|  |  | **PyQt5: GUi design** | April 2022 |
| --- | --- | --- | --- |
|  |  |  | |
| **INSIDE**  \_\_  **Installation and running.**  \_\_  **Using new form dialogue box and main window.**  \_\_  **Qt designer vs Hand Coding** |  | PyQt5 Tutorial 2022, Create Python GUIs with Qt | |
|  |  | Introduction PyQt5 is a set of Python bindings for Qt5 application framework from Digia. Qt library is one of the most powerful GUI libraries. PyQt5 is the latest version of a GUI widgets toolkit developed by Riverbank Computing. It is a Python interface for **Qt**, one of the most powerful, and popular cross-platform GUI library. PyQt5 is a blend of Python programming language and the Qt library.  Qt Designer also allows you to preview your GUIs using different styles and resolutions, connect [signals and slots](https://www.riverbankcomputing.com/static/Docs/PyQt5/signals_slots.html), create menus and toolbars, and more.    **Installing and running Qt Designer**         **Using Qt Designer’s New Form Dialog** When Qt Designer is set to run, the application’s main window and the *New Form* dialog is presented. In this dialog, you can select from five available GUI templates. These templates include options to create dialogs, main windows, and custom widgets:   | Template | Form Type | Widgets | | --- | --- | --- | | Dialog with Buttons Bottom | Dialog | *OK* and *Cancel* buttons laid out horizontally on the bottom-right corner | | Dialog with Buttons Right | Dialog | *OK* and *Cancel* buttons laid out vertically on the top-right corner | | Dialog without Buttons | Dialog | No | | Main Window | Main Window | A menu bar at the top and a status bar at the bottom | | Widget | Widget | No |   A Sample Form GUI created from the same.       **Working With Qt Designer’s Main Window** Qt Designer’s **main window** consists of a menu bar including options for saving and managing forms, editing forms and altering the edit mode, laying out and previewing forms, and also for tuning the application’s settings and using its help documentation:        The main window also provides a toolbar that displays commonly used options. You’ll use most of these options when you’re editing and laying out your forms. These options are also available in the main menu, especially in the *File*, *Edit*, and *Form* menus:      Qt Designer’s main window also consists of a few **dock windows** that provide an indifferent set of features and tools:   * Widget Box * Object Inspector * Property Editor * Resource Browser * Action Editor * Signal/Slot Editor   ~The **Widget Box** provides a selection of layout managers, spacers, standard widgets, and other objects that you can use to create a GUI for your dialogs and windows.  ~The **Object Inspector** provides a [tree](https://doc.qt.io/qt-5/objecttrees.html) view of all the objects on the current form.  ~The **Property Editor** holds a two-column table with the active object’s properties and their values.  ~**Resource Browser** provides a quick way of adding resources, such as icons, translation files, images, and other binary files to your applications  **~Action Editor** provides a way to create actions and add them to your forms  **~Signal/Slot Editor** provides a way to connect signals and slots in your forms. **Using Qt Designer vs Hand Coding Your GUIs** With PyQt, there are at least two options for developing the GUI of a window or dialog: Qt Designer can be used, or you can hand code the GUI in Python code. Both options have their pros and cons. Sometimes it’s difficult to decide when to use one or the other.  Qt Designer broadens the options of a user-friendly graphical interface that allows you to quickly create GUIs. This can boost your productivity as a developer and shorten your development cycles.  Hand coding your GUIs, additionally, can provide much more control over them. With this approach, extending new components and features doesn’t require any additional tools beyond your [code editor or IDE](https://realpython.com/python-ides-code-editors-guide/), which can be quite convenient in some development environments.    Whether you use Qt Designer or hand code your GUIs is a personal decision. Here are some general considerations on both approaches:  ~**Developer’s productivity** is comparatively higher in Qt designer where as **control over GUI components** happens to be more powerful in hand coding.  ~**The separation from GUI logic from business logic** is evidently interpreted through Qt designer rather than hand coding which provides **flexibility for exploration ,learning ,prototyping, and sketching.**   **Conclusion.** Beyond these concerns, if you’re just starting with PyQt, then Qt Designer can help you discover available widgets, layout managers, base classes, properties and their typical values, and so on.  One last difference between using Qt Designer and hand coding a GUI is that you need to run an extra step when using Qt Designer: translating .ui files into Python code | |
|  |  |  | |

|  | |  |  |
| --- | --- | --- | --- |
|  | |  |  |
|  |  | | |