pyqtribbon

Release 0.5.0

WANG Hailin

Mar 13, 2023

CONTENTS:

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

- GitHub Repository: github.com/haiiliin/pyqtribbon.
- PyPI: pypi.org/project/pyqtribbon.
- Documentation: pyqtribbon.haiiliin.com/en/stable.
- Read the Docs: readthedocs.org/projects/pyqtribbon.

CHAPTER

ONE

GETTING STARTED

1.1 Installation

PyQtRibbon is distribued 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
```

CHAPTER

TWO

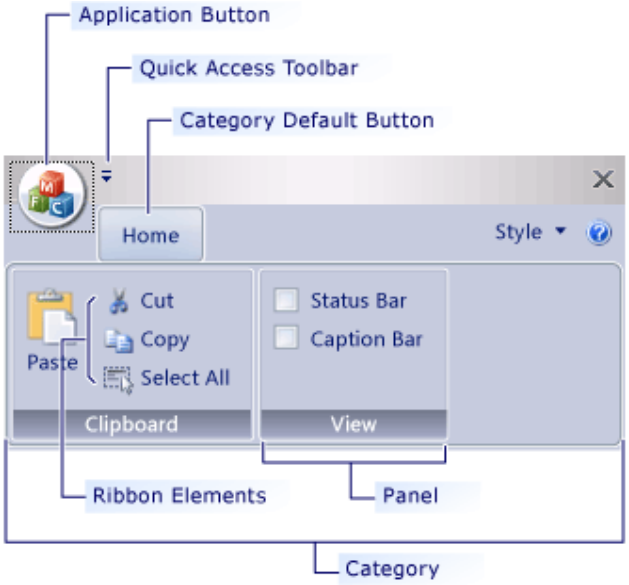
THE RIBBON BAR

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:

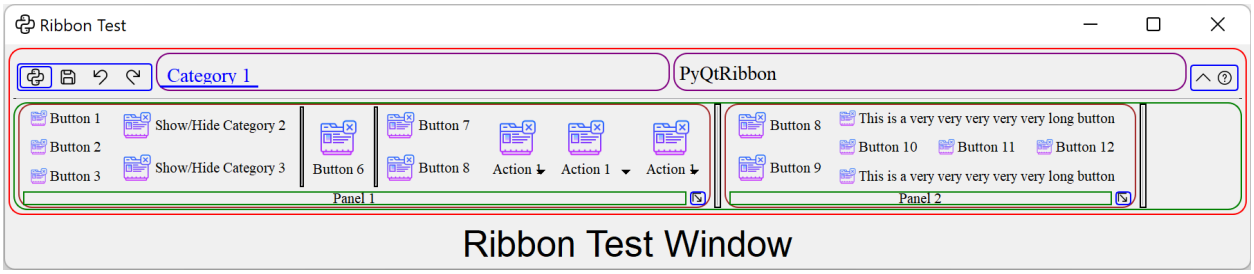
Note: A ribbon is a user interface (UI) element that organizes commands into logical groups. These groups appear on separate tabs in a strip across the top of the window. The ribbon replaces the menu bar and toolbars. A ribbon can significantly improve application usability. For more information, see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



CHAPTER

THREE

USER MANUAL

3.1 The RibbonScreenShotWindow Class

The `RibbonScreenShotWindow` class is just for taking a screenshot of the window, the window will be closed 0.1s after it is shown. It is just used for documenting the window.

class pyqtribbon.screenshotwindow.**RibbonScreenShotWindow**(*fileName: str = 'shot.jpg'*, **args*, ***kwargs*)

This class is just for taking a screenshot of the window, the window will be closed 0.1s after it is shown.

Methods

setScreenShotFileName(fileName)	Set the file name for the screenshot.
takeScreenShot()	Take a screenshot of the window.

setScreenShotFileName(*fileName: str*)

Set the file name for the screenshot.

Parameters **fileName** – The file name for the screenshot.

takeScreenShot()

Take a screenshot of the window.

3.2 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.

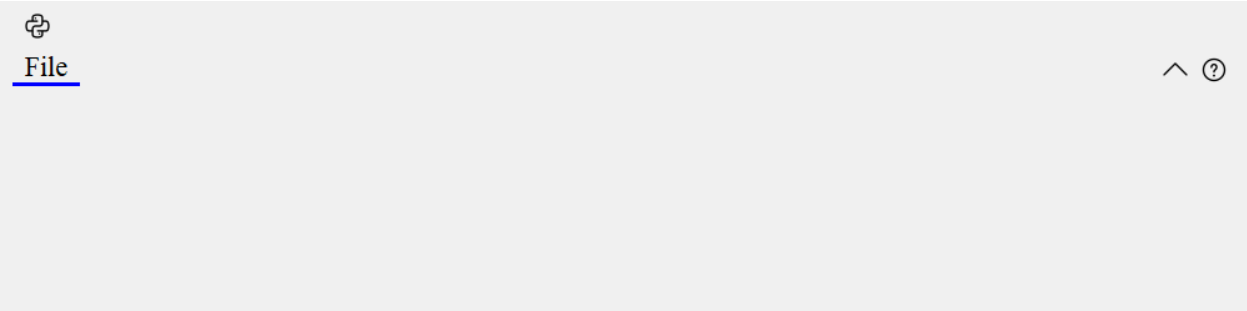
```
...
from pyqtribbon import RibbonBar

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

3.2.1 Example

For example, using the following code,

```
import sys

from qtpy import QtWidgets, QtGui

from pyqtribbon import RibbonBar
from pyqtribbon.screenshotwindow import RibbonScreenShotWindow

if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    app.setFont(QtGui.QFont("Times New Roman", 8))
    window = RibbonScreenShotWindow('ribbonbar.png')

    # Ribbon bar
    ribbonbar = RibbonBar()
    window.setMenuBar(ribbonbar)

    # Show the window
    window.resize(1000, 250)
    window.show()

    sys.exit(app.exec_())
```

You can get a window like this:



3.3 Customize Ribbon Bar

3.3.1 General Setups

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

3.3.2 Setup Application Button

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

3.3.3 Setup Title

RibbonBar.title()	Return the title of the ribbon.
RibbonBar.setTitle(title)	Set the title of the ribbon.
RibbonBar.addTitleWidget(widget)	Add a widget to the title widget.
RibbonBar.insertTitleWidget(index, widget)	Insert a widget to the title widget.
RibbonBar.removeTitleWidget(widget)	Remove a widget from the title widget.

3.3.4 Setup Category Tab Bar

RibbonBar.tabBar()	Return the tab bar of the ribbon.

3.3.5 Setup Quick Access Bar

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

3.3.6 Setup Right Tool Bar

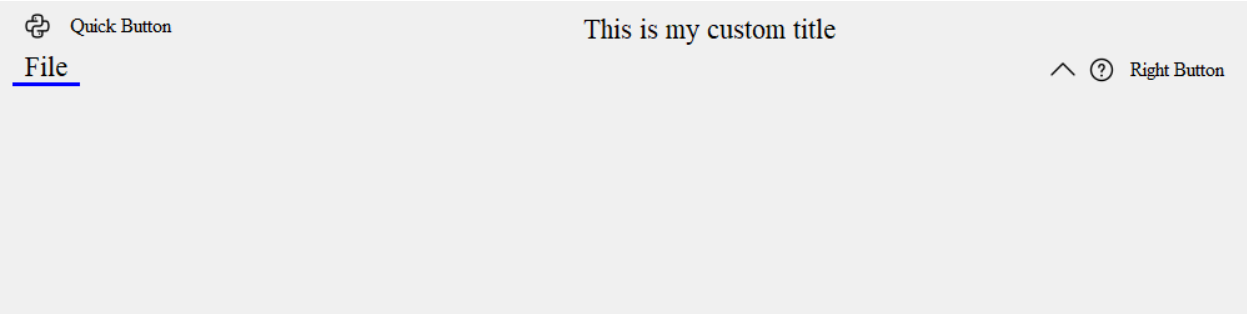
RibbonBar.rightToolBar()	Return the right toolbar of the ribbon.
RibbonBar.addRightToolButton(button)	Add a widget to the right button bar.
RibbonBar.setRightToolBarHeight([height])	Set the height of the right buttons.
RibbonBar.setHelpButtonIcon(icon)	Set the icon of the help button.
RibbonBar.removeHelpButton()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

3.3.7 Example

For example, using the following code,

```
import sys

from qtpy import QtGui
from qtpy.QtWidgets import QApplication, QToolButton

from pyqtribbon import RibbonBar
from pyqtribbon.screenshotwindow import RibbonScreenShotWindow

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setFont(QtGui.QFont("Times New Roman", 8))
    window = RibbonScreenShotWindow('ribbonbar-customize.png')

    # Ribbon bar
    ribbonbar = RibbonBar()
    window.setMenuBar(ribbonbar)

    # Title of the ribbon
    ribbonbar.setTitle('This is my custom title')

    # Quick Access Bar
    qbutton = QToolButton()
    qbutton.setText('Quick Button')
    ribbonbar.addQuickAccessButton(qbutton)

    # Right toolbar
    rbutton = QToolButton()
    rbutton.setText('Right Button')
    ribbonbar.addRightToolButton(rbutton)

    # Show the window
    window.resize(1000, 250)
    window.show()

    sys.exit(app.exec_())
```

You can get a window like this:



3.3.8 Manage Categories

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

3.4 Customize Categories

3.4.1 Setup Styles

RibbonCategory.categoryStyle()	Return the button style of the category.
RibbonCategory.setCategoryStyle(style)	Set the button style of the category.

3.4.2 Manage Panels

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

3.4.3 Example

For example, using the following code,

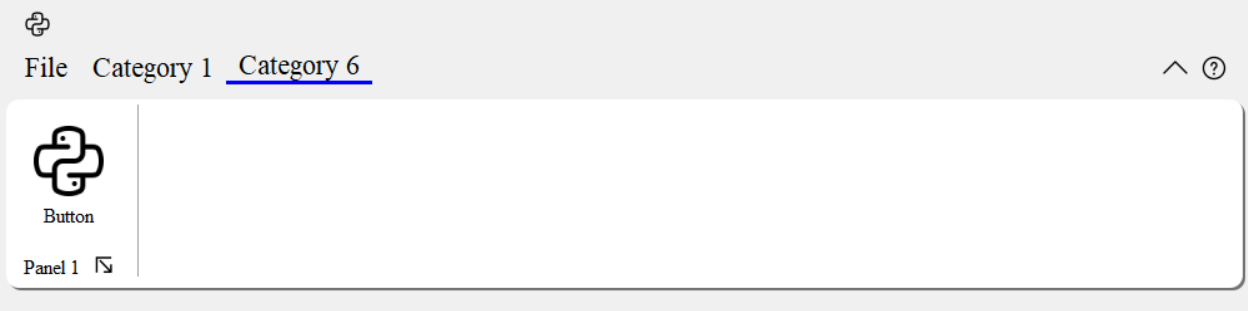
```
import sys

from qtpy import QtGui
from qtpy.QtWidgets import QApplication
from qtpy.QtGui import QIcon

from pyqtribbon import RibbonBar, RibbonCategoryStyle
from pyqtribbon.screenshotwindow import RibbonScreenShotWindow
```

(continues on next page)

(continued from previous page)

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setFont(QtGui.QFont("Times New Roman", 8))
    window = RibbonScreenShotWindow('category.png')

    # Ribbon bar
    ribbonbar = RibbonBar()
    window.setMenuBar(ribbonbar)

    # Categories
    category1 = ribbonbar.addCategory('Category 1')
    panel1 = category1.addPanel('Panel 1')
    panel1.addLargeButton('Large Button 1', QIcon('python.png'))

    category2 = ribbonbar.addContextCategory('Category 2')
    panel12 = category2.addPanel('Panel 2')
    panel12.addLargeButton('Large Button 2', QIcon('python.png'))

    categories = ribbonbar.addCategoriesBy({
        'Category 6': {
            "style": RibbonCategoryStyle.Normal,
            "panels": {
                "Panel 1": {
                    "showPanelOptionButton": True,
                    "widgets": {
                        "Button 1": {
                            "type": "Button",
                            "arguments": {
                                "icon": QIcon("python.png"),
                                "text": "Button",
                                "tooltip": "This is a tooltip",
                            }
                        },
                    }
                },
            }
        }
    })
    ribbonbar.setCurrentCategory(categories['Category 6'])

    # Show the window
    window.resize(1000, 250)
    window.show()

    sys.exit(app.exec_())
```

You can get a window like this:



3.5 Customize Panels

3.5.1 Setup Title Label

RibbonPanel.title()	Get the title of the panel.
RibbonPanel.setTitle(title)	Set the title of the panel.

3.5.2 Setup Panel Option Button

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

3.5.3 Add Widgets to Panels

RibbonPanel.addWidget(widget[, rowSpan, ...])	Add a widget to the panel.
RibbonPanel.addWidgetsBy(data)	Add widgets to the panel.
RibbonPanel.removeWidget(widget)	Remove a widget from the panel.
RibbonPanel.widget(index)	Get the widget at the given index.
RibbonPanel.widgets()	Get all the widgets in the panel.
RibbonPanel.addSmallWidget(widget[, mode, ...])	Add a small widget to the panel.
RibbonPanel.addMediumWidget(widget[, mode, ...])	Add a medium widget to the panel.
RibbonPanel.addLargeWidget(widget[, mode, ...])	Add a large widget to the panel.
RibbonPanel.addButton([text, icon, style, ...])	Add a button to the panel.
RibbonPanel.addSmallButton([text, icon, ...])	Add a small button to the panel.
RibbonPanel.addMediumButton([text, icon, ...])	Add a medium button to the panel.
RibbonPanel.addLargeButton([text, icon, ...])	Add a large button to the panel.
RibbonPanel.addToggleButton([text, icon, ...])	Add a toggle button to the panel.
RibbonPanel.addSmallToggleButton([text, ...])	Add a small toggle button to the panel.
RibbonPanel.addMediumToggleButton([text, ...])	Add a medium toggle button to the panel.
RibbonPanel.addLargeToggleButton([text, ...])	Add a large toggle button to the panel.
RibbonPanel.addComboBox(items[, rowSpan, ...])	Add a combo box to the panel.
RibbonPanel.addFontComboBox([rowSpan, ...])	Add a font combo box to the panel.
RibbonPanel.addLineEdit([rowSpan, colSpan, ...])	Add a line edit to the panel.

continues on next page

Table 1 – continued from previous page

RibbonPanel.addTextEdit([rowSpan, colSpan, ...])	Add a text edit to the panel.
RibbonPanel.addPlainTextEdit([rowSpan, ...])	Add a plain text edit to the panel.
RibbonPanel.addLabel(text[, rowSpan, ...])	Add a label to the panel.
RibbonPanel.addProgressBar([rowSpan, ...])	Add a progress bar to the panel.
RibbonPanel.addSlider([rowSpan, colSpan, ...])	Add a slider to the panel.
RibbonPanel.addSpinBox([rowSpan, colSpan, ...])	Add a spin box to the panel.
RibbonPanel.addDoubleSpinBox([rowSpan, ...])	Add a double spin box to the panel.
RibbonPanel.addDateEdit([rowSpan, colSpan, ...])	Add a date edit to the panel.
RibbonPanel.addTimeEdit([rowSpan, colSpan, ...])	Add a time edit to the panel.
RibbonPanel.addDateTimeEdit([rowSpan, ...])	Add a date time edit to the panel.
RibbonPanel.addTableWidget([rowSpan, ...])	Add a table widget to the panel.
RibbonPanel.addTreeWidget([rowSpan, ...])	Add a tree widget to the panel.
RibbonPanel.addListWidget([rowSpan, ...])	Add a list widget to the panel.
RibbonPanel.addCalendarWidget([rowSpan, ...])	Add a calendar widget to the panel.
RibbonPanel.addSeparator([orientation, ...])	Add a separator to the panel.
RibbonPanel.addHorizontalSeparator([width, ...])	Add a horizontal separator to the panel.
RibbonPanel.addVerticalSeparator([width, ...])	Add a vertical separator to the panel.
RibbonPanel.addGallery([minimumWidth, ...])	Add a gallery to the panel.

3.5.4 Example

For example, using the following code,

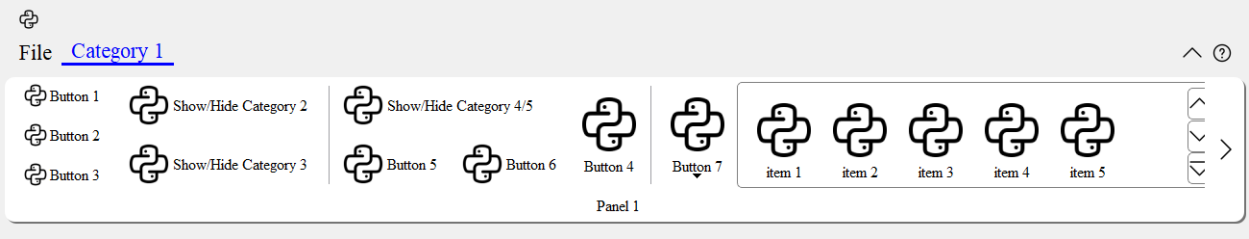
```
import sys

from qtpy import QtGui
from qtpy.QtWidgets import QApplication, QToolButton, QMenu, QLabel, QLineEdit
from qtpy.QtGui import QIcon
from qtpy.QtCore import Qt

from pyqtribbon import RibbonBar
from pyqtribbon.screenshotwindow import RibbonScreenShotWindow

if __name__ == '__main__':
    app = QApplication(sys.argv)
    app.setFont(QtGui.QFont("Times New Roman", 8))
    window = RibbonScreenShotWindow('panel.png')

    # Ribbon bar
    ribbonbar = RibbonBar()
    window.setMenuBar(ribbonbar)

    category1 = ribbonbar.addCategory("Category 1")
    panel = category1.addPanel("Panel 1", showPanelOptionButton=False)
    panel.addSmallButton("Button 1", icon=QIcon("python.png"))
    panel.addSmallButton("Button 2", icon=QIcon("python.png"))
    panel.addSmallButton("Button 3", icon=QIcon("python.png"))
    panel.addMediumToggleButton("Show/Hide Category 2", icon=QIcon("python.png"))
    panel.addVerticalSeparator()
    panel.addMediumToggleButton("Show/Hide Category 3", icon=QIcon("python.png"))
```

(continues on next page)

(continued from previous page)

```
    panel.addMediumToggleButton("Show/Hide Category 4/5", icon=QIcon("python.png"),
                                                          colSpan=2, alignment=Qt.AlignLeft)
    panel.addLargeButton("Button 4", icon=QIcon("python.png"))
    panel.addVerticalSeparator()
    panel.addMediumButton("Button 5", icon=QIcon("python.png"))
    panel.addMediumButton("Button 6", icon=QIcon("python.png"))

    button = panel.addLargeButton("Button 7", icon=QIcon("python.png"))
    menu = QMenu()
    menu.addAction(QIcon("python.png"), "Action 1")
    menu.addAction(QIcon("python.png"), "Action 2")
    menu.addAction(QIcon("python.png"), "Action 3")
    button.setMenu(menu)
    button.setPopupMode(QToolButton.InstantPopup)
    panel.addWidget(button, rowSpan=6)

    gallery = panel.addGallery(minimumWidth=500, popupHideOnClick=True)
    for i in range(100):
        gallery.addToggleButton(f'item {i+1}', QIcon("python.png"))
    popupMenu = gallery.popupMenu()
    submenu = popupMenu.addMenu(QIcon("python.png"), 'Submenu')
    submenu.addAction(QIcon("python.png"), "Action 4")
    popupMenu.addAction(QIcon("python.png"), "Action 1")
    popupMenu.addAction(QIcon("python.png"), "Action 2")
    popupMenu.addSeparator()
    popupMenu.addWidget(QLabel("This is a custom widget"))
    formLayout = popupMenu.addFormLayoutWidget()
    formLayout.addRow(QLabel("Row 1"), QLineEdit())

    # Show the window
    window.resize(1300, 250)
    window.show()

    sys.exit(app.exec_())
```

You can get a window like this:

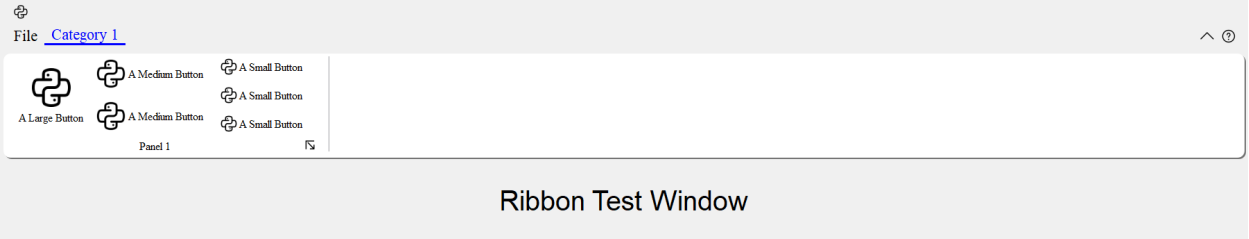


3.6 A Complete Example

The following code snippet is a complete example.

```
import sys

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

from pyqtribbon import RibbonBar
from pyqtribbon.screenshotwindow import RibbonScreenShotWindow
from pyqtribbon.utils import data_file_path

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

    # Central widget
    window = RibbonScreenShotWindow('tutorial-ribbonbar.png')
    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_())
```

You can get a window like this:


File
Category 1
A Large Button
A Medium Button
A Medium Button
A Small Button
A Small Button
A Small Button
Panel 1
Ribbon Test Window

CHAPTER

FOUR

API REFERENCES

4.1 Ribbon Bar

4.1.1 RibbonBar

class pyqtribbon.ribbonbar.**RibbonBar**(*title: str = ''*, *maxRows=6*, *parent=None*)

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

Bases: QMenuBar

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

Methods

actionAt(self, a0)	
actionGeometry(self, a0)	
activeAction(self)	
addAction(-> QAction)	
addCategoriesBy(data)	Add categories from a dict.
addCategory(title[, style, color])	Add a new category to the ribbon.
addContextCategories(name, titles[, color])	Add a group of context categories with the same tab color to the ribbon.
addContextCategory(title[, color])	Add a new context category to the ribbon.
addFileMenu()	Add a file menu to the ribbon.
addMenu(-> QAction -> QMenu)	
addNormalCategory(title)	Add a new category to the ribbon.
addQuickAccessButton(button)	Add a button to the quick access bar.
addRightToolButton(button)	Add a widget to the right button bar.
addSeparator(self)	
addTitleWidget(widget)	Add a widget to the title widget.
applicationOptionButton()	Return the application button.
categories()	Return a list of categories of the ribbon.
category(name)	Return the category with the given name.

continues on next page

Table 1 – continued from previous page

categoryVisible(category)	Return whether the category is shown.
clear(self)	
collapseRibbonButton()	Return the collapse ribbon button.
cornerWidget(self[, corner])	
currentCategory()	Return the current category.
helpRibbonButton()	Return the help button of the ribbon.
hideContextCategory(category)	Hide the given category or categories, if it is not a context category, nothing happens.
hideRibbon()	Hide the ribbon.
insertMenu(self, before, menu)	
insertSeparator(self, before)	
insertTitleWidget(index, widget)	Insert a widget to the title widget.
isDefaultUp(self)	
isNativeMenuBar(self)	
minimumSizeHint()	Return the minimum size hint of the widget.
quickAccessToolBar()	Return the quick access toolbar of the ribbon.
removeCategories(categories)	Remove a list of categories from the ribbon.
removeCategory(category)	Remove a category from the ribbon.
removeCollapseButton()	Remove the min button from the ribbon.
removeHelpButton()	Remove the help button from the ribbon.
removeTitleWidget(widget)	Remove a widget from the title widget.
ribbonHeight()	Get the total height of the ribbon.
ribbonVisible()	Get the visibility of the ribbon.
rightToolBar()	Return the right toolbar of the ribbon.
setActiveAction(self, action)	
setApplicationIcon(icon)	Set the application icon.
setCollapseButtonIcon(icon)	Set the icon of the min button.
setCornerWidget(self, widget[, corner])	
setCurrentCategory(category)	Set the current category.
setDefaultUp(self, a0)	
setHelpButtonIcon(icon)	Set the icon of the help button.
setNativeMenuBar(self, nativeMenuBar)	
setQuickAccessButtonHeight([height])	Set the height of the quick access buttons.
setRibbonHeight(height)	Set the total height of the ribbon.
setRibbonStyle(style)	Set the style of the ribbon.
setRibbonVisible(visible)	Set the visibility of the ribbon.
setRightToolBarHeight([height])	Set the height of the right buttons.
setTitle(title)	Set the title of the ribbon.
showCategoryByIndex(index)	Show category by tab index
showContextCategory(category)	Show the given category or categories, if it is not a context category, nothing happens.

continues on next page

Table 1 – continued from previous page

showRibbon()	Show the ribbon.
tabBar()	Return the tab bar of the ribbon.
title()	Return the title of the ribbon.

fileButtonClicked	
helpButtonClicked	

actionAt(*self*, *a0: QPoint*) → QAction

actionGeometry(*self*, *a0: QAction*) → QRect

activeAction(*self*) → QAction

addAction(*self*, *action: QAction*)

addAction(*self*, *text: str*) → QAction

addAction(*self*, *text: str*, *slot: 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: Union[QColor, GlobalColor] = 9*) → *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: Union[QColor, GlobalColor] = 9*) → *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*, *menu: QMenu*) → QAction

addMenu(*self*, *title: str*) → QMenu

addMenu(*self*, *icon: QIcon*, *title: 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

addTitleWidget(*widget: QWidget*)

Add a widget to the title widget.

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

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.

fileButtonClicked

Signal, the file button was clicked.

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*, *before: QAction*, *menu: QMenu*) → QAction

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

insertTitleWidget(*index: int*, *widget: QWidget*)

Insert a widget to the title widget.

Parameters

- **index** – The index to insert the widget.

- **widget** – The widget to insert.

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.

removeTitleWidget(*widget: QWidget*)

Remove a widget from the title widget.

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

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*, *action: 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*, *widget: QWidget*, *corner: Corner = Qt.TopRightCorner*)

setCurrentCategory(*category:* RibbonCategory)

Set the current category.

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

setDefaultUp(*self*, *a0: bool*)

setHelpButtonIcon(*icon: QIcon*)

Set the icon of the help button.

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

setNativeMenuBar(*self*, *nativeMenuBar: bool*)

setQuickAccessButtonHeight(*height: int = 30*)

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.

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.

title() → str

Return the title of the ribbon.

Returns The title of the ribbon.

4.2 Ribbon Title

4.2.1 RibbonApplicationButton

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

Bases: QTabBar

The TabBar for the title widget.

Methods

addAssociatedTabs(name, texts, color)	Add associated multiple tabs which have the same color to the tab bar.
addTab(text[, color])	Add a new tab to the tab bar.
currentTabColor()	Current tab color
indexOf(tabName)	Return the index of the tab with the given name.
paintEvent(a0)	Paint the tab bar.
removeAssociatedTabs(titles)	Remove tabs with the given titles.
tabTitles()	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 pyqtribbon.titlewidget.**RibbonTitleLabel**

Bases: QLabel

Title label in the ribbon bar.

4.2.4 RibbonTitleWidget

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

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

Bases: QFrame

The title widget of the ribbon.

Methods

addQuickAccessButton(button)	Add a widget to the quick access bar.
addRightToolButton(button)	Add a widget to the right button bar.
addTitleWidget(widget)	Add a widget to the title layout.
applicationButton()	Return the application button.
collapseRibbonButton()	Return the collapse ribbon button.
helpRibbonButton()	Return the help ribbon button.
insertTitleWidget(index, widget)	Insert a widget to the title layout.
quickAccessButtons()	Return the quick access buttons of the ribbon.
quickAccessToolBar()	Return the quick access toolbar of the ribbon.
removeCollapseButton()	Remove the min button from the ribbon.
removeHelpButton()	Remove the help button from the ribbon.
removeTitleWidget(widget)	Remove a widget from the title layout.
rightToolBar()	Return the right toolbar of the ribbon.
setApplicationIcon(icon)	Set the application icon.
setCollapseButtonIcon(icon)	Set the icon of the min button.
setHelpButtonIcon(icon)	Set the icon of the help button.
setQuickAccessButtonHeight([height])	Set the height of the quick access buttons.
setRightToolBarHeight([height])	Set the height of the right buttons.
setTitle(title)	Set the title of the ribbon.
tabBar()	Return the tab bar of the ribbon.
title()	Return the title of the ribbon.

collapseRibbonButtonClicked	
helpButtonClicked	

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.

addTitleWidget(*widget: QWidget*)

Add a widget to the title layout.

Parameters **widget** – The widget 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 wa clicked.

helpButtonClicked(*bool*)

Signal, the help button was clicked.

helpRibbonButton() → QToolButton

Return the help ribbon button.

Returns The help ribbon button.

insertTitleWidget(*index: int*, *widget: QWidget*)

Insert a widget to the title layout.

Parameters

- **index** – The index to insert the widget.
- **widget** – The widget to insert.

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.

removeTitleWidget(*widget: QWidget*)

Remove a widget from the title layout.

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

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 = 30*)

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.

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.

title() → str

Return the title of the ribbon.

Returns The title of the ribbon.

Ribbon Stacked Widget

4.2.5 RibbonStackedWidget

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

Bases: QStackedWidget

Stacked widget that is used to display the ribbon.

4.3 Ribbon Category

4.3.1 RibbonCategory

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

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

Bases: QFrame

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

Methods

addPanel(title[, showPanelOptionButton])	Add a new panel to the category.
addPanelsBy(data)	Add panels from a dictionary.
categoryStyle()	Return the button style of the category.
panel(title)	Return a panel from the category.
panels()	Return all panels in the category.
removePanel(title)	Remove a panel from the category.
setCategoryStyle(style)	Set the button style of the category.
setMaximumRows(rows)	Set the maximum number of rows.
takePanel(title)	Remove and return a panel from the category.
title()	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, "widgets": {

"widget-name": { "type": "Button", "arguments": {

"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.

setMaximumRows(*rows: int*)

Set the maximum number of rows.

Parameters **rows** – The maximum number of rows.

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

Bases: *RibbonCategory*

A normal category.

Methods

setCategoryStyle(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 pyqtribbon.category.**RibbonContextCategory**(*title: str*, *color: QColor*, *parent: QWidget*)

Bases: *RibbonCategory*

A context category.

Methods

categoryVisible()	Return whether the category is shown.
color()	Return the color of the context category.
hideContextCategory()	Hide the given category, if it is not a context category, nothing happens.
setCategoryStyle(style)	Set the button style of the category.
setCategoryVisible(visible)	Set the state of the category.
setColor(color)	Set the color of the context category.
showContextCategory()	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 pyqtribbon.category.**RibbonContextCategories**(*name: str*, *color: QColor*, *categories: Dict[str,* RibbonContextCategory*]*, *ribbon*)

Bases: Dict[str, *RibbonContextCategory*]

A list of context categories.

Methods

categoriesVisible()	Return whether the categories are shown.
color()	Return the color of the context categories.
hideContextCategories()	Hide the categories
name()	Return the name of the context categories.
setCategoriesVisible(visible)	Set the state of the categories.
setColor(color)	Set the color of the context categories.
setName(name)	Set the name of the context categories.
showContextCategories()	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 pyqtribbon.category.**RibbonCategoryStyle**(*value*)

Bases: IntEnum

The button style of a category.

4.3.6 RibbonCategoryScrollArea

4.3.7 RibbonCategoryScrollAreaContents

4.3.8 RibbonCategoryLayoutButton

4.3.9 RibbonCategoryLayoutWidget

4.4 Ribbon Panel

4.4.1 RibbonPanel

class pyqtribbon.panel.**RibbonPanel**(*title: str = ''*, *maxRows: int = 6*, *showPanelOptionButton=True*, *parent=None*)

class pyqtribbon.panel.**RibbonPanel**(*parent=None*)

Bases: QFrame

Panel in the ribbon category.

Methods

addButton([text, icon, style, showText, ...])	Add a button to the panel.
addCalendarWidget([rowSpan, colSpan, mode, ...])	Add a calendar widget to the panel.
addComboBox(items[, rowSpan, colSpan, mode, ...])	Add a combo box to the panel.
addDateEdit([rowSpan, colSpan, mode, ...])	Add a date edit to the panel.
addDateTimeEdit([rowSpan, colSpan, mode, ...])	Add a date time edit to the panel.
addDoubleSpinBox([rowSpan, colSpan, mode, ...])	Add a double spin box to the panel.
addFontComboBox([rowSpan, colSpan, mode, ...])	Add a font combo box to the panel.
addGallery([minimumWidth, popupHideOnClick, ...])	Add a gallery to the panel.
addHorizontalSeparator([width, rowSpan, ...])	Add a horizontal separator to the panel.

continues on next page

Table 2 – continued from previous page

addLabel(text[, rowSpan, colSpan, mode, ...])	Add a label to the panel.
addLargeButton([text, icon, showText, ...])	Add a large button to the panel.
addLargeToggleButton([text, icon, showText, ...])	Add a large toggle button to the panel.
addLargeWidget(widget[, mode, alignment, ...])	Add a large widget to the panel.
addLineEdit([rowSpan, colSpan, mode, ...])	Add a line edit to the panel.
addListWidget([rowSpan, colSpan, mode, ...])	Add a list widget to the panel.
addMediumButton([text, icon, showText, ...])	Add a medium button to the panel.
addMediumToggleButton([text, icon, ...])	Add a medium toggle button to the panel.
addMediumWidget(widget[, mode, alignment, ...])	Add a medium widget to the panel.
addPlainTextEdit([rowSpan, colSpan, mode, ...])	Add a plain text edit to the panel.
addProgressBar([rowSpan, colSpan, mode, ...])	Add a progress bar to the panel.
addSeparator([orientation, width, rowSpan, ...])	Add a separator to the panel.
addSlider([rowSpan, colSpan, mode, ...])	Add a slider to the panel.
addSmallButton([text, icon, showText, ...])	Add a small button to the panel.
addSmallToggleButton([text, icon, showText, ...])	Add a small toggle button to the panel.
addSmallWidget(widget[, mode, alignment, ...])	Add a small widget to the panel.
addSpinBox([rowSpan, colSpan, mode, ...])	Add a spin box to the panel.
addTableWidget([rowSpan, colSpan, mode, ...])	Add a table widget to the panel.
addTextEdit([rowSpan, colSpan, mode, ...])	Add a text edit to the panel.
addTimeEdit([rowSpan, colSpan, mode, ...])	Add a time edit to the panel.
addToggleButton([text, icon, style, ...])	Add a toggle button to the panel.
addTreeWidget([rowSpan, colSpan, mode, ...])	Add a tree widget to the panel.
addVerticalSeparator([width, rowSpan, ...])	Add a vertical separator to the panel.
addWidget(widget[, rowSpan, colSpan, mode, ...])	Add a widget to the panel.
addWidgetsBy(data)	Add widgets to the panel.
defaultRowSpan(rowSpan)	Return the number of span rows for the given widget type.
largeRows()	Return the number of span rows for large widgets.
maximumRows()	Return the maximal number of rows in the panel.
mediumRows()	Return the number of span rows for medium widgets.
panelOptionButton()	Return the panel option button.
removeWidget(widget)	Remove a widget from the panel.
rowHeight()	Return the height of a row.
setLargeRows(rows)	Set the number of span rows for large widgets.
setMaximumRows(maxRows)	Set the maximal number of rows in the panel.
setMediumRows(rows)	Set the number of span rows for medium widgets.
setPanelOptionToolTip(text)	Set the tooltip of the panel option button.
setSmallRows(rows)	Set the number of span rows for small widgets.
setTitle(title)	Set the title of the panel.
smallRows()	Return the number of span rows for small widgets.
title()	Get the title of the panel.
widget(index)	Get the widget at the given index.
widgets()	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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The button that was added.

addCalendarWidget(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The calendar widget that was added.

addComboBox(*items: List[str]*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The combo box that was added.

addDateEdit(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The date edit that was added.

addDateTimeEdit(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The date time edit that was added.

addDoubleSpinBox(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The double spin box that was added.

addFontComboBox(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The combo box that was added.

addGallery(*minimumWidth=800*, *popupHideOnClick=False*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the

maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The gallery.

addHorizontalSeparator(*width=6*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 2*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The separator.

addLabel(*text: str*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The button that was added.

addLargeWidget(*widget: QWidget*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*)

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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

addLineEdit(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The line edit that was added.

addListWidget(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 <

percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The button that was added.

addMediumWidget(*widget: QWidget*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*)

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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

addPlainTextEdit(*rowSpan: Union[int,* RibbonButtonStyle*] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The text edit that was added.

addProgressBar(*rowSpan: Union[int,* RibbonButtonStyle*] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the

maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The progress bar that was added.

addSeparator(*orientation=2*, *width=6*, *rowSpan: Union[int,* RibbonButtonStyle*] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The separator.

addSlider(*rowSpan: Union[int,* RibbonButtonStyle*] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The button that was added.

addSmallWidget(*widget: QWidget*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*)

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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The widget that was added.

addSpinBox(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The spin box that was added.

addTableWidget(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The table widget that was added.

addTextEdit(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The text edit that was added.

addTimeEdit(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

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*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The button that was added.

addTreeWidget(*rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → 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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The tree widget that was added.

addVerticalSeparator(*width=6*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Large*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*) → *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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 <

percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

Returns The separator.

addWidget(*widget: QWidget*, *rowSpan: Union[int, RibbonButtonStyle] = RibbonButtonStyle.Small*, *colSpan: int = 1*, *mode=RibbonSpaceFindMode.ColumnWise*, *alignment=132*, *fixedHeight: Union[bool, float] = False*)

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.
- **fixedHeight** – Whether to fix the height of the widget, it can be a boolean, a percentage or a fixed height, when a boolean is given, the height is fixed to the maximum height allowed if the value is True, when a percentage is given (0 < percentage < 1) the height is calculated from the height of the maximum height allowed, depends on the number of rows to span. The minimum height is 40% of the maximum height allowed.

addWidgetsBy(*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.

defaultRowSpan(*rowSpan: Union[int, RibbonButtonStyle]*)

Return the number of span rows for the given widget type.

Parameters **rowSpan** – row span or type.

Returns The number of span rows for the given widget type.

largeRows() → int

Return the number of span rows for large widgets.

Returns The number of span rows for large widgets.

maximumRows() → int

Return the maximal number of rows in the panel.

Returns The maximal number of rows in the panel.

mediumRows() → int

Return the number of span rows for medium widgets.

Returns The number of span rows for medium 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.

setLargeRows(*rows: int*)

Set the number of span rows for large widgets.

Parameters **rows** – The number of span rows for large widgets.

setMaximumRows(*maxRows: int*)

Set the maximal number of rows in the panel.

Parameters **maxRows** – The maximal number of rows in the panel.

setMediumRows(*rows: int*)

Set the number of span rows for medium widgets.

Parameters **rows** – The number of span rows for medium widgets.

setPanelOptionToolTip(*text: str*)

Set the tooltip of the panel option button.

Parameters **text** – The tooltip text.

setSmallRows(*rows: int*)

Set the number of span rows for small widgets.

Parameters **rows** – The number of span rows for small widgets.

setTitle(*title: str*)

Set the title of the panel.

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

smallRows() → int

Return the number of span rows for small widgets.

Returns The number of span rows for small widgets.

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 pyqtribbon.panel.**RibbonPanelItemWidget**(*parent=None*)

Bases: QFrame

Widget to display a panel item.

Methods

addWidget(widget)	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 pyqtribbon.panel.**RibbonSpaceFindMode**(*value*)

Bases: IntEnum

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

4.4.4 RibbonGridLayoutManager

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

Bases: object

Grid Layout Manager.

Methods

request_cells([rowSpan, colSpan, mode])	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 pyqtribbon.gallery.**RibbonGallery**(*minimumWidth=800*, *popupHideOnClick=False*, *parent=None*)

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

Bases: QFrame

A widget that displays a gallery of buttons.

Methods

addButton([text, icon, slot, shortcut, ...])	Add a button to the gallery
addToggleButton([text, icon, slot, ...])	Add a toggle button to the gallery
hidePopupWidget()	Hide the popup window
popupMenu()	Return the popup menu.
popupWindowSize()	Return the size of the popup window
resizeEvent(a0)	Resize the gallery.
setPopupHideOnClick(popupHideOnClick)	Set the hide on click flag
setPopupWindowSize(size)	Set the size of the popup window
setSelectedButton()	Set the selected button
showPopup()	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 pyqtribbon.gallery.**RibbonGalleryListWidget**(*parent=None*)

Bases: QListWidget

Gallery list widget.

Methods

resizeEvent(e)	Resize the list widget.
scrollToNextRow()	Scroll to the next row.
scrollToPreviousRow()	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 pyqtribbon.gallery.**RibbonGalleryButton**

Bases: QToolButton

Gallery button.

4.5.4 RibbonGalleryPopupListWidget

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

Bases: *RibbonGalleryListWidget*

Gallery popup list widget.

4.5.5 RibbonPopupWidget

class pyqtribbon.gallery.**RibbonPopupWidget**

Bases: QFrame

The popup widget for the gallery widget.

4.6 Ribbon Tool Button

4.6.1 RibbonToolButton

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

Bases: QToolButton

Tool button that is showed in the ribbon.

Methods

addRibbonMenu()	Add a ribbon menu for the button.
buttonStyle()	Get the button style of the button.
maximumIconSize()	Get the maximum icon size of the button.
setButtonStyle(style)	Set the button style of the button.
setMaximumIconSize(size)	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 pyqtribbon.toolbutton.**RibbonButtonStyle**(*value*)

Bases: IntEnum

Button style, Small, Medium, or Large.

4.7 Ribbon Separator

4.7.1 RibbonSeparator

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

class pyqtribbon.separator.**RibbonSeparator**(*parent=None*)

Bases: QFrame

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

Methods

paintEvent(event)	Paint the separator.
setTopBottomMargins(top, bottom)	Set the top and bottom margins.
sizeHint()	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 pyqtribbon.separator.**RibbonHorizontalSeparator**(*width: int = 6*, *parent=None*)

Bases: *RibbonSeparator*

Horizontal separator.

4.7.3 RibbonVerticalSeparator

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

Bases: *RibbonSeparator*

Vertical separator.

4.8 Ribbon Menu

4.8.1 RibbonMenu

class pyqtribbon.menu.**RibbonMenu**(*title: str = ''*, *parent=None*)

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

Bases: QMenu

Methods

addFormLayoutWidget()	Add a form layout widget to the menu.
addGridLayoutWidget()	Add a grid layout widget to the menu.
addHorizontalLayoutWidget()	Add a horizontal layout widget to the menu.
addLabel([text, alignment])	Add a label to the menu.
addSpacing([spacing])	Add spacing to the menu.
addVerticalLayoutWidget()	Add a vertical layout widget to the menu.
addWidget(widget)	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 pyqtribbon.menu.**RibbonPermanentMenu**(*title: str = ''*, *parent=None*)

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

Bases: *RibbonMenu*

A permanent menu.

Methods

actionEvent(self, a0)
hideEvent(self, a0)

actionAdded	

actionEvent(*self*, *a0: QActionEvent*)

hideEvent(*self*, *a0: QHideEvent*)

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

A

B

C

D

S

T

W