
domdf-wxpython-tools

Release 0.3.0.post1

Tools and widgets for wxPython.

Dominic Davis-Foster

Jun 05, 2023

Contents

1	Installation	1
1.1	from PyPI	1
1.2	from GitHub	1
2	API Reference	3
2.1	ColourPickerPanel	3
2.2	StylePickerPanel	4
2.3	WebView	6
2.4	border_config	7
2.5	chartpanel	8
2.6	clearable_textctrl	10
2.7	dialogs	20
2.8	editable_listbox	23
2.9	events	28
2.10	filebrowsectrl	30
2.11	icons	35
2.12	imagepanel	35
2.13	keyboard	37
2.14	list_dialog	38
2.15	logctrl	39
2.16	picker	42
2.17	projections	45
2.18	style_picker	49
2.19	tabbable_textctrl	51
2.20	textctrlwrapper	51
2.21	timer_thread	56
2.22	utils	57
2.23	validators	58
2.24	panel_listctrl	60
3	Contributing	67
3.1	Overview	67
3.2	Coding style	67
3.3	Automated tests	67
3.4	Type Annotations	67
3.5	Build documentation locally	68
3.6	Downloading source code	68
	Python Module Index	71
	Index	73

Installation

1.1 from PyPI

```
$ python3 -m pip install domdf_wxpython_tools --user
```

1.2 from GitHub

```
$ python3 -m pip install git+https://github.com/domdfcoding/domdf_wxpython_tools@master --user
```


API Reference

2.1 ColourPickerPanel

Based on StylePickerPanel, a Panel for selecting a list of colours, and their order

Classes:

<code>ColourPickerPanel(parent[, id, pos, size, ...])</code>	Based on StylePickerPanel, a Panel for selecting a list of colours, and their order.
--	--

```
class ColourPickerPanel (parent, id=- 1, pos=(- 1, - 1), size=(- 1, - 1), style=524288,
                        name=b'panel', label='Choose Colours: ', picker_choices=None,
                        selection_choices=None)
```

Bases: `StylePickerPanel`

Based on StylePickerPanel, a Panel for selecting a list of colours, and their order.

Parameters

- **parent** (`Window`) – The parent window.
- **id** (`int`) – An identifier for the panel. `wx.ID_ANY` is taken to mean a default. Default `-1`.
- **pos** (`Point`) – The panel position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **size** (`Size`) – The panel size. The value `wx.DefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **style** (`int`) – The window style. See `wxPanel`. Default `524288`.
- **name** (`str`) – The window name. Default `b'panel'`.
- **label** (`str`) – Label for the panel. Default `'Choose Colours: '`.
- **picker_choices** (`Optional[List[str]]`) – A list of hex value choices to populate the ‘picker’ side of the panel with. Default `None`.
- **selection_choices** (`Optional[List[str]]`) – A list of hex value choices to populate the ‘selection’ side of the panel with. Default `None`.

Methods:

<code>GetSelection()</code>	Returns a list of the currently selected colours
<code>add(event)</code>	Event handler for adding the colour currently selected in the ‘picker’ to the ‘selection’
<code>get_selection()</code>	Returns a list of the currently selected colours

continues on next page

Table 2 – continued from previous page

<code>pick(*args)</code>	Open a <code>wx.ColourDialog</code> to edit the colour currently selected in the picker.
<code>remove(event)</code>	Event handler for removing the colour currently selected in the ‘selection’
<code>update_preview(list_obj, axes)</code>	Update the preview from the given list.

GetSelection()

Returns a list of the currently selected colours

Return type `List[str]`

add(event)

Event handler for adding the colour currently selected in the ‘picker’ to the ‘selection’

get_selection()

Returns a list of the currently selected colours

Return type `List[str]`

pick(*args)

Open a `wx.ColourDialog` to edit the colour currently selected in the picker.

remove(event)

Event handler for removing the colour currently selected in the ‘selection’

update_preview(list_obj, axes)

Update the preview from the given list.

Parameters

- **list_obj** (`ListBox`) – The list to update the preview for.
- **axes** (`Axes`) – The preview axes to update.

2.2 StylePickerPanel

Classes:

<code>StylePickerPanel(parent[, id, pos, size, ...])</code>	Based on <code>StylePickerPanel</code> , a Panel for selecting a list of colours, and their order.
---	--

```
class StylePickerPanel (parent, id=- 1, pos=(- 1, - 1), size=(- 1, - 1), style=524288, name=b'panel',  
                        label='Choose Styles: ', selection_choices=None)
```

Bases: `Panel`

Based on `StylePickerPanel`, a Panel for selecting a list of colours, and their order.

Parameters

- **parent** (`Window`) – The parent window.
- **id** – An identifier for the panel. `wx.ID_ANY` is taken to mean a default. Default `-1`.

- **pos** – The panel position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **size** – The panel size. The value `wx.DefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **style** – The window style. See `wxPanel`. Default `524288`.
- **name** – The window name. Default `b'panel'`.
- **label** – Label for the panel. Default `'Choose Styles: '`.
- **selection_choices** (`Optional[List[str]]`) – A list of hex value choices to populate the 'selection' side of the panel with. Default `None`.

Methods:

`add(event)`

`do_layout()`

`get_selection()`

`move([direction])`

`move_down(event)`

`move_up(event)`

`remove(event)`

`set_properties()`

`update_picker_preview(*_)`

`update_preview(list_obj, axes)`

`update_selection_preview(*_)`

`add (event)``do_layout ()``get_selection ()``move (direction=1)``move_down (event)``move_up (event)``remove (event)``set_properties ()``update_picker_preview (*_)``update_preview (list_obj, axes)``update_selection_preview (*_)`

2.3 WebView

Set the emulation level for wxWidgets WebView purely in Python.

Notes:

- The highest emulation level may be used even when the corresponding browser version is not installed.
- Using the *_FORCE options is not recommended.
- The `wxWEBVIEWIE_EMU_DEFAULT` can be used to reset the emulation level to the system default.

The values of the constants were taken from https://msdn.microsoft.com/library/ee330730.aspx#browser_emulation and must not be changed.

Functions:

<code>MSWSetEmulationLevel([level, program_name])</code>	Sets the emulation level for wxWidgets WebView.
--	---

Data:

<code>wxWEBVIEWIE_EMU_DEFAULT</code>	The system default browser emulation level.
<code>wxWEBVIEWIE_EMU_IE10</code>	Emulate Internet Explorer 10
<code>wxWEBVIEWIE_EMU_IE10_FORCE</code>	Emulate Internet Explorer 10 (force)
<code>wxWEBVIEWIE_EMU_IE11</code>	Emulate Internet Explorer 11
<code>wxWEBVIEWIE_EMU_IE11_FORCE</code>	Emulate Internet Explorer 12 (force)
<code>wxWEBVIEWIE_EMU_IE7</code>	Emulate Internet Explorer 7
<code>wxWEBVIEWIE_EMU_IE8</code>	Emulate Internet Explorer 8
<code>wxWEBVIEWIE_EMU_IE8_FORCE</code>	Emulate Internet Explorer 8 (force)
<code>wxWEBVIEWIE_EMU_IE9</code>	Emulate Internet Explorer 9
<code>wxWEBVIEWIE_EMU_IE9_FORCE</code>	Emulate Internet Explorer 9 (force)

MSWSetEmulationLevel (*level=0, program_name=None*)

Sets the emulation level for wxWidgets WebView.

Parameters

- **level** – The emulation level to use. Default 0.
- **program_name** – The name of the program to set the emulation level for. Defaults to the Python executable.

Returns Whether the operation completed successfully.

wxWEBVIEWIE_EMU_DEFAULT = 0

Type: `int`

The system default browser emulation level.

wxWEBVIEWIE_EMU_IE10 = 10000

Type: `int`

Emulate Internet Explorer 10

wxWEBVIEWIE_EMU_IE10_FORCE = 10001

Type: `int`

Emulate Internet Explorer 10 (force)

wxWEBVIEWIE_EMU_IE11 = 11000

Type: `int`

Emulate Internet Explorer 11

wxWEBVIEWIE_EMU_IE11_FORCE = 11001

Type: `int`

Emulate Internet Explorer 12 (force)

wxWEBVIEWIE_EMU_IE7 = 7000

Type: `int`

Emulate Internet Explorer 7

wxWEBVIEWIE_EMU_IE8 = 8000

Type: `int`

Emulate Internet Explorer 8

wxWEBVIEWIE_EMU_IE8_FORCE = 8888

Type: `int`

Emulate Internet Explorer 8 (force)

wxWEBVIEWIE_EMU_IE9 = 9000

Type: `int`

Emulate Internet Explorer 9

wxWEBVIEWIE_EMU_IE9_FORCE = 9999

Type: `int`

Emulate Internet Explorer 9 (force)

2.4 border_config

Dialog for configuring borders for charts.

Classes:

border_config(parent, chromatogram_figure, ...)

type parent `Window`

class border_config (parent, chromatogram_figure, *args, **kws)

Bases: `Dialog`

Parameters

- **parent** (`Window`)
- **chromatogram_figure**
- ***args**
- ****kwds**

Methods:

`apply_tight_layout(_)`

`close_dialog(_)`

`update_borders(_)`

`apply_tight_layout (_)``close_dialog (_)``update_borders (_)`

2.5 chartpanel

A canvas for displaying a chart within a wxPython window

Classes:

<code>ChartPanelBase</code> (parent, fig, ax[, id, pos, ...])	Panel that contains a matplotlib plotting window, used for displaying an image.
---	---

class ChartPanelBase (parent, fig, ax, id=-1, pos=(-1, -1), size=(-1, -1), style=0, name=b'panel')

Bases: `Panel`

Panel that contains a matplotlib plotting window, used for displaying an image. The image can be right clicked to bring up a context menu allowing copying, pasting and saving of the image. The image can be panned by holding the left mouse button and moving the mouse, and zoomed in and out using the scrollwheel on the mouse.

Parameters

- **parent** (`Window`) – The parent window.
- **fig** (`Figure`)
- **ax** (`Axes`)
- **id** (`int`) – An identifier for the panel. `wx.ID_ANY` is taken to mean a default. Default `-1`.
- **pos** (`Point`) – The panel position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **size** (`Size`) – The panel size. The value `::wxDefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **style** (`int`) – The window style. See `wxPanel`. Default `0`.

- **name** (*str*) – Window name. Default b'panel'.

Methods:

<code>configure_borders([event])</code>	Open the Configure Borders dialog.
<code>constrain_zoom([key])</code>	Constrain zoom to the x axis only.
<code>on_size_change(_)</code>	Event handler for size change events
<code>pan([enable])</code>	Enable the Pan tool.
<code>previous_view(*_)</code>	Go to the previous view of the chart.
<code>reset_view(*_)</code>	Reset the view of the chart.
<code>setup_scrollwheel_zooming([scale])</code>	Allow zooming of the chart with the scrollwheel.
<code>setup_ylim_refresher(y_data, x_data)</code>	Setup the function for updating the ylim whenever the xlim changes.
<code>size_change()</code>	Internal function that runs whenever the window is resized.
<code>zoom([enable])</code>	Enable the Zoom tool.

configure_borders (*event=None*)
Open the Configure Borders dialog.

constrain_zoom (*key='x'*)
Constrain zoom to the x axis only.

Parameters **key** (*str*) – Default 'x'.

on_size_change (*_*)
Event handler for size change events

pan (*enable=True*)
Enable the Pan tool.

previous_view (**_*)
Go to the previous view of the chart.

reset_view (**_*)
Reset the view of the chart.

setup_scrollwheel_zooming (*scale=1.1*)
Allow zooming of the chart with the scrollwheel.

Parameters **scale** (*float*) – Default 1.1.

setup_ylim_refresher (*y_data, x_data*)
Setup the function for updating the ylim whenever the xlim changes.

Parameters

- **y_data**
- **x_data**

size_change (*_*)
Internal function that runs whenever the window is resized.

zoom (*enable=True*)
Enable the Zoom tool.

2.6 clearable_textctrl

A TextCtrl with a button to clear its contents

Classes:

<i>CTCWidget</i> (parent, value, style, validator)	Text control used by <i>ClearableTextCtrl</i> .
<i>ClearButton</i> (parent, eventType, bmp)	Clear button for the <i>ClearableTextCtrl</i> .
<i>ClearableTextCtrl</i> (parent[, id, value, pos, ...])	TextCtrl with a button to clear its contents.

Data:

<i>clear_btn</i>	XPM button icon for clearing the text control.
------------------	--

class **CTCWidget** (*parent, value, style, validator*)

Bases: *TextCtrl*

Text control used by *ClearableTextCtrl*.

Parameters

- **parent** (*ClearableTextCtrl*) – The parent window.
- **value** (*str*) – The initial value of the text control
- **style** (*int*) – The style of the text control
- **validator**

Methods:

<i>GetMainWindowOfCompositeControl</i> ()	Returns the parent object.
<i>OnText</i> (event)	Event handler for text being entered in the control.
<i>OnTextEnter</i> (_)	Event handler for the enter / return key being pressed

GetMainWindowOfCompositeControl ()

Returns the parent object.

Return type *ClearableTextCtrl*

OnText (*event*)

Event handler for text being entered in the control.

Parameters **event** – The wxPython event.

OnTextEnter (_)

Event handler for the enter / return key being pressed

Parameters **event** – The wxPython event.

class ClearButton (*parent, eventType, bmp*)

Bases: *Control*

Clear button for the *ClearableTextCtrl*.

Parameters

- **parent** (*ClearableTextCtrl*) – The parent window.
- **eventType** (*PyEventBinder*)
- **bmp** (*Bitmap*)

Methods:

<i>AcceptsFocusFromKeyboard()</i>	Always returns <i>False</i> .
<i>GetBestSize()</i>	Returns the best size for the control.
<i>GetMainWindowOfCompositeControl()</i>	Returns the parent object.
<i>OnLeftUp(_)</i>	Event Handler for left mouse button being released.
<i>OnPaint(_)</i>	Event Handler for widget being painted.
<i>SetBitmapLabel(label)</i>	Set bitmap for the button.

AcceptsFocusFromKeyboard()

Always returns *False*.

Return type *bool*

GetBestSize()

Returns the best size for the control.

Return type *Size*

GetMainWindowOfCompositeControl()

Returns the parent object.

Return type *ClearableTextCtrl*

OnLeftUp(_)

Event Handler for left mouse button being released.

OnPaint(_)

Event Handler for widget being painted.

SetBitmapLabel(label)

Set bitmap for the button.

Parameters **label** (*Bitmap*) – Bitmap to set for the button.

class ClearableTextCtrl (*parent, id=-1, value="", pos=(-1, -1), size=(-1, -1), style=0, validator=<wx.Validator object>, name='ClearableTextCtrl'*)

Bases: *TextCtrlWrapper, Panel*

TextCtrl with a button to clear its contents.

Parameters

- **parent** (*Window*) – The parent window.

- **id** (*int*) – An identifier for the control. `wx.ID_ANY` is taken to mean a default. Default `-1`.
- **value** (*str*) – Default text value. Default `' '`.
- **pos** (*Point*) – The control position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **size** (*Size*) – The control size. The value `wx.DefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **style** (*int*) – The window style. See `wx.TextCtrl`. Default `0`.
- **validator** (*Validator*) – Window validator. Default `<wx.Validator object at 0x7f0d47ba8b50>`.
- **name** (*str*) – Window name. Default `'ClearableTextCtrl'`.

Methods:

<i>AutoComplete</i> (completer)	Enable auto-completion using the provided completer object.
<i>AutoCompleteDirectories</i> ()	Call this function to enable auto-completion of the text using the file system directories.
<i>AutoCompleteFileNames</i> ()	Call this function to enable auto-completion of the text typed in a single-line text control using all valid file system paths.
<i>ChangeValue</i> (value)	Sets the new text control value.
<i>DiscardEdits</i> ()	Resets the internal modified flag as if the current changes had been saved.
<i>EmulateKeyPress</i> (event)	Inserts into the control the character which would have been inserted if the given key event had occurred in the text control.
<i>GetBestClientSize</i> ()	Returns the best size for the control.
<i>GetCompositeWindowParts</i> ()	Returns a list of <code>wx.Window</code> objects that make up this control.
<i>GetDefaultStyle</i> ()	Returns the style currently used for new text.
<i>GetInsertionPoint</i> ()	Returns the insertion point, or cursor, position.
<i>GetLineLength</i> (lineNo)	Gets the length of the specified line, not including any trailing newline character(s).
<i>GetLineText</i> (lineNo)	Returns the contents of a given line in the text control, not including any trailing newline character(s).
<i>GetNumberOfLines</i> ()	Returns the number of lines in the text control buffer.
<i>GetRange</i> (from_, to_)	Returns the string containing the text starting in the positions from and up to to in the control.
<i>GetStyle</i> (position, style)	Returns the style at this position in the text control.
<i>HitTest</i> (pt)	Finds the row and column of the character at the specified point.
<i>HitTestPos</i> (pt)	Finds the position of the character at the specified point.
<i>IsClearButtonVisible</i> ()	Returns the clear button's visibility state.
<i>IsModified</i> ()	Returns whether the text has been modified by user.
<i>IsMultiLine</i> ()	Returns whether this is a multi line control.
<i>IsSingleLine</i> ()	Returns whether this is a single line control.
<i>LayoutControls</i> ()	Lays out the child controls.

continues on next page

Table 15 – continued from previous page

<i>MarkDirty()</i>	Mark the control as modified (dirty).
<i>OnClearButton(event)</i>	Event handler for clear button being pressed
<i>OnSize(event)</i>	Event handler for the size of the control being changed.
<i>PositionToXY(pos)</i>	Converts given position to a zero-based column, line number pair.
<i>SetBackgroundColour(colour)</i>	Sets the background colour of the control
<i>SetClearBitmap(bitmap)</i>	Sets the bitmap for the clear button.
<i>SetDefaultStyle(style)</i>	Changes the default style to use for the new text which is going to be added to the control.
<i>SetEditable(editable)</i>	Makes the text item editable or read-only, overriding the <code>wx.TE_READONLY</code> flag.
<i>SetFont(font)</i>	Sets the font for the control.
<i>SetInsertionPoint(pos)</i>	Sets the insertion point at the given position.
<i>SetInsertionPointEnd()</i>	Sets the insertion point at the end of the text control.
<i>SetMaxLength(length)</i>	This function sets the maximum number of characters the user can enter into the control.
<i>SetModified(modified)</i>	Marks the control as being modified by the user or not.
<i>SetStyle(start, end, style)</i>	Changes the style of the given range.
<i>ShouldInheritColours()</i>	
rtype <code>bool</code>	
<i>ShowPosition(pos)</i>	Makes the line containing the given position visible.
<i>XYToPosition(x, y)</i>	Converts the given zero-based column and line number to a position.
Attributes:	
<i>default_clear_bitmap</i>	Returns the default clear button bitmap for the control.

AutoComplete (*completer*)

Enable auto-completion using the provided completer object.

The specified completer object will be used to retrieve the list of possible completions for the already entered text and will be deleted by `wx.TextEntry` itself when it's not needed any longer.

Parameters **completer** (`TextCompleter`) – The object to be used for generating completions if not `None`. If it is `None`, auto-completion is disabled. The `wx.TextEntry` object takes ownership of this pointer and will delete it in any case (i.e. even if this method return `False`).

Return type `bool`

Returns `True` if the auto-completion was enabled or `False` if the operation failed, typically because auto-completion is not supported by the current platform.

AutoCompleteDirectories ()

Call this function to enable auto-completion of the text using the file system directories.

Unlike `AutoCompleteFileNames()`, which completes both file names and directories, this function only completes the directory names.

Note: This function is only implemented in wxMSW port and does nothing under the other platforms.

Return type `bool`

Returns `True` if the auto-completion was enabled or `False` if the operation failed, typically because auto-completion is not supported by the current platform.

AutoCompleteFileNames ()

Call this function to enable auto-completion of the text typed in a single-line text control using all valid file system paths.

Note: This function is only implemented in wxMSW port and does nothing under the other platforms.

Return type `bool`

Returns `True` if the auto-completion was enabled or `False` if the operation failed, typically because auto-completion is not supported by the current platform.

ChangeValue (value)

Sets the new text control value.

It also marks the control as not-modified which means that `IsModified()` would return `False` immediately after the call to `ChangeValue()`.

The insertion point is set to the start of the control (i.e. position 0) by this function.

This functions does not generate the `wx.EVT_TEXT` event but otherwise is identical to `SetValue()`.

Parameters **value** (`str`) – The new value to set. It may contain newline characters if the text control is multiline.

DiscardEdits ()

Resets the internal modified flag as if the current changes had been saved.

EmulateKeyPress (event)

Inserts into the control the character which would have been inserted if the given key event had occurred in the text control.

The event object should be the same as the one passed to `wx.EVT_KEY_DOWN` handler previously by `wxWidgets`.

Note: This function doesn't currently work correctly for all keys under any platform but MSW.

Parameters **event** (`KeyEvent`)

Return type `bool`

Returns `True` if the event resulted in a change to the control, `False` otherwise.

GetBestClientSize()

Returns the best size for the control.

Return type `Size`

GetCompositeWindowParts()

Returns a list of `wx.Window` objects that make up this control.

Return type `List[Window]`

GetDefaultStyle()

Returns the style currently used for new text.

Return type `TextAttr`

GetInsertionPoint()

Returns the insertion point, or cursor, position.

This is defined as the zero based index of the character position to the right of the insertion point. For example, if the insertion point is at the end of the single-line text control, it is equal to `GetLastPosition()`.

Return type `int`

GetLineLength(*lineNo*)

Gets the length of the specified line, not including any trailing newline character(s).

Parameters `lineNo` (`int`) – Line number (starting from zero).

Return type `int`

Returns The length of the line, or -1 if `lineNo` was invalid.

GetLineText(*lineNo*)

Returns the contents of a given line in the text control, not including any trailing newline character(s).

Parameters `lineNo` (`int`) – Line number (starting from zero).

Return type `str`

GetNumberOfLines()

Returns the number of lines in the text control buffer.

Return type `int`

GetRange(*from_*, *to_*)

Returns the string containing the text starting in the positions `from` and up to `to` in the control.

The positions must have been returned by another `wx.TextCtrl` method.

Note: The positions in a multiline `wx.TextCtrl` do not correspond to the indices in the string returned by `GetValue` because of the different new line representations (`CR` or `CR LF`) and so this method should be used to obtain the correct results instead of extracting parts of the entire value. It may also be more efficient, especially if the control contains a lot of data.

Parameters

- `from_ (int)`
- `to_ (int)`

Return type `str`

GetStyle (*position*, *style*)

Returns the style at this position in the text control.

Not all platforms support this function.

Parameters

- **position** (`int`)
- **style** (`TextAttr`)

Return type `bool`

Returns `True` on success, `False` if an error occurred (this may also mean that the styles are not supported under this platform).

HitTest (*pt*)

Finds the row and column of the character at the specified point.

If the return code is not `wx.TE_HT_UNKNOWN` the row and column of the character closest to this position are returned, otherwise the output parameters are not modified.

Note: This function is currently only implemented in Univ, wxMSW and wxGTK ports and always returns `wx.TE_HT_UNKNOWN` in the other ports.

NB: *pt* is in device coords (not adjusted for the client area origin nor scrolling).

Parameters *pt*

Return type `TextCtrlHitTestResult`

HitTestPos (*pt*)

Finds the position of the character at the specified point.

If the return code is not `wx.TE_HT_UNKNOWN` the position of the character closest to this position is returned, otherwise the output parameter is not modified.

Note: This function is currently only implemented in Univ, wxMSW and wxGTK ports and always returns `wx.TE_HT_UNKNOWN` in the other ports.

Parameters *pt*

Return type `wxTextCtrlHitTestResult`

IsClearButtonVisible ()

Returns the clear button's visibility state.

Return type `bool`

IsModified()

Returns whether the text has been modified by user.

Note: Calling `SetValue()` doesn't make the control modified.

Return type `bool`

IsMultiLine()

Returns whether this is a multi line control.

Return type `bool`

IsSingleLine()

Returns whether this is a single line control.

Return type `bool`

LayoutControls()

Lays out the child controls.

MarkDirty()

Mark the control as modified (dirty).

OnClearButton(event)

Event handler for clear button being pressed

Parameters `event` (`CommandEvent`)

OnSize(event)

Event handler for the size of the control being changed.

Parameters `event` (`SizeEvent`)

PositionToXY(pos)

Converts given position to a zero-based column, line number pair.

Parameters `pos` (`int`) – Position

Return type `Tuple[int, int]`

SetBackgroundColour(colour)

Sets the background colour of the control

Parameters `colour` (`Colour`) – The colour to be used as the background colour; pass `wx.NullColour` to reset to the default colour.

Note: You may want to use `wx.SystemSettings.GetColour` to retrieve a suitable colour to use rather than setting an hard-coded one.

Return type `bool`

Returns `True` if the operation completes successfully, `False` otherwise.

SetClearBitmap (*bitmap*)

Sets the bitmap for the clear button.

Parameters **bitmap** (`Bitmap`)

SetDefaultStyle (*style*)

Changes the default style to use for the new text which is going to be added to the control.

This applies both to the text added programmatically using `WriteText()` or `AppendText()` and to the text entered by the user interactively.

If either of the font, foreground, or background colour is not set in style, the values of the previous default style are used for them. If the previous default style didn't set them either then the global font or colours of the text control itself are used as fall back.

However, if the style parameter is the default `wx.TextAttr`, then the default style is just reset (instead of being combined with the new style which wouldn't change it at all).

Parameters **style** (`TextAttr`) – The style for the new text

Return type `bool`

Returns `True` on success, `False` if an error occurred (this may also mean that the styles are not supported under this platform).

SetEditable (*editable*)

Makes the text item editable or read-only, overriding the `wx.TE_READONLY` flag.

Parameters **editable** (`bool`) – If `True`, the control should be editable. If `False`, the control should be read-only.

SetFont (*font*)

Sets the font for the control.

Parameters **font** (`Font`) – Font to associate with this control. Pass `wx.NullFont` to reset to the default font.

Return type `bool`

Returns `True` if the operation completes successfully, `False` otherwise.

SetInsertionPoint (*pos*)

Sets the insertion point at the given position.

Parameters **pos** (`int`) – Position to set, in the range from 0 to `GetLastPosition()` inclusive.

SetInsertionPointEnd ()

Sets the insertion point at the end of the text control.

This is equivalent to calling `SetInsertionPoint()` with `GetLastPosition()` as the argument.

SetMaxLength (*length*)

This function sets the maximum number of characters the user can enter into the control.

In other words, it allows limiting the text value length to len not counting the terminating NUL character.

If `len` is 0, the previously set max length limit, if any, is discarded, and the user may enter as much text as the underlying native text control widget supports (typically at least 32Kb).

If the user tries to enter more characters into the text control when it is already filled up to the maximal length, a `wx.EVT_TEXT_MAXLEN` event is sent to notify the program about it (giving it the possibility to show an explanatory message, for example) and the extra input is discarded.

Note that in wxGTK this function may only be used with single line text controls.

Parameters `length` (`int`)

SetModified (`modified`)

Marks the control as being modified by the user or not.

Parameters `modified` (`bool`)

SetStyle (`start`, `end`, `style`)

Changes the style of the given range.

If any attribute within style is not set, the corresponding attribute from `GetDefaultStyle` is used.

Parameters

- **start** (`int`) – The start of the range to change.
- **end** (`int`) – The end of the range to change.
- **style** (`TextAttr`) – The new style for the range.

Return type `bool`

Returns `True` on success, `False` if an error occurred (this may also mean the styles are not supported under this platform).

ShouldInheritColours ()

Return type `bool`

ShowPosition (`pos`)

Makes the line containing the given position visible.

Parameters `pos` (`int`) – The position that should be visible.

XYToPosition (`x`, `y`)

Converts the given zero-based column and line number to a position.

Parameters

- **x** (`int`) – The column number
- **y** (`int`) – The line number

Return type `int`

property default_clear_bitmap

Returns the default clear button bitmap for the control.

Return type `Bitmap`

```
textctrl
    Type: TextCtrl
```

```
clear_btn = [b'25 19 2 1', b' c None', b'+ c #000000', b' ', b' ', b' ', b' ', b' ', b' ', b' ',
    Type: List[bytes]
    XPM button icon for clearing the text control.
```

2.7 dialogs

Several dialog classes and helper functions for file/folder dialogs

Classes:

<i>FloatEntryDialog</i> (*args, **kwargs)	Alternative to wx.NumberEntryDialog that provides a TextCtrl which only allows numbers and decimal points to be entered.
<i>IntEntryDialog</i> (*args, **kwargs)	Alternative to wx.NumberEntryDialog that provides a TextCtrl which only allows numbers to be entered.
<i>Wildcards</i> ()	Class to generate glob wildcards for wx.FileDialog

Functions:

<i>file_dialog</i> (*args, **kwargs)	Create a wx.FileDialog with for the extension and file-typestring given, and return the filename selected.
<i>file_dialog_multiple</i> (parent, extension, ...)	Create a wx.FileDialog with the extension and file-typestring given, and return a list of the files selected.
<i>file_dialog_wildcard</i> (parent, title, wildcard)	Create a wx.FileDialog with the wildcard string given, and return a list of the files selected.

```
class FloatEntryDialog(*args, **kwargs)
```

```
    Bases: TextEntryDialog
```

Alternative to wx.NumberEntryDialog that provides a TextCtrl which only allows numbers and decimal points to be entered.

Based on <http://wxpython-users.1045709.n5.nabble.com/Adding-Validation-to-wx-TextEntryDialog-td2371082.html>

Methods:

<i>GetValue</i> ()

```
    GetValue()
```

```
class IntEntryDialog(*args, **kwargs)
```

```
    Bases: TextEntryDialog
```

Alternative to wx.NumberEntryDialog that provides a TextCtrl which only allows numbers to be entered.

Based on <http://wxpython-users.1045709.n5.nabble.com/Adding-Validation-to-wx-TextEntryDialog-td2371082.html>

Methods:

`GetValue()`

GetValue ()**class Wildcards**Bases: `object`Class to generate glob wildcards for `wx.FileDialog`**Methods:**

<code>__repr__()</code>	Return <code>repr(self)</code> .
<code>__str__()</code>	Return <code>str(self)</code> .
<code>add_all_files_wildcard([hint_format])</code>	Add a wildcard for 'All Files'.
<code>add_common_filetype(filetype[, hint_format, ...])</code>	Add a common filetype.
<code>add_filetype(description[, extensions, ...])</code>	Add a filetype to the wildcards
<code>add_image_wildcard([value_format])</code>	Add a wildcard for all image filetypes.

Attributes:

<code>wildcard</code>	Returns a string representing the wildcards for use in <code>wx.FileDialog</code> or <code>file_dialog_wildcards</code>
-----------------------	---

__repr__ ()
Return `repr(self)`.

__str__ ()
Return `str(self)`.

add_all_files_wildcard (*hint_format=0*)
Add a wildcard for 'All Files'.

Parameters **hint_format** (*int*) – How the hints should be formatted. Valid values are `None` and `style_hidden`. Default 0.

add_common_filetype (*filetype, hint_format=8, value_format=12*)
Add a common filetype.

Parameters

- **filetype** (*str*) – The name of the filetype, Possible values are in `common_filetypes`
- **hint_format** (*int*) – How the hints should be formatted. Default 8.
- **value_format** (*int*) – How the values should be formatted. Default 12.

Valid values for *hint_format* and *value_format* are `style_uppercase`, `style_lowercase` and `style_hidden`, which can be combined using the `|` operator.

add_filetype (*description, extensions=None, hint_format=8, value_format=12*)
Add a filetype to the wildcards

Parameters

- **description** (*str*) – Description of the filetype
- **extensions** (*Optional[Sequence[str]]*) – A list of valid file extensions for the filetype. Default *None*.
- **hint_format** (*int*) – How the hints should be formatted. Default 8.
- **value_format** (*int*) – How the values should be formatted. Default 12.

Valid values for *hint_format* and *value_format* are *style_uppercase*, *style_lowercase* and *style_hidden*, which can be combined using the *|* operator.

add_image_wildcard (*value_format=12*)

Add a wildcard for all image filetypes.

Parameters **value_format** (*int*) – How the values should be formatted. Default 12.

Valid values for *value_format* are *style_uppercase*, *style_lowercase* and *style_hidden*, which can be combined using the *|* operator.

property wildcard

Returns a string representing the wildcards for use in *wx.FileDialog* or *file_dialog_wildcards*

Return type *str*

file_dialog (**args, **kwargs*)

Create a *wx.FileDialog* with for the extension and filetypestring given, and return the filename selected.

Parameters

- **parent**
- **extension**
- **title**
- **filetypestring**
- **style**
- ****kwargs**

Return type *Optional[str]*

Returns The filename selected in the dialog. If *wx.FD_MULTIPLE* is in the style, this will be the first filename selected.

file_dialog_multiple (*parent, extension, title, filetypestring, style=6, **kwargs*)

Create a *wx.FileDialog* with the extension and filetypestring given, and return a list of the files selected.

Parameters

- **parent** (*Window*) – Parent window. Should not be *None*.
- **extension** (*str*)
- **title** (*str*)
- **filetypestring** (*str*)
- **style** – Default 6.

- ****kwargs**

Return type `Optional[List[str]]`

Returns List of filenames for the selected files. If `wx.FD_MULTIPLE` is not in the style, the list will contain only one element.

file_dialog_wildcard (*parent, title, wildcard, style=6, **kwargs*)

Create a `wx.FileDialog` with the wildcard string given, and return a list of the files selected.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **wildcard** (`str`)
- **title** (`str`)
- **style** (`int`) – Default 6.
- **kwargs**

Returns List of filenames for the selected files. If `wx.FD_MULTIPLE` is not in the style, the list will contain only one element

Return type list of str

2.8 editable_listbox

A Python implementation of `wx.EditableListBox`, a `ListBox` with editable items.

Available in two flavours:

- Vanilla, that accepts any string value; and
- **Numerical, that only accepts numerical values.** Those could be str, int, float or decimal.Decimal, but decimal.Decimal is used internally and is the type that will be returned.

Classes:

<code>CleverListCtrl</code> (parent[, id, pos, size, ...])	list control with auto-resizable column:
<code>EditableListBox</code> (parent[, id, label, pos, ...])	This class provides a composite control that lets the user easily enter and edit a list of strings.
<code>EditableNumericalListBox</code> (parent[, id, ...])	
type parent <code>Window</code>	

class CleverListCtrl (*parent, id=-1, pos=(-1, -1), size=(-1, -1), style=4, validator=<wx.Validator object>, name=b'listCtrl'*)

Bases: `ListCtrl`

list control with auto-resizable column:

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** (`int`) – Default -1.

- **pos** (`Point`) – Default `(-1, -1)`.
- **size** (`Size`) – Default `(-1, -1)`.
- **style** (`int`) – Default `4`.
- **validator** (`Validator`) – Default `<wx.Validator object at 0x7f0d47ba8b50>`.
- **name** (`str`) – Default `b'listCtrl'`.

Methods:

`CreateColumns()`

`OnSize(event)`

param event The wxPython event.

`SizeColumns()`

CreateColumns()**OnSize** (*event*)

Parameters **event** – The wxPython event.

SizeColumns()

```
class EditableListBox (parent, id=-1, label='', pos=(-1, -1), size=(-1, -1), style=1792,  
                        name=b'editableListBox')
```

Bases: `Panel`

This class provides a composite control that lets the user easily enter and edit a list of strings.

Styles supported:

- `wx.adv.EL_ALLOW_NEW` - Allow user to create new items.
- `wx.adv.EL_ALLOW_EDIT` - Allow user to edit text in the control.
- `wx.adv.EL_ALLOW_DELETE` - Allow user to delete text from the control.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** (`int`) – Default `-1`.
- **label** (`str`) – Default `''`.
- **pos** (`Point`) – Default `(-1, -1)`.
- **size** (`Size`) – Default `(-1, -1)`.
- **style** (`int`) – Default `1792`.
- **name** (`str`) – Default `b'editableListBox'`.

Methods:

<i>GetDelButton()</i>	Retrieves a reference to the BitmapButton that is used as the ‘delete’ button in the control.
<i>GetDownButton()</i>	Retrieves a reference to the BitmapButton that is used as the ‘down’ button in the control.
<i>GetEditButton()</i>	Retrieves a reference to the BitmapButton that is used as the ‘edit’ button in the control.
<i>GetListCtrl()</i>	Returns a reference to the actual list control portion of the custom control.
<i>GetNewButton()</i>	Retrieves a reference to the BitmapButton that is used as the ‘new’ button in the control.
<i>GetStrings()</i>	Returns a list of the current contents of the control.
<i>GetUpButton()</i>	Retrieves a reference to the BitmapButton that is used as the ‘up’ button in the control.
<i>OnBeginLabelEdit(event)</i>	
	param event The wxPython event.
<i>OnDelItem(_)</i>	
<i>OnDownItem(_)</i>	
<i>OnEditItem(_)</i>	
<i>OnEndLabelEdit(event)</i>	
	param event The wxPython event.
<i>OnItemActivated(evt)</i>	
<i>OnItemSelected(event)</i>	
<i>OnNewItem(event)</i>	
	param event The wxPython event.
<i>OnUpItem(_)</i>	
<i>SetStrings(strings)</i>	Replaces current contents with given strings.
<i>SetupEditControl()</i>	
<i>SwapItems(i1, i2)</i>	
	type i1 <code>int</code>
<i>on_value_changed(event)</i>	

GetDelButton ()

Retrieves a reference to the BitmapButton that is used as the ‘delete’ button in the control.

GetDownButton ()

Retrieves a reference to the BitmapButton that is used as the ‘down’ button in the control.

GetEditButton ()

Retrieves a reference to the BitmapButton that is used as the ‘edit’ button in the control.

GetListCtrl ()

Returns a reference to the actual list control portion of the custom control.

Returns

Return type `wx.ListCtrl`

GetNewButton ()

Retrieves a reference to the BitmapButton that is used as the ‘new’ button in the control.

GetStrings ()

Returns a list of the current contents of the control.

Returns list of strings

Return type list of str

GetUpButton ()

Retrieves a reference to the BitmapButton that is used as the ‘up’ button in the control.

OnBeginLabelEdit (event)

Parameters **event** – The wxPython event.

OnDelItem (_)

OnDownItem (_)

OnEditItem (_)

OnEndLabelEdit (event)

Parameters **event** – The wxPython event.

OnItemActivated (evt)

OnItemSelected (event)

OnNewItem (event)

Parameters **event** – The wxPython event.

OnUpItem (_)

SetStrings (strings)

Replaces current contents with given strings.

Parameters **strings** (`List[str]`) – list of strings.

SetupEditControl ()

SwapItems (i1, i2)

Parameters

- **i1** (`int`)
- **i2** (`int`)

`on_value_changed` (*event*)

class EditableNumericalListBox (*parent*, *id*=- 1, *label*="", *decimal_places*=- 1, *pos*=(- 1, - 1), *size*=(- 1, - 1), *style*=1792, *name*=b'editableListBox')

Bases: *EditableListBox*

Parameters

- **parent** (*Window*) – Parent window. Should not be *None*.
- **id** (*int*) – Default -1.
- **label** (*str*) – Default ''.
- **pos** (*Point*) – Default (-1, -1).
- **size** (*Size*) – Default (-1, -1).
- **style** (*int*) – Default 1792.
- **name** (*str*) – Default b'editableListBox'.

Methods:

<i>GetDecimalPlaces()</i>	
<i>GetStrings()</i>	Returns a list of the current contents of the control.
<i>GetValues()</i>	Returns a list of the current contents of the control.
<i>SetDecimalPlaces</i> (<i>_decimal_places</i>)	
<i>SetStrings</i> (<i>strings</i>)	Replaces current contents with given strings.
<i>SetValues</i> (<i>values</i>)	Replaces current contents with given values.
<i>SetupEditControl()</i>	
<i>on_value_changed</i> (<i>event</i>)	

Attributes:

<i>decimal_places</i>

GetDecimalPlaces ()

GetStrings ()

Returns a list of the current contents of the control.

Returns list of strings

Return type list of str

GetValues ()

Returns a list of the current contents of the control.

Returns

Return type

SetDecimalPlaces (*_decimal_places*)

SetStrings (*strings*)

Replaces current contents with given strings.

Parameters **strings** – list of strings.

SetValues (*values*)

Replaces current contents with given values.

Parameters **values** (*Sequence[Union[float, Decimal]]*) – list of values

SetupEditControl ()

property **decimal_places**

on_value_changed (*event*)

2.9 events

Reusable code for simple events

Usage:

```
>>> from domdf_wxpython_tools import events
```

```
>>> # Create the event outside of the class
>>> myEVT = events.SimpleEvent()
```

```
>>> # Set the receiver to the class you want.
>>> # This is usually done from within the receiver class
>>> class MyClass(object):
...     def __init__(self):
...         myEVT.set_receiver(self)
...
...     # Then bind the event to a handler
...     myEVT.Bind(self.handler)
...
...     def handler(self):
...         '''Handler for myEVT'''
...         pass
```

```
>>> # From within the thread, trigger the event with the following syntax:
```

```
>>> myEVT.trigger()
```

Classes:

<i>PayloadEvent</i> (etype, eid, value)	Event containing a message payload.
---	-------------------------------------

<i>SimpleEvent</i> ([receiver, name])

param receiver

class **PayloadEvent** (*etype, eid, value*)

Bases: *PyCommandEvent*

Event containing a message payload.

Methods:

<code>GetValue()</code>	Returns the value from the event.
-------------------------	-----------------------------------

GetValue()

Returns the value from the event. @return: the value of this event

class SimpleEvent (*receiver=None, name='Event'*)

Bases: `object`

Parameters

- **receiver** – Default `None`.
- **name** – Default `'Event'`.

Methods:

<code>Bind(handler[, receiver])</code>	Bind the event to the handler.
<code>Unbind([receiver])</code>	Unbind the event from the handler
<code>__repr__()</code>	Return a nicely formatted representation string.
<code>set_receiver(receiver)</code>	Set the class that is to receive the event trigger.
<code>trigger([value])</code>	Trigger the event

Bind (*handler, receiver=None, **kwargs*)

Bind the event to the handler.

Parameters

- **handler** – handler to bind the event to.
- **kwargs** – keyword arguments to pass through to receiver's Bind method.

Unbind (*receiver=None, **kwargs*)

Unbind the event from the handler

Parameters **kwargs** – keyword arguments to pass through to receiver's Unbind method

`__repr__()`

Return a nicely formatted representation string.

set_receiver (*receiver*)

Set the class that is to receive the event trigger.

Parameters **receiver**

trigger (*value=None*)

Trigger the event

2.10 filebrowsectrl

Classes:

<code>DirBrowseCtrl</code> (parent[, id, pos, size, ...])	
<code>FileBrowseCtrl</code> (parent[, id, pos, size, ...])	A control to allow the user to type in a filename or browse with the standard file dialog to select file.
<code>FileBrowseCtrlWithHistory</code> (*arguments, ...)	with following additions:

```
class DirBrowseCtrl (parent, id=- 1, pos=(- 1, - 1), size=(- 1, - 1), style=537401408, labelText='Select
a directory:', buttonText='Browse', toolTip='Type directory name or browse to
select', dialogTitle="", initialValue=None, changeCallback=None,
name='DirBrowseCtrl')
```

Bases: `FileBrowseCtrl`

Methods:

<code>OnBrowse</code> ([ev])	Going to browse for file...
------------------------------	-----------------------------

Attributes:

—

OnBrowse (ev=None)
Going to browse for file...

textctrl
Type: `TextCtrl`

```
class FileBrowseCtrl (parent, id=-1, pos=(-1, -1), size=(-1, -1), style=524289, labelText='File
Entry:', buttonText='Browse', toolTip='Type a filename or click the browse
button to choose a file', dialogTitle='Choose a file', initialValue="",
changeCallback=<function ' <lambda>'>, labelWidth=0,
name='fileBrowseButton', show_cancel_btn=True, fileMask='All files
(*.*)*.*', dialog_title='File Picker', **kwargs)
```

Bases: `TextCtrlWrapper`, `FileBrowseButton`

A control to allow the user to type in a filename or browse with the standard file dialog to select file.

Based on and subclassed from `wx.lib.filebrowsebutton.FileBrowseButton` but with a `wx.SearchCtrl` in place of the `wx.TextCtrl` to provide the cancel/clear button and with an icon on the browse button.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** (`int`) – Control identifier. A value of `-1` denotes a default value. Default `-1`.
- **pos** (`Point`) – Control position. Default `(-1, -1)`.
- **size** (`Size`) – Control size. Default `(-1, -1)`.
- **style** (`int`) – Window style. See `wx.Window` and `ClearableTextCtrl` for supported styles. Default `524289`.
- **labelText** (`str`) – Text for label to left of text field. Default `'File Entry: '`.

- **buttonText** (*str*) – Text for button which launches the file dialog. Default 'Browse'.
- **toolTip** (*str*) – Help text. Default 'Type a filename or click the browse button to choose a file'.
- **dialogTitle** (*str*) – Title used in file dialog. Default 'Choose a file'.
- **initialValue** (*str*) – The initial value of the TextCtrl. Default ''.
- **changeCallback** – Optional callback called for all changes in value of the control. Default `domdf_wxpython_tools.filebrowsectrl..`
- **labelWidth** – Width of the label. Default 0.
- **name** – Default 'fileBrowseButton'.
- **show_cancel_btn** (*bool*) – Whether to show or hide the cancel button. Default `True`.
- **dialog_title** (*str*) – The title of the FileDialog. Default 'File Picker'.
- **fileMask** (*str*) – File mask (glob pattern, such as *.*) to use in file dialog. See `wx.FileDialog` for more information. Default 'All files (*.*)|*.*'.

Methods:

<i>ChangeValue</i> (value)	Sets the new text control value.
<i>GetLabel</i> ()	Retrieve the label's current text.
<i>GetLineLength</i> (lineNo)	Gets the length of the specified line, not including any trailing newline character(s).
<i>GetLineText</i> (lineNo)	Returns the contents of a given line in the text control, not including any trailing newline character(s).
<i>GetNumberOfLines</i> ()	Returns the number of lines in the text control buffer.
<i>GetRange</i> (from_, to_)	Returns the string containing the text starting in the positions from and up to in the control.
<i>IsEditable</i> ()	Returns <code>True</code> if the controls contents may be edited by user (note that it always can be changed by the program).
<i>IsModified</i> ()	Returns <code>True</code> if the text has been modified by user.
<i>IsMultiLine</i> ()	Returns <code>True</code> if this is a multi line edit control and <code>False</code> otherwise.
<i>IsSingleLine</i> ()	Returns <code>True</code> if this is a single line edit control and <code>False</code> otherwise.
<i>MarkDirty</i> ()	Mark text as modified (dirty).
<i>OnBrowse</i> ([event])	Going to browse for file...
<i>SetLabel</i> (value)	Set the label's current text.
<i>SetModified</i> (modified)	Marks the control as being modified by the user or not.
<i>SetValue</i> (value[, callBack])	Sets the new text control value.
<i>createBrowseButton</i> ()	Create the browse-button control.
<i>createTextControl</i> ()	Create the text control.

Attributes:

ChangeValue (*value*)
Sets the new text control value.

It also marks the control as not-modified which means that `IsModified()` would return `False` immediately after the call to `ChangeValue` .

The insertion point is set to the start of the control (i.e. position 0) by this function.

This functions does not generate the `wxEVT_TEXT` event but otherwise is identical to `SetValue` .

Parameters `value` (`str`) – The new value to set. It may contain newline characters if the text control is multi-line.

GetLabel ()

Retrieve the label's current text.

GetLineLength (lineNo)

Gets the length of the specified line, not including any trailing newline character(s).

Parameters `lineNo` (`int`) – Line number (starting from zero).

Return type `int`

Returns The length of the line, or `-1` if `lineNo` was invalid.

GetLineText (lineNo)

Returns the contents of a given line in the text control, not including any trailing newline character(s).

Parameters `lineNo` (`int`) – Line number (starting from zero).

Return type `str`

Returns The contents of the line.

GetNumberOfLines ()

Returns the number of lines in the text control buffer.

Returns

Return type `int`

GetRange (from_, to_)

Returns the string containing the text starting in the positions `from` and up to in the control.

The positions must have been returned by another `wx.TextCtrl` method.

Parameters

- `from_` (`int`)
- `to_` (`int`)

Return type `str`

IsEditable ()

Returns `True` if the controls contents may be edited by user (note that it always can be changed by the program).

In other words, this functions returns `True` if the control hasn't been put in read-only mode by a previous call to `SetEditable` .

Return type `bool`

IsModified()

Returns `True` if the text has been modified by user.

Note that calling `SetValue` doesn't make the control modified.

Returns

Return type `bool`

IsMultiLine()

Returns `True` if this is a multi line edit control and `False` otherwise.

Returns

Return type `bool`

IsSingleLine()

Returns `True` if this is a single line edit control and `False` otherwise.

Returns

Return type `bool`

MarkDirty()

Mark text as modified (dirty).

Returns

Return type

OnBrowse (event=None)

Going to browse for file...

SetLabel (value)

Set the label's current text.

SetModified (modified)

Marks the control as being modified by the user or not.

Parameters **modified** (`bool`)

SetValue (value, callBack=1)

Sets the new text control value.

It also marks the control as not-modified which means that `IsModified()` would return `False` immediately after the call to `SetValue`.

The insertion point is set to the start of the control (i.e. position 0) by this function unless the control value doesn't change at all, in which case the insertion point is left at its original position.

Note that, unlike most other functions changing the controls values, this function generates a `wx-EVT_TEXT` event. To avoid this you can use `ChangeValue` instead.

Parameters: `value` (string) – The new value to set. It may contain newline characters if the text control is multi-line.

createBrowseButton()

Create the browse-button control.

createTextControl()
Create the text control.

textctrl
Type: `TextCtrl`

class FileBrowseCtrlWithHistory (*arguments, **namedarguments)
Bases: `FileBrowseCtrl`

with following additions: `__init__(..., history=None)`

history – optional list of paths for initial history drop-down (must be passed by name, not a positional argument) If history is callable it will must return a list used for the history drop-down

changeCallback – as for `FileBrowseCtrl`, but with a work-around for win32 systems which don't appear to create `wx.EVT_COMBOBOX` events properly. There is a (slight) chance that this work-around will cause some systems to create two events for each Combobox selection. If you discover this condition, please report it!

As for a `FileBrowseCtrl.__init__` otherwise.

GetHistoryControl() Return reference to the control which implements interfaces required for manipulating the history list. See `GetHistoryControl` documentation for description of what that interface is.

GetHistory() Return current history list

SetHistory(value=(), selectedIndex = None) Set current history list, if `selectedIndex` is not `None`, select that index

Methods:

<code>GetHistory()</code>	Return the current history list
<code>GetHistoryControl()</code>	Return a pointer to the control which provides (at least) the following methods for manipulating the history list:
<code>OnSetFocus(event)</code>	When the history scroll is selected, update the history
<code>SetHistory([value, selectedIndex, control])</code>	Set the current history list
<code>createTextControl()</code>	Create the text control.

Attributes:

GetHistory()
Return the current history list

GetHistoryControl()
Return a pointer to the control which provides (at least) the following methods for manipulating the history list:

`Append(item)` – add item `Clear()` – clear all items `Delete(index)` – 0-based index to delete from list `SetSelection(index)` – 0-based index to select in list

Semantics of the methods follow those for the `wxComboBox` control

OnSetFocus (*event*)

When the history scroll is selected, update the history

SetHistory (*value=()*, *selectionIndex=None*, *control=None*)

Set the current history list

createTextControl ()

Create the text control.

textctrl

Type: `TextCtrl`

2.11 icons

Functions:

<code>GetStockBitmap(art_id[, art_client])</code>	Get a stock bitmap from its wx.ART_xxx ID
<code>GetStockToolbarBitmap(art_id)</code>	
<code>get_button_icon(icon_name[, size])</code>	
<code>get_toolbar_icon(icon_name[, size])</code>	

GetStockBitmap (*art_id*, *art_client=None*)

Get a stock bitmap from its wx.ART_xxx ID

GetStockToolbarBitmap (*art_id*)

get_button_icon (*icon_name*, *size=24*)

get_toolbar_icon (*icon_name*, *size=24*)

2.12 imagepanel

Based on ChartPanelBase, a canvas for displaying an image within a wxPython window using PIL and matplotlib, with a right click menu with some basic options

Classes:

<code>EvtImgPanelChanged(windowID, obj)</code>	Custom Event for an image in the ImagePanel being changed.
<code>ImagePanel(parent[, image, id, pos, size, ...])</code>	Panel that contains a matplotlib plotting window, used for displaying an image.

class EvtImgPanelChanged (*windowID*, *obj*)

Bases: `PyCommandEvent`

Custom Event for an image in the ImagePanel being changed.

Parameters

- **windowID** (*int*)
- **obj**

Attributes:

eventType

eventType = 0
Type: *int*

class ImagePanel (*parent, image=None, id=- 1, pos=(- 1, - 1), size=(- 1, - 1), style=0, name=b'panel'*)
Bases: *ChartPanelBase*

Panel that contains a matplotlib plotting window, used for displaying an image. The image can be right clicked to bring up a context menu allowing copying, pasting and saving of the image. The image can be panned by holding the left mouse button and moving the mouse, and zoomed in and out using the scrollwheel on the mouse.

Parameters

- **parent** (*Window*) – The parent window.
- **image** – Default *None*.
- **id** (*int*) – An identifier for the panel. *wx.ID_ANY* is taken to mean a default. Default *-1*.
- **pos** (*Point*) – The panel position. The value *wx.DefaultPosition* indicates a default position, chosen by either the windowing system or *wxWidgets*, depending on platform. Default *(-1, -1)*.
- **size** (*Size*) – The panel size. The value *::wxDefaultSize* indicates a default size, chosen by either the windowing system or *wxWidgets*, depending on platform. Default *(-1, -1)*.
- **style** (*int*) – The window style. See *wxPanel*. Default *0*.
- **name** (*str*) – Window name. Default *b'panel'*.

Methods:

<i>clear</i> ([<i>event</i>])	Clear the image from the control
<i>copy</i> ([_])	Copy the image to the clipboard.
<i>load_image</i> ([<i>new_image</i> , <i>suppress_event</i>])	Load the 'new_image' into the control.
<i>on_context_menu</i> (<i>event</i>)	Event Handler for bringing up right click context menu
<i>on_load</i> ([<i>event</i>])	Load the image into the dialog from the file selected in the dialog
<i>on_save</i> ([<i>event</i>])	Save the image to the location selected in the dialog
<i>paste</i> ([<i>event</i>])	Paste the image from the clipboard into the control.
<i>reset_view</i> (*_)	Reset the view of the image.

Attributes:

default_image

image Returns the image being displayed in the control

clear (*event=None*)

Clear the image from the control

copy (*_=None*)

Copy the image to the clipboard.

default_image = ('RGB', (640, 480), (240, 240, 240))

Type: `tuple`

property image

Returns the image being displayed in the control

Return type `PIL.Image.Image`

load_image (*new_image=None, suppress_event=False*)

Load the 'new_image' into the control.

Parameters

- **new_image** (`Union[Image, None, str, Path, PathLike]`) – The image to load, or a string pointing to the image on a filesystem. Default `None`.
- **suppress_event** (`bool`) – Whether the event that the image has changed should be suppressed. Default `False`.

on_context_menu (*event*)

Event Handler for bringing up right click context menu

on_load (*event=None*)

Load the image into the dialog from the file selected in the dialog

on_save (*event=None*)

Save the image to the location selected in the dialog

paste (*event=None*)

Paste the image from the clipboard into the control.

reset_view (**_*)

Reset the view of the image.

2.13 keyboard

Functions:

`gen_keymap()`

gen_keymap ()

2.14 list_dialog

Classes:

<code>list_dialog</code> (parent[, id, title, label, ...])	A dialog containing a <code>wx.ListBox</code> .
--	---

```
class list_dialog (parent, id=- 1, title='Choose', label='Choose: ', choices=None, pos=(- 1, - 1),
                    size=(- 1, - 1), style=536877056, name=b'dialog')
```

Bases: `Dialog`

A dialog containing a `wx.ListBox`.

Parameters

- **parent** (`Window`) – The parent window. Can be `None`, a frame or another dialog box
- **id** (`int`) – An identifier for the dialog. `wx.ID_ANY` is taken to mean a default. Default `-1`.
- **title** (`str`) – The title of the dialog. Default `'Choose'`.
- **label** (`str`) – The label for the `wx.ListBox`. Default `'Choose: '`.
- **choices** (`Optional[List[str]]`) – A list of choices for the `wx.ListBox`. Default `None`.
- **pos** (`Point`) – The dialog position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **size** (`Size`) – The dialog size. The value `::wxDefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default `(-1, -1)`.
- **style** (`int`) – The window style. See `wxPanel`. Default `536877056`.
- **name** (`str`) – Window name. Default `b'dialog'`.

Methods:

<code>do_cancel</code> (_)	Event Handler for dialog being cancelled
<code>do_select</code> (_)	Event Handler for item in list being selected

do_cancel (_)
Event Handler for dialog being cancelled

do_select (_)
Event Handler for item in list being selected

2.15 logctrl

Log Control, supporting text copying and zoom.

Classes:

<i>LogCtrl</i> (parent[, id, pos, size, style, name])	Log Window based on StyledTextCtrl.
---	-------------------------------------

class LogCtrl (parent, id=-1, pos=(-1, -1), size=(-1, -1), style=138412032, name='Log')

Bases: *StyledTextCtrl*

Log Window based on *StyledTextCtrl*.

Parameters

- **parent** (*Window*) – The parent window.
- **id** (*int*) – An identifier for the Log window. `wx.ID_ANY` is taken to mean a default. Default -1.
- **pos** (*Point*) – The Log window position. The value `wx.DefaultPosition` indicates a default position, chosen by either the windowing system or `wxWidgets`, depending on platform. Default (-1, -1).
- **size** (*Size*) – The Log window size. The value `::wxDefaultSize` indicates a default size, chosen by either the windowing system or `wxWidgets`, depending on platform. Default (-1, -1).
- **style** (*int*) – The window style. See `wxPanel`. Default 138412032.
- **name** (*str*) – Window name. Default 'Log'.

Methods:

<i>Append</i> (text[, c])	Add the text to the end of the control using colour <code>c</code> which should be suitable for feeding directly to <code>wx.NamedColour</code> .
<i>AppendStderr</i> (text)	Add the stderr text to the end of the control using colour “red”
<i>CanCopy</i> ()	Returns <code>True</code> if text is selected and can be copied, <code>False</code> otherwise.
<i>Copy</i> ()	Copy the selection and place it on the clipboard.
<i>DoFindNext</i> (findData[, findDlg])	
<i>GetContextMenu</i> ()	Create and return a context menu for the log.
<i>GetLastPosition</i> ()	
<i>GetRange</i> (start, end)	
<i>GetSelection</i> ()	
<i>OnContextMenu</i> (_)	vent Handler for showing the context menu
<i>OnFindClose</i> (_)	
<i>OnFindText</i> (*_)	
<i>OnZoomDefault</i> (*_)	Event Handler for resetting the zoom
<i>OnZoomIn</i> (*_)	Event Handler for zooming in
<i>OnZoomOut</i> (*_)	Event Handler for zooming out
<i>ShowPosition</i> (pos)	
<i>ToggleLineNumbers</i> (*_)	
<i>ToggleWrap</i> (*_)	Toggle word wrap.

continues on next page

Table 47 – continued from previous page

<code>bufferHasChanged()</code>	
<code>bufferSave()</code>	
<code>fixLineEndings(text)</code>	Return text with line endings replaced by OS-specific endings.
<code>getStyle([c])</code>	Returns a style for a given colour if one exists.
<code>onKeyPress(event)</code>	Event Handler for key being pressed.
<code>setDisplayLineNumbers(state)</code>	
<code>setStyles(faces)</code>	Configure font size, typeface and color for lexer.
<code>wrap([wrap])</code>	Set whether text is word wrapped.
<code>write(text)</code>	Display text in the log.

Attributes:

Append (*text*, *c=None*)

Add the text to the end of the control using colour *c* which should be suitable for feeding directly to `wx.NamedColour`.

Parameters

- **text** – ... Should be a unicode string or contain only ascii data.
- **c** – Default `None`.

AppendStderr (*text*)

Add the stderr text to the end of the control using colour “red”

Parameters **text** – ... Should be a unicode string or contain only ascii data.

CanCopy ()

Returns `True` if text is selected and can be copied, `False` otherwise.

Return type `bool`

Copy ()

Copy the selection and place it on the clipboard.

DoFindNext (*findData*, *findDlg=None*)**GetContextMenu** ()

Create and return a context menu for the log. This is used instead of the scintilla default menu in order to correctly respect our immutable buffer.

GetLastPosition ()**GetRange** (*start*, *end*)**GetSelection** ()**OnContextMenu** (_)

vent Handler for showing the context menu

OnFindClose (*_*)

OnFindText (**_*)

OnZoomDefault (**_*)

Event Handler for resetting the zoom

OnZoomIn (**_*)

Event Handler for zooming in

OnZoomOut (**_*)

Event Handler for zooming out

ShowPosition (*pos*)

ToggleLineNumbers (**_*)

ToggleWrap (**_*)

Toggle word wrap.

bufferHasChanged ()

bufferSave ()

findDlg

Type: `Optional[FindReplaceDialog]`

fixLineEndings (*text*)

Return text with line endings replaced by OS-specific endings.

Parameters *text*

getStyle (*c='black'*)

Returns a style for a given colour if one exists.

If no style exists for the colour, make a new style.

If we run out of styles, (only 32 allowed here) we go to the top of the list and reuse previous styles.

Parameters *c* (*str*) – Default 'black'.

onKeyPress (*event*)

Event Handler for key being pressed.

setDisplayLineNumbers (*state*)

setStyles (*faces*)

Configure font size, typeface and color for lexer.

Parameters *faces*

wrap (*wrap=True*)

Set whether text is word wrapped.

Parameters `wrap` (`bool`) – Whether the text should be word wrapped. Default `True`.

write (`text`)

Display text in the log.

Replace line endings with OS-specific endings.

Parameters `text`

2.16 picker

Classes:

`dir_picker`(parent[, id, value, pos, size, ...])

type parent `Window`

`file_folder_picker`(parent[, id, value, pos, ...])

type parent `Window`

`file_picker`(parent[, id, value, pos, size, ...])

type parent `Window`

class `dir_picker` (`parent`, `id=-1`, `value=""`, `pos=(-1, -1)`, `size=(-1, -1)`, `style=524288`,
 `name=b'dir_picker'`)

Bases: `TextCtrlWrapper`, `Panel`

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** – Default `-1`.
- **value** – Default `''`.
- **pos** – Default `(-1, -1)`.
- **size** – Default `(-1, -1)`.
- **style** – Default `524288`.
- **name** – Default `b'dir_picker'`.

Methods:

<code>Browse(*_)</code>	
<code>Clear(*_)</code>	Clears the text in the control.
<code>GetInsertionPoint()</code>	Returns the insertion point, or cursor, position.
<code>ResetValue()</code>	Resets the text ctrl to the initial value, either specified in <code>__init__</code> or set with <code>SetInitialValue</code> .
<code>SetHeight(height)</code>	Set the height of the widgets
<code>SetInitialValue(value)</code>	Sets the initial value for the text ctrl.

continues on next page

Table 50 – continued from previous page

<i>SetInsertionPoint</i> (pos)	Sets the insertion point at the given position.
<i>SetInsertionPointEnd</i> ()	Sets the insertion point at the end of the text control.
<i>SetTextWidth</i> (width)	Sets the width of the TextCtrl.
<i>get_value</i> ()	
<i>reset_value</i> ()	
<i>set_height</i> (height)	
<i>set_textctrl_width</i> (width)	

Attributes:**Browse** (*_)**Clear** (*_)

Clears the text in the control.

Note that this function will generate a wxEVT_TEXT event, i.e. its effect is identical to calling SetValue("").

GetInsertionPoint ()

Returns the insertion point, or cursor, position.

This is defined as the zero based index of the character position to the right of the insertion point. For example, if the insertion point is at the end of the single-line text control, it is equal to GetLastPosition .

Return type long

ResetValue ()

Resets the text ctrl to the initial value, either specified in __init__ or set with SetInitialValue.

SetHeight (height)

Set the height of the widgets

Parameters **height** (int) – Height of the widgets

SetInitialValue (value)

Sets the initial value for the text ctrl.

Parameters **value** (str) – Initial value for the text ctrl.

SetInsertionPoint (pos)

Sets the insertion point at the given position.

Parameters **pos** (int) – Position to set, in the range from 0 to *GetLastPosition()* inclusive.

SetInsertionPointEnd ()

Sets the insertion point at the end of the text control.

This is equivalent to calling wx.TextCtrl.SetInsertionPoint with wx.TextCtrl.GetLastPosition argument.

SetTextWidth (width)

Sets the width of the TextCtrl.

Parameters **width** (*int*) – The width of the TextCtrl.

get_value()

reset_value()

set_height (*height*)

set_textctrl_width (*width*)

textctrl

Type: `TextCtrl`

class file_folder_picker (*parent*, *id*=- 1, *value*="", *pos*=(- 1, - 1), *size*=(- 1, - 1), *style*=524294, *name*=b'file_picker', *extension*='*', *title*='File Picker', *filetypestring*='All Files', *start_as_files*=True, ***kwargs*)

Bases: `dir_picker`

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** – Default -1.
- **value** – Default ''.
- **pos** – Default (-1, -1).
- **size** – Default (-1, -1).
- **style** – Default 524294.
- **name** – Default b'file_picker'.
- **extension** – Default '*'.
- **title** – Default 'File Picker'.
- **filetypestring** – Default 'All Files'.
- **kwargs**

Methods:

`Browse(*_)`

`set_files_mode()`

`set_folders_mode()`

`toggle_mode()`

Attributes:

—

Browse (*_)

set_files_mode ()

set_folders_mode ()


```
textctrl
    Type: TextCtrl
```

```
toggle_mode()
```

```
class file_picker(parent, id=-1, value="", pos=(-1, -1), size=(-1, -1), style=524294,
                  name=b'file_picker', extension='*', title='File Picker', filetypestring='All Files',
                  **kwargs)
```

Bases: `dir_picker`

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** – Default `-1`.
- **value** – Default `' '`.
- **pos** – Default `(-1, -1)`.
- **size** – Default `(-1, -1)`.
- **style** – Default `524294`.
- **name** – Default `b'file_picker'`.
- **extension** – Default `'*'`.
- **title** – Default `'File Picker'`.
- **filetypestring** – Default `'All Files'`.
- **kwargs**

Methods:

```
Browse(*_)
```

Attributes:

—

```
Browse(*_)
```

```
textctrl
    Type: TextCtrl
```

2.17 projections

Classes:

<code>NoZoom(fig, *args[, facecolor, frameon, ...])</code>	Prevent zooming in pan mode.
<code>XPanAxes(fig, *args[, facecolor, frameon, ...])</code>	Constrain pan to x-axis.
<code>XPanAxes_NoZoom(fig, *args[, facecolor, ...])</code>	Constrain pan to x-axis and prevent zooming.

```
class NoZoom (fig, *args, facecolor=None, frameon=True, sharex=None, sharey=None, label="",
              xscale=None, yscale=None, box_aspect=None, **kwargs)
```

Bases: `Axes`

Prevent zooming in pan mode.

Methods:

```
drag_pan(button, key, x, y)
```

param button The pressed mouse button.

```
set(*[, adjustable, agg_filter, alpha, ...])
```

Set multiple properties at once.

Attributes:

```
name
```

drag_pan (button, key, x, y)

Parameters

- **button** – The pressed mouse button.
- **key** (`Optional[str]`) – The pressed key, if any.
- **x** (`float`) – The mouse coordinates in display coords.
- **y** (`float`) – The mouse coordinates in display coords.

name = 'NoZoom'

Type: `str`

```
set (*, adjustable=<UNSET>, agg_filter=<UNSET>, alpha=<UNSET>, anchor=<UNSET>,
      animated=<UNSET>, aspect=<UNSET>, autoscale_on=<UNSET>, autoscalex_on=<UNSET>,
      autoscaley_on=<UNSET>, axes_locator=<UNSET>, axisbelow=<UNSET>,
      box_aspect=<UNSET>, clip_box=<UNSET>, clip_on=<UNSET>, clip_path=<UNSET>,
      facecolor=<UNSET>, frame_on=<UNSET>, gid=<UNSET>, in_layout=<UNSET>,
      label=<UNSET>, mouseover=<UNSET>, navigate=<UNSET>, path_effects=<UNSET>,
      picker=<UNSET>, position=<UNSET>, prop_cycle=<UNSET>,
      rasterization_zorder=<UNSET>, rasterized=<UNSET>, sketch_params=<UNSET>,
      snap=<UNSET>, subplotspec=<UNSET>, title=<UNSET>, transform=<UNSET>,
      url=<UNSET>, visible=<UNSET>, xbound=<UNSET>, xlabel=<UNSET>, xlim=<UNSET>,
      xmargin=<UNSET>, xscale=<UNSET>, xticklabels=<UNSET>, xticks=<UNSET>,
      ybound=<UNSET>, ylabel=<UNSET>, ylim=<UNSET>, ymargin=<UNSET>,
      yscale=<UNSET>, yticklabels=<UNSET>, yticks=<UNSET>, zorder=<UNSET>)
```

Set multiple properties at once.

Supported properties are

Properties: adjustable: {'box', 'datalim'} agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array and two offsets from the bottom left corner of the image alpha: scalar or None anchor: (float, float) or {'C', 'SW', 'S', 'SE', 'E', 'NE', ...} animated: bool aspect: {'auto', 'equal'} or float autoscale_on: bool autoscalex_on: unknown autoscaley_on: unknown axes_locator: Callable[[Axes, Renderer], Bbox] axisbelow: bool or 'line' box_aspect: float or None clip_box: *.Bbox* clip_on: bool clip_path: Patch or (Path, Transform) or None facecolor:

or fc: color figure: *.Figure* frame_on: bool gid: str in_layout: bool label: object mouseover: bool navigate: bool navigate_mode: unknown path_effects: *.AbstractPathEffect* picker: None or bool or float or callable position: [left, bottom, width, height] or *~matplotlib.transforms.Bbox* prop_cycle: unknown rasterization_zorder: float or None rasterized: bool sketch_params: (scale: float, length: float, randomness: float) snap: bool or None subplotspec: unknown title: str transform: *.Transform* url: str visible: bool xbound: unknown xlabel: str xlim: (bottom: float, top: float) xmargin: float greater than -0.5 xscale: unknown xticklabels: unknown xticks: unknown ybound: unknown ylabel: str ylim: (bottom: float, top: float) ymargin: float greater than -0.5 yscale: unknown yticklabels: unknown yticks: unknown zorder: float

```
class XPanAxes (fig, *args, facecolor=None, frameon=True, sharex=None, sharey=None, label="",
               xscale=None, yscale=None, box_aspect=None, **kwargs)
```

Bases: *Axes*

Constrain pan to x-axis.

Methods:

```
drag_pan(button, key, x, y)
```

param button The pressed mouse button.

```
set(*[, adjustable, agg_filter, alpha, ...])
```

Set multiple properties at once.

Attributes:

```
name
```

drag_pan (*button, key, x, y*)

Parameters

- **button** – The pressed mouse button.
- **key** (*Optional[str]*) – The pressed key, if any.
- **x** (*float*) – The mouse coordinates in display coords.
- **y** (*float*) – The mouse coordinates in display coords.

```
name = 'XPanAxes'
```

Type: *str*

```

set (*, adjustable=<UNSET>, agg_filter=<UNSET>, alpha=<UNSET>, anchor=<UNSET>,
    animated=<UNSET>, aspect=<UNSET>, autoscale_on=<UNSET>, autoscalex_on=<UNSET>,
    autoscaley_on=<UNSET>, axes_locator=<UNSET>, axisbelow=<UNSET>,
    box_aspect=<UNSET>, clip_box=<UNSET>, clip_on=<UNSET>, clip_path=<UNSET>,
    facecolor=<UNSET>, frame_on=<UNSET>, gid=<UNSET>, in_layout=<UNSET>,
    label=<UNSET>, mouseover=<UNSET>, navigate=<UNSET>, path_effects=<UNSET>,
    picker=<UNSET>, position=<UNSET>, prop_cycle=<UNSET>,
    rasterization_zorder=<UNSET>, rasterized=<UNSET>, sketch_params=<UNSET>,
    snap=<UNSET>, subplotspec=<UNSET>, title=<UNSET>, transform=<UNSET>,
    url=<UNSET>, visible=<UNSET>, xbound=<UNSET>, xlabel=<UNSET>, xlim=<UNSET>,
    xmargin=<UNSET>, xscale=<UNSET>, xticklabels=<UNSET>, xticks=<UNSET>,
    ybound=<UNSET>, ylabel=<UNSET>, ylim=<UNSET>, ymargin=<UNSET>,
    yscale=<UNSET>, yticklabels=<UNSET>, yticks=<UNSET>, zorder=<UNSET>)

```

Set multiple properties at once.

Supported properties are

Properties: adjustable: {'box', 'datalim'} agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array and two offsets from the bottom left corner of the image alpha: scalar or None anchor: (float, float) or {'C', 'SW', 'S', 'SE', 'E', 'NE', ...} animated: bool aspect: {'auto', 'equal'} or float autoscale_on: bool autoscalex_on: unknown autoscaley_on: unknown axes_locator: Callable[[Axes, Renderer], Bbox] axisbelow: bool or 'line' box_aspect: float or None clip_box: *Bbox* clip_on: bool clip_path: Patch or (Path, Transform) or None facecolor or fc: color figure: *Figure* frame_on: bool gid: str in_layout: bool label: object mouseover: bool navigate: bool navigate_mode: unknown path_effects: *AbstractPathEffect* picker: None or bool or float or callable position: [left, bottom, width, height] or *~matplotlib.transforms.Bbox* prop_cycle: unknown rasterization_zorder: float or None rasterized: bool sketch_params: (scale: float, length: float, randomness: float) snap: bool or None subplotspec: unknown title: str transform: *Transform* url: str visible: bool xbound: unknown xlabel: str xlim: (bottom: float, top: float) xmargin: float greater than -0.5 xscale: unknown xticklabels: unknown xticks: unknown ybound: unknown ylabel: str ylim: (bottom: float, top: float) ymargin: float greater than -0.5 yscale: unknown yticklabels: unknown yticks: unknown zorder: float

```

class XPanAxes_NoZoom (fig, *args, facecolor=None, frameon=True, sharex=None, sharey=None,
    label="", xscale=None, yscale=None, box_aspect=None, **kwargs)

```

Bases: [Axes](#)

Constrain pan to x-axis and prevent zooming.

Methods:

[*drag_pan*](#)(button, key, x, y)

param button The pressed mouse button.

[*set*](#)(*[, adjustable, agg_filter, alpha, ...])

Set multiple properties at once.

Attributes:

[*name*](#)

drag_pan (*button*, *key*, *x*, *y*)

Parameters

- **button** – The pressed mouse button.
- **key** (`Optional[str]`) – The pressed key, if any.
- **x** (`float`) – The mouse coordinates in display coords.
- **y** (`float`) – The mouse coordinates in display coords.

name = 'XPanAxes_NoZoom'

Type: `str`

set (*, *adjustable*=<UNSET>, *agg_filter*=<UNSET>, *alpha*=<UNSET>, *anchor*=<UNSET>, *animated*=<UNSET>, *aspect*=<UNSET>, *autoscale_on*=<UNSET>, *autoscalex_on*=<UNSET>, *autoscaley_on*=<UNSET>, *axes_locator*=<UNSET>, *axisbelow*=<UNSET>, *box_aspect*=<UNSET>, *clip_box*=<UNSET>, *clip_on*=<UNSET>, *clip_path*=<UNSET>, *facecolor*=<UNSET>, *frame_on*=<UNSET>, *gid*=<UNSET>, *in_layout*=<UNSET>, *label*=<UNSET>, *mouseover*=<UNSET>, *navigate*=<UNSET>, *path_effects*=<UNSET>, *picker*=<UNSET>, *position*=<UNSET>, *prop_cycle*=<UNSET>, *rasterization_zorder*=<UNSET>, *rasterized*=<UNSET>, *sketch_params*=<UNSET>, *snap*=<UNSET>, *subplotspec*=<UNSET>, *title*=<UNSET>, *transform*=<UNSET>, *url*=<UNSET>, *visible*=<UNSET>, *xbound*=<UNSET>, *xlabel*=<UNSET>, *xlim*=<UNSET>, *xmargin*=<UNSET>, *xscale*=<UNSET>, *xticklabels*=<UNSET>, *xticks*=<UNSET>, *ybound*=<UNSET>, *ylabel*=<UNSET>, *ylim*=<UNSET>, *ymargin*=<UNSET>, *yscale*=<UNSET>, *yticklabels*=<UNSET>, *yticks*=<UNSET>, *zorder*=<UNSET>)

Set multiple properties at once.

Supported properties are

Properties: *adjustable*: {'box', 'datalim'} *agg_filter*: a filter function, which takes a (m, n, 3) float array and a dpi value, and returns a (m, n, 3) array and two offsets from the bottom left corner of the image *alpha*: scalar or None *anchor*: (float, float) or {'C', 'SW', 'S', 'SE', 'E', 'NE', ...} *animated*: bool *aspect*: {'auto', 'equal'} or float *autoscale_on*: bool *autoscalex_on*: unknown *autoscaley_on*: unknown *axes_locator*: Callable[[Axes, Renderer], Bbox] *axisbelow*: bool or 'line' *box_aspect*: float or None *clip_box*: *Bbox* *clip_on*: bool *clip_path*: Patch or (Path, Transform) or None *facecolor* or *fc*: color *figure*: *Figure* *frame_on*: bool *gid*: str *in_layout*: bool *label*: object *mouseover*: bool *navigate*: bool *navigate_mode*: unknown *path_effects*: *AbstractPathEffect* *picker*: None or bool or float or callable *position*: [left, bottom, width, height] or *~matplotlib.transforms.Bbox* *prop_cycle*: unknown *rasterization_zorder*: float or None *rasterized*: bool *sketch_params*: (scale: float, length: float, randomness: float) *snap*: bool or None *subplotspec*: unknown *title*: str *transform*: *Transform* *url*: str *visible*: bool *xbound*: unknown *xlabel*: str *xlim*: (bottom: float, top: float) *xmargin*: float greater than -0.5 *xscale*: unknown *xticklabels*: unknown *xticks*: unknown *ybound*: unknown *ylabel*: str *ylim*: (bottom: float, top: float) *ymargin*: float greater than -0.5 *yscale*: unknown *yticklabels*: unknown *yticks*: unknown *zorder*: float

2.18 style_picker

Dialogs for choosing Matplotlib colours and styles.

Classes:

<code>ColourPicker</code> (parent[, title, label, ...])	Dialog for choosing Matplotlib colours.
<code>StylePicker</code> (parent[, title, label, ...])	Dialog for choosing Matplotlib marker styles.

```
class ColourPicker (parent, title='Choose Colours', label='Choose Colours: ', picker_choices=None,
                    selection_choices=None, *args, **kwds)
```

Bases: `StylePicker`

Dialog for choosing Matplotlib colours.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **title** (`str`) – The dialog title. Default 'Choose Colours'.
- **label** (`str`) – The dialog label. Default 'Choose Colours: '.
- **selection_choices** (`Optional[List]`) – Default `None`.
- ***args** – Additional arguments passed to `wx.Dialog`.
- ****kwds** – Additional keyword arguments passed to `wx.Dialog`.

Methods:

<code>apply(event)</code>	Called when the user presses the <i>Apply</i> button.
---------------------------	---

apply (*event*)

Called when the user presses the *Apply* button.

Parameters **event** – The wxPython event.

```
class StylePicker (parent, title='Choose Styles', label='Choose Styles: ', selection_choices=None,
                  *args, **kwds)
```

Bases: `Dialog`

Dialog for choosing Matplotlib marker styles.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **title** (`str`) – The dialog title. Default 'Choose Styles'.
- **label** (`str`) – The dialog label. Default 'Choose Styles: '.
- **selection_choices** (`Optional[List]`) – Default `None`.
- ***args** – Additional arguments passed to `wx.Dialog`.
- ****kwds** – Additional keyword arguments passed to `wx.Dialog`.

Methods:

<code>apply(event)</code>	Called when the user presses the <i>Apply</i> button.
<code>cancel(_)</code>	Called when the user presses the <i>Cancel</i> button.

apply (*event*)

Called when the user presses the *Apply* button.

Parameters **event** – The wxPython event.

cancel (*_*)

Called when the user presses the *Cancel* button.

2.19 tabbable_textctrl

Multiline `wx.TextCtrl` that allows tabbing to the next or previous control.

Classes:

<code>TabbableTextCtrl</code> (parent[, id, value, pos, ...])	Multiline <code>wx.TextCtrl</code> that allows tabbing to the next or previous control.
---	---

```
class TabbableTextCtrl (parent, id=-1, value="", pos=(-1, -1), size=(-1, -1), style=0,
                        validator=<wx.Validator object>, name=b'text')
```

Bases: `TextCtrl`

Multiline `wx.TextCtrl` that allows tabbing to the next or previous control.

Parameters

- **parent** (`Window`) – Parent window. Should not be `None`.
- **id** (`int`) – Control identifier. A value of `-1` denotes a default value. Default `-1`.
- **value** (`str`) – Default text value. Default `' '`.
- **pos** (`Point`) – Text control position. Default `(-1, -1)`.
- **size** (`Size`) – Text control size. Default `(-1, -1)`.
- **style** (`int`) – Window style. See `wx.TextCtrl`. Default `0`.
- **validator** (`Validator`) – Window validator. Default `wx.DefaultValidator`.
- **name** (`AnyStr`) – Window name. Default `b'text'`.

Methods:

<code>on_char</code> (event)	Event handler for key being pressed, to allow for navigating between controls with <code>TAB</code> .
------------------------------	---

```
static on_char (event)
```

Event handler for key being pressed, to allow for navigating between controls with `TAB`.

2.20 textctrlwrapper

Classes:

<code>TextCtrlWrapper</code> ()	Base class for wrappers around <code>wx.TextCtrl</code> .
---------------------------------	---

```
class TextCtrlWrapper
```

Bases: `object`

Base class for wrappers around `wx.TextCtrl`.

Subclasses must set the value of `textctrl`.

Methods:

<code>AppendText(text)</code>	Appends the given text to the end of the text control.
<code>CanCopy()</code>	Returns <code>True</code> if the selection can be copied to the clipboard.
<code>CanCut()</code>	Returns <code>True</code> if the selection can be cut to the clipboard.
<code>CanPaste()</code>	Returns <code>True</code> if the contents of the clipboard can be pasted into the text control.
<code>CanRedo()</code>	Returns <code>True</code> if there is a redo facility available, and the last operation can be redone.
<code>CanUndo()</code>	Returns <code>True</code> if there is an undo facility available, and the last operation can be undone.
<code>Clear()</code>	Clears the text in the control.
<code>Copy()</code>	Copies the selected text to the clipboard.
<code>Cut()</code>	Copies the selected text to the clipboard and removes it from the control.
<code>GetLastPosition()</code>	Returns the zero based index of the last position in the text control, which is equal to the number of characters in the control.
<code>GetSelection()</code>	Gets the current selection span.
<code>GetStringSelection()</code>	Returns the text currently selected in the control.
<code>GetValue()</code>	Gets the contents of the control.
<code>IsEditable()</code>	Returns <code>True</code> if the controls contents may be edited by user (note that it always can be changed by the program).
<code>IsEmpty()</code>	Returns whether the control is currently empty.
<code>Paste()</code>	Pastes the clipboard contents into the control.
<code>Redo()</code>	If there is a redo facility and the last operation can be redone, redoes the last operation.
<code>Remove(from_, to_)</code>	Removes the text starting at the first given position up to (but not including) the character at the last position.
<code>Replace(from_, to_, value)</code>	Replaces the text starting at the first position up to (but not including) the character at the last position with the given text.
<code>SelectAll()</code>	Selects all text in the control.
<code>SelectNone()</code>	Deselects selected text in the control.
<code>SetSelection(from_, to_)</code>	Selects the text starting at the first position up to (but not including) the character at the last position.
<code>SetValue(value)</code>	Sets the new text control value.
<code>Undo()</code>	If there is an undo facility, and the last operation can be undone, undoes the last operation.
<code>WriteText(text)</code>	Writes the text into the text control at the current insertion position.

Attributes:

`textctrl`

The `wx.TextCtrl` being wrapped.

AppendText (*text*)

Appends the given text to the end of the text control.

Note: After the text is appended, the insertion point will be at the end of the text control. If this behaviour is not desired, the programmer should use `GetInsertionPoint` and `SetInsertionPoint`.

Parameters **text** (`str`)

CanCopy ()

Returns `True` if the selection can be copied to the clipboard.

Return type `bool`

CanCut ()

Returns `True` if the selection can be cut to the clipboard.

Return type `bool`

CanPaste ()

Returns `True` if the contents of the clipboard can be pasted into the text control.

On some platforms (Motif, GTK) this is an approximation and returns `True` if the control is editable, `False` otherwise.

Return type `bool`

CanRedo ()

Returns `True` if there is a redo facility available, and the last operation can be redone.

Return type `bool`

CanUndo ()

Returns `True` if there is an undo facility available, and the last operation can be undone.

Return type `bool`

Clear ()

Clears the text in the control.

Note that this function will generate a `wx.wxEVT_TEXT` event, i.e. its effect is identical to calling `SetValue('')`.

Copy ()

Copies the selected text to the clipboard.

Cut ()

Copies the selected text to the clipboard and removes it from the control.

GetLastPosition()

Returns the zero based index of the last position in the text control, which is equal to the number of characters in the control.

Return type `wx.TextPos`

GetSelection()

Gets the current selection span.

If the returned values are equal, there was no selection.

Note: The indices returned may be used with the other `wx.TextCtrl` methods but don't necessarily represent the correct indices into the string returned by `GetValue` for multiline controls under Windows (at least,) you should use `GetStringSelection` to get the selected text.

Return type `Tuple[int, int]`

GetStringSelection()

Returns the text currently selected in the control.

If there is no selection, the returned string is empty.

Return type `str`

GetValue()

Gets the contents of the control.

Note: For a multiline text control, the lines will be separated by (Unix-style) `\n` characters, even under Windows where they are separated by a `\r\n` sequence in the native control.

Return type `str`

IsEditable()

Returns `True` if the controls contents may be edited by user (note that it always can be changed by the program).

In other words, this functions returns `True` if the control hasn't been put in read-only mode by a previous call to `SetEditable()`.

Return type `bool`

IsEmpty()

Returns whether the control is currently empty.

Return type `bool`

Paste()

Pastes the clipboard contents into the control.

Redo()

If there is a redo facility and the last operation can be redone, redoes the last operation.

Does nothing if there is no redo facility.

Remove (*from_*, *to_*)

Removes the text starting at the first given position up to (but not including) the character at the last position.

This function puts the current insertion point position at *to* as a side effect.

Parameters

- **from_** (*int*) – The first position
- **to_** (*int*) – The last position

Replace (*from_*, *to_*, *value*)

Replaces the text starting at the first position up to (but not including) the character at the last position with the given text.

This function puts the current insertion point position at *to* as a side effect.

Parameters

- **from_** (*int*) – The first position
- **to_** (*int*) – The last position
- **value** – The value to replace the existing text with.

Return type *str*

SelectAll ()

Selects all text in the control.

SelectNone ()

Deselects selected text in the control.

SetSelection (*from_*, *to_*)

Selects the text starting at the first position up to (but not including) the character at the last position.

If both parameters are equal to -1 all text in the control is selected.

Notice that the insertion point will be moved to *from* by this function.

Parameters

- **from_** (*int*) – The first position
- **to_** (*int*) – The last position

SetValue (*value*)

Sets the new text control value.

It also marks the control as not-modified which means that `IsModified()` would return `False` immediately after the call to `SetValue()`.

The insertion point is set to the start of the control (i.e. position 0) by this function unless the control value doesn't change at all, in which case the insertion point is left at its original position.

Note: Unlike most other functions changing the control's values, this function generates a `wx.EVT_TEXT` event. To avoid this you can use `ChangeValue()` instead.

Parameters `value` (`str`) – The new value to set. It may contain newline characters if the text control is multiline.

Undo()

If there is an undo facility, and the last operation can be undone, undoes the last operation.

Does nothing if there is no undo facility.

WriteText (`text`)

Writes the text into the text control at the current insertion position.

Parameters `text` (`str`) – Text to write to the text control

textctrl

Type: `TextCtrl`

The `wx.TextCtrl` being wrapped.

2.21 timer_thread

Background thread that sends an event after the specified interval.

Useful for timeouts or updating timers, clocks etc.

Classes:

<code>Timer</code> (parent[, interval])	Background Timer Class.
---	-------------------------

Data:

<code>timer_event</code>	An instance of <code>domdf_python_tools.events.SimpleEvent</code> called Timer .
--------------------------	---

class `Timer` (`parent`, `interval=1.0`)

Bases: `Thread`

Background Timer Class.

Parameters

- **parent** (`Window`) – Class to send event updates to.
- **interval** (`float`) – Interval to trigger events at, in seconds. Default 1.0.

Methods:

<code>join</code> ([timeout])	Stop the thread and wait for it to end.
-------------------------------	---

continues on next page

Table 73 – continued from previous page

<code>run()</code>	Run the timer thread.
--------------------	-----------------------

join (*timeout=None*)

Stop the thread and wait for it to end.

Parameters **timeout** – Default `None`.

run ()

Run the timer thread.

timer_event = **SimpleEvent** (**name=Timer**)

Type: `SimpleEvent`

An instance of `domdf_python_tools.events.SimpleEvent` called **Timer**.

This event is triggered when the timer has expired.

2.22 utils

General utility functions.

Functions:

<code>collapse_label</code> (<i>text</i> [, <i>collapsed</i>])	Constructs the label to display on a collapsible section.
<code>coming_soon</code> (<i>[msg]</i>)	Displays a message box informing the user that the desired feature has not been implemented.
<code>generate_faces</code> ()	Returns a platform dependent set of typefaces.
<code>toggle</code> (<i>control</i>)	Toggle value of the given control.

collapse_label (*text*, *collapsed=True*)

Constructs the label to display on a collapsible section.

Parameters

- **text** (*str*) – The text of the label.
- **collapsed** (*bool*) – Whether the section is collapsed. Default `True`.

Return type `str`

coming_soon (*msg='This feature has not been implemented yet'*)

Displays a message box informing the user that the desired feature has not been implemented.

Parameters **msg** (*str*) – The message to display. Default `'This feature has not been implemented yet'`.

generate_faces ()

Returns a platform dependent set of typefaces.

Return type `Dict[str, Any]`

toggle (*control*)

Toggle value of the given control.

Usually used for checkboxes.

Parameters `control` (`CheckBox`)

Return type `bool`

2.23 validators

Various validator classes

Classes:

<code>CharValidator(flag)</code>	A Validator that only allows the type of characters selected to be entered.
<code>FloatValidator(flag)</code>	A Validator that only allows numbers and decimal points to be entered.
<code>ValidatorBase()</code>	Base class for Validators.

class `CharValidator` (*flag*)

Bases: `ValidatorBase`

A Validator that only allows the type of characters selected to be entered.

The possible flags are:

- int-only - only the numbers 0123456789 can be entered.
- float-only - only numbers and decimal points can be entered.

Methods:

<code>Clone()</code>	Clones the <code>CharValidator</code> .
<code>OnChar(event)</code>	Event handler for text being entered in the control.
<code>Validate(win)</code>	Validate the control.

Clone ()

Clones the `CharValidator`.

OnChar (*event*)

Event handler for text being entered in the control.

Parameters `event` – The wxPython event.

Validate (*win*)

Validate the control.

Parameters `win`

Return type `bool`

class `FloatValidator` (*flag*)

Bases: `CharValidator`

A Validator that only allows numbers and decimal points to be entered. If a decimal point has already been entered, a second one cannot be entered. The argument *flag* is used to limit the number of decimal places that can be entered.

Methods:

<code>OnChar(event)</code>	Event handler for text being entered in the control.
----------------------------	--

OnChar (*event*)

Event handler for text being entered in the control.

Parameters **event** – The wxPython event.

class ValidatorBase

Bases: `Validator`

Base class for Validators.

Methods:

<code>Clone()</code>	Clones the <code>wx.Validator</code> .
<code>TransferFromWindow()</code>	Transfer data from window to validator.
<code>TransferToWindow()</code>	Transfer data from validator to window.
<code>reset_ctrl()</code>	Resets the control's background colour.
<code>set_warning()</code>	Set the control's background colour to pink.

Clone ()

Clones the `wx.Validator`.

TransferFromWindow ()

Transfer data from window to validator.

The default implementation returns `False`, indicating that an error occurred. We simply return `True`, as we don't do any data transfer.

Return type `bool`

TransferToWindow ()

Transfer data from validator to window.

The default implementation returns `False`, indicating that an error occurred. We simply return `True`, as we don't do any data transfer.

Return type `bool`

reset_ctrl ()

Resets the control's background colour.

Return type `bool`

set_warning ()

Set the control's background colour to pink.

Return type `bool`

2.24 panel_listctrl

2.24.1 constants

Constants for *panel_listctrl*.

2.24.2 css_parser

Functions:

<code>parse_css(css_data)</code>	Parse the stylesheet from the given string
<code>parse_css_file(filename)</code>	Parse the stylesheet in the given file.

parse_css (*css_data*)

Parse the stylesheet from the given string

Parameters **css_data** (`str`) – A string representing a CSS stylesheet

Return type `Dict`

Returns Parsed CSS stylesheet

parse_css_file (*filename*)

Parse the stylesheet in the given file.

Parameters **filename** (`Union[str, Path, PathLike]`) – The filename of the stylesheet to parse.

Return type `Dict`

Returns Parsed CSS stylesheet.

2.24.3 font_parser

Functions:

<code>freezeargs(func)</code>	Makes a mutable dictionary immutable, so it can be used as an argument for <code>slru_cache()</code> .
<code>parse_font(style_dict)</code>	Parse the font from the <code>style_dict</code> .

freezeargs (*func*)

Makes a mutable dictionary immutable, so it can be used as an argument for `slru_cache()`.

Return type `Callable`

parse_font (*style_dict*)

Parse the font from the `style_dict`.

Parameters **style_dict** (`Dict`) – Dictionary containing styling information for the font

Return type `Tuple[str, Dict]`

Returns Tuple containing the colour of the font, and the font properties

The font properties dictionary returned will contain the following keys:

- `family (wx.FontFamily)` – The font family: a generic portable way of referring to fonts without specifying a facename. This parameter must be one of the `wx.FontFamily` enumeration values. If the `faceName` argument is provided, then it overrides the font family.
- `style (wx.FontStyle)` – One of `wx.FONTSTYLE_NORMAL`, `wx.FONTSTYLE_SLANT` and `wx.FONTSTYLE_ITALIC`.
- `weight (wx.FontWeight)` – Font weight, sometimes also referred to as font boldness. One of the `wx.FontWeight` enumeration values.
- `underline (wx.bool)` – The value can be `True` or `False`. At present this has an effect on Windows and Motif 2.x only.
- `faceName (str)` – An optional string specifying the face name to be used. If it is an empty string, a default face name will be chosen based on the family.
- `encoding (wx.FontEncoding)` – An encoding which may be one of the enumeration values of `wx.FontEncoding`. If the specified encoding isn't available, no font is created (see also Font Encodings).

and one of:

- `pointSize (int)` – Size in points. See `SetPointSize` for more info. Notice that, for historical reasons, the value 70 here is interpreted as `DEFAULT` and results in creation of the font with the default size and not of a font with the size of 70pt. If you really need the latter, please use `SetPointSize(70)`. Note that this constructor and the matching `Create()` method overload are the only places in `wx.Font` API handling `DEFAULT` specially: neither `SetPointSize` nor the constructor taking `wx.FontInfo` handle this value in this way.
- `pixelSize (wx.Size)` – Size in pixels. See `SetPixelSize` for more info.

depending on the font size specified in `'style_dict'`

2.24.4 panel_listctrl

A custom Panel that acts as a `ListCtrl` for other `wx.Panel` objects.

An example `ListItem` exists that provides two `StaticText` fields and can be used as the basis for custom list items

Classes:

`PanelListCtrl`(parent[, id, pos, size, ...])

`PanelListItem`(parent, text_dict, style_data)

type parent `PanelListCtrl`

```
class PanelListCtrl (parent, id=- 1, pos=(- 1, - 1), size=(- 1, - 1), style=524288, name=b'panel',
                    left_padding=32)
```

Bases: `ScrolledWindow`

Methods:

`AcceptsFocus()`

`AcceptsFocusFromKeyboard()`

continues on next page

Table 82 – continued from previous page

<i>Append</i> (panel_list_item)	Append a 'PanelListItem' object, or an instance of a custom subclass, to the control.
<i>AppendNewItem</i> (text_dict, style_data)	Append a new 'PanelListItem' object to the control, passing the 'text_dict' and 'style_data' parameters to the new object.
<i>Clear</i> ()	Removes all items from the control
<i>DeleteItem</i> (item)	Deletes the specified item from the control.
<i>DeselectAll</i> ()	Deselect all items.
<i>Focus</i> (idx)	Set Focus to the the given item.
<i>GetColumnCount</i> ()	Returns the number of columns.
<i>GetFirstSelected</i> (*_)	Returns the first selected item, or -1 when none is selected.
<i>GetFocusedItem</i> ()	Gets the currently focused item or -1 if none is focused.
<i>GetItem</i> (itemIdx, *_)	Returns information about the item.
<i>GetItemBackgroundColour</i> (item)	Returns the colour for this item.
<i>GetItemCount</i> ()	Returns the number of items in the list control.
<i>GetItemPosition</i> (item)	Returns the position of the item, or -1 if it is not found.
<i>GetNextSelected</i> (item)	Returns subsequent selected items, or -1 when no more are selected.
<i>GetSelectedItemCount</i> ()	Returns the number of selected items in the list control.
<i>IsEmpty</i> ()	Returns true if the control doesn't currently contain any items.
<i>IsSelected</i> (idx)	Returns :py:obj:`True` if the item is selected.
<i>RefreshItem</i> (item)	Redraws the given item.
<i>RefreshItems</i> (itemFrom, itemTo)	Redraws the items between itemFrom and itemTo.
<i>Select</i> (idx[, on])	Selects/deselects an item.
<i>SetSelection</i> (idx)	Set the current selection to the item at the given index.

Attributes:

<i>ColumnCount</i>	Returns the number of columns.
<i>FocusedItem</i>	Gets the currently focused item or -1 if none is focused.
<i>ItemCount</i>	Returns the number of items in the list control.

AcceptsFocus()**AcceptsFocusFromKeyboard()****Append** (panel_list_item)

Append a 'PanelListItem' object, or an instance of a custom subclass, to the control.

AppendNewItem (text_dict, style_data)

Append a new 'PanelListItem' object to the control, passing the 'text_dict' and 'style_data' parameters to the new object.

Parameters

- **text_dict** (*Dict*)
- **style_data**

Return type *PanelListItem*

Returns The new PanelListItem object that was added to the control

Clear()

Removes all items from the control

property ColumnCount

Returns the number of columns.

Return type *int*

DeleteItem (*item*)

Deletes the specified item from the control.

Parameters *item*

Return type *bool*

Returns *True* if the item was removed, *False* otherwise (usually because the item wasn't in the control)

DeselectAll()

Deselect all items.

Focus (*idx*)

Set Focus to the the given item.

Parameters *idx*

property FocusedItem

Gets the currently focused item or -1 if none is focused.

Returns

Return type

GetColumnCount()

Returns the number of columns.

GetFirstSelected (**_*)

Returns the first selected item, or -1 when none is selected.

Return type *int*

GetFocusedItem()

Gets the currently focused item or -1 if none is focused.

Returns

Return type

GetItem (*itemIdx*, *_)

Returns information about the item. See `SetItem()` for more information.

Parameters *itemIdx*

GetItemBackgroundColour (*item*)

Returns the colour for this item.

Return type `Colour`

GetItemCount ()

Returns the number of items in the list control.

Return type `int`

GetItemPosition (*item*)

Returns the position of the item, or -1 if it is not found.

Return type `int`

GetNextSelected (*item*)

Returns subsequent selected items, or -1 when no more are selected.

Parameters *item*

GetSelectedItemCount ()

Returns the number of selected items in the list control.

Return type `int`

IsEmpty ()

Returns true if the control doesn't currently contain any items.

Returns

Return type

IsSelected (*idx*)

Returns `:py:obj:`True`` if the item is selected.

Parameters *idx*

property ItemCount

Returns the number of items in the list control.

Return type `int`

RefreshItem (*item*)

Redraws the given item.

Parameters *item*

RefreshItems (*itemFrom*, *itemTo*)

Redraws the items between *itemFrom* and *itemTo*.

Parameters

- **itemFrom**
- **itemTo**

Select (*idx*, *on=1*)

Selects/deselects an item.

Parameters

- **idx**
- **on** (*int*) – Default 1.

SetSelection (*idx*)

Set the current selection to the item at the given index.

Parameters **idx** (*int*) – index of the item to select.

class PanelListItem (*parent*, *text_dict*, *style_data*, *id=-1*, *style=0*, *name=b'panel'*, *left_padding=32*)

Bases: `Panel`

Parameters

- **parent** (*PanelListCtrl*) – The PanelListCtrl the item is to go into
- **text_dict** (*Dict*)
- **style_data**
- **id** (*int*) – An identifier for the panel. ID_ANY is taken to mean a default. Default -1.
- **style** (*int*) – The window style. See wx.Panel. Default 0.
- **name** (*str*) – Window name. Default b'panel'.
- **left_padding** (*int*) – the spacing to the left of the text in the control. Default 32.

Methods:

<code>DeselectItem()</code>	
<code>GetBackgroundColour()</code>	
<code>GetContents()</code>	
<code>GetCurrentBackgroundColour()</code>	Returns the current background colour of the :class`wx.Panel` class.
<code>IsSelected()</code>	Returns whether the <i>PanelListItem</i> is selected.
<code>OnClick(_)</code>	
<code>OnDoubleClick(_)</code>	
<code>OnKeyDown(event)</code>	
param event The wxPython event.	
<code>OnMiddleClick(_)</code>	
<code>OnRightClick(_)</code>	
<code>Refresh(**kwargs)</code>	
<code>SelectItem([select])</code>	Select (or deselect) the given item.
<code>SetBackgroundColour(colour)</code>	Set the background colour for the item.
<code>SetSelectedBackgroundColour(colour)</code>	Set the background colour for the item when it is selected.

DeselectItem()

GetBackgroundColour()

GetContents()

GetCurrentBackgroundColour()

Returns the current background colour of the `:class`wx.Panel`` class.

IsSelected()

Returns whether the *PanelListItem* is selected.

Return type `bool`

OnClick(_)

OnDoubleClick(_)

OnKeyDown(event)

Parameters `event` – The wxPython event.

OnMiddleClick(_)

OnRightClick(_)

Refresh(kwargs)**

SelectItem(select=True)

Select (or deselect) the given item.

Parameters `select` (`bool`) – If `False` the item is deselected. Default `True`.

SetBackgroundColour(colour)

Set the background colour for the item.

Parameters `colour`

SetSelectedBackgroundColour(colour)

Set the background colour for the item when it is selected.

Parameters `colour`

Contributing

3.1 Overview

`domdf_wxpython_tools` uses `tox` to automate testing and packaging, and `pre-commit` to maintain code quality.

Install `pre-commit` with `pip` and install the git hook:

```
$ python -m pip install pre-commit
$ pre-commit install
```

3.2 Coding style

`formate` is used for code formatting.

It can be run manually via `pre-commit`:

```
$ pre-commit run formate -a
```

Or, to run the complete autoformatting suite:

```
$ pre-commit run -a
```

3.3 Automated tests

Tests are run with `tox` and `pytest`. To run tests for a specific Python version, such as Python 3.6:

```
$ tox -e py36
```

To run tests for all Python versions, simply run:

```
$ tox
```

3.4 Type Annotations

Type annotations are checked using `mypy`. Run `mypy` using `tox`:

```
$ tox -e mypy
```

3.5 Build documentation locally

The documentation is powered by Sphinx. A local copy of the documentation can be built with `tox`:

```
$ tox -e docs
```

3.6 Downloading source code

The `domdf_wxpython_tools` source code is available on GitHub, and can be accessed from the following URL:
https://github.com/domdfcoding/domdf_wxpython_tools

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/domdf_wxpython_tools
```

```
Cloning into 'domdf_wxpython_tools'...
remote: Enumerating objects: 47, done.
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (41/41), done.
remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a ‘zip’ file by clicking:

Clone or download → Download Zip

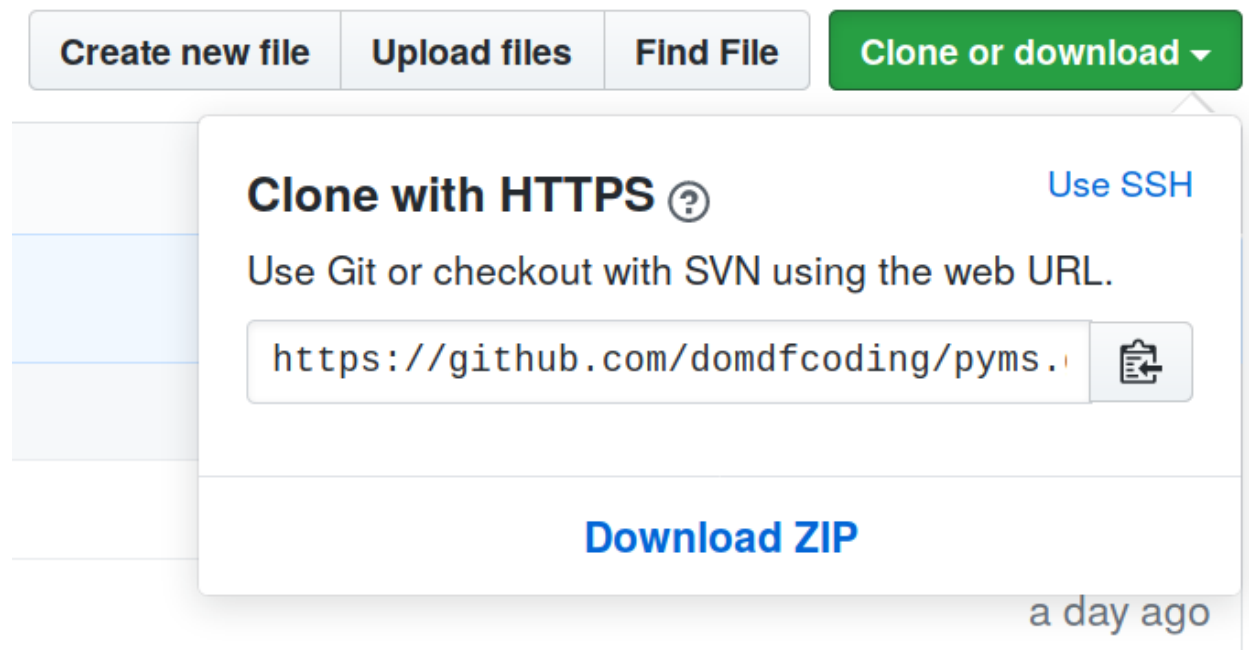


Fig. 1: Downloading a ‘zip’ file of the source code

3.6.1 Building from source

The recommended way to build `domdf_wxpython_tools` is to use `tox`:

```
$ tox -e build
```

The source and wheel distributions will be in the directory `dist`.

If you wish, you may also use `pep517.build` or another **PEP 517**-compatible build tool.

Python Module Index

d

- `domdf_wxpython_tools.border_config`, 7
- `domdf_wxpython_tools.chartpanel`, 8
- `domdf_wxpython_tools.clearable_textctrl`,
10
- `domdf_wxpython_tools.ColourPickerPanel`,
3
- `domdf_wxpython_tools.dialogs`, 20
- `domdf_wxpython_tools.editable_listbox`,
23
- `domdf_wxpython_tools.events`, 28
- `domdf_wxpython_tools.filebrowsectrl`, 30
- `domdf_wxpython_tools.icons`, 35
- `domdf_wxpython_tools.imagepanel`, 35
- `domdf_wxpython_tools.keyboard`, 37
- `domdf_wxpython_tools.list_dialog`, 38
- `domdf_wxpython_tools.logctrl`, 39
- `domdf_wxpython_tools.panel_listctrl.constants`,
60
- `domdf_wxpython_tools.panel_listctrl.css_parser`,
60
- `domdf_wxpython_tools.panel_listctrl.font_parser`,
60
- `domdf_wxpython_tools.panel_listctrl.panel_listctrl`,
61
- `domdf_wxpython_tools.picker`, 42
- `domdf_wxpython_tools.projections`, 45
- `domdf_wxpython_tools.style_picker`, 49
- `domdf_wxpython_tools.StylePickerPanel`,
4
- `domdf_wxpython_tools.tabbable_textctrl`,
51
- `domdf_wxpython_tools.textctrlwrapper`,
51
- `domdf_wxpython_tools.timer_thread`, 56
- `domdf_wxpython_tools.utils`, 57
- `domdf_wxpython_tools.validators`, 58
- `domdf_wxpython_tools.WebView`, 6

Symbols

`__repr__()` (*SimpleEvent method*), 29
`__repr__()` (*Wildcards method*), 21
`__str__()` (*Wildcards method*), 21

A

`AcceptsFocus()` (*PanelListCtrl method*), 62
`AcceptsFocusFromKeyboard()` (*ClearButton method*), 11
`AcceptsFocusFromKeyboard()` (*PanelListCtrl method*), 62
`add()` (*ColourPickerPanel method*), 4
`add()` (*StylePickerPanel method*), 5
`add_all_files_wildcard()` (*Wildcards method*), 21
`add_common_filetype()` (*Wildcards method*), 21
`add_filetype()` (*Wildcards method*), 21
`add_image_wildcard()` (*Wildcards method*), 22
`Append()` (*LogCtrl method*), 40
`Append()` (*PanelListCtrl method*), 62
`AppendNewItem()` (*PanelListCtrl method*), 62
`AppendStderr()` (*LogCtrl method*), 40
`AppendText()` (*TextCtrlWrapper method*), 53
`apply()` (*ColourPicker method*), 50
`apply()` (*StylePicker method*), 50
`apply_tight_layout()` (*border_config method*), 8
`AutoComplete()` (*ClearableTextCtrl method*), 13
`AutoCompletedDirectories()` (*ClearableTextCtrl method*), 13
`AutoCompleteFileNames()` (*ClearableTextCtrl method*), 14

B

`Bind()` (*SimpleEvent method*), 29
`border_config` (class in *domdf_wxpython_tools.border_config*), 7
`Browse()` (*dir_picker method*), 43
`Browse()` (*file_folder_picker method*), 44
`Browse()` (*file_picker method*), 45
`bufferHasChanged()` (*LogCtrl method*), 41
`bufferSave()` (*LogCtrl method*), 41

C

`cancel()` (*StylePicker method*), 50

`CanCopy()` (*LogCtrl method*), 40
`CanCopy()` (*TextCtrlWrapper method*), 53
`CanCut()` (*TextCtrlWrapper method*), 53
`CanPaste()` (*TextCtrlWrapper method*), 53
`CanRedo()` (*TextCtrlWrapper method*), 53
`CanUndo()` (*TextCtrlWrapper method*), 53
`ChangeValue()` (*ClearableTextCtrl method*), 14
`ChangeValue()` (*FileBrowseCtrl method*), 31
`ChartPanelBase` (class in *domdf_wxpython_tools.chartpanel*), 8
`CharValidator` (class in *domdf_wxpython_tools.validators*), 58
`Clear()` (*dir_picker method*), 43
`clear()` (*ImagePanel method*), 36
`Clear()` (*PanelListCtrl method*), 63
`Clear()` (*TextCtrlWrapper method*), 53
`clear_btn` (in module *domdf_wxpython_tools.clearable_textctrl*), 20
`ClearableTextCtrl` (class in *domdf_wxpython_tools.clearable_textctrl*), 11
`ClearButton` (class in *domdf_wxpython_tools.clearable_textctrl*), 11
`CleverListCtrl` (class in *domdf_wxpython_tools.editable_listbox*), 23
`Clone()` (*CharValidator method*), 58
`Clone()` (*ValidatorBase method*), 59
`close_dialog()` (*border_config method*), 8
`collapse_label()` (in module *domdf_wxpython_tools.utils*), 57
`ColourPicker` (class in *domdf_wxpython_tools.style_picker*), 49
`ColourPickerPanel` (class in *domdf_wxpython_tools.ColourPickerPanel*), 3
`ColumnCount()` (*PanelListCtrl property*), 63
`coming_soon()` (in module *domdf_wxpython_tools.utils*), 57
`configure_borders()` (*ChartPanelBase method*), 9
`constrain_zoom()` (*ChartPanelBase method*), 9
`copy()` (*ImagePanel method*), 37
`Copy()` (*LogCtrl method*), 40
`Copy()` (*TextCtrlWrapper method*), 53
`createBrowseButton()` (*FileBrowseCtrl method*), 33

CreateColumns() (*CleverListCtrl method*), 24
 createTextControl() (*FileBrowseCtrl method*), 33
 createTextControl() (*FileBrowseCtrlWithHistory method*), 35
 CTCWidget (class in *domdf_wxpython_tools.clearable_textctrl*), 10
 Cut() (*TextCtrlWrapper method*), 53

D

decimal_places() (*EditableNumericalListBox property*), 28
 default_clear_bitmap() (*ClearableTextCtrl property*), 19
 default_image (*ImagePanel attribute*), 37
 DeleteItem() (*PanelListCtrl method*), 63
 DeselectAll() (*PanelListCtrl method*), 63
 DeselectItem() (*PanelListItem method*), 65
 dir_picker (class in *domdf_wxpython_tools.picker*), 42
 DirBrowseCtrl (class in *domdf_wxpython_tools.filebrowsectrl*), 30
 DiscardEdits() (*ClearableTextCtrl method*), 14
 do_cancel() (*list_dialog method*), 38
 do_layout() (*StylePickerPanel method*), 5
 do_select() (*list_dialog method*), 38
 DoFindNext() (*LogCtrl method*), 40
 domdf_wxpython_tools.border_config module, 7
 domdf_wxpython_tools.chartpanel module, 8
 domdf_wxpython_tools.clearable_textctrl module, 10
 domdf_wxpython_tools.ColourPickerPanel module, 3
 domdf_wxpython_tools.dialogs module, 20
 domdf_wxpython_tools.editable_listbox module, 23
 domdf_wxpython_tools.events module, 28
 domdf_wxpython_tools.filebrowsectrl module, 30
 domdf_wxpython_tools.icons module, 35
 domdf_wxpython_tools.imagepanel module, 35
 domdf_wxpython_tools.keyboard module, 37
 domdf_wxpython_tools.list_dialog module, 38
 domdf_wxpython_tools.logctrl module, 39
 domdf_wxpython_tools.panel_listctrl.constants

module, 60
 domdf_wxpython_tools.panel_listctrl.css_parser module, 60
 domdf_wxpython_tools.panel_listctrl.font_parser module, 60
 domdf_wxpython_tools.panel_listctrl.panel_listctrl module, 61
 domdf_wxpython_tools.picker module, 42
 domdf_wxpython_tools.projections module, 45
 domdf_wxpython_tools.style_picker module, 49
 domdf_wxpython_tools.StylePickerPanel module, 4
 domdf_wxpython_tools.tabbable_textctrl module, 51
 domdf_wxpython_tools.textctrlwrapper module, 51
 domdf_wxpython_tools.timer_thread module, 56
 domdf_wxpython_tools.utils module, 57
 domdf_wxpython_tools.validators module, 58
 domdf_wxpython_tools.WebView module, 6
 drag_pan() (*NoZoom method*), 46
 drag_pan() (*XPanAxes method*), 47
 drag_pan() (*XPanAxes_NoZoom method*), 48

E

EditableListBox (class in *domdf_wxpython_tools.editable_listbox*), 24
 EditableNumericalListBox (class in *domdf_wxpython_tools.editable_listbox*), 27
 EmulateKeyPress() (*ClearableTextCtrl method*), 14
 eventType (*EvtImgPanelChanged attribute*), 36
 EvtImgPanelChanged (class in *domdf_wxpython_tools.imagepanel*), 35

F

file_dialog() (in module *domdf_wxpython_tools.dialogs*), 22
 file_dialog_multiple() (in module *domdf_wxpython_tools.dialogs*), 22
 file_dialog_wildcard() (in module *domdf_wxpython_tools.dialogs*), 23
 file_folder_picker (class in *domdf_wxpython_tools.picker*), 44
 file_picker (class in *domdf_wxpython_tools.picker*), 45
 FileBrowseCtrl (class in *domdf_wxpython_tools.filebrowsectrl*), 30

FileBrowseCtrlWithHistory (*class in domdf_wxpython_tools.filebrowsectrl*), 34
 findDlg (*LogCtrl attribute*), 41
 fixLineEndings () (*LogCtrl method*), 41
 FloatEntryDialog (*class in domdf_wxpython_tools.dialogs*), 20
 FloatValidator (*class in domdf_wxpython_tools.validators*), 58
 Focus () (*PanelListCtrl method*), 63
 FocusedItem () (*PanelListCtrl property*), 63
 freezeargs () (*in module domdf_wxpython_tools.panel_listctrl.font_parser*), 60

G

gen_keymap () (*in module domdf_wxpython_tools.keyboard*), 37
 generate_faces () (*in module domdf_wxpython_tools.utils*), 57
 get_button_icon () (*in module domdf_wxpython_tools.icons*), 35
 get_selection () (*ColourPickerPanel method*), 4
 get_selection () (*StylePickerPanel method*), 5
 get_toolbar_icon () (*in module domdf_wxpython_tools.icons*), 35
 get_value () (*dir_picker method*), 44
 GetBackgroundColour () (*PanelListItem method*), 66
 GetBestClientSize () (*ClearableTextCtrl method*), 14
 GetBestSize () (*ClearButton method*), 11
 GetColumnCount () (*PanelListCtrl method*), 63
 GetCompositeWindowParts () (*ClearableTextCtrl method*), 15
 GetContents () (*PanelListItem method*), 66
 GetContextMenu () (*LogCtrl method*), 40
 GetCurrentBackgroundColour () (*PanelListItem method*), 66
 GetDecimalPlaces () (*EditableNumericalListBox method*), 27
 GetDefaultStyle () (*ClearableTextCtrl method*), 15
 GetDelButton () (*EditableListBox method*), 25
 GetDownButton () (*EditableListBox method*), 25
 GetEditButton () (*EditableListBox method*), 25
 GetFirstSelected () (*PanelListCtrl method*), 63
 GetFocusedItem () (*PanelListCtrl method*), 63
 GetHistory () (*FileBrowseCtrlWithHistory method*), 34
 GetHistoryControl () (*FileBrowseCtrlWithHistory method*), 34
 GetInsertionPoint () (*ClearableTextCtrl method*), 15
 GetInsertionPoint () (*dir_picker method*), 43
 GetItem () (*PanelListCtrl method*), 63

GetItemBackgroundColour () (*PanelListCtrl method*), 64
 GetItemCount () (*PanelListCtrl method*), 64
 GetItemPosition () (*PanelListCtrl method*), 64
 GetLabel () (*FileBrowseCtrl method*), 32
 GetLastPosition () (*LogCtrl method*), 40
 GetLastPosition () (*TextCtrlWrapper method*), 53
 GetLineLength () (*ClearableTextCtrl method*), 15
 GetLineLength () (*FileBrowseCtrl method*), 32
 GetLineText () (*ClearableTextCtrl method*), 15
 GetLineText () (*FileBrowseCtrl method*), 32
 GetListCtrl () (*EditableListBox method*), 25
 GetMainWindowOfCompositeControl () (*ClearButton method*), 11
 GetMainWindowOfCompositeControl () (*CTCWidget method*), 10
 GetNewButton () (*EditableListBox method*), 25
 GetNextSelected () (*PanelListCtrl method*), 64
 GetNumberOfLines () (*ClearableTextCtrl method*), 15
 GetNumberOfLines () (*FileBrowseCtrl method*), 32
 GetRange () (*ClearableTextCtrl method*), 15
 GetRange () (*FileBrowseCtrl method*), 32
 GetRange () (*LogCtrl method*), 40
 GetSelectedItemCount () (*PanelListCtrl method*), 64
 GetSelection () (*ColourPickerPanel method*), 4
 GetSelection () (*LogCtrl method*), 40
 GetSelection () (*TextCtrlWrapper method*), 54
 GetStockBitmap () (*in module domdf_wxpython_tools.icons*), 35
 GetStockToolbarBitmap () (*in module domdf_wxpython_tools.icons*), 35
 GetStrings () (*EditableListBox method*), 26
 GetStrings () (*EditableNumericalListBox method*), 27
 GetStringSelection () (*TextCtrlWrapper method*), 54
 GetStyle () (*ClearableTextCtrl method*), 16
 getStyle () (*LogCtrl method*), 41
 GetUpButton () (*EditableListBox method*), 26
 GetValue () (*FloatEntryDialog method*), 20
 GetValue () (*IntEntryDialog method*), 21
 GetValue () (*PayloadEvent method*), 29
 GetValue () (*TextCtrlWrapper method*), 54
 GetValues () (*EditableNumericalListBox method*), 27

H

HitTest () (*ClearableTextCtrl method*), 16
 HitTestPos () (*ClearableTextCtrl method*), 16

I

image () (*ImagePanel property*), 37

ImagePanel (*class in*
 domdf_wxpython_tools.imagepanel), 36
 IntEntryDialog (*class in*
 domdf_wxpython_tools.dialogs), 20
 IsClearButtonVisible() (*ClearableTextCtrl*
 method), 16
 IsEditable() (*FileBrowseCtrl method*), 32
 IsEditable() (*TextCtrlWrapper method*), 54
 IsEmpty() (*PanelListCtrl method*), 64
 IsEmpty() (*TextCtrlWrapper method*), 54
 IsModified() (*ClearableTextCtrl method*), 16
 IsModified() (*FileBrowseCtrl method*), 32
 IsMultiLine() (*ClearableTextCtrl method*), 17
 IsMultiLine() (*FileBrowseCtrl method*), 33
 IsSelected() (*PanelListCtrl method*), 64
 IsSelected() (*PanelListItem method*), 66
 IsSingleLine() (*ClearableTextCtrl method*), 17
 IsSingleLine() (*FileBrowseCtrl method*), 33
 ItemCount() (*PanelListCtrl property*), 64

J

join() (*Timer method*), 57

L

LayoutControls() (*ClearableTextCtrl method*), 17
 list_dialog (*class in*
 domdf_wxpython_tools.list_dialog), 38
 load_image() (*ImagePanel method*), 37
 LogCtrl (*class in domdf_wxpython_tools.logctrl*), 39

M

MarkDirty() (*ClearableTextCtrl method*), 17
 MarkDirty() (*FileBrowseCtrl method*), 33
 module

 domdf_wxpython_tools.border_config,
 7
 domdf_wxpython_tools.chartpanel, 8
 domdf_wxpython_tools.clearable_textctrl,
 10
 domdf_wxpython_tools.ColourPickerPanel,
 3
 domdf_wxpython_tools.dialogs, 20
 domdf_wxpython_tools.editable_listbox,
 23
 domdf_wxpython_tools.events, 28
 domdf_wxpython_tools.filebrowsectrl,
 30
 domdf_wxpython_tools.icons, 35
 domdf_wxpython_tools.imagepanel, 35
 domdf_wxpython_tools.keyboard, 37
 domdf_wxpython_tools.list_dialog, 38
 domdf_wxpython_tools.logctrl, 39
 domdf_wxpython_tools.panel_listctrl.constants,
 60

 domdf_wxpython_tools.panel_listctrl.css_parser,
 60
 domdf_wxpython_tools.panel_listctrl.font_parser,
 60
 domdf_wxpython_tools.panel_listctrl.panel_listctrl,
 61
 domdf_wxpython_tools.picker, 42
 domdf_wxpython_tools.projections, 45
 domdf_wxpython_tools.style_picker,
 49
 domdf_wxpython_tools.StylePickerPanel,
 4
 domdf_wxpython_tools.tabbable_textctrl,
 51
 domdf_wxpython_tools.textctrlwrapper,
 51
 domdf_wxpython_tools.timer_thread,
 56
 domdf_wxpython_tools.utils, 57
 domdf_wxpython_tools.validators, 58
 domdf_wxpython_tools.WebView, 6

move() (*StylePickerPanel method*), 5
 move_down() (*StylePickerPanel method*), 5
 move_up() (*StylePickerPanel method*), 5
 MSWSetEmulationLevel() (*in module*
 domdf_wxpython_tools.WebView), 6

N

name (*NoZoom attribute*), 46
 name (*XPanAxes attribute*), 47
 name (*XPanAxes_NoZoom attribute*), 49
 NoZoom (*class in domdf_wxpython_tools.projections*),
 45

O

on_char() (*TabbableTextCtrl static method*), 51
 on_context_menu() (*ImagePanel method*), 37
 on_load() (*ImagePanel method*), 37
 on_save() (*ImagePanel method*), 37
 on_size_change() (*ChartPanelBase method*), 9
 on_value_changed() (*EditableListBox method*), 27
 on_value_changed() (*EditableNumericalListBox*
 method), 28
 OnBeginLabelEdit() (*EditableListBox method*), 26
 OnBrowse() (*DirBrowseCtrl method*), 30
 OnBrowse() (*FileBrowseCtrl method*), 33
 OnChar() (*CharValidator method*), 58
 OnChar() (*FloatValidator method*), 59
 OnClearButton() (*ClearableTextCtrl method*), 17
 OnClick() (*PanelListItem method*), 66
 OnContextMenu() (*LogCtrl method*), 40
 OnDelItem() (*EditableListBox method*), 26
 OnDoubleClick() (*PanelListItem method*), 66
 OnDownItem() (*EditableListBox method*), 26

OnEditItem() (*EditableListBox method*), 26
 OnEndLabelEdit() (*EditableListBox method*), 26
 OnFindClose() (*LogCtrl method*), 40
 OnFindText() (*LogCtrl method*), 41
 OnItemActivated() (*EditableListBox method*), 26
 OnItemSelected() (*EditableListBox method*), 26
 OnKeyDown() (*PanelListItem method*), 66
 onKeyPress() (*LogCtrl method*), 41
 OnLeftUp() (*ClearButton method*), 11
 OnMiddleClick() (*PanelListItem method*), 66
 OnNewItem() (*EditableListBox method*), 26
 OnPaint() (*ClearButton method*), 11
 OnRightClick() (*PanelListItem method*), 66
 OnSetFocus() (*FileBrowseCtrlWithHistory method*), 34
 OnSize() (*ClearableTextCtrl method*), 17
 OnSize() (*CleverListCtrl method*), 24
 OnText() (*CTCWidget method*), 10
 OnTextEnter() (*CTCWidget method*), 10
 OnUpItem() (*EditableListBox method*), 26
 OnZoomDefault() (*LogCtrl method*), 41
 OnZoomIn() (*LogCtrl method*), 41
 OnZoomOut() (*LogCtrl method*), 41

P

pan() (*ChartPanelBase method*), 9
 PanelListCtrl (class in
 domdf_wxpython_tools.panel_listctrl.panel_listctrl), 61
 PanelListItem (class in
 domdf_wxpython_tools.panel_listctrl.panel_listctrl), 65
 parse_css() (in module
 domdf_wxpython_tools.panel_listctrl.css_parser), 60
 parse_css_file() (in module
 domdf_wxpython_tools.panel_listctrl.css_parser), 60
 parse_font() (in module
 domdf_wxpython_tools.panel_listctrl.font_parser), 60
 paste() (*ImagePanel method*), 37
 Paste() (*TextCtrlWrapper method*), 54
 PayloadEvent (class in
 domdf_wxpython_tools.events), 28
 pick() (*ColourPickerPanel method*), 4
 PositionToXY() (*ClearableTextCtrl method*), 17
 previous_view() (*ChartPanelBase method*), 9
 Python Enhancement Proposals
 PEP 517, 69

R

Redo() (*TextCtrlWrapper method*), 54
 Refresh() (*PanelListItem method*), 66

RefreshItem() (*PanelListCtrl method*), 64
 RefreshItems() (*PanelListCtrl method*), 64
 remove() (*ColourPickerPanel method*), 4
 remove() (*StylePickerPanel method*), 5
 Remove() (*TextCtrlWrapper method*), 55
 Replace() (*TextCtrlWrapper method*), 55
 reset_ctrl() (*ValidatorBase method*), 59
 reset_value() (*dir_picker method*), 44
 reset_view() (*ChartPanelBase method*), 9
 reset_view() (*ImagePanel method*), 37
 ResetValue() (*dir_picker method*), 43
 run() (*Timer method*), 57

S

Select() (*PanelListCtrl method*), 65
 SelectAll() (*TextCtrlWrapper method*), 55
 SelectItem() (*PanelListItem method*), 66
 SelectNone() (*TextCtrlWrapper method*), 55
 set() (*NoZoom method*), 46
 set() (*XPanAxes method*), 47
 set() (*XPanAxes_NoZoom method*), 49
 set_files_mode() (*file_folder_picker method*), 44
 set_folders_mode() (*file_folder_picker method*), 44
 set_height() (*dir_picker method*), 44
 set_properties() (*StylePickerPanel method*), 5
 set_receiver() (*SimpleEvent method*), 29
 set_textctrl_width() (*dir_picker method*), 44
 set_warning() (*ValidatorBase method*), 59
 SetBackgroundColour() (*ClearableTextCtrl method*), 17
 SetBackgroundColour() (*PanelListItem method*), 66
 SetBitmapLabel() (*ClearButton method*), 11
 SetClearBitmap() (*ClearableTextCtrl method*), 18
 SetDecimalPlaces() (*EditableNumericalListBox method*), 27
 SetDefaultStyle() (*ClearableTextCtrl method*), 18
 setDisplayLineNumbers() (*LogCtrl method*), 41
 SetEditable() (*ClearableTextCtrl method*), 18
 SetFont() (*ClearableTextCtrl method*), 18
 SetHeight() (*dir_picker method*), 43
 SetHistory() (*FileBrowseCtrlWithHistory method*), 35
 SetInitialValue() (*dir_picker method*), 43
 SetInsertionPoint() (*ClearableTextCtrl method*), 18
 SetInsertionPoint() (*dir_picker method*), 43
 SetInsertionPointEnd() (*ClearableTextCtrl method*), 18
 SetInsertionPointEnd() (*dir_picker method*), 43
 SetLabel() (*FileBrowseCtrl method*), 33
 SetMaxLength() (*ClearableTextCtrl method*), 18
 SetModified() (*ClearableTextCtrl method*), 19

SetModified() (*FileBrowseCtrl method*), 33
 SetSelectedBackgroundColour() (*PanelListItem method*), 66
 SetSelection() (*PanelListCtrl method*), 65
 SetSelection() (*TextCtrlWrapper method*), 55
 SetStrings() (*EditableListBox method*), 26
 SetStrings() (*EditableNumericalListBox method*), 27
 SetStyle() (*ClearableTextCtrl method*), 19
 setStyles() (*LogCtrl method*), 41
 SetTextWidth() (*dir_picker method*), 43
 setup_scrollwheel_zooming() (*ChartPanelBase method*), 9
 setup_ylim_refresher() (*ChartPanelBase method*), 9
 SetupEditControl() (*EditableListBox method*), 26
 SetupEditControl() (*EditableNumericalListBox method*), 28
 SetValue() (*FileBrowseCtrl method*), 33
 SetValue() (*TextCtrlWrapper method*), 55
 SetValues() (*EditableNumericalListBox method*), 28
 ShouldInheritColours() (*ClearableTextCtrl method*), 19
 ShowPosition() (*ClearableTextCtrl method*), 19
 ShowPosition() (*LogCtrl method*), 41
 SimpleEvent (class in *domdf_wxpython_tools.events*), 29
 size_change() (*ChartPanelBase method*), 9
 SizeColumns() (*CleverListCtrl method*), 24
 StylePicker (class in *domdf_wxpython_tools.style_picker*), 50
 StylePickerPanel (class in *domdf_wxpython_tools.StylePickerPanel*), 4
 SwapItems() (*EditableListBox method*), 26

T

TabbableTextCtrl (class in *domdf_wxpython_tools.tabbable_textctrl*), 51
 textctrl (*ClearableTextCtrl attribute*), 19
 textctrl (*dir_picker attribute*), 44
 textctrl (*DirBrowseCtrl attribute*), 30
 textctrl (*file_folder_picker attribute*), 45
 textctrl (*file_picker attribute*), 45
 textctrl (*FileBrowseCtrl attribute*), 34
 textctrl (*FileBrowseCtrlWithHistory attribute*), 35
 textctrl (*TextCtrlWrapper attribute*), 56
 TextCtrlWrapper (class in *domdf_wxpython_tools.textctrlwrapper*), 51
 Timer (class in *domdf_wxpython_tools.timer_thread*), 56
 timer_event (in module *domdf_wxpython_tools.timer_thread*), 57
 toggle() (in module *domdf_wxpython_tools.utils*), 57
 toggle_mode() (*file_folder_picker method*), 45

ToggleLineNumbers() (*LogCtrl method*), 41
 ToggleWrap() (*LogCtrl method*), 41
 TransferFromWindow() (*ValidatorBase method*), 59
 TransferToWindow() (*ValidatorBase method*), 59
 trigger() (*SimpleEvent method*), 29

U

Unbind() (*SimpleEvent method*), 29
 Undo() (*TextCtrlWrapper method*), 56
 update_borders() (*border_config method*), 8
 update_picker_preview() (*StylePickerPanel method*), 5
 update_preview() (*ColourPickerPanel method*), 4
 update_preview() (*StylePickerPanel method*), 5
 update_selection_preview() (*StylePickerPanel method*), 5

V

Validate() (*CharValidator method*), 58
 ValidatorBase (class in *domdf_wxpython_tools.validators*), 59

W

wildcard() (*Wildcards property*), 22
 Wildcards (class in *domdf_wxpython_tools.dialogs*), 21
 wrap() (*LogCtrl method*), 41
 write() (*LogCtrl method*), 42
 WriteText() (*TextCtrlWrapper method*), 56
 wxWEBVIEWIE_EMU_DEFAULT (in module *domdf_wxpython_tools.WebView*), 6
 wxWEBVIEWIE_EMU_IE10 (in module *domdf_wxpython_tools.WebView*), 6
 wxWEBVIEWIE_EMU_IE10_FORCE (in module *domdf_wxpython_tools.WebView*), 6
 wxWEBVIEWIE_EMU_IE11 (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE11_FORCE (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE7 (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE8 (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE8_FORCE (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE9 (in module *domdf_wxpython_tools.WebView*), 7
 wxWEBVIEWIE_EMU_IE9_FORCE (in module *domdf_wxpython_tools.WebView*), 7

X

XPanAxes (class in *domdf_wxpython_tools.projections*), 47

XPanAxes_NoZoom (*class in*
 domdf_wxpython_tools.projections), [48](#)
XYToPosition() (*ClearableTextCtrl method*), [19](#)

Z

zoom() (*ChartPanelBase method*), [9](#)