# PyQt vs. PySide

PyQt and PySide are both sets of Python bindings for the Qt framework. While they serve similar purposes, there are key differences between the two.

## General Comparison

|  |  |  |
| --- | --- | --- |
|  | **PyQt** | **PySide** |
| **Licensing** | - Available under the **GPL** (General Public License) and a commercial license.  - If you develop an application using PyQt and distribute it without purchasing a commercial license, your application must also be open-source under the GPL. | - Available under **LGPL** (Lesser General Public License).  - Allows use in proprietary applications without needing to open-source your code. |
| **Development and Maintenance** | - Developed and maintained by **Riverbank Computing**.  - Longer history and established presence in the community. | - Officially developed by **Qt Company**.  - Often receives updates in alignment with new Qt releases. |
| **API Differences** | - Slightly different API compared to PySide.  - Some classes and methods may have different names or parameters. | - Aims for a more consistent API with the underlying Qt framework, making it easier for those familiar with C++. |
| **Documentation** | - Extensive documentation, tutorials, and community support due to its longer presence.  - Many existing resources and examples available. | - Good documentation maintained by the Qt Company.  - Community support is growing, especially as it gains popularity. |

## Module Comparison

*This is Poe GPT answer (I double-check with Google, and there are some incorrect info, so just take the table below as a reference only)*:

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **PySide** | **PyQt** | **Notes** |
| QtCore | ✔️ | ✔️ | Core non-GUI functionality |
| QtGui | ✔️ | ✔️ | Basic graphics and windowing capabilities |
| QtWidgets | ✔️ | ✔️ | Essential GUI components and widgets |
| QtMultimedia | ✔️ | ✔️ | Audio and video playback |
| QtMultimediaWidgets | ✔️ | ✔️ | Widgets for multimedia applications |
| QtNetwork | ✔️ | ✔️ | Networking capabilities |
| QtWebEngine | ✔️ | ✔️ | Web content rendering based on Chromium |
| QtWebSockets | ✔️ | ✔️ | WebSocket protocol support |
| QtXml | ✔️ | ✔️ | XML parsing and handling |
| QtSql | ✔️ | ✔️ | Database interaction and SQL support |
| QtQml | ✔️ | ✔️ | QML language bindings and integration |
| QtQuick | ✔️ | ✔️ | Quick application development |
| Qt3D | ✔️ | ✔️ | Support for 3D graphics and rendering |
| QtPrintSupport | ✔️ | ✔️ | Printing capabilities |
| QtStyle | ✔️ | ✔️ | Custom styles and theming |
| QtCharts | ✔️ | ❌ | Interactive charts and visualizations (PySide only) |
| QtDataVisualization | ✔️ | ❌ | 3D data visualization (PySide only) |
| QtDataVisualization | ✔️ | ❌ | 3D data visualization (PySide only) |
| QtScript | ❌ | ✔️ | Scripting language support (PyQt only) |
| QtConcurrent | ❌ | ✔️ | Support for concurrent programming (PyQt only) |
| QtTest | ✔️ | ✔️ | Testing framework for Qt applications |
| QtLottie | ✔️ | ❌ | Animation support using Lottie (PySide only) |
| QtLocation | ✔️ | ❌ | Geolocation support (PySide only) |
| QtHelp | ✔️ | ❌ | Help system support (PySide only) |
| QtPositioning | ✔️ | ❌ | Positioning capabilities (PySide only) |
| QtRemoteObjects | ✔️ | ❌ | Remote object communication (PySide only) |
| QtWebEngineWidgets | ❌ | ✔️ | Additional features in PyQt |
| Commercial Modules | ❌ | ✔️ | Some specialized commercial modules (PyQt only) |
|  |  |  |  |
| Qt Creator | ✔️ | ✔️ |  |

Other resources:

<https://machinekoder.com/pyqt-vs-qt-for-python-pyside2-pyside/>

<https://www.e-education.psu.edu/geog489/node/2225>

# PySide Tools

PySide provides a helpful tool set to support Python development with Qt. Most of these tools are originated from the Qt framework, but they’re specically designed to use with Python instead of C++ like the Qt framework's counterparts. So, don't supprise if you find minor differences between them.

|  |  |  |
| --- | --- | --- |
| **Tool Name** | **Purpose** | **Note** |
| **pyside6-assistant.exe** | Launches Qt Assistant for viewing Qt **documentation**. |  |
| **pyside6-balsam.exe** | Part of the Balsam framework for creating responsive UIs in Qt. |  |
| **pyside6-balsamui.exe** | A UI tool for Balsam, allowing visual interface design. |  |
| **pyside6-deploy.exe** | **Packages** and deploys PySide6 applications for distribution. |  |
| **pyside6-designer.exe** | Opens Qt Designer for **designing GUI forms** with a **drag-and-drop** interface. | The most important and mostly-used |
| **pyside6-genpyi.exe** | Generates Python type stubs (.pyi files) for type hinting support in IDEs. |  |
| **pyside6-linguist.exe** | Launches Qt Linguist for translating Qt applications. |  |
| **pyside6-lrelease.exe** | Compiles .ts translation files into .qm files for application use. |  |
| **pyside6-lupdate.exe** | Updates **translation** files by scanning source code for translatable strings. |  |
| **pyside6-metaobjectdump.exe** | Dumps meta-object information for Qt objects, useful for debugging. |  |
| **pyside6-qmlls.exe** | Starts a **QML** language server for IDE support and code completion. |  |
| **pyside6-qml.exe** | Executes **QML** files and applications directly. |  |
| **pyside6-qmlcachegen.exe** | Generates **QML** cache files to improve loading performance. |  |
| **pyside6-qmlformat.exe** | Formats **QML** files according to style guidelines. |  |
| **pyside6-qmlimportscanner.exe** | Scans **QML** files for imported modules to manage dependencies. |  |
| **pyside6-qsb.exe** | Tool for building and managing Qt projects. |  |
| **pyside6-rrc.exe** | **Compiles resource files** (.qrc) into a binary format for use in applications. | Used a lot |
| **pyside6-qmltyperegistrar.exe** | Registers QML types with the QML engine for custom type usage. |  |
| **pyside6-qmllint.exe** | Lints QML files to catch errors and enforce coding standards. |  |
| **pyside6-svgtoqml.exe** | Converts SVG files to QML format for application use. |  |
| **pyside6-uic.exe** | **Compiles UI files from Qt Designer into Python** code. | Used a lot |

# Qt Designer in PySide

## Installation

Although the **standalone** version of Qt Designer can be [installed from Qt installer package](https://www.pythonguis.com/installation/install-qt-designer-standalone/), it’s not what we can use because it only supports C++.

Luckily, PySide6 includes Qt Designer automatically when you install it with pip. This version has Python support and is what we will use. After installation, you can run Qt Designer from the command line using the built-in launcher of PySide6:

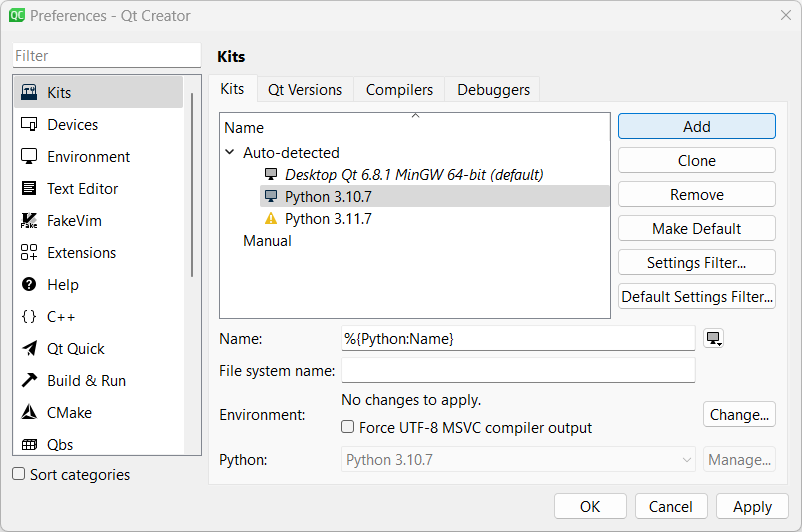
$ pyside6-designer

## Usage

Check the Qt Designer.docx tutorial.

# Qt Creator in PySide

To set up a Python interpreter in Qt Creator:



# Modern GUIs

## Qt Designer

Themes:

* <https://github.com/Wanderson-Magalhaes/Modern_GUI_PyDracula_PySide6_or_PyQt6>
* <https://github.com/Wanderson-Magalhaes/PyOneDark_Qt_Widgets_Modern_GUI>
* <https://github.com/gmarull/qtmodern>

Custom widgets: <https://github.com/KhamisiKibet/QT-PyQt-PySide-Custom-Widgets>

## QML

<https://github.com/zhuzichu520/FluentUI2/tree/main/FluentUI> and <https://www.youtube.com/watch?v=HoFjD62QCtw>

# Graphs and Charts

## Matplotlib

<https://matplotlib.org/>

## PyQtGraph

<https://www.pyqtgraph.org/>

[PyQtGraph — pyqtgraph 0.14.0dev0 documentation](https://pyqtgraph.readthedocs.io/en/latest/)

# References

<https://www.pythonguis.com/pyside6-tutorial/>

<https://doc.qt.io/qtforpython-6/gettingstarted.html>

<https://github.com/mfitzp/books/tree/main/create-gui-applications/pyside6>