

---

# **pyqtribbon**

***Release 0.3.1***

**WANG Hailin**

**Jul 28, 2022**



**CONTENTS:**

<b>1</b>	<b>Getting Started</b>	<b>3</b>
1.1	Installation . . . . .	3
<b>2</b>	<b>The Ribbon Bar</b>	<b>5</b>
2.1	Introduction . . . . .	5
2.2	Definitions of Ribbon Elements . . . . .	6
2.3	Ribbon Elements in PyQtRibbon . . . . .	6
<b>3</b>	<b>User Manual</b>	<b>7</b>
3.1	Instantiate a Ribbon Bar . . . . .	7
3.2	Customize Ribbon Bar . . . . .	7
3.3	Customize Categories . . . . .	9
3.4	Customize Panels . . . . .	9
3.5	A Complete Example . . . . .	11
<b>4</b>	<b>API References</b>	<b>13</b>
4.1	Ribbon Bar . . . . .	13
4.2	Ribbon Title . . . . .	19
4.3	Ribbon Category . . . . .	23
4.4	Ribbon Panel . . . . .	28
4.5	Ribbon Gallery . . . . .	39
4.6	Ribbon Tool Button . . . . .	41
4.7	Ribbon Separator . . . . .	42
4.8	Ribbon Menu . . . . .	43
<b>5</b>	<b>Indices and tables</b>	<b>45</b>
	<b>Index</b>	<b>47</b>



PyQtRibbon is a Qt-based application framework for building user interfaces.

- GitHub Repository: [github.com/haibiliin/pyqtribbon](https://github.com/haibiliin/pyqtribbon).
- PyPI: [pypi.org/project/pyqtribbon](https://pypi.org/project/pyqtribbon).
- Documentation: [pyqtribbon.haibiliin.com](https://pyqtribbon.haibiliin.com).
- Read the Docs: [readthedocs.org/projects/pyqtribbon](https://readthedocs.org/projects/pyqtribbon).



## GETTING STARTED

### 1.1 Installation

PyQtRibbon is distributed to [PyPI](#), you can use pip to install it:

```
pip install pyqtribbon
```

You can also install the package from source:

```
pip install git+https://github.com/haiiliin/pyqtribbon.git@main
```





## THE RIBBON BAR

### 2.1 Introduction

The ribbon is first introduced by Microsoft in the 2000's. It is a toolbar with a tabbed interface. According to [Microsoft](#):

**Note:** A ribbon is a user interface (UI) element that organizes commands into logical groups. These groups appear on separate tabs in a strip across the top of the window. The ribbon replaces the menu bar and toolbars. A ribbon can significantly improve application usability. For more information, see [Ribbons](#). The following illustration shows a ribbon. A ribbon can significantly improve application usability. For more information, see [Ribbons](#). The following illustration shows a ribbon.

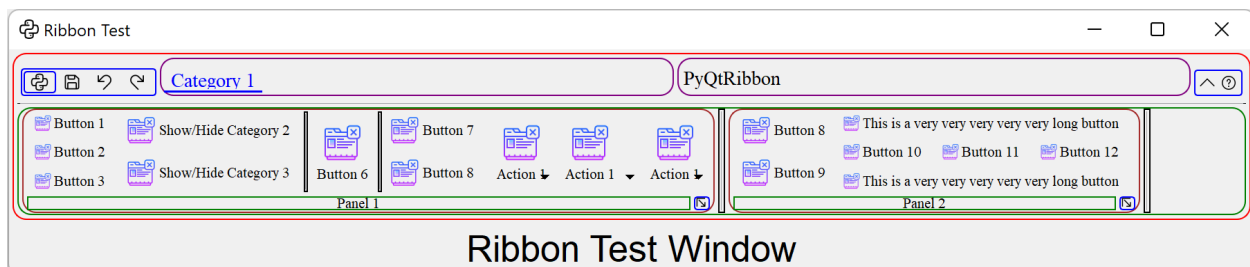


## 2.2 Definitions of Ribbon Elements



- **Application button**: The button that appears on the upper-left corner of a ribbon. The Application button replaces the File menu and is visible even when the ribbon is minimized. When the button is clicked, a menu that has a list of commands is displayed.
- **Quick Access toolbar**: A small, customizable toolbar that displays frequently used commands.
- **Category**: The logical grouping that represents the contents of a ribbon tab.
- **Category Default button**: The button that appears on the ribbon when the ribbon is minimized. When the button is clicked, the category reappears as a menu.
- **Panel**: An area of the ribbon bar that displays a group of related controls. Every ribbon category contains one or more ribbon panels.
- **Ribbon elements**: Controls in the panels, for example, buttons and combo boxes. To see the various controls that can be hosted on a ribbon, see RibbonGadgets Sample: Ribbon Gadgets Application.

## 2.3 Ribbon Elements in PyQtRibbon



## 3.1 Instantiate a Ribbon Bar

*RibbonBar* is inherited from *QMenuBar*, you can use the *setMenuBar* method of *QMainWindow* to set the ribbon bar as the main menu bar.

```
...
from ribbon import RibbonBar

window = QtWidgets.QMainWindow()
ribbon = RibbonBar()
window.setMenuBar(ribbon)
...
```

## 3.2 Customize Ribbon Bar

### 3.2.1 General Setups

<i>RibbonBar.setRibbonStyle</i> (style)	Set the style of the ribbon.
<i>RibbonBar.ribbonHeight</i> ()	Get the total height of the ribbon.
<i>RibbonBar.setRibbonHeight</i> (height)	Set the total height of the ribbon.
<i>RibbonBar.showRibbon</i> ()	Show the ribbon.
<i>RibbonBar.hideRibbon</i> ()	Hide the ribbon.
<i>RibbonBar.ribbonVisible</i> ()	Get the visibility of the ribbon.
<i>RibbonBar.setRibbonVisible</i> (visible)	Set the visibility of the ribbon.

### 3.2.2 Setup Application Button

<i>RibbonBar.applicationOptionButton</i> ()	Return the application button.
<i>RibbonBar.setApplicationIcon</i> (icon)	Set the application icon.
<i>RibbonBar.addFileMenu</i> ()	Add a file menu to the ribbon.

### 3.2.3 Setup Title

<code>RibbonBar.title()</code>	Return the title of the ribbon.
<code>RibbonBar.setTitle(title)</code>	Set the title of the ribbon.

### 3.2.4 Setup Category Tab Bar

<code>RibbonBar.tabBar()</code>	Return the tab bar of the ribbon.
<code>RibbonBar.tabBarHeight()</code>	Get the height of the tab bar.
<code>RibbonBar.setTabBarHeight([height])</code>	Set the height of the tab bar.

### 3.2.5 Setup Quick Access Bar

<code>RibbonBar.quickAccessToolBar()</code>	Return the quick access toolbar of the ribbon.
<code>RibbonBar.addQuickAccessButton(button)</code>	Add a button to the quick access bar.
<code>RibbonBar.setQuickAccessButtonHeight([height])</code>	Set the height of the quick access buttons.

### 3.2.6 Setup Right Tool Bar

<code>RibbonBar.rightToolBar()</code>	Return the right toolbar of the ribbon.
<code>RibbonBar.addRightToolButton(button)</code>	Add a widget to the right button bar.
<code>RibbonBar.setRightToolBarHeight([height])</code>	Set the height of the right buttons.
<code>RibbonBar.setHelpButtonIcon(icon)</code>	Set the icon of the help button.
<code>RibbonBar.removeHelpButton()</code>	Remove the help button from the ribbon.
<code>RibbonBar.helpButtonClicked(bool)</code>	Signal, the help button was clicked.
<code>RibbonBar.collapseRibbonButton()</code>	Return the collapse ribbon button.
<code>RibbonBar.setCollapseButtonIcon(icon)</code>	Set the icon of the min button.
<code>RibbonBar.removeCollapseButton()</code>	Remove the min button from the ribbon.

### 3.2.7 Manage Categories

<i>RibbonBar.categories()</i>	Return a list of categories of the ribbon.
<i>RibbonBar.addCategory</i> (title[, style, color])	Add a new category to the ribbon.
<i>RibbonBar.addCategoriesBy</i> (data)	Add categories from a dict.
<i>RibbonBar.addNormalCategory</i> (title)	Add a new category to the ribbon.
<i>RibbonBar.addContextCategory</i> (title[, color])	Add a new context category to the ribbon.
<i>RibbonBar.addContextCategories</i> (name, titles)	Add a group of context categories with the same tab color to the ribbon.
<i>RibbonBar.showContextCategory</i> (category)	Show the given category or categories, if it is not a context category, nothing happens.
<i>RibbonBar.hideContextCategory</i> (category)	Hide the given category or categories, if it is not a context category, nothing happens.
<i>RibbonBar.removeCategory</i> (category)	Remove a category from the ribbon.
<i>RibbonBar.setCurrentCategory</i> (category)	Set the current category.
<i>RibbonBar.currentCategory</i> ()	Return the current category.
<i>RibbonBar.showCategoryByIndex</i> (index)	Show category by tab index

## 3.3 Customize Categories

### 3.3.1 Setup Styles

<i>RibbonCategory.categoryStyle</i> ()	Return the button style of the category.
<i>RibbonCategory.setCategoryStyle</i> (style)	Set the button style of the category.

### 3.3.2 Manage Panels

<i>RibbonCategory.addPanel</i> (title[, ...])	Add a new panel to the category.
<i>RibbonCategory.addPanelsBy</i> (data)	Add panels from a dictionary.
<i>RibbonCategory.removePanel</i> (title)	Remove a panel from the category.
<i>RibbonCategory.takePanel</i> (title)	Remove and return a panel from the category.
<i>RibbonCategory.panel</i> (title)	Return a panel from the category.
<i>RibbonCategory.panels</i> ()	Return all panels in the category.

## 3.4 Customize Panels

### 3.4.1 Setup Title Label

<i>RibbonPanel.title</i> ()	Get the title of the panel.
<i>RibbonPanel.setTitle</i> (title)	Set the title of the panel.

### 3.4.2 Setup Panel Option Button

<code>RibbonPanel.panelOptionButton()</code>	Return the panel option button.
<code>RibbonPanel.setPanelOptionToolTip(text)</code>	Set the tooltip of the panel option button.
<code>RibbonPanel.panelOptionClicked(bool)</code>	<code>pyqtSignal(*types, name: str = ..., revision: int = ..., arguments: Sequence = ...) -&gt; PYQT_SIGNAL</code>

### 3.4.3 Add Widgets to Panels

<code>RibbonPanel.addWidget(widget[, rowSpan, ...])</code>	Add a widget to the panel.
<code>RibbonPanel.addWidgetsBy(data)</code>	Add widgets to the panel.
<code>RibbonPanel.removeWidget(widget)</code>	Remove a widget from the panel.
<code>RibbonPanel.widget(index)</code>	Get the widget at the given index.
<code>RibbonPanel.widgets()</code>	Get all the widgets in the panel.
<code>RibbonPanel.addSmallWidget(widget[, mode, ...])</code>	Add a small widget to the panel.
<code>RibbonPanel.addMediumWidget(widget[, mode, ...])</code>	Add a medium widget to the panel.
<code>RibbonPanel.addLargeWidget(widget[, mode, ...])</code>	Add a large widget to the panel.
<code>RibbonPanel.addButton([text, icon, style, ...])</code>	Add a button to the panel.
<code>RibbonPanel.addSmallButton([text, icon, ...])</code>	Add a small button to the panel.
<code>RibbonPanel.addMediumButton([text, icon, ...])</code>	Add a medium button to the panel.
<code>RibbonPanel.addLargeButton([text, icon, ...])</code>	Add a large button to the panel.
<code>RibbonPanel.addToggleButton([text, icon, ...])</code>	Add a toggle button to the panel.
<code>RibbonPanel.addSmallToggleButton([text, ...])</code>	Add a small toggle button to the panel.
<code>RibbonPanel.addMediumToggleButton([text, ...])</code>	Add a medium toggle button to the panel.
<code>RibbonPanel.addLargeToggleButton([text, ...])</code>	Add a large toggle button to the panel.
<code>RibbonPanel.addComboBox(items[, rowSpan, ...])</code>	Add a combo box to the panel.
<code>RibbonPanel.addFontComboBox([rowSpan, ...])</code>	Add a font combo box to the panel.
<code>RibbonPanel.addLineEdit([rowSpan, colSpan, ...])</code>	Add a line edit to the panel.
<code>RibbonPanel.addTextEdit([rowSpan, colSpan, ...])</code>	Add a text edit to the panel.
<code>RibbonPanel.addPlainTextEdit([rowSpan, ...])</code>	Add a plain text edit to the panel.
<code>RibbonPanel.addLabel(text[, rowSpan, ...])</code>	Add a label to the panel.
<code>RibbonPanel.addProgressBar([rowSpan, ...])</code>	Add a progress bar to the panel.
<code>RibbonPanel.addSlider([rowSpan, colSpan, ...])</code>	Add a slider to the panel.
<code>RibbonPanel.addSpinBox([rowSpan, colSpan, ...])</code>	Add a spin box to the panel.
<code>RibbonPanel.addDoubleSpinBox([rowSpan, ...])</code>	Add a double spin box to the panel.
<code>RibbonPanel.addDateEdit([rowSpan, colSpan, ...])</code>	Add a date edit to the panel.
<code>RibbonPanel.addTimeEdit([rowSpan, colSpan, ...])</code>	Add a time edit to the panel.
<code>RibbonPanel.addDateTimeEdit([rowSpan, ...])</code>	Add a date time edit to the panel.
<code>RibbonPanel.addTableWidget([rowSpan, ...])</code>	Add a table widget to the panel.
<code>RibbonPanel.addTreeWidget([rowSpan, ...])</code>	Add a tree widget to the panel.
<code>RibbonPanel.addListWidget([rowSpan, ...])</code>	Add a list widget to the panel.
<code>RibbonPanel.addCalendarWidget([rowSpan, ...])</code>	Add a calendar widget to the panel.
<code>RibbonPanel.addSeparator([orientation, ...])</code>	Add a separator to the panel.
<code>RibbonPanel.addHorizontalSeparator([width, ...])</code>	Add a horizontal separator to the panel.
<code>RibbonPanel.addVerticalSeparator([width, ...])</code>	Add a vertical separator to the panel.
<code>RibbonPanel.addGallery([minimumWidth, ...])</code>	Add a gallery to the panel.

## 3.5 A Complete Example

The following code snippet is a complete example.

```
import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel, QWidget, QVBoxLayout
from PyQt5.QtGui import QIcon, QFont
from PyQt5.QtCore import Qt

from ribbon import RibbonBar
from ribbon.utils import data_file_path

if __name__ == "__main__":
    app = QApplication(sys.argv)
    app.setFont(QFont("Times New Roman", 8))

    # Central widget
    window = QMainWindow()
    window.setWindowIcon(QIcon(data_file_path("icons/python.png")))
    centralWidget = QWidget()
    window.setCentralWidget(centralWidget)
    layout = QVBoxLayout(centralWidget)

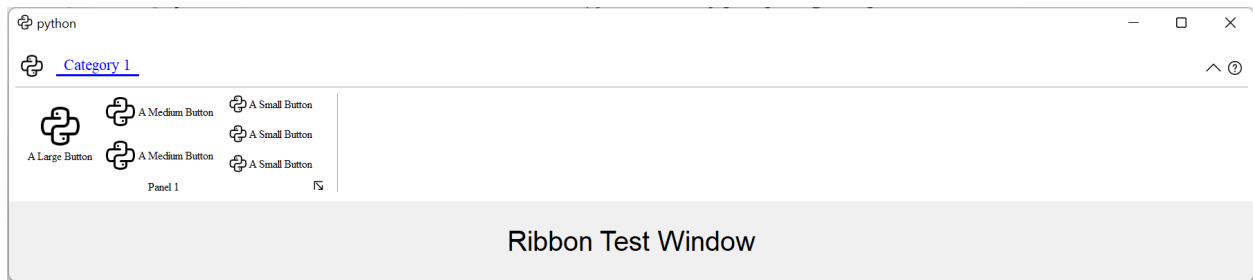
    # Ribbon bar
    ribbonbar = RibbonBar()
    window.setMenuBar(ribbonbar)
    category = ribbonbar.addCategory("Category 1")
    panel = category.addPanel("Panel 1")
    panel.addLargeButton("A Large Button", QIcon(data_file_path("icons/python.png")))
    panel.addMediumButton("A Medium Button", QIcon(data_file_path("icons/python.png")))
    panel.addMediumButton("A Medium Button", QIcon(data_file_path("icons/python.png")))
    panel.addSmallButton("A Small Button", QIcon(data_file_path("icons/python.png")))
    panel.addSmallButton("A Small Button", QIcon(data_file_path("icons/python.png")))
    panel.addSmallButton("A Small Button", QIcon(data_file_path("icons/python.png")))

    # Display a label in the main window
    label = QLabel("Ribbon Test Window")
    label.setFont(QFont("Arial", 20))
    label.setAlignment(Qt.AlignCenter)

    # Add the ribbon bar and label to the layout
    layout.addWidget(label, 1)

    # Show the window
    window.resize(1800, 350)
    window.show()
    sys.exit(app.exec_())
```

It would be rendered as follows:





## API REFERENCES

### 4.1 Ribbon Bar

#### 4.1.1 RibbonBar

**class** ribbon.ribbonbar.**RibbonBar**(*title: str = "", parent=None*)

**class** ribbon.ribbonbar.**RibbonBar**(*parent=None*)

Bases: `QMenuBar`

The `RibbonBar` class is the top level widget that contains the ribbon.

#### Methods

<hr/> <i>actionAt</i> (self, <code>QPoint</code> ) <hr/>	
<hr/> <i>actionGeometry</i> (self, <code>QAction</code> ) <hr/>	
<hr/> <i>activeAction</i> (self) <hr/>	
<hr/> <i>addAction</i> (-> <code>QAction</code> ) <hr/>	
<hr/> <i>addCategoriesBy</i> (data)	Add categories from a dict.
<hr/> <i>addCategory</i> (title[, style, color])	Add a new category to the ribbon.
<hr/> <i>addContextCategories</i> (name, titles[, color])	Add a group of context categories with the same tab color to the ribbon.
<hr/> <i>addContextCategory</i> (title[, color])	Add a new context category to the ribbon.
<hr/> <i>addFileMenu</i> ()	Add a file menu to the ribbon.
<hr/> <i>addMenu</i> (-> <code>QAction</code> -> <code>QMenu</code> ) <hr/>	
<hr/> <i>addNormalCategory</i> (title)	Add a new category to the ribbon.
<hr/> <i>addQuickAccessButton</i> (button)	Add a button to the quick access bar.
<hr/> <i>addRightToolButton</i> (button)	Add a widget to the right button bar.
<hr/> <i>addSeparator</i> (self) <hr/>	
<hr/> <i>applicationOptionButton</i> ()	Return the application button.
<hr/> <i>categories</i> ()	Return a list of categories of the ribbon.
<hr/> <i>category</i> (name)	Return the category with the given name.
<hr/> <i>categoryVisible</i> (category)	Return whether the category is shown.

continues on next page

Table 1 – continued from previous page

<i>clear(self)</i>	
<i>collapseRibbonButton()</i>	Return the collapse ribbon button.
<i>cornerWidget(self[, corner])</i>	
<i>currentCategory()</i>	Return the current category.
<i>helpRibbonButton()</i>	Return the help button of the ribbon.
<i>hideContextCategory(category)</i>	Hide the given category or categories, if it is not a context category, nothing happens.
<i>hideRibbon()</i>	Hide the ribbon.
<i>insertMenu(self, QAction, QMenu)</i>	
<i>insertSeparator(self, QAction)</i>	
<i>isDefaultUp(self)</i>	
<i>isNativeMenuBar(self)</i>	
<i>minimumSizeHint()</i>	Return the minimum size hint of the widget.
<i>quickAccessToolBar()</i>	Return the quick access toolbar of the ribbon.
<i>removeCategories(categories)</i>	Remove a list of categories from the ribbon.
<i>removeCategory(category)</i>	Remove a category from the ribbon.
<i>removeCollapseButton()</i>	Remove the min button from the ribbon.
<i>removeHelpButton()</i>	Remove the help button from the ribbon.
<i>ribbonHeight()</i>	Get the total height of the ribbon.
<i>ribbonVisible()</i>	Get the visibility of the ribbon.
<i>rightToolBar()</i>	Return the right toolbar of the ribbon.
<i>setActiveAction(self, QAction)</i>	
<i>setApplicationIcon(icon)</i>	Set the application icon.
<i>setCollapseButtonIcon(icon)</i>	Set the icon of the min button.
<i>setCornerWidget(self, QWidget[, corner])</i>	
<i>setCurrentCategory(category)</i>	Set the current category.
<i>setDefaultUp(self, bool)</i>	
<i>setHelpButtonIcon(icon)</i>	Set the icon of the help button.
<i>setNativeMenuBar(self, bool)</i>	
<i>setQuickAccessButtonHeight([height])</i>	Set the height of the quick access buttons.
<i>setRibbonHeight(height)</i>	Set the total height of the ribbon.
<i>setRibbonStyle(style)</i>	Set the style of the ribbon.
<i>setRibbonVisible(visible)</i>	Set the visibility of the ribbon.
<i>setRightToolBarHeight([height])</i>	Set the height of the right buttons.
<i>setTabBarHeight([height])</i>	Set the height of the tab bar.
<i>setTitle(title)</i>	Set the title of the ribbon.
<i>showCategoryByIndex(index)</i>	Show category by tab index
<i>showContextCategory(category)</i>	Show the given category or categories, if it is not a context category, nothing happens.
<i>showRibbon()</i>	Show the ribbon.
<i>tabBar()</i>	Return the tab bar of the ribbon.

continues on next page

Table 1 – continued from previous page

<code>tabBarHeight()</code>	Get the height of the tab bar.
<code>title()</code>	Return the title of the ribbon.

helpButtonClicked	
-------------------	--

**actionAt**(*self*, *QPoint*) → *QAction***actionGeometry**(*self*, *QAction*) → *QRect***activeAction**(*self*) → *QAction***addAction**(*self*, *QAction*)**addAction**(*self*, *str*) → *QAction***addAction**(*self*, *str*, *PYQT\_SLOT*) → *QAction***addCategoriesBy**(*data*: *Dict[str, Dict]*) → *Dict[str, RibbonCategory]*

Add categories from a dict.

**Parameters** **data** – The dict of categories. The dict is of the form:

```
{
    "category-title": { "style": RibbonCategoryStyle.Normal, "color": QtCore.Qt.red, "panels":
        "panel-title": { "showPanelOptionButton": True, "widgets": {
            "widget-name": { "type": "Button", "arguments": {
                "key1": "value1", "key2": "value2"
            }
        },
    },
    },
    },
}
```

**Returns** A dict of categories of the ribbon.**addCategory**(*title*: *str*, *style*=*RibbonCategoryStyle.Normal*, *color*: *Optional[QColor]* = *None*) → *Union[RibbonNormalCategory, RibbonContextCategory]*

Add a new category to the ribbon.

**Parameters**

- **title** – The title of the category.
- **style** – The button style of the category.
- **color** – The color of the context category, only used if style is Context, if None, the default color will be used.

**Returns** The newly created category.

**addContextCategories**(*name: str, titles: List[str], color: Optional[Union[QColor, GlobalColor]] = None*) → *RibbonContextCategories*

Add a group of context categories with the same tab color to the ribbon.

**Parameters**

- **name** – The name of the context categories.
- **titles** – The title of the category.
- **color** – The color of the context category, if None, the default color will be used.

**Returns** The newly created category.

**addContextCategory**(*title: str, color: Optional[Union[QColor, GlobalColor]] = None*) → *RibbonContextCategory*

Add a new context category to the ribbon.

**Parameters**

- **title** – The title of the category.
- **color** – The color of the context category, if None, the default color will be used.

**Returns** The newly created category.

**addFileMenu**() → *RibbonMenu*

Add a file menu to the ribbon.

**addMenu**(*self, QMenu*) → *QAction*

**addMenu**(*self, str*) → *QMenu*

**addMenu**(*self, QIcon, str*) → *QMenu*

**addNormalCategory**(*title: str*) → *RibbonNormalCategory*

Add a new category to the ribbon.

**Parameters** **title** – The title of the category.

**Returns** The newly created category.

**addQuickAccessButton**(*button: QToolButton*)

Add a button to the quick access bar.

**Parameters** **button** – The button to add.

**addRightToolButton**(*button: QToolButton*)

Add a widget to the right button bar.

**Parameters** **button** – The button to add.

**addSeparator**(*self*) → *QAction*

**applicationOptionButton**() → *RibbonApplicationButton*

Return the application button.

**categories**() → Dict[str, *RibbonCategory*]

Return a list of categories of the ribbon.

**Returns** A dict of categories of the ribbon.

**category**(*name: str*) → *RibbonCategory*

Return the category with the given name.

**Parameters** **name** – The name of the category.

**Returns** The category with the given name.

**categoryVisible**(*category: RibbonCategory*) → bool

Return whether the category is shown.

**Parameters** **category** – The category to check.

**Returns** Whether the category is shown.

**clear**(*self*)

**collapseRibbonButton**() → *QToolButton*

Return the collapse ribbon button.

**Returns** The collapse ribbon button.

**cornerWidget**(*self*, *corner*: *Corner* = *Qt.TopRightCorner*) → *QWidget*

**currentCategory**() → *RibbonCategory*

Return the current category.

**Returns** The current category.

**helpButtonClicked**(*bool*)

Signal, the help button was clicked.

**helpRibbonButton**() → *QToolButton*

Return the help button of the ribbon.

**Returns** The help button of the ribbon.

**hideContextCategory**(*category*: *Union*[*RibbonContextCategory*, *RibbonContextCategories*])

Hide the given category or categories, if it is not a context category, nothing happens.

**Parameters** **category** – The category to hide.

**hideRibbon**()

Hide the ribbon.

**insertMenu**(*self*, *QAction*, *QMenu*) → *QAction*

**insertSeparator**(*self*, *QAction*) → *QAction*

**isDefaultUp**(*self*) → *bool*

**isNativeMenuBar**(*self*) → *bool*

**minimumSizeHint**() → *QSize*

Return the minimum size hint of the widget.

**Returns** The minimum size hint.

**quickAccessToolBar**() → *QToolBar*

Return the quick access toolbar of the ribbon.

**Returns** The quick access toolbar of the ribbon.

**removeCategories**(*categories*: *RibbonContextCategories*)

Remove a list of categories from the ribbon.

**Parameters** **categories** – The categories to remove.

**removeCategory**(*category*: *RibbonCategory*)

Remove a category from the ribbon.

**Parameters** **category** – The category to remove.

**removeCollapseButton**()

Remove the min button from the ribbon.

**removeHelpButton**()

Remove the help button from the ribbon.

**ribbonHeight()** → int

Get the total height of the ribbon.

**Returns** The height of the ribbon.

**ribbonVisible()** → bool

Get the visibility of the ribbon.

**Returns** True if the ribbon is visible, False otherwise.

**rightToolBar()** → [QToolBar](#)

Return the right toolbar of the ribbon.

**Returns** The right toolbar of the ribbon.

**setActiveAction(self, QAction)**

**setApplicationIcon(icon: QIcon)**

Set the application icon.

**Parameters** **icon** – The icon to set.

**setCollapseButtonIcon(icon: QIcon)**

Set the icon of the min button.

**Parameters** **icon** – The icon to set.

**setCornerWidget(self, QWidget, corner: Corner = Qt.TopRightCorner)**

**setCurrentCategory(category: RibbonCategory)**

Set the current category.

**Parameters** **category** – The category to set.

**setDefaultUp(self, bool)**

**setHelpButtonIcon(icon: QIcon)**

Set the icon of the help button.

**Parameters** **icon** – The icon to set.

**setNativeMenuBar(self, bool)**

**setQuickAccessButtonHeight(height: int = 40)**

Set the height of the quick access buttons.

**Parameters** **height** – The height to set.

**setRibbonHeight(height: int)**

Set the total height of the ribbon.

**Parameters** **height** – The height to set.

**setRibbonStyle(style: RibbonStyle)**

Set the style of the ribbon.

**Parameters** **style** – The style to set.

**setRibbonVisible(visible: bool)**

Set the visibility of the ribbon.

**Parameters** **visible** – True to show the ribbon, False to hide it.

**setRightToolBarHeight(height: int = 24)**

Set the height of the right buttons.

**Parameters** **height** – The height to set.

**setTabBarHeight**(*height: int = 50*)

Set the height of the tab bar.

**Parameters** **height** – The height to set.

**setTitle**(*title: str*)

Set the title of the ribbon.

**Parameters** **title** – The title to set.

**showCategoryByIndex**(*index: int*)

Show category by tab index

**Parameters** **index** – tab index

**showContextCategory**(*category: Union[RibbonContextCategory, RibbonContextCategories]*)

Show the given category or categories, if it is not a context category, nothing happens.

**Parameters** **category** – The category to show.

**showRibbon**()

Show the ribbon.

**tabBar**() → *RibbonTabBar*

Return the tab bar of the ribbon.

**Returns** The tab bar of the ribbon.

**tabBarHeight**() → int

Get the height of the tab bar.

**Returns** The height of the tab bar.

**title**() → str

Return the title of the ribbon.

**Returns** The title of the ribbon.

## 4.2 Ribbon Title

### 4.2.1 RibbonApplicationButton

**class** ribbon.titlewidget.**RibbonApplicationButton**

Bases: *QToolButton*

Application button in the ribbon bar.

#### Methods

---

*addFileMenu*()

Add a new ribbon menu to the application button.

---

**addFileMenu**() → *RibbonMenu*

Add a new ribbon menu to the application button.

**Returns** The new ribbon menu.

## 4.2.2 RibbonTabBar

**class** ribbon.tabbar.RibbonTabBar(*parent=None*)

Bases: [QTabBar](#)

The TabBar for the title widget.

### Methods

<a href="#">addAssociatedTabs</a> (name, texts, color)	Add associated multiple tabs which have the same color to the tab bar.
<a href="#">addTab</a> (text[, color])	Add a new tab to the tab bar.
<a href="#">currentTabColor</a> ()	Current tab color
<a href="#">indexOf</a> (tabName)	Return the index of the tab with the given name.
<a href="#">paintEvent</a> (a0)	Paint the tab bar.
<a href="#">removeAssociatedTabs</a> (titles)	Remove tabs with the given titles.
<a href="#">tabTitles</a> ()	Return the titles of all tabs.

**addAssociatedTabs**(*name: str, texts: List[str], color: QColor*) → List[int]

Add associated multiple tabs which have the same color to the tab bar.

**Parameters**

- **name** – The name of the context category.
- **texts** – The texts of the tabs.
- **color** – The color of the tabs.

**Returns** The indices of the tabs.

**addTab**(*text: str, color: Optional[QColor] = None*) → int

Add a new tab to the tab bar.

**Parameters**

- **text** – The text of the tab.
- **color** – The color of the tab.

**Returns** The index of the tab.

**currentTabColor**() → QColor

Current tab color

**Returns** Current tab color

**indexOf**(*tabName: str*) → int

Return the index of the tab with the given name.

**Parameters** **tabName** – The name of the tab.

**Returns** The index of the tab.

**paintEvent**(*a0: QPaintEvent*) → None

Paint the tab bar.

**removeAssociatedTabs**(*titles: List[str]*) → None

Remove tabs with the given titles.

**Parameters** **titles** – The titles of the tabs to remove.

**tabTitles**() → List[str]

Return the titles of all tabs.

**Returns** The titles of all tabs.



### 4.2.3 RibbonTitleLabel

**class** ribbon.titlewidget.RibbonTitleLabel

Bases: QLabel

Title label in the ribbon bar.

### 4.2.4 RibbonTitleWidget

**class** ribbon.titlewidget.RibbonTitleWidget(*title='PyQtRibbon', parent=None*)

**class** ribbon.titlewidget.RibbonTitleWidget(*parent=None*)

Bases: QFrame

The title widget of the ribbon.

#### Methods

<a href="#"><i>addQuickAccessButton</i></a> (button)	Add a widget to the quick access bar.
<a href="#"><i>addRightToolButton</i></a> (button)	Add a widget to the right button bar.
<a href="#"><i>applicationButton</i></a> ()	Return the application button.
<a href="#"><i>collapseRibbonButton</i></a> ()	Return the collapse ribbon button.
<a href="#"><i>helpRibbonButton</i></a> ()	Return the help ribbon button.
<a href="#"><i>quickAccessButtons</i></a> ()	Return the quick access buttons of the ribbon.
<a href="#"><i>quickAccessToolBar</i></a> ()	Return the quick access toolbar of the ribbon.
<a href="#"><i>removeCollapseButton</i></a> ()	Remove the min button from the ribbon.
<a href="#"><i>removeHelpButton</i></a> ()	Remove the help button from the ribbon.
<a href="#"><i>rightToolBar</i></a> ()	Return the right toolbar of the ribbon.
<a href="#"><i>setApplicationIcon</i></a> (icon)	Set the application icon.
<a href="#"><i>setCollapseButtonIcon</i></a> (icon)	Set the icon of the min button.
<a href="#"><i>setHelpButtonIcon</i></a> (icon)	Set the icon of the help button.
<a href="#"><i>setQuickAccessButtonHeight</i></a> ([height])	Set the height of the quick access buttons.
<a href="#"><i>setRightToolBarHeight</i></a> ([height])	Set the height of the right buttons.
<a href="#"><i>setTabBarHeight</i></a> ([height])	Set the height of the tab bar.
<a href="#"><i>setTitle</i></a> (title)	Set the title of the ribbon.
<a href="#"><i>tabBar</i></a> ()	Return the tab bar of the ribbon.
<a href="#"><i>tabBarHeight</i></a> ()	Get the height of the tab bar.
<a href="#"><i>title</i></a> ()	Return the title of the ribbon.

<b>collapseRibbonButtonClicked</b>	
<b>helpButtonClicked</b>	

**addQuickAccessButton**(*button: QToolButton*)

Add a widget to the quick access bar.

**Parameters** **button** – The button to add.

**addRightToolButton**(*button: QToolButton*)

Add a widget to the right button bar.

**Parameters** **button** – The button to add.

**applicationButton()** → *RibbonApplicationButton*

Return the application button.

**collapseRibbonButton()** → *QToolButton*

Return the collapse ribbon button.

**Returns** The collapse ribbon button.

**collapseRibbonButtonClicked(*bool*)**

Signal, the collapse button was clicked.

**helpButtonClicked(*bool*)**

Signal, the help button was clicked.

**helpRibbonButton()** → *QToolButton*

Return the help ribbon button.

**Returns** The help ribbon button.

**quickAccessButtons()** → List[*QToolButton*]

Return the quick access buttons of the ribbon.

**Returns** The quick access buttons of the ribbon.

**quickAccessToolBar()** → *QToolBar*

Return the quick access toolbar of the ribbon.

**Returns** The quick access toolbar of the ribbon.

**removeCollapseButton()**

Remove the min button from the ribbon.

**removeHelpButton()**

Remove the help button from the ribbon.

**rightToolBar()** → *QToolBar*

Return the right toolbar of the ribbon.

**Returns** The right toolbar of the ribbon.

**setApplicationIcon(*icon: QIcon*)**

Set the application icon.

**Parameters** **icon** – The icon to set.

**setCollapseButtonIcon(*icon: QIcon*)**

Set the icon of the min button.

**Parameters** **icon** – The icon to set.

**setHelpButtonIcon(*icon: QIcon*)**

Set the icon of the help button.

**Parameters** **icon** – The icon to set.

**setQuickAccessButtonHeight(*height: int = 40*)**

Set the height of the quick access buttons.

**Parameters** **height** – The height to set.

**setRightToolBarHeight(*height: int = 24*)**

Set the height of the right buttons.

**Parameters** **height** – The height to set.

**setTabBarHeight(*height: int = 50*)**

Set the height of the tab bar.

**Parameters** **height** – The height to set.

**setTitle**(*title: str*)

Set the title of the ribbon.

**Parameters** *title* – The title to set.

**tabBar**() → *RibbonTabBar*

Return the tab bar of the ribbon.

**Returns** The tab bar of the ribbon.

**tabBarHeight**() → int

Get the height of the tab bar.

**Returns** The height of the tab bar.

**title**() → str

Return the title of the ribbon.

**Returns** The title of the ribbon.

Ribbon Stacked Widget

## 4.2.5 RibbonStackedWidget

**class** ribbon.ribbonbar.**RibbonStackedWidget**(*parent=None*)

Bases: *QStackedWidget*

Stacked widget that is used to display the ribbon.

## 4.3 Ribbon Category

### 4.3.1 RibbonCategory

**class** ribbon.category.**RibbonCategory**(*title: str = "", style: RibbonCategoryStyle = RibbonCategoryStyle.Normal, color: QColor = None, parent=None*)

**class** ribbon.category.**RibbonCategory**(*parent=None*)

Bases: *QFrame*

The RibbonCategory is the logical grouping that represents the contents of a ribbon tab.

#### Methods

<i>addPanel</i> ( <i>title[, showPanelOptionButton]</i> )	Add a new panel to the category.
<i>addPanelsBy</i> ( <i>data</i> )	Add panels from a dictionary.
<i>categoryStyle</i> ()	Return the button style of the category.
<i>panel</i> ( <i>title</i> )	Return a panel from the category.
<i>panels</i> ()	Return all panels in the category.
<i>removePanel</i> ( <i>title</i> )	Remove a panel from the category.
<i>setCategoryStyle</i> ( <i>style</i> )	Set the button style of the category.
<i>takePanel</i> ( <i>title</i> )	Remove and return a panel from the category.
<i>title</i> ()	Return the title of the category.

**addPanel**(*title: str, showPanelOptionButton=True*) → *RibbonPanel*

Add a new panel to the category.

**Parameters**

- **title** – The title of the panel.
- **showPanelOptionButton** – Whether to show the panel option button.

**Returns** The newly created panel.

**addPanelsBy**(*data: Dict[str, Dict]*) → Dict[str, *RibbonPanel*]

Add panels from a dictionary.

**Parameters** **data** – The dictionary. The keys are the titles of the panels. The value is a dictionary of arguments. the argument `showPanelOptionButton` is a boolean to decide whether to show the panel option button, the rest arguments are passed to the `RibbonPanel.addWidgetsBy()` method. The dict is of the form: {

```
    "panel-title": { "showPanelOptionButton": True, "wid-
                    gets": {
                        "widget-name": { "type": "Button", "argu-
                                         ments": {
                                             "key1": "value1", "key2": "value2"
                                         }
                    },
                },
            }
```

**Returns** A dictionary of the newly created panels.

**categoryStyle**() → *RibbonCategoryStyle*

Return the button style of the category.

**Returns** The button style.

**panel**(*title: str*) → *RibbonPanel*

Return a panel from the category.

**Parameters** **title** – The title of the panel.

**Returns** The panel.

**panels**() → Dict[str, *RibbonPanel*]

Return all panels in the category.

**Returns** The panels.

**removePanel**(*title: str*)

Remove a panel from the category.

**Parameters** **title** – The title of the panel.

**setCategoryStyle**(*style: RibbonCategoryStyle*)

Set the button style of the category.

**Parameters** **style** – The button style.

**takePanel**(*title: str*) → *RibbonPanel*

Remove and return a panel from the category.

**Parameters** **title** – The title of the panel.

**Returns** The removed panel.

**title**() → str

Return the title of the category.

### 4.3.2 RibbonNormalCategory

**class** ribbon.category.RibbonNormalCategory(*title: str, parent: QWidget*)

Bases: *RibbonCategory*

A normal category.

#### Methods

<i>setCategoryStyle</i> (style)	Set the button style of the category.
---------------------------------	---------------------------------------

**setCategoryStyle**(*style: RibbonCategoryStyle*)

Set the button style of the category.

**Parameters** **style** – The button style.

### 4.3.3 RibbonContextCategory

**class** ribbon.category.RibbonContextCategory(*title: str, color: QColor, parent: QWidget*)

Bases: *RibbonCategory*

A context category.

#### Methods

<i>categoryVisible</i> ()	Return whether the category is shown.
<i>color</i> ()	Return the color of the context category.
<i>hideContextCategory</i> ()	Hide the given category, if it is not a context category, nothing happens.
<i>setCategoryStyle</i> (style)	Set the button style of the category.
<i>setCategoryVisible</i> (visible)	Set the state of the category.
<i>setColor</i> (color)	Set the color of the context category.
<i>showContextCategory</i> ()	Show the given category, if it is not a context category, nothing happens.

**categoryVisible**() → bool

Return whether the category is shown.

**Returns** Whether the category is shown.

**color**() → QColor

Return the color of the context category.

**Returns** The color of the context category.

**hideContextCategory**()

Hide the given category, if it is not a context category, nothing happens.

**setCategoryStyle**(*style: RibbonCategoryStyle*)

Set the button style of the category.

**Parameters** **style** – The button style.

**setCategoryVisible**(*visible: bool*)

Set the state of the category.

**Parameters** **visible** – The state.

**setColor**(*color: QColor*)

Set the color of the context category.

**Parameters** **color** – The color of the context category.

**showContextCategory**()

Show the given category, if it is not a context category, nothing happens.

### 4.3.4 RibbonContextCategories

**class** ribbon.category.**RibbonContextCategories**(*name: str, color: QColor, categories: Dict[str, RibbonContextCategory], ribbon*)

Bases: Dict[str, [RibbonContextCategory](#)]

A list of context categories.

#### Methods

<a href="#">categoriesVisible</a> ()	Return whether the categories are shown.
<a href="#">color</a> ()	Return the color of the context categories.
<a href="#">hideContextCategories</a> ()	Hide the categories
<a href="#">name</a> ()	Return the name of the context categories.
<a href="#">setCategoriesVisible</a> ( <i>visible</i> )	Set the state of the categories.
<a href="#">setColor</a> ( <i>color</i> )	Set the color of the context categories.
<a href="#">setName</a> ( <i>name</i> )	Set the name of the context categories.
<a href="#">showContextCategories</a> ()	Show the categories

**categoriesVisible**() → bool

Return whether the categories are shown.

**color**() → [QColor](#)

Return the color of the context categories.

**hideContextCategories**()

Hide the categories

**name**() → str

Return the name of the context categories.

**setCategoriesVisible**(*visible: bool*)

Set the state of the categories.

**setColor**(*color: QColor*)

Set the color of the context categories.

**setName**(*name: str*)

Set the name of the context categories.

**showContextCategories**()

Show the categories

### 4.3.5 RibbonCategoryStyle

**class** ribbon.category.RibbonCategoryStyle(*value*)

Bases: IntEnum

The button style of a category.

### 4.3.6 RibbonCategoryScrollArea

**class** ribbon.categorylayoutwidget.RibbonCategoryScrollArea

Bases: QScrollArea

Scroll area for the gallery

### 4.3.7 RibbonCategoryScrollAreaContents

**class** ribbon.categorylayoutwidget.RibbonCategoryScrollAreaContents

Bases: QFrame

Scroll area contents for the gallery

### 4.3.8 RibbonCategoryLayoutButton

**class** ribbon.categorylayoutwidget.RibbonCategoryLayoutButton

Bases: QToolButton

Previous/Next buttons in the category when the size is not enough for the widgets.

### 4.3.9 RibbonCategoryLayoutWidget

**class** ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget(*parent=None*)

Bases: QFrame

The category layout widget's category scroll area to arrange the widgets in the category.

#### Methods

<a href="#"><i>addWidget(widget)</i></a>	Add a widget to the category layout.
<a href="#"><i>autoSetScrollButtonsVisible()</i></a>	Set the visibility of the scroll buttons.
<a href="#"><i>paintEvent(a0)</i></a>	Override the paint event to draw the background.
<a href="#"><i>removeWidget(widget)</i></a>	Remove a widget from the category layout.
<a href="#"><i>resizeEvent(a0)</i></a>	Override the resize event to resize the scroll area.
<a href="#"><i>scrollNext()</i></a>	Scroll the category to the next widget.
<a href="#"><i>scrollPrevious()</i></a>	Scroll the category to the previous widget.
<a href="#"><i>takeWidget(widget)</i></a>	Remove and return a widget from the category layout.

displayOptionsButtonClicked	
-----------------------------	--

**addWidget**(*widget: QWidget*)

Add a widget to the category layout.

**Parameters** **widget** – The widget to add.

**autoSetScrollButtonsVisible**()

Set the visibility of the scroll buttons.

**paintEvent**(*a0: QPaintEvent*) → None

Override the paint event to draw the background.

**removeWidget**(*widget: QWidget*)

Remove a widget from the category layout.

**Parameters** **widget** – The widget to remove.

**resizeEvent**(*a0: QResizeEvent*) → None

Override the resize event to resize the scroll area.

**scrollNext**()

Scroll the category to the next widget.

**scrollPrevious**()

Scroll the category to the previous widget.

**takeWidget**(*widget: QWidget*) → *QWidget*

Remove and return a widget from the category layout.

**Parameters** **widget** – The widget to remove.

**Returns** The widget that was removed.

## 4.4 Ribbon Panel

### 4.4.1 RibbonPanel

```
class ribbon.panel.RibbonPanel(title: str = "", maxRows: int = 6, showPanelOptionButton=True,  
                               parent=None)
```

```
class ribbon.panel.RibbonPanel(parent=None)
```

Bases: *QFrame*

Panel in the ribbon category.

#### Methods

<a href="#"><i>addButton</i></a> ([ <i>text, icon, style, showText, ...</i> ])	Add a button to the panel.
<a href="#"><i>addCalendarWidget</i></a> ([ <i>rowSpan, colSpan, mode, ...</i> ])	Add a calendar widget to the panel.
<a href="#"><i>addComboBox</i></a> ([ <i>items, rowSpan, colSpan, mode, ...</i> ])	Add a combo box to the panel.
<a href="#"><i>addDateEdit</i></a> ([ <i>rowSpan, colSpan, mode, alignment</i> ])	Add a date edit to the panel.
<a href="#"><i>addDateTimeEdit</i></a> ([ <i>rowSpan, colSpan, mode, ...</i> ])	Add a date time edit to the panel.
<a href="#"><i>addDoubleSpinBox</i></a> ([ <i>rowSpan, colSpan, mode, ...</i> ])	Add a double spin box to the panel.
<a href="#"><i>addFontComboBox</i></a> ([ <i>rowSpan, colSpan, mode, ...</i> ])	Add a font combo box to the panel.

continues on next page



Table 2 – continued from previous page

<code>addGallery</code> ([minimumWidth, popupHideOnClick, ...])	Add a gallery to the panel.
<code>addHorizontalSeparator</code> ([width, rowSpan, ...])	Add a horizontal separator to the panel.
<code>addLabel</code> (text[, rowSpan, colSpan, mode, ...])	Add a label to the panel.
<code>addLargeButton</code> ([text, icon, showText, ...])	Add a large button to the panel.
<code>addLargeToggleButton</code> ([text, icon, showText, ...])	Add a large toggle button to the panel.
<code>addLargeWidget</code> (widget[, mode, alignment])	Add a large widget to the panel.
<code>addLineEdit</code> ([rowSpan, colSpan, mode, alignment])	Add a line edit to the panel.
<code>addListWidget</code> ([rowSpan, colSpan, mode, ...])	Add a list widget to the panel.
<code>addMediumButton</code> ([text, icon, showText, ...])	Add a medium button to the panel.
<code>addMediumToggleButton</code> ([text, icon, ...])	Add a medium toggle button to the panel.
<code>addMediumWidget</code> (widget[, mode, alignment])	Add a medium widget to the panel.
<code>addPlainTextEdit</code> ([rowSpan, colSpan, mode, ...])	Add a plain text edit to the panel.
<code>addProgressBar</code> ([rowSpan, colSpan, mode, ...])	Add a progress bar to the panel.
<code>addSeparator</code> ([orientation, width, rowSpan, ...])	Add a separator to the panel.
<code>addSlider</code> ([rowSpan, colSpan, mode, alignment])	Add a slider to the panel.
<code>addSmallButton</code> ([text, icon, showText, ...])	Add a small button to the panel.
<code>addSmallToggleButton</code> ([text, icon, showText, ...])	Add a small toggle button to the panel.
<code>addSmallWidget</code> (widget[, mode, alignment])	Add a small widget to the panel.
<code>addSpinBox</code> ([rowSpan, colSpan, mode, alignment])	Add a spin box to the panel.
<code>addTableWidget</code> ([rowSpan, colSpan, mode, ...])	Add a table widget to the panel.
<code>addTextEdit</code> ([rowSpan, colSpan, mode, alignment])	Add a text edit to the panel.
<code>addTimeEdit</code> ([rowSpan, colSpan, mode, alignment])	Add a time edit to the panel.
<code>addToggleButton</code> ([text, icon, style, ...])	Add a toggle button to the panel.
<code>addTreeWidget</code> ([rowSpan, colSpan, mode, ...])	Add a tree widget to the panel.
<code>addVerticalSeparator</code> ([width, rowSpan, ...])	Add a vertical separator to the panel.
<code>addWidget</code> (widget[, rowSpan, colSpan, mode, ...])	Add a widget to the panel.
<code>addWidgetBy</code> (data)	Add widgets to the panel.
<code>panelOptionButton</code> ()	Return the panel option button.
<code>removeWidget</code> (widget)	Remove a widget from the panel.
<code>rowHeight</code> ()	Return the height of a row.
<code>setPanelOptionToolTip</code> (text)	Set the tooltip of the panel option button.
<code>setTitle</code> (title)	Set the title of the panel.
<code>title</code> ()	Get the title of the panel.
<code>widget</code> (index)	Get the widget at the given index.
<code>widgets</code> ()	Get all the widgets in the panel.

panelOptionClicked	
--------------------	--

**addButton**(text: Optional[str] = None, icon: Optional[QIcon] = None, style: RibbonButtonStyle = RibbonButtonStyle.Large, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132) → RibbonToolButton

Add a button to the panel.

#### Parameters

- **text** – The text of the button.
- **icon** – The icon of the button.

- **style** – The style of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addCalendarWidget**(*rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QCalendarWidget](#)

Add a calendar widget to the panel.

**Parameters**

- **rowSpan** – The number of rows the calendar widget should span.
- **colSpan** – The number of columns the calendar widget should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the calendar widget.

**Returns** The calendar widget that was added.

**addComboBox**(*items: List[str], rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QComboBox](#)

Add a combo box to the panel.

**Parameters**

- **items** – The items of the combo box.
- **rowSpan** – The number of rows the combo box should span.
- **colSpan** – The number of columns the combo box should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the combo box.

**Returns** The combo box that was added.

**addDateEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QDateEdit](#)

Add a date edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the date edit should span.
- **colSpan** – The number of columns the date edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the date edit.

**Returns** The date edit that was added.

**addDateTimeEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QDateTimeEdit](#)

Add a date time edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the date time edit should span.
- **colSpan** – The number of columns the date time edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the date time edit.

**Returns** The date time edit that was added.

**addDoubleSpinBox**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QDoubleSpinBox](#)

Add a double spin box to the panel.

**Parameters**

- **rowSpan** – The number of rows the double spin box should span.
- **colSpan** – The number of columns the double spin box should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the double spin box.

**Returns** The double spin box that was added.

**addFontComboBox**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QFontComboBox](#)

Add a font combo box to the panel.

**Parameters**

- **rowSpan** – The number of rows the combo box should span.
- **colSpan** – The number of columns the combo box should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the combo box.

**Returns** The combo box that was added.

**addGallery**(*minimumWidth=800, popupHideOnClick=False, rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [RibbonGallery](#)

Add a gallery to the panel.

**Parameters**

- **minimumWidth** – The minimum width of the gallery.
- **popupHideOnClick** – Whether the gallery popup should be hidden when a user clicks on it.
- **rowSpan** – The number of rows the gallery spans.
- **colSpan** – The number of columns the gallery spans.
- **mode** – The mode of the gallery.
- **alignment** – The alignment of the gallery.

**Returns** The gallery.

**addHorizontalSeparator**(*width=6, rowSpan: int = 1, colSpan: int = 2, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [RibbonHorizontalSeparator](#)

Add a horizontal separator to the panel.

**Parameters**

- **width** – The width of the separator.
- **rowSpan** – The number of rows the separator spans.
- **colSpan** – The number of columns the separator spans.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the separator.

**Returns** The separator.

**addLabel**(*text: str, rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QLabel](#)

Add a label to the panel.

**Parameters**

- **text** – The text of the label.
- **rowSpan** – The number of rows the label should span.
- **colSpan** – The number of columns the label should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the label.

**Returns** The label that was added.

**addLargeButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [RibbonToolButton](#)

Add a large button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addLargeToggleButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *RibbonToolButton*

Add a large toggle button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addLargeWidget**(*widget: QWidget, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*)

Add a large widget to the panel.

**Parameters**

- **widget** – The widget to add.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the widget.

**addLineEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *QLineEdit*

Add a line edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the line edit should span.
- **colSpan** – The number of columns the line edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the line edit.

**Returns** The line edit that was added.

**addListWidget**(*rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *QListWidget*

Add a list widget to the panel.

**Parameters**

- **rowSpan** – The number of rows the list widget should span.

- **colSpan** – The number of columns the list widget should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the list widget.

**Returns** The list widget that was added.

**addMediumButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *RibbonToolButton*

Add a medium button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addMediumToggleButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *RibbonToolButton*

Add a medium toggle button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addMediumWidget**(*widget: QWidget, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*)

Add a medium widget to the panel.

**Parameters**

- **widget** – The widget to add.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the widget.

**addPlainTextEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → *QPlainTextEdit*

Add a plain text edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the text edit should span.
- **colSpan** – The number of columns the text edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the text edit.

**Returns** The text edit that was added.

**addProgressBar**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QProgressBar](#)

Add a progress bar to the panel.

**Parameters**

- **rowSpan** – The number of rows the progress bar should span.
- **colSpan** – The number of columns the progress bar should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the progress bar.

**Returns** The progress bar that was added.

**addSeparator**(*orientation=2, width=6, rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [Union\[RibbonHorizontalSeparator, RibbonVerticalSeparator\]](#)

Add a separator to the panel.

**Parameters**

- **orientation** – The orientation of the separator.
- **width** – The width of the separator.
- **rowSpan** – The number of rows the separator spans.
- **colSpan** – The number of columns the separator spans.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the separator.

**Returns** The separator.

**addSlider**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QSlider](#)

Add a slider to the panel.

**Parameters**

- **rowSpan** – The number of rows the slider should span.
- **colSpan** – The number of columns the slider should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the slider.

**Returns** The slider that was added.

**addSmallButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [RibbonToolButton](#)

Add a small button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addSmallToggleButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, showText: bool = True, colSpan: int = 1, slot=None, shortcut=None, tooltip=None, statusTip=None, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [RibbonToolButton](#)

Add a small toggle button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

**addSmallWidget**(*widget: QWidget, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*)

Add a small widget to the panel.

**Parameters**

- **widget** – The widget to add.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the widget.

**Returns** The widget that was added.

**addSpinBox**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QSpinBox](#)

Add a spin box to the panel.

**Parameters**

- **rowSpan** – The number of rows the spin box should span.
- **colSpan** – The number of columns the spin box should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the spin box.

**Returns** The spin box that was added.

**addTableWidget**(*rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QTableWidget](#)

Add a table widget to the panel.

**Parameters**

- **rowSpan** – The number of rows the table widget should span.
- **colSpan** – The number of columns the table widget should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the table widget.

**Returns** The table widget that was added.

**addTextEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QTextEdit](#)

Add a text edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the text edit should span.
- **colSpan** – The number of columns the text edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the text edit.

**Returns** The text edit that was added.

**addTimeEdit**(*rowSpan: int = 2, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise, alignment=132*) → [QTimeEdit](#)



Add a time edit to the panel.

**Parameters**

- **rowSpan** – The number of rows the time edit should span.
- **colSpan** – The number of columns the time edit should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the time edit.

**Returns** The time edit that was added.

```
addToggleButton(text: Optional[str] = None, icon: Optional[QIcon] = None, style: RibbonButtonStyle =  
RibbonButtonStyle.Large, showText: bool = True, colSpan: int = 1, slot=None,  
shortcut=None, tooltip=None, statusTip=None,  
mode=RibbonSpaceFindMode.ColumnWise, alignment=132) → RibbonToolButton
```

Add a toggle button to the panel.

**Parameters**

- **text** – The text of the button.
- **icon** – The icon of the button.
- **style** – The style of the button.
- **showText** – Whether to show the text of the button.
- **colSpan** – The number of columns the button should span.
- **slot** – The slot to call when the button is clicked.
- **shortcut** – The shortcut of the button.
- **tooltip** – The tooltip of the button.
- **statusTip** – The status tip of the button.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the button.

**Returns** The button that was added.

```
addTreeWidget(rowSpan: int = 6, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise,  
alignment=132) → QTreeWidget
```

Add a tree widget to the panel.

**Parameters**

- **rowSpan** – The number of rows the tree widget should span.
- **colSpan** – The number of columns the tree widget should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the tree widget.

**Returns** The tree widget that was added.

```
addVerticalSeparator(width=6, rowSpan: int = 6, colSpan: int = 1,  
mode=RibbonSpaceFindMode.ColumnWise, alignment=132) →  
RibbonVerticalSeparator
```

Add a vertical separator to the panel.

**Parameters**

- **width** – The width of the separator.
- **rowSpan** – The number of rows the separator spans.
- **colSpan** – The number of columns the separator spans.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the separator.

**Returns** The separator.

```
addWidget(widget: QWidget, rowSpan: int = 2, colSpan: int = 1,  
mode=RibbonSpaceFindMode.ColumnWise, alignment=132)
```

Add a widget to the panel.

**Parameters**

- **widget** – The widget to add.



- **rowSpan** – The number of rows the widget should span, 2: small, 3: medium, 6: large.
- **colSpan** – The number of columns the widget should span.
- **mode** – The mode to find spaces.
- **alignment** – The alignment of the widget.

**addWidgetBy**(*data: Dict[str, Dict]*) → Dict[str, [QWidget](#)]

Add widgets to the panel.

**Parameters** **data** – The data to add. The dict is of the form: {

```
"widget-name": { "type": "Button", "arguments": {
                    "key1": "value1", "key2": "value2"
                }
            },
```

} Possible types are: Button, SmallButton, MediumButton, LargeButton, ToggleButton, SmallToggleButton, MediumToggleButton, LargeToggleButton, ComboBox, FontComboBox, LineEdit, TextEdit, PlainTextEdit, Label, ProgressBar, SpinBox, DoubleSpinBox, DataEdit, TimeEdit, DateTimeEdit, TableWidget, TreeWidget, ListWidget, CalendarWidget, Separator, HorizontalSeparator, VerticalSeparator, Gallery.

**Returns** A dictionary of the added widgets.

**panelOptionButton**() → [RibbonPanelOptionButton](#)

Return the panel option button.

**Returns** The panel option button.

**removeWidget**(*widget: QWidget*)

Remove a widget from the panel.

**rowHeight**() → int

Return the height of a row.

**setPanelOptionToolTip**(*text: str*)

Set the tooltip of the panel option button.

**Parameters** **text** – The tooltip text.

**setTitle**(*title: str*)

Set the title of the panel.

**Parameters** **title** – The title to set.

**title**()

Get the title of the panel.

**Returns** The title.

**widget**(*index: int*) → [QWidget](#)

Get the widget at the given index.

**Parameters** **index** – The index of the widget, starting from 0.

**Returns** The widget at the given index.

**widgets**() → List[[QWidget](#)]

Get all the widgets in the panel.

**Returns** A list of all the widgets in the panel.

### 4.4.2 RibbonPanelItemWidget

**class** ribbon.panel.RibbonPanelItemWidget(*parent=None*)

Bases: `QFrame`

Widget to display a panel item.

#### Methods

---

<code>addWidget(widget)</code>	Add a widget to the panel item.
--------------------------------	---------------------------------

---

**addWidget**(*widget*)

Add a widget to the panel item.

**Parameters** **widget** – The widget to add.

### 4.4.3 RibbonSpaceFindMode

**class** ribbon.panel.RibbonSpaceFindMode(*value*)

Bases: `IntEnum`

Mode to find available space in a grid layout, ColumnWise or RowWise.

### 4.4.4 RibbonGridLayoutManager

**class** ribbon.panel.RibbonGridLayoutManager(*rows: int*)

Bases: `object`

Grid Layout Manager.

#### Methods

---

<code>request_cells([rowSpan, colSpan, mode])</code>	Request a number of available cells from the grid.
--	--

---

**request\_cells**(*rowSpan: int = 1, colSpan: int = 1, mode=RibbonSpaceFindMode.ColumnWise*)

Request a number of available cells from the grid.

**Parameters**

- **rowSpan** – The number of rows the cell should span.
- **colSpan** – The number of columns the cell should span.
- **mode** – The mode of the grid.

**Returns** row, col, the row and column of the requested cell.

## 4.5 Ribbon Gallery

### 4.5.1 RibbonGallery

**class** ribbon.gallery.**RibbonGallery**(*minimumWidth=800, popupHideOnClick=False, parent=None*)

**class** ribbon.gallery.**RibbonGallery**(*parent=None*)

Bases: [QFrame](#)

A widget that displays a gallery of buttons.

#### Methods

<a href="#"><i>addButton</i></a> ([ <i>text, icon, slot, shortcut, ...</i> ])	Add a button to the gallery
<a href="#"><i>addToggleButton</i></a> ([ <i>text, icon, slot, ...</i> ])	Add a toggle button to the gallery
<a href="#"><i>hidePopupWidget</i></a> ()	Hide the popup window
<a href="#"><i>popupMenu</i></a> ()	Return the popup menu.
<a href="#"><i>popupWindowSize</i></a> ()	Return the size of the popup window
<a href="#"><i>resizeEvent</i></a> (a0)	Resize the gallery.
<a href="#"><i>setPopupHideOnClick</i></a> (popupHideOnClick)	Set the hide on click flag
<a href="#"><i>setPopupWindowSize</i></a> (size)	Set the size of the popup window
<a href="#"><i>setSelectedButton</i></a> ()	Set the selected button
<a href="#"><i>showPopup</i></a> ()	Show the popup window

**addButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, slot=None, shortcut=None, tooltip=None, statusTip=None, checkable=False*) → [\*RibbonToolButton\*](#)

Add a button to the gallery

#### Parameters

- **text** – text of the button
- **icon** – icon of the button
- **slot** – slot to call when the button is clicked
- **shortcut** – shortcut of the button
- **tooltip** – tooltip of the button
- **statusTip** – status tip of the button
- **checkable** – checkable flag of the button

**Returns** the button added

**addToggleButton**(*text: Optional[str] = None, icon: Optional[QIcon] = None, slot=None, shortcut=None, tooltip=None, statusTip=None*) → [\*RibbonToolButton\*](#)

Add a toggle button to the gallery

#### Parameters

- **text** – text of the button
- **icon** – icon of the button
- **slot** – slot to call when the button is clicked
- **shortcut** – shortcut of the button
- **tooltip** – tooltip of the button
- **statusTip** – status tip of the button

**Returns** the button added

**hidePopupWidget**()

Hide the popup window

**popupMenu()** → *RibbonPermanentMenu*

Return the popup menu.

**popupWindowSize()**

Return the size of the popup window

**Returns** size of the popup window

**resizeEvent**(*a0: QResizeEvent*) → None

Resize the gallery.

**setPopupHideOnClick**(*popupHideOnClick: bool*)

Set the hide on click flag

**Parameters** **popupHideOnClick** – hide on click flag

**setPopupWindowSize**(*size: QSize*)

Set the size of the popup window

**Parameters** **size** – size of the popup window

**setSelectedButton()**

Set the selected button

**showPopup()**

Show the popup window

## 4.5.2 RibbonGalleryListWidget

**class** ribbon.gallery.RibbonGalleryListWidget(*parent=None*)

Bases: *QWidget*

Gallery list widget.

### Methods

<i>resizeEvent</i> ( <i>e</i> )	Resize the list widget.
<i>scrollToNextRow</i> ()	Scroll to the next row.
<i>scrollToPreviousRow</i> ()	Scroll to the previous row.

**resizeEvent**(*e: QResizeEvent*) → None

Resize the list widget.

**scrollToNextRow**() → None

Scroll to the next row.

**scrollToPreviousRow**() → None

Scroll to the previous row.

### 4.5.3 RibbonGalleryButton

**class** ribbon.gallery.RibbonGalleryButton

Bases: *QToolButton*

Gallery button.

### 4.5.4 RibbonGalleryPopupListWidget

**class** ribbon.gallery.RibbonGalleryPopupListWidget(*parent=None*)

Bases: *RibbonGalleryListWidget*

Gallery popup list widget.

### 4.5.5 RibbonPopupWidget

**class** ribbon.gallery.RibbonPopupWidget

Bases: *QFrame*

The popup widget for the gallery widget.

## 4.6 Ribbon Tool Button

### 4.6.1 RibbonToolButton

**class** ribbon.toolbutton.RibbonToolButton(*parent=None*)

Bases: *QToolButton*

Tool button that is showed in the ribbon.

#### Methods

<i>addRibbonMenu()</i>	Add a ribbon menu for the button.
<i>buttonStyle()</i>	Get the button style of the button.
<i>maximumIconSize()</i>	Get the maximum icon size of the button.
<i>setButtonStyle(style)</i>	Set the button style of the button.
<i>setMaximumIconSize(size)</i>	Set the maximum icon size of the button.

**addRibbonMenu()** → *RibbonMenu*

Add a ribbon menu for the button.

**Returns** The added ribbon menu.

**buttonStyle()** → *RibbonButtonStyle*

Get the button style of the button.

**Returns** The button style of the button.

**maximumIconSize()** → int

Get the maximum icon size of the button.

**Returns** The maximum icon size of the button.

**setButtonStyle**(*style*: [RibbonButtonStyle](#))

Set the button style of the button.

**Parameters** **style** – The button style of the button.

**setMaximumIconSize**(*size*: *int*)

Set the maximum icon size of the button.

**Parameters** **size** – The maximum icon size of the button.

## 4.6.2 RibbonButtonStyle

**class** `ribbon.toolbar.RibbonButtonStyle`(*value*)

Bases: `IntEnum`

Button style, Small, Medium, or Large.

## 4.7 Ribbon Separator

### 4.7.1 RibbonSeparator

**class** `ribbon.separator.RibbonSeparator`(*orientation*=`QtCore.Qt.Vertical`, *width*=6, *parent*=*None*)

**class** `ribbon.separator.RibbonSeparator`(*parent*=*None*)

Bases: `QFrame`

The `RibbonSeparator` is a separator that can be used to separate widgets in a ribbon.

#### Methods

<code>paintEvent</code> ( <i>event</i> )	Paint the separator.
<code>setTopBottomMargins</code> ( <i>top</i> , <i>bottom</i> )	Set the top and bottom margins.
<code>sizeHint</code> ()	Return the size hint.

**paintEvent**(*event*: `QPaintEvent`) → `None`

Paint the separator.

**setTopBottomMargins**(*top*: *int*, *bottom*: *int*) → `None`

Set the top and bottom margins.

**sizeHint**() → `QSize`

Return the size hint.

### 4.7.2 RibbonHorizontalSeparator

**class** `ribbon.separator.RibbonHorizontalSeparator`(*width*: *int* = 6, *parent*=*None*)

Bases: [RibbonSeparator](#)

Horizontal separator.

### 4.7.3 RibbonVerticalSeparator

**class** ribbon.separator.RibbonVerticalSeparator(*width: int = 6, parent=None*)

Bases: [RibbonSeparator](#)

Vertical separator.

## 4.8 Ribbon Menu

### 4.8.1 RibbonMenu

**class** ribbon.menu.RibbonMenu(*title: str = "", parent=None*)

**class** ribbon.menu.RibbonMenu(*parent=None*)

Bases: [QMenu](#)

#### Methods

<a href="#">addFormLayoutWidget()</a>	Add a form layout widget to the menu.
<a href="#">addGridLayoutWidget()</a>	Add a grid layout widget to the menu.
<a href="#">addHorizontalLayoutWidget()</a>	Add a horizontal layout widget to the menu.
<a href="#">addLabel([text, alignment])</a>	Add a label to the menu.
<a href="#">addSpacing([spacing])</a>	Add spacing to the menu.
<a href="#">addVerticalLayoutWidget()</a>	Add a vertical layout widget to the menu.
<a href="#">addWidget(widget)</a>	Add a widget to the menu.

**addFormLayoutWidget()** → [QFormLayout](#)

Add a form layout widget to the menu.

**Returns** The form layout.

**addGridLayoutWidget()** → [QGridLayout](#)

Add a grid layout widget to the menu.

**Returns** The grid layout.

**addHorizontalLayoutWidget()** → [QHBoxLayout](#)

Add a horizontal layout widget to the menu.

**Returns** The horizontal layout.

**addLabel**(*text: str = "", alignment: [Alignment](#) = 1*)

Add a label to the menu.

**Parameters**

- **text** – The text of the label.
- **alignment** – The alignment of the label.

**addSpacing**(*spacing: int = 5*)

Add spacing to the menu.

**Parameters** **spacing** – The spacing.

**addVerticalLayoutWidget()** → [QVBoxLayout](#)

Add a vertical layout widget to the menu.

**Returns** The vertical layout.

**addWidget**(*widget*: *QWidget*)

Add a widget to the menu.

**Parameters** **widget** – The widget to add.

## 4.8.2 RibbonPermanentMenu

**class** ribbon.menu.RibbonPermanentMenu(*title*: *str* = "", *parent*=None)

**class** ribbon.menu.RibbonPermanentMenu(*parent*=None)

Bases: *RibbonMenu*

A permanent menu.

### Methods

---

*actionEvent*(self, QActionEvent)

---

*hideEvent*(self, QHideEvent)

---

actionAdded	
-------------	--

**actionEvent**(*self*, QActionEvent)

**hideEvent**(*self*, QHideEvent)



## INDICES AND TABLES

- genindex
- modindex
- search



## A

- `actionAt()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `actionEvent()` (*ribbon.menu.RibbonPermanentMenu method*), 44
- `actionGeometry()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `activeAction()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `addAction()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `addAssociatedTabs()` (*ribbon.tabbar.RibbonTabBar method*), 20
- `addButton()` (*ribbon.gallery.RibbonGallery method*), 39
- `addButton()` (*ribbon.panel.RibbonPanel method*), 29
- `addCalendarWidget()` (*ribbon.panel.RibbonPanel method*), 30
- `addCategoriesBy()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `addCategory()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `addComboBox()` (*ribbon.panel.RibbonPanel method*), 30
- `addContextCategories()` (*ribbon.ribbonbar.RibbonBar method*), 15
- `addContextCategory()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addDateEdit()` (*ribbon.panel.RibbonPanel method*), 30
- `addDateTimeEdit()` (*ribbon.panel.RibbonPanel method*), 30
- `addDoubleSpinBox()` (*ribbon.panel.RibbonPanel method*), 30
- `addFileMenu()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addFileMenu()` (*ribbon.titlewidget.RibbonApplicationButton method*), 19
- `addFontComboBox()` (*ribbon.panel.RibbonPanel method*), 31
- `addFormLayoutWidget()` (*ribbon.menu.RibbonMenu method*), 43
- `addGallery()` (*ribbon.panel.RibbonPanel method*), 31
- `addGridLayoutWidget()` (*ribbon.menu.RibbonMenu method*), 43
- `addHorizontalLayoutWidget()` (*ribbon.menu.RibbonMenu method*), 43
- `addHorizontalSeparator()` (*ribbon.panel.RibbonPanel method*), 31
- `addLabel()` (*ribbon.menu.RibbonMenu method*), 43
- `addLabel()` (*ribbon.panel.RibbonPanel method*), 31
- `addLargeButton()` (*ribbon.panel.RibbonPanel method*), 31
- `addLargeToggleButton()` (*ribbon.panel.RibbonPanel method*), 32
- `addLargeWidget()` (*ribbon.panel.RibbonPanel method*), 32
- `addLineEdit()` (*ribbon.panel.RibbonPanel method*), 32
- `addListWidget()` (*ribbon.panel.RibbonPanel method*), 32
- `addMediumButton()` (*ribbon.panel.RibbonPanel method*), 33
- `addMediumToggleButton()` (*ribbon.panel.RibbonPanel method*), 33
- `addMediumWidget()` (*ribbon.panel.RibbonPanel method*), 33
- `addMenu()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addNormalCategory()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addPanel()` (*ribbon.category.RibbonCategory method*), 23
- `addPanelsBy()` (*ribbon.category.RibbonCategory method*), 24
- `addPlainTextEdit()` (*ribbon.panel.RibbonPanel method*), 33
- `addProgressBar()` (*ribbon.panel.RibbonPanel method*), 34
- `addQuickAccessButton()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addQuickAccessButton()` (*ribbon.titlewidget.RibbonTitleWidget method*), 21
- `addRibbonMenu()` (*ribbon.toolbutton.RibbonToolButton method*), 41
- `addRightToolButton()` (*ribbon.ribbonbar.RibbonBar method*), 16
- `addRightToolButton()` (*ribbon.titlewidget.RibbonTitleWidget method*), 21

- 21  
 addSeparator() (*ribbon.panel.RibbonPanel* method), 34  
 addSeparator() (*ribbon.ribbonbar.RibbonBar* method), 16  
 addSlider() (*ribbon.panel.RibbonPanel* method), 34  
 addSmallButton() (*ribbon.panel.RibbonPanel* method), 34  
 addSmallToggleButton() (*ribbon.panel.RibbonPanel* method), 34  
 addSmallWidget() (*ribbon.panel.RibbonPanel* method), 35  
 addSpacing() (*ribbon.menu.RibbonMenu* method), 43  
 addSpinBox() (*ribbon.panel.RibbonPanel* method), 35  
 addTab() (*ribbon.tabbar.RibbonTabBar* method), 20  
 addTableWidget() (*ribbon.panel.RibbonPanel* method), 35  
 addTextEdit() (*ribbon.panel.RibbonPanel* method), 35  
 addTimeEdit() (*ribbon.panel.RibbonPanel* method), 35  
 addToggleButton() (*ribbon.gallery.RibbonGallery* method), 39  
 addToggleButton() (*ribbon.panel.RibbonPanel* method), 36  
 addTreeWidget() (*ribbon.panel.RibbonPanel* method), 36  
 addVerticalLayoutWidget() (*ribbon.menu.RibbonMenu* method), 43  
 addVerticalSeparator() (*ribbon.panel.RibbonPanel* method), 36  
 addWidget() (*ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget* method), 27  
 addWidget() (*ribbon.menu.RibbonMenu* method), 43  
 addWidget() (*ribbon.panel.RibbonPanel* method), 36  
 addWidget() (*ribbon.panel.RibbonPanelItemWidget* method), 38  
 addWidgetsBy() (*ribbon.panel.RibbonPanel* method), 37  
 applicationButton() (*ribbon.titlewidget.RibbonTitleWidget* method), 21  
 applicationOptionButton() (*ribbon.ribbonbar.RibbonBar* method), 16  
 autoSetScrollButtonsVisible() (*ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget* method), 28
- ## B
- buttonStyle() (*ribbon.toolbutton.RibbonToolButton* method), 41
- ## C
- categories() (*ribbon.ribbonbar.RibbonBar* method), 16  
 categoriesVisible() (*ribbon.category.RibbonContextCategories* method), 26  
 category() (*ribbon.ribbonbar.RibbonBar* method), 16  
 categoryStyle() (*ribbon.category.RibbonCategory* method), 24  
 categoryVisible() (*ribbon.category.RibbonContextCategory* method), 25  
 categoryVisible() (*ribbon.ribbonbar.RibbonBar* method), 16  
 clear() (*ribbon.ribbonbar.RibbonBar* method), 16  
 collapseRibbonButton() (*ribbon.ribbonbar.RibbonBar* method), 17  
 collapseRibbonButton() (*ribbon.titlewidget.RibbonTitleWidget* method), 22  
 collapseRibbonButtonClicked (*ribbon.titlewidget.RibbonTitleWidget* attribute), 22  
 color() (*ribbon.category.RibbonContextCategories* method), 26  
 color() (*ribbon.category.RibbonContextCategory* method), 25  
 cornerWidget() (*ribbon.ribbonbar.RibbonBar* method), 17  
 currentCategory() (*ribbon.ribbonbar.RibbonBar* method), 17  
 currentTabColor() (*ribbon.tabbar.RibbonTabBar* method), 20
- ## H
- helpButtonClicked (*ribbon.ribbonbar.RibbonBar* attribute), 17  
 helpButtonClicked (*ribbon.titlewidget.RibbonTitleWidget* attribute), 22  
 helpRibbonButton() (*ribbon.ribbonbar.RibbonBar* method), 17  
 helpRibbonButton() (*ribbon.titlewidget.RibbonTitleWidget* method), 22  
 hideContextCategories() (*ribbon.category.RibbonContextCategories* method), 26  
 hideContextCategory() (*ribbon.category.RibbonContextCategory* method), 25  
 hideContextCategory() (*ribbon.ribbonbar.RibbonBar* method), 17  
 hideEvent() (*ribbon.menu.RibbonPermanentMenu* method), 44  
 hidePopupWidget() (*ribbon.gallery.RibbonGallery* method), 39

- hideRibbon() (*ribbon.ribbonbar.RibbonBar* method), 17
- I
- indexOf() (*ribbon.tabbar.RibbonTabBar* method), 20
- insertMenu() (*ribbon.ribbonbar.RibbonBar* method), 17
- insertSeparator() (*ribbon.ribbonbar.RibbonBar* method), 17
- isDefaultUp() (*ribbon.ribbonbar.RibbonBar* method), 17
- isNativeMenuBar() (*ribbon.ribbonbar.RibbonBar* method), 17
- M
- maximumIconSize() (*ribbon.toolbox.RibbonToolButton* method), 41
- minimumSizeHint() (*ribbon.ribbonbar.RibbonBar* method), 17
- N
- name() (*ribbon.category.RibbonContextCategories* method), 26
- P
- paintEvent() (*ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget* method), 28
- paintEvent() (*ribbon.separator.RibbonSeparator* method), 42
- paintEvent() (*ribbon.tabbar.RibbonTabBar* method), 20
- panel() (*ribbon.category.RibbonCategory* method), 24
- panelOptionButton() (*ribbon.panel.RibbonPanel* method), 37
- panels() (*ribbon.category.RibbonCategory* method), 24
- popupMenu() (*ribbon.gallery.RibbonGallery* method), 39
- popupWindowSize() (*ribbon.gallery.RibbonGallery* method), 40
- Q
- quickAccessButtons() (*ribbon.titlewidget.RibbonTitleWidget* method), 22
- quickAccessToolBar() (*ribbon.ribbonbar.RibbonBar* method), 17
- quickAccessToolBar() (*ribbon.titlewidget.RibbonTitleWidget* method), 22
- R
- removeAssociatedTabs() (*ribbon.tabbar.RibbonTabBar* method), 20
- removeCategories() (*ribbon.ribbonbar.RibbonBar* method), 17
- removeCategory() (*ribbon.ribbonbar.RibbonBar* method), 17
- removeCollapseButton() (*ribbon.ribbonbar.RibbonBar* method), 17
- removeCollapseButton() (*ribbon.titlewidget.RibbonTitleWidget* method), 22
- removeHelpButton() (*ribbon.ribbonbar.RibbonBar* method), 17
- removeHelpButton() (*ribbon.titlewidget.RibbonTitleWidget* method), 22
- removePanel() (*ribbon.category.RibbonCategory* method), 24
- removeWidget() (*ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget* method), 28
- removeWidget() (*ribbon.panel.RibbonPanel* method), 37
- request\_cells() (*ribbon.panel.RibbonGridLayoutManager* method), 38
- resizeEvent() (*ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget* method), 28
- resizeEvent() (*ribbon.gallery.RibbonGallery* method), 40
- resizeEvent() (*ribbon.gallery.RibbonGalleryListWidget* method), 40
- RibbonApplicationButton (class in *ribbon.titlewidget*), 19
- RibbonBar (class in *ribbon.ribbonbar*), 13
- RibbonButtonStyle (class in *ribbon.toolbox*), 42
- RibbonCategory (class in *ribbon.category*), 23
- RibbonCategoryLayoutButton (class in *ribbon.categorylayoutwidget*), 27
- RibbonCategoryLayoutWidget (class in *ribbon.categorylayoutwidget*), 27
- RibbonCategoryScrollArea (class in *ribbon.categorylayoutwidget*), 27
- RibbonCategoryScrollAreaContents (class in *ribbon.categorylayoutwidget*), 27
- RibbonCategoryStyle (class in *ribbon.category*), 27
- RibbonContextCategories (class in *ribbon.category*), 26
- RibbonContextCategory (class in *ribbon.category*), 25
- RibbonGallery (class in *ribbon.gallery*), 39
- RibbonGalleryButton (class in *ribbon.gallery*), 41
- RibbonGalleryListWidget (class in *ribbon.gallery*), 40
- RibbonGalleryPopupListWidget (class in *ribbon.gallery*), 41
- RibbonGridLayoutManager (class in *ribbon.panel*), 38

ribbonHeight() (ribbon.ribbonbar.RibbonBar method), 17  
 RibbonHorizontalSeparator (class in ribbon.separator), 42  
 RibbonMenu (class in ribbon.menu), 43  
 RibbonNormalCategory (class in ribbon.category), 25  
 RibbonPanel (class in ribbon.panel), 28  
 RibbonPanelItemWidget (class in ribbon.panel), 38  
 RibbonPermanentMenu (class in ribbon.menu), 44  
 RibbonPopupWidget (class in ribbon.gallery), 41  
 RibbonSeparator (class in ribbon.separator), 42  
 RibbonSpaceFindMode (class in ribbon.panel), 38  
 RibbonStackedWidget (class in ribbon.ribbonbar), 23  
 RibbonTabBar (class in ribbon.tabbar), 20  
 RibbonTitleLabel (class in ribbon.titlewidget), 21  
 RibbonTitleWidget (class in ribbon.titlewidget), 21  
 RibbonToolButton (class in ribbon.toolbutton), 41  
 RibbonVerticalSeparator (class in ribbon.separator), 43  
 ribbonVisible() (ribbon.ribbonbar.RibbonBar method), 18  
 rightToolBar() (ribbon.ribbonbar.RibbonBar method), 18  
 rightToolBar() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 rowHeight() (ribbon.panel.RibbonPanel method), 37

## S

scrollNext() (ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget method), 28  
 scrollPrevious() (ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget method), 28  
 scrollToNextRow() (ribbon.gallery.RibbonGalleryListWidget method), 40  
 scrollToPreviousRow() (ribbon.gallery.RibbonGalleryListWidget method), 40  
 setActiveAction() (ribbon.ribbonbar.RibbonBar method), 18  
 setApplicationIcon() (ribbon.ribbonbar.RibbonBar method), 18  
 setApplicationIcon() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setButtonStyle() (ribbon.toolbutton.RibbonToolButton method), 41  
 setCategoriesVisible() (ribbon.category.RibbonContextCategories method), 26  
 setCategoryStyle() (ribbon.category.RibbonCategory method), 24  
 setCategoryStyle() (ribbon.category.RibbonContextCategory method), 25  
 setCategoryStyle() (ribbon.category.RibbonNormalCategory method), 25  
 setCategoryVisible() (ribbon.category.RibbonContextCategory method), 25  
 setCollapseButtonIcon() (ribbon.ribbonbar.RibbonBar method), 18  
 setCollapseButtonIcon() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setColor() (ribbon.category.RibbonContextCategories method), 26  
 setColor() (ribbon.category.RibbonContextCategory method), 26  
 setCornerWidget() (ribbon.ribbonbar.RibbonBar method), 18  
 setCurrentCategory() (ribbon.ribbonbar.RibbonBar method), 18  
 setDefaultUp() (ribbon.ribbonbar.RibbonBar method), 18  
 setHelpButtonIcon() (ribbon.ribbonbar.RibbonBar method), 18  
 setHelpButtonIcon() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setMaximumIconSize() (ribbon.toolbutton.RibbonToolButton method), 42  
 setName() (ribbon.category.RibbonContextCategories method), 26  
 setNativeMenuBar() (ribbon.ribbonbar.RibbonBar method), 18  
 setPanelOptionToolTip() (ribbon.panel.RibbonPanel method), 37  
 setPopupHideOnClick() (ribbon.gallery.RibbonGallery method), 40  
 setPopupWindowSize() (ribbon.gallery.RibbonGallery method), 40  
 setQuickAccessButtonHeight() (ribbon.ribbonbar.RibbonBar method), 18  
 setQuickAccessButtonHeight() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setRibbonHeight() (ribbon.ribbonbar.RibbonBar method), 18  
 setRibbonStyle() (ribbon.ribbonbar.RibbonBar method), 18  
 setRibbonVisible() (ribbon.ribbonbar.RibbonBar method), 18

setRightToolBarHeight() (ribbon.ribbonbar.RibbonBar method), 18  
 setRightToolBarHeight() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setSelectedButton() (ribbon.gallery.RibbonGallery method), 40  
 setTabBarHeight() (ribbon.ribbonbar.RibbonBar method), 18  
 setTabBarHeight() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setTitle() (ribbon.panel.RibbonPanel method), 37  
 setTitle() (ribbon.ribbonbar.RibbonBar method), 19  
 setTitle() (ribbon.titlewidget.RibbonTitleWidget method), 22  
 setTopBottomMargins() (ribbon.separator.RibbonSeparator method), 42  
 showCategoryByIndex() (ribbon.ribbonbar.RibbonBar method), 19  
 showContextCategories() (ribbon.category.RibbonContextCategories method), 26  
 showContextCategory() (ribbon.category.RibbonContextCategory method), 26  
 showContextCategory() (ribbon.ribbonbar.RibbonBar method), 19  
 showPopup() (ribbon.gallery.RibbonGallery method), 40  
 showRibbon() (ribbon.ribbonbar.RibbonBar method), 19  
 sizeHint() (ribbon.separator.RibbonSeparator method), 42

## T

tabBar() (ribbon.ribbonbar.RibbonBar method), 19  
 tabBar() (ribbon.titlewidget.RibbonTitleWidget method), 23  
 tabBarHeight() (ribbon.ribbonbar.RibbonBar method), 19  
 tabBarHeight() (ribbon.titlewidget.RibbonTitleWidget method), 23  
 tabTitles() (ribbon.tabbar.RibbonTabBar method), 20  
 takePanel() (ribbon.category.RibbonCategory method), 24  
 takeWidget() (ribbon.categorylayoutwidget.RibbonCategoryLayoutWidget method), 28  
 title() (ribbon.category.RibbonCategory method), 24  
 title() (ribbon.panel.RibbonPanel method), 37  
 title() (ribbon.ribbonbar.RibbonBar method), 19  
 title() (ribbon.titlewidget.RibbonTitleWidget method), 23

## W

widget() (ribbon.panel.RibbonPanel method), 37  
 widgets() (ribbon.panel.RibbonPanel method), 37