

---

# **tk\_tools Documentation**

**Jason R. Jones**

**Jun 01, 2022**



---

## Contents

---

<b>1 Installation</b>	<b>1</b>
1.1 Pip . . . . .	1
1.2 Setup.py . . . . .	1
1.3 Dependencies . . . . .	1
<b>2 Widget Groups</b>	<b>3</b>
2.1 LabelGrid . . . . .	3
2.2 EntryGrid . . . . .	4
2.3 ButtonGrid . . . . .	4
2.4 KeyValueEntry . . . . .	5
2.5 Calendar . . . . .	6
2.6 MultiSlotFrame . . . . .	8
2.7 SevenSegment . . . . .	9
<b>3 Canvas Widgets</b>	<b>11</b>
3.1 RotaryScale . . . . .	11
3.2 Gauge . . . . .	12
3.3 Graph . . . . .	13
3.4 LED . . . . .	15
<b>4 Smart Widgets</b>	<b>17</b>
4.1 SmartOptionMenu . . . . .	17
4.2 SmartSpinBox . . . . .	18
4.3 SmartCheckbutton . . . . .	18
4.4 SmartListBox . . . . .	18
4.5 BinaryLabel . . . . .	19
4.6 ByteLabel . . . . .	20
<b>5 Tool Tips</b>	<b>23</b>
5.1 ToolTip . . . . .	23
<b>6 Introduction</b>	<b>25</b>
<b>7 Tkinter Setup</b>	<b>27</b>
<b>8 Indices and tables</b>	<b>29</b>
<b>Index</b>	<b>31</b>

---



# CHAPTER 1

---

## Installation

---

### 1.1 Pip

To install, simply `pip install tk_tools`. All images and other source material are included as packages within `python`, so you shouldn't have to do any funky workarounds even when using this package in `pyinstaller` or other static execution environments. Some environments may require some basic modification to this, such as the use of `pip3` instead of `pip`.

### 1.2 Setup.py

Clone the git repository, navigate to the cloned directory, and `python3 setup.py install`.

### 1.3 Dependencies

The `tk_tools` package is written with Python 3.5+ in mind! It uses type hints so that your IDE - such as PyCharm - can easily identify potential issues with your code as you write it. If you want this to support a different python version, create an issue and I'm sure that we can work something out easily enough.



# CHAPTER 2

---

## Widget Groups

---

Widget Groups consist of groups of other widgets.

### 2.1 LabelGrid

Column0	Column1	Column2
1	2	3
1	2	3
1	2	3
1	2	3
1	2	3

```
class groups.LabelGrid(parent, num_of_columns: int, headers: list = None, **options)
```

A table-like display widget.

#### Parameters

- **parent** – the tk parent element of this frame
- **num\_of\_columns** – the number of columns contained of the grid
- **headers** – a list containing the names of the column headers

```
add_row(data: list)
```

Add a row of data to the current widget

**Parameters** **data** – a row of data

**Returns** None

## 2.2 EntryGrid

L0	L1	L2

**class** groups.EntryGrid(*parent*, *num\_of\_columns*: int, *headers*: list = None, \*\**options*)

Add a spreadsheet-like grid of entry widgets.

### Parameters

- **parent** – the tk parent element of this frame
- **num\_of\_columns** – the number of columns contained of the grid
- **headers** – a list containing the names of the column headers

**add\_row**(*data*: list = None)

Add a row of data to the current widget, add a <Tab> binding to the last element of the last row, and set the focus at the beginning of the next row.

**Parameters** **data** – a row of data

**Returns** None

**read**(*as\_dicts*=True)

Read the data from the entry fields

**Parameters** **as\_dicts** – True if list of dicts required, else False

**Returns** entries as a dict or table

## 2.3 ButtonGrid

Column0	Column1	Column2
A1	B1	C1
A2	B2	C2
A3	B3	C3

**class** groups.ButtonGrid(*parent*, *num\_of\_columns*: int, *headers*: list = None, \*\**options*)

A grid of buttons.

### Parameters

- **parent** – the tk parent element of this frame
- **num\_of\_columns** – the number of columns contained of the grid

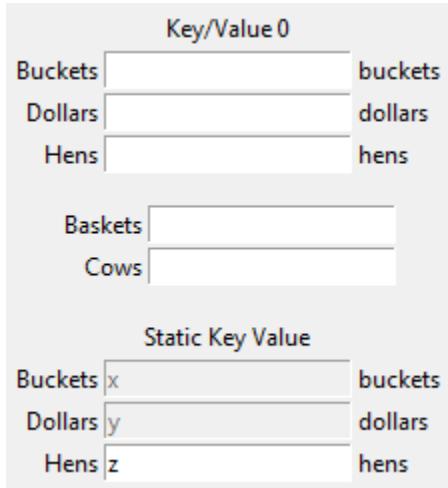
- **headers** – a list containing the names of the column headers

**add\_row** (*data*: list, *row\_label*: str = None)

Add a row of buttons each with their own callbacks to the current widget. Each element in *data* will consist of a label and a command. :param *data*: a list of tuples of the form ('label', <callback>) :return: None

## 2.4 KeyValueEntry

The screenshot consists of three individual examples of KeyValueEntry widgets.



```
class groups.KeyValueEntry(parent, keys: list, defaults: list = None, unit_labels: list = None, enables: list = None, title: str = None, on_change_callback: callable = None, **options)
```

Creates a key-value input/output frame.

### Parameters

- **parent** – the parent frame
- **keys** – the keys represented
- **defaults** – default values for each key
- **unit\_labels** – unit labels for each key (to the right of the value)
- **enables** – True/False for each key
- **title** – The title of the block
- **on\_change\_callback** – a function callback when any element is changed
- **options** – frame tk options

**add\_row** (*key*: str, *default*: str = None, *unit\_label*: str = None, *enable*: bool = None)

Add a single row and re-draw as necessary

### Parameters

- **key** – the name and dict accessor
- **default** – the default value
- **unit\_label** – the label that should be applied at the right of the entry
- **enable** – the ‘enabled’ state (defaults to True)

**Returns**

**change\_enables** (*enables\_list: list*)

Enable/disable inputs.

**Parameters** **enables\_list** – list containing enables for each key

**Returns** None

**get()**

Retrieve the GUI elements for program use.

**Returns** a dictionary containing all of the data from the key/value entries

**load** (*data: dict*)

Load values into the key/values via dict.

**Parameters** **data** – dict containing the key/values that should be inserted

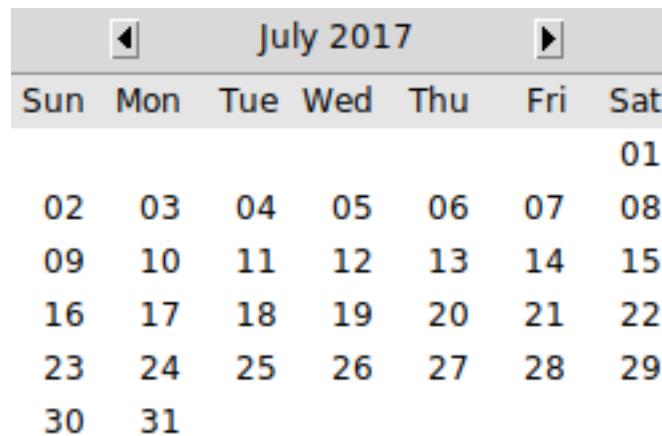
**Returns** None

**reset()**

Clears all entries.

**Returns** None

## 2.5 Calendar



**class** `groups.Calendar` (*parent, callback: callable = None, year: int = None, month: int = None, day: int = None, \*\*kwargs*)

Graphical date selection widget, with callbacks. To change the language, use the `locale` library with the appropriate settings for the target language. For instance, to display the `Calendar` widget in German, you might use:

```
locale.setlocale(locale.LC_ALL, 'deu_deu')
```

**Parameters**

- **parent** – the parent frame
- **callback** – the callable to be executed on selection
- **year** – the year as an integer, i.e. 2020
- **month** – the month as an integer; not zero-indexed; i.e.

“1” will translate to “January” :param day: the day as an integer; not zero-indexed :param kwargs: tkinter.frame keyword arguments

**add\_callback** (*callback: callable*)

Adds a callback to call when the user clicks on a date

**Parameters** **callback** – a callable function

**Returns** None

**class datetime** (*year, month, day[, hour[, minute[, second[, microsecond[, tzinfo]]]]]*)

The year, month and day arguments are required. tzinfo may be None, or an instance of a tzinfo subclass. The remaining arguments may be ints.

**astimezone** ()

tz -> convert to local time in new timezone tz

**combine** ()

date, time -> datetime with same date and time fields

**ctime** ()

Return ctime() style string.

**date** ()

Return date object with same year, month and day.

**dst** ()

Return self.tzinfo.dst(self).

**fromisoformat** ()

string -> datetime from datetime.isoformat() output

**fromtimestamp** ()

timestamp[, tz] -> tz’s local time from POSIX timestamp.

**isoformat** ()

[sep] -> string in ISO 8601 format, YYYY-MM-DDT[HH[:MM[:SS[.mmm[uuu]]]]][+HH:MM]. sep is used to separate the year from the time, and defaults to ‘T’. timespec specifies what components of the time to include (allowed values are ‘auto’, ‘hours’, ‘minutes’, ‘seconds’, ‘milliseconds’, and ‘microseconds’).

**now** ()

Returns new datetime object representing current time local to tz.

**tz** Timezone object.

If no tz is specified, uses local timezone.

**replace** ()

Return datetime with new specified fields.

**strptime** ()

string, format -> new datetime parsed from a string (like time.strptime()).

**time** ()

Return time object with same time but with tzinfo=None.

**timestamp** ()

Return POSIX timestamp as float.

**timetuple** ()

Return time tuple, compatible with time.localtime().

**timetz** ()

Return time object with same time and tzinfo.

**tzname()**

Return self.tzinfo.tzname(self).

**utcfromtimestamp()**

Construct a naive UTC datetime from a POSIX timestamp.

**utcnow()**

Return a new datetime representing UTC day and time.

**utcoffset()**

Return self.tzinfo.utcoffset(self).

**utctimetuple()**

Return UTC time tuple, compatible with time.localtime().

**selection**

Return a datetime representing the current selected date.

**class timedelta**

Difference between two datetime values.

timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

All arguments are optional and default to 0. Arguments may be integers or floats, and may be positive or negative.

**days**

Number of days.

**microseconds**

Number of microseconds (>= 0 and less than 1 second).

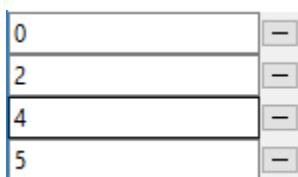
**seconds**

Number of seconds (>= 0 and less than 1 day).

**total\_seconds()**

Total seconds in the duration.

## 2.6 MultiSlotFrame



**class groups.MultiSlotFrame(parent, columns: int = 1)**

Can hold several removable elements, such as a list of files, directories, or a checklist.:

```
# create and grid the frame
msf = tk_tools.MultiSlotFrame(root)
msf.grid()

# add some items
msf.add('item 1')
msf.add('item 2')
```

(continues on next page)

(continued from previous page)

```
# get any user-entered or modified values
print(msf.get())
```

**Parameters**

- **parent** – the tk parent frame
- **columns** – the number of user columns (defaults to 1)

**add**(*string: (<class 'str'>, <class 'list'>)*)

Add a new slot to the multi-frame containing the string. :param string: a string to insert :return: None

**clear**()

Clear out the multi-frame :return:

**get**()

Retrieve and return the values in the multi-frame :return: A list of values containing the contents of the GUI

## 2.7 SevenSegment

**class** groups.SevenSegment(*parent, height: int = 50, digit\_color='black', background='white'*)

Creates a single seven-segment display which may be used to emulate a numeric display of old:

```
# create and grid the frame
ss = tk_tools.SevenSegment(root)
ss.grid()

# set the value
ss.set_value(2)

# set the value with a period
ss.set_value(6.0)
```

**Parameters**

- **parent** – the tk parent frame
- **height** – the widget height (defaults to 50)
- **digit\_color** – the digit color (ex: ‘black’, ‘#ff0000’)
- **background** – the background color (ex: ‘black’, ‘#ff0000’)

**class** groups.SevenSegmentDigits(*parent, digits: int = 1, height: int = 50, digit\_color='black', background='white'*)

Creates a single seven-segment display which may be used to emulate a numeric display of old:

```
# create and grid the frame
ss = tk_tools.SevenSegment(root)
ss.grid()

# set the value
ss.set_value(2)

# set the value with a period
ss.set_value(6.0)
```

### Parameters

- **parent** – the tk parent frame
- **height** – the widget height (defaults to 50)
- **digit\_color** – the digit color (ex: ‘black’, ‘#ff0000’)
- **background** – the background color (ex: ‘black’, ‘#ff0000’)

# CHAPTER 3

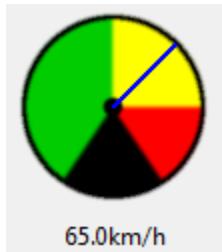
---

## Canvas Widgets

---

These widgets provide visual feedback to the user using the canvas.

### 3.1 RotaryScale



```
class canvas.RotaryScale(parent, max_value: (<class 'float'>, <class 'int'>) = 100.0, size: (<class 'float'>, <class 'int'>) = 100, unit: str = None, img_data: str = None, needle_color='blue', needle_thickness=0, **options)
```

Shows a rotary scale, much like a speedometer.:

```
rs = tk_tools.RotaryScale(root, max_value=100.0, size=100, unit='km/h')
rs.grid(row=0, column=0)

rs.set_value(10)
```

#### Parameters

- **parent** – tkinter parent frame
- **max\_value** – the value corresponding to the maximum value on the scale
- **size** – the size in pixels
- **options** – the frame options

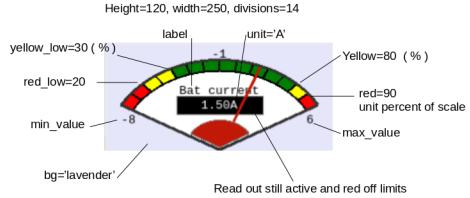
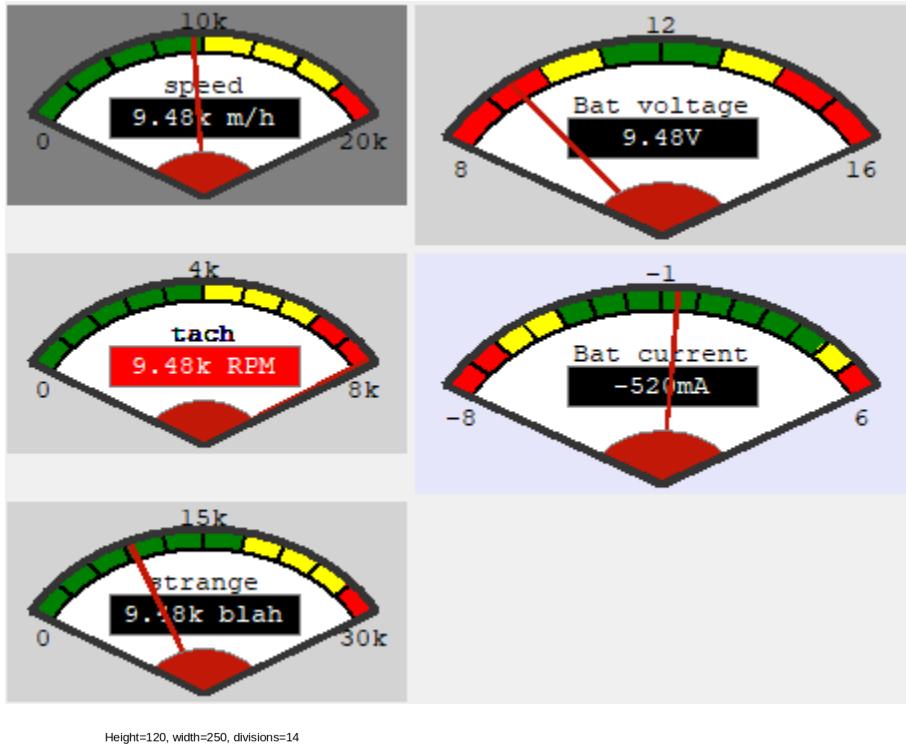
**set\_value** (number: (<class 'float'>, <class 'int'>))

Sets the value of the graphic

**Parameters** **number** – the number (must be between 0 and ‘max\_range’ or the scale will peg the limits)

**Returns** None

## 3.2 Gauge



```
class canvas.Gauge (parent, width: int = 200, height: int = 100, min_value=0.0, max_value=100.0,
                    label="", unit="", divisions=8, yellow=50, red=80, yellow_low=0, red_low=0,
                    bg='lightgrey')
```

Shows a gauge, much like the RotaryGauge.:

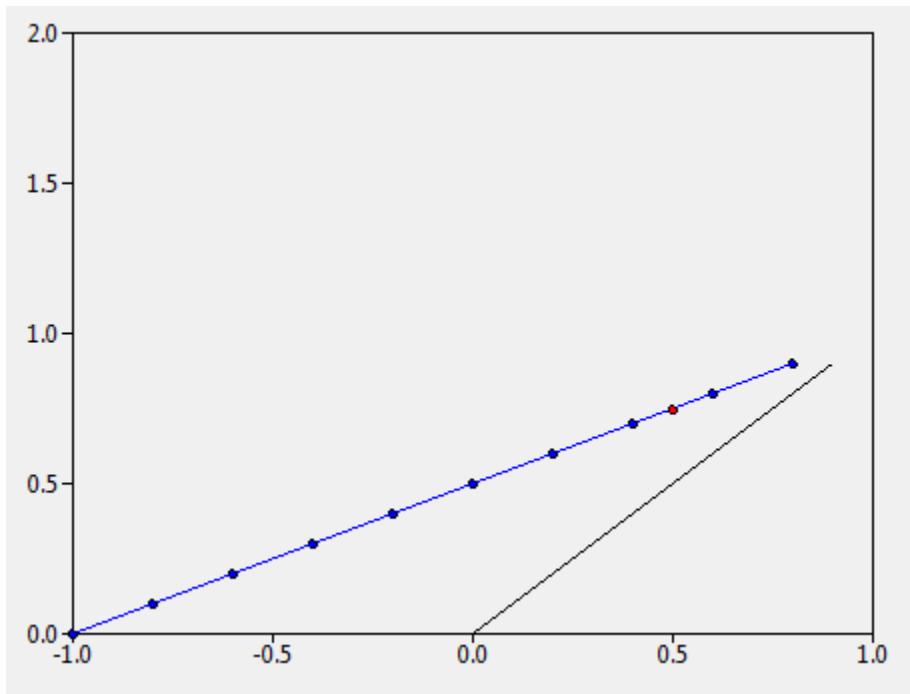
```
gauge = tk_tools.Gauge(root, max_value=100.0,
                      label='speed', unit='km/h')
gauge.grid()
gauge.set_value(10)
```

### Parameters

- **parent** – tkinter parent frame
- **width** – canvas width

- **height** – canvas height
- **min\_value** – the minimum value
- **max\_value** – the maximum value
- **label** – the label on the scale
- **unit** – the unit to show on the scale
- **divisions** – the number of divisions on the scale
- **yellow** – the beginning of the yellow (warning) zone in percent
- **red** – the beginning of the red (danger) zone in percent
- **yellow\_low** – in percent warning for low values
- **red\_low** – in percent if very low values are a danger
- **bg** – background

### 3.3 Graph



```
class canvas.Graph(parent, x_min: float, x_max: float, y_min: float, y_max: float, x_tick: float, y_tick: float, options)
```

Tkinter native graph (pretty basic, but doesn't require heavy install).:

```
graph = tk_tools.Graph(
    parent=root,
    x_min=-1.0,
    x_max=1.0,
    y_min=0.0,
    y_max=2.0,
    x_tick=0.2,
```

(continues on next page)

(continued from previous page)

```
y_tick=0.2,  
width=500,  
height=400  
)  
  
graph.grid(row=0, column=0)  
  
# create an initial line  
line_0 = [(x/10, x/10) for x in range(10)]  
graph.plot_line(line_0)
```

### Parameters

- **parent** – the parent frame
- **x\_min** – the x minimum
- **x\_max** – the x maximum
- **y\_min** – the y minimum
- **y\_max** – the y maximum
- **x\_tick** – the ‘tick’ on the x-axis
- **y\_tick** – the ‘tick’ on the y-axis
- **options** – additional valid tkinter.canvas options

### draw\_axes()

Removes all existing series and re-draws the axes.

#### Returns

None

### static frange (start, stop, step, digits\_to\_round=3)

Works like range for doubles

### Parameters

- **start** – starting value
- **stop** – ending value
- **step** – the increment\_value
- **digits\_to\_round** – the digits to which to round (makes floating-point numbers much easier to work with)

#### Returns

generator

### plot\_line (points: list, color='black', point\_visibility=False)

Plot a line of points

### Parameters

- **points** – a list of tuples, each tuple containing an (x, y) point
- **color** – the color of the line
- **point\_visibility** – True if the points should be individually visible

#### Returns

None

**plot\_point** (*x*, *y*, *visible=True*, *color='black'*, *size=5*)

Places a single point on the grid

#### Parameters

- **x** – the x coordinate
- **y** – the y coordinate
- **visible** – True if the individual point should be visible
- **color** – the color of the point
- **size** – the point size in pixels

**Returns** The absolute coordinates as a tuple

## 3.4 LED

```
class canvas.Led(parent, size: int = 100, on_click_callback: callable = None, toggle_on_click: bool = False, **options)
```

Create an LED-like interface for the user.:

```
led = tk_tools.Led(root, size=50)
led.pack()

led.to_red()
led.to_green(on=True)
```

The user also has the option of adding an *on\_click\_callback* function. When the button is clicked, the button will change state and the on-click callback will be executed. The callback must accept a single boolean parameter, *on*, which indicates if the LED was just turned on or off.

#### Parameters

- **parent** – the parent frame
- **size** – the size in pixels
- **on\_click\_callback** – a callback which accepts a boolean parameter ‘on’
- **options** – the frame options

**to\_green** (*on: bool = False*)

Change the LED to green (on or off).

**Parameters** **on** – True or False

**Returns** None

**to\_grey** (*on: bool = False*)

Change the LED to grey.

**Parameters** **on** – Unused, here for API consistency with the other states

**Returns** None

**to\_red** (*on: bool = False*)

Change the LED to red (on or off) :param on: True or False :return: None

**to\_yellow** (*on: bool = False*)

Change the LED to yellow (on or off) :param on: True or False :return: None



# CHAPTER 4

---

## Smart Widgets

---

Smart widgets consist of existing widgets with improved API. In most cases, these widgets will simply incorporate the appropriate type of `xVar` for the widget type. For instance, imaging providing for an `OptionMenu` without having to use a `StringVar`. These widgets generally appear the same as their ordinary counterparts that are already present within the library.

### 4.1 SmartOptionMenu

```
class widgets.SmartOptionMenu(parent, options: list, initial_value: str = None, callback: callable
                               = None)
```

Classic drop down entry with built-in tracing variable.:

```
# create the dropdown and grid
som = SmartOptionMenu(root, ['one', 'two', 'three'])
som.grid()

# define a callback function that retrieves
# the currently selected option
def callback():
    print(som.get())

# add the callback function to the dropdown
som.add_callback(callback)
```

#### Parameters

- **data** – the tk parent frame
- **options** – a list containing the drop down options
- **initial\_value** – the initial value of the dropdown
- **callback** – a function

## 4.2 SmartSpinBox

```
class widgets.SmartSpinBox(parent, entry_type: str = 'float', callback: callable = None, **options)
```

Easy-to-use spinbox. Takes most options that work with a normal SpinBox. Attempts to call your callback function - if assigned - whenever there is a change to the spinbox.:

```
# create a callback function
def callback(value):
    print('the new value is: ', value)

# create the smart spinbox and grid
ssb = SmartSpinBox(root, from_=0, to=5, callback=callback)
ssb.grid()
```

### Parameters

- **parent** – the tk parent frame
- **entry\_type** – ‘str’, ‘int’, ‘float’
- **callback** – python callable
- **options** – any options that are valid for tkinter.SpinBox

## 4.3 SmartCheckbutton

```
class widgets.SmartCheckbutton(parent, callback: callable = None, **options)
```

Easy-to-use check button. Takes most options that work with a normal CheckButton. Attempts to call your callback function - if assigned - whenever there is a change to the check button.:

```
# create the smart spinbox and grid
scb = SmartCheckbutton(root)
scb.grid()

# define a callback function that retrieves
# the currently selected option
def callback():
    print(scb.get())

# add the callback function to the checkbutton
scb.add_callback(callback)
```

### Parameters

- **parent** – the tk parent frame
- **callback** – python callable
- **options** – any options that are valid for tkinter.Checkbutton

## 4.4 SmartListBox

```
class widgets.SmartListBox(parent, options: List[str], width: int = 12, height: int = 5,
                           on_select_callback: callable = None, selectmode: str = 'browse')
```

Easy-to-use List Box. Takes most options that work with a normal CheckButton. Attempts to call your callback

function - if assigned - whenever there is a change to the list box selections.:

```
# create the smart spinbox and grid
scb = SmartListBox(root, options=['one', 'two', 'three'])
scb.grid()

# define a callback function that retrieves
# the currently selected option
def callback():
    print(scb.get_selected())

# add the callback function to the checkbutton
scb.add_callback(callback)
```

### Parameters

- **parent** – the tk parent frame
- **options** – any options that are valid for tkinter.Checkbutton
- **on\_select\_callback** – python callable
- **selectmode** – the selector mode (supports “browse” and “multiple”)

**add\_callback** (*callback: callable*)

Associates a callback function when the user makes a selection.

**Parameters** **callback** – a callable function

## 4.5 BinaryLabel

d1:10011001

**class** `wIDGETS.BINARYLABEL` (*parent, value: int = 0, prefix: str = ”, bit\_width: int = 8, \*\*options*)

Displays a value binary. Provides methods for easy manipulation of bit values.:

```
# create the label and grid
bl = BinaryLabel(root, 255)
bl.grid()

# toggle highest bit
bl.toggle_msb()
```

### Parameters

- **parent** – the tk parent frame
- **value** – the initial value, default is 0
- **options** – prefix string for identifiers

**clear\_bit** (*position: int*)

Clears the value at position

**Parameters** **position** – integer between 0 and 7, inclusive

**Returns** None

```
clear_lsb()
    Clears the least significant bit :return: None

clear_msb()
    Clears the most significant bit :return: None

get()
    Return the current value

    Returns the current integer value

get_bit (position: int)
    Returns the bit value at position

    Parameters position – integer between 0 and <width>, inclusive

    Returns the value at position as a integer

get_lsb()
    Returns the least significant bit as an integer :return: the LSB

get_msb()
    Returns the most significant bit as an integer :return: the MSB

set (value: int)
    Set the current value

    Parameters value –

    Returns None

set_bit (position: int)
    Sets the value at position

    Parameters position – integer between 0 and 7, inclusive

    Returns None

set_lsb()
    Sets the least significant bit :return: None

set_msb()
    Sets the most significant bit :return: None

toggle_bit (position: int)
    Toggles the value at position

    Parameters position – integer between 0 and 7, inclusive

    Returns None

toggle_lsb()
    Toggles the least significant bit :return:

toggle_msb()
    Changes the most significant bit :return: None
```

## 4.6 ByteLabel

d1:10011001

**class** widgets.**ByteLabel** (*parent*, *value*: int = 0, *prefix*: str = "", *bit\_width*: int = 8, \*\**options*)

Has been replaced with more general BinaryLabel. Still here for backwards compatibility.



# CHAPTER 5

---

## Tool Tips

---

### 5.1 ToolTip

```
class tooltips.ToolTip(widget, text: str = 'widget info', time: int = 4000)
```

Add a tooltip to any widget.:

```
entry = tk.Entry(root)
entry.grid()

# creates a tooltip
tk_tools.ToolTip(entry, 'enter a value between 1 and 10')
```

#### Parameters

- **widget** – the widget on which to hover
- **text** – the text to display
- **time** – the time to display the text, in milliseconds



# CHAPTER 6

---

## Introduction

---

The `tk_tools` package exists in a space like other packages. In many cases, the `tkinter` interface leaves some API to be desired while, in other cases, it leaves out some room for fairly standard visualizations. This is a collection of widgets and tools that have been developed over the course of creating GUI elements as a means to simplify and enhance the process and results.

There are three categories of widgets:

- groups of widgets that are useful as a group
- visual aids using the canvas
- useful improvements on existing widgets



# CHAPTER 7

---

## Tkinter Setup

---

Each of the code examples assumes a structure similar to the below in order to setup the root environment.:

```
import tkinter as tk
import tk_tools

root = tk.Tk()

# -----
# ----- your GUI widget(s) here -----
# -----


root.mainloop()
```



# CHAPTER 8

---

## Indices and tables

---

- genindex



---

## Index

---

### A

add() (*groups.MultiSlotFrame method*), 9  
add\_callback() (*groups.Calendar method*), 7  
add\_callback() (*widgets.SmartListBox method*), 19  
add\_row() (*groups.ButtonGrid method*), 5  
add\_row() (*groups.EntryGrid method*), 4  
add\_row() (*groups.KeyValueEntry method*), 5  
add\_row() (*groups.LabelGrid method*), 3  
astimezone() (*groups.Calendar.datetime method*), 7

### B

BinaryLabel (*class in widgets*), 19  
ButtonGrid (*class in groups*), 4  
ByteLabel (*class in widgets*), 20

### C

Calendar (*class in groups*), 6  
Calendar.datetime (*class in groups*), 7  
Calendar.timedelta (*class in groups*), 8  
change\_enables() (*groups.KeyValueEntry method*),  
    6  
clear() (*groups.MultiSlotFrame method*), 9  
clear\_bit() (*widgets.BinaryLabel method*), 19  
clear\_lsb() (*widgets.BinaryLabel method*), 19  
clear\_msb() (*widgets.BinaryLabel method*), 20  
combine() (*groups.Calendar.datetime method*), 7  
ctime() (*groups.Calendar.datetime method*), 7

### D

date() (*groups.Calendar.datetime method*), 7  
days (*groups.Calendar.timedelta attribute*), 8  
draw\_axes() (*canvas.Graph method*), 14  
dst() (*groups.Calendar.datetime method*), 7

### E

EntryGrid (*class in groups*), 4

### F

frange() (*canvas.Graph static method*), 14

fromisoformat() (*groups.Calendar.datetime method*), 7  
fromtimestamp() (*groups.Calendar.datetime method*), 7

### G

Gauge (*class in canvas*), 12  
get() (*groups.KeyValueEntry method*), 6  
get() (*groups.MultiSlotFrame method*), 9  
get() (*widgets.BinaryLabel method*), 20  
get\_bit() (*widgets.BinaryLabel method*), 20  
get\_lsb() (*widgets.BinaryLabel method*), 20  
get\_msb() (*widgets.BinaryLabel method*), 20  
Graph (*class in canvas*), 13

### I

isoformat() (*groups.Calendar.datetime method*), 7

### K

KeyValueEntry (*class in groups*), 5

### L

LabelGrid (*class in groups*), 3  
Led (*class in canvas*), 15  
load() (*groups.KeyValueEntry method*), 6

### M

microseconds (*groups.Calendar.timedelta attribute*),  
    8  
MultiSlotFrame (*class in groups*), 8

### N

now() (*groups.Calendar.datetime method*), 7

### P

plot\_line() (*canvas.Graph method*), 14  
plot\_point() (*canvas.Graph method*), 14

## R

read() (*groups.EntryGrid method*), 4  
replace() (*groups.Calendar.datetime method*), 7  
reset() (*groups.KeyValueEntry method*), 6  
RotaryScale (*class in canvas*), 11

## S

seconds (*groups.Calendar.timedelta attribute*), 8  
selection (*groups.Calendar attribute*), 8  
set() (*widgets.BinaryLabel method*), 20  
set\_bit() (*widgets.BinaryLabel method*), 20  
set\_lsb() (*widgets.BinaryLabel method*), 20  
set\_msb() (*widgets.BinaryLabel method*), 20  
set\_value() (*canvas.RotaryScale method*), 11  
SevenSegment (*class in groups*), 9  
SevenSegmentDigits (*class in groups*), 9  
SmartCheckbutton (*class in widgets*), 18  
SmartListBox (*class in widgets*), 18  
SmartOptionMenu (*class in widgets*), 17  
SmartSpinBox (*class in widgets*), 18  
strptime() (*groups.Calendar.datetime method*), 7

## T

time() (*groups.Calendar.datetime method*), 7  
timestamp() (*groups.Calendar.datetime method*), 7  
timetuple() (*groups.Calendar.datetime method*), 7  
timetz() (*groups.Calendar.datetime method*), 7  
to\_green() (*canvas.Led method*), 15  
to\_grey() (*canvas.Led method*), 15  
to\_red() (*canvas.Led method*), 15  
to\_yellow() (*canvas.Led method*), 15  
toggle\_bit() (*widgets.BinaryLabel method*), 20  
toggle\_lsb() (*widgets.BinaryLabel method*), 20  
toggle\_msb() (*widgets.BinaryLabel method*), 20  
ToolTip (*class in tooltips*), 23  
total\_seconds() (*groups.Calendar.timedelta method*), 8  
tzname() (*groups.Calendar.datetime method*), 7

## U

utcfromtimestamp() (*groups.Calendar.datetime method*), 8  
utcnow() (*groups.Calendar.datetime method*), 8  
utcoffset() (*groups.Calendar.datetime method*), 8  
utctimetuple() (*groups.Calendar.datetime method*), 8