth=**None**,

progress\_meter\_style=**None**,

progress\_meter\_relief=**None**,

progress\_meter\_color=**None**,

progress\_meter\_size=**None**,

text\_justification=**None**,

background\_color=**None**,

element\_background\_color=**None**,

text\_element\_background\_color=**None**,

input\_elements\_background\_color=**None**,

input\_text\_color=**None**,

scrollbar\_color=**None**,

text\_color=**None**,

element\_text\_color=**None**,

debug\_win\_size=(**None**, **None**),

window\_location=(**None**, **None**),

error\_button\_color=(**None**, **None**),

tooltip\_time=**None**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| icon | filename of icon used for taskbar and title bar |
| button\_color | button color (foreground, background) |
| element\_size | Tuple[int, int] element size (width, height) in characters |
| button\_element\_size | Tuple[int, int] |
| margins | tkinter margins around outsize (Default = (None)) |
| element\_padding | (Default = (None)) |
| auto\_size\_text | True if size should fit the text length |
| auto\_size\_buttons |  |
| font | specifies the font family, size, etc |
| border\_width | width of border around element |
| slider\_border\_width |  |
| slider\_relief |  |
| slider\_orientation |  |
| autoclose\_time |  |
| message\_box\_line\_width |  |
| progress\_meter\_border\_depth |  |
| progress\_meter\_style |  |
| progress\_meter\_relief |  |
| progress\_meter\_color |  |
| progress\_meter\_size | Tuple[int, int] |
| text\_justification |  |
| background\_color | color of background |
| element\_background\_color |  |
| text\_element\_background\_color |  |
| input\_elements\_background\_color |  |
| input\_text\_color |  |
| scrollbar\_color |  |
| text\_color | color of the text |
| element\_text\_color |  |
| debug\_win\_size | Tuple[int, int] (Default = (None)) |
| window\_location | (Default = (None)) |
| error\_button\_color | (Default = (None)) |
| tooltip\_time | time in milliseconds to wait before showing a tooltip. Default is 400ms |

sgprint(args,

size=(**None**, **None**),

end=**None**,

sep=**None**,

location=(**None**, **None**),

font=**None**,

no\_titlebar=**False**,

no\_button=**False**,

grab\_anywhere=**False**,

keep\_on\_top=**False**,

do\_not\_reroute\_stdout=**True**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args |  |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| end |  |
| sep |  |
| location | Location on screen to display |
| font | specifies the font family, size, etc |
| no\_titlebar | (Default = False) |
| no\_button | (Default = False) |
| grab\_anywhere | If True can grab anywhere to move the window (Default = False) |
| do\_not\_reroute\_stdout | (Default = True) |

**sgprint\_close**()

Shows the smaller "popout" window. Default location is the upper right corner of your screen

show\_debugger\_popout\_window(location=(None, None), args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| location | Tuple[int, int] Locations (x,y) on the screen to place upper left corner of the window |
| \*args | Not used |

Shows the large main debugger window

show\_debugger\_window(location=(None, None), args)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| location | Tuple[int, int] Locations (x,y) on the screen to place upper left corner of the window |
| \*args | Not used |

Show a scrolled Popup window containing the user's text that was supplied. Use with as many items to print as you want, just like a print statement.

sprint(args,

title=**None**,

button\_color=**None**,

yes\_no=**False**,

auto\_close=**False**,

auto\_close\_duration=**None**,

size=(**None**, **None**),

location=(**None**, **None**),

non\_blocking=**False**)

Parameter Descriptions:

| **Name** | **Meaning** |
| --- | --- |
| \*args | (Any) Variable number of items to display |
| title | (str) Title to display in the window. |
| button\_color | Tuple[str, str] button color (foreground, background) |
| yes\_no | (bool) If True, displays Yes and No buttons instead of Ok |
| auto\_close | (bool) if True window will close itself |
| auto\_close\_duration | Union[int, float] Older versions only accept int. Time in seconds until window will close |
| size | Tuple[int, int] (w,h) w=characters-wide, h=rows-high |
| location | Tuple[int, int] Location on the screen to place the upper left corner of the window |
| non\_blocking | (bool) if True the call will immediately return rather than waiting on user input |
|  |  |
| **return** | Union[str, None, TIMEOUT\_KEY] Returns text of the button that was pressed. None will be returned if user closed window with X |

The PySimpleGUI "Test Harness". This is meant to be a super-quick test of the Elements.

**test**()