Einführung - PyQT in Python

Definition:

PyQt ist eine Sammlung von Python-Bindings für die Qt-Bibliotheken. Es ermöglicht Python-Entwicklern, die Funktionen von Qt zur plattformübergreifenden GUI-Entwicklung direkt in Python zu nutzen.

Entwickelt von: Riverbank Computing.

PyQt wurde von Firma Riverbank Computing (Großbritannien) entwickelt, und die erste Version wurde im Jahr 1998 veröffentlicht.

Riverbank Computing ist bekannt für die Entwicklung von Python-Bindings für Qt und hat im Laufe der Jahre kontinuierlich daran gearbeitet, PyQt zu aktualisieren und zu verbessern, um mit den neuesten Versionen von Qt Schritt zu halten.

Einführung - PyQT in Python

Was ist Qt?

Qt ist ein freies und Open-Source-Widget-Toolkit zur Erstellung grafischer Benutzeroberflächen sowie plattformübergreifender Anwendungen, die auf verschiedenen Software- und Hardwareplattformen wie Linux, Windows, macOS, Android oder eingebetteten Systemen mit wenig oder keiner Änderung des zugrunde liegenden Codes laufen.

- Entwickelt von: Ursprünglich von Trolltech entwickelt, jetzt von The Qt Company gepflegt.
- Programmiersprache: Hauptsächlich in C++ geschrieben.
- Kernkomponenten: Qt umfasst eine umfassende Bibliothekssammlung für die Erstellung von GUI, Datenverarbeitung, Netzwerk und mehr.
- Modular: Qt's modularer Aufbau ermöglicht es Entwicklern, nur die benötigten Bibliotheken in ihre Anwendungen zu integrieren. Dies führt zu effizienterem Code und einer besseren Strukturierung der Projekte.

Einführung - PyQT in Python

Verbindung zwischen PyQt und Qt

Binding-Framework:

PyQt übersetzt das Qt-Framework von C++ in Python. Das bedeutet, dass Sie die leistungsstarken Funktionen der Qt-Bibliothek nutzen können, während Sie Ihre Anwendung in Python schreiben.

Ähnliche API:

PyQt strebt an, eine ähnliche API wie Qt beizubehalten. Das bedeutet, wenn Sie wissen, wie man Qt in C++ verwendet, können Sie sich leicht an PyQt in Python anpassen. Die meisten Klassen und Methoden haben dieselben Namen und Funktionen.

Verwendung der Qt-Bibliotheken:

Wenn Sie eine PyQt-Anwendung schreiben, verwenden Sie im Wesentlichen die Qt-Bibliotheken im Hintergrund. Die PyQt-Schicht ermöglicht es Ihnen, Python-Code zu schreiben, der mit diesen Bibliotheken interagiert.

Einführung - PyQT in Python

Warum PyQt verwenden?

Benutzerfreundlichkeit:

Python ist im Allgemeinen einfacher zu erlernen und zu verwenden als **C++**. PyQt bringt die Leistungsfähigkeit von Qt zu Python-Entwicklern.

Schnelle Entwicklung:

Die dynamische Natur von **Python** und die leistungsstarken Standardbibliotheken ermöglichen eine schnelle Entwicklung.

Riesige Auswahl an Widgets und integrierten Funktionen:

Viele Widgets und eingebaute Funktionen für jedes GUI-Projekt

Plattformübergreifend:

PyQt erbt die Fähigkeit von Qt, plattformübergreifende Anwendungen zu erstellen.

Einführung - PyQT in Python

Hauptmerkmale von PyQT5:

Plattformübergreifend:

Windows, macOS, Linux.

Umfassend:

Bietet eine breite Palette von Widgets und Tools.

Integrierte Entwicklung:

Unterstützung für Ereignisbehandlung, Signale und Slots.

High-DPI-Unterstützung:

Bessere Anzeige auf modernen Bildschirmen.

Einführung - PyQT in Python

PyQt Versionen & Installation

Versionen: PyQt4 (Bindings für Qt4 - Wird nicht mehr aktiv gepflegt. Qt4 selbst gilt als veraltet.)
PyQt5 (Bindings für Qt5) - ab Jahr 2016
PyQt6 (Bindings für Qt6) - seit 2021.

Lizenzierung: Verfügbar unter GPL- General Public License und kommerziellen Lizenzen.

In diesem Kurs verwenden wir PyQt5, da es stabil, weit verbreitet und für Einsteiger einfacher verständlich ist. Die Konzepte sind später auf PyQt6 übertragbar – es kann Unterschiede bei Klassennamen und Konstanten geben, aber die Grundkonzepte bleiben gleich. Die meisten Online-Ressourcen, YouTube-Tutorials und Fachbücher basieren auf PyQt5.

Installation von PyQt5: pip install PyQt5
Installation PyQt6: pip install PyQt6
Installation überprüfen: pip show PyQt5 oder pip show PyQt6

Wichtigste Konzepte in PyQT (für Beginner!)

QApplication:

Notwendig für jede GUI-Anwendung, um die Ereignisschleife zu starten und das GUI-Subsystem zu initialisieren. QApplication ist die Klasse, die jede Qt GUI-Anwendung benötigt. Sie initialisiert das GUI-Subsystem und startet die Ereignisschleife. Jede GUI-Anwendung benötigt eine Instanz von QApplication, um korrekt zu funktionieren.

QMainWindow:

Bietet eine umfassende und flexible Struktur für Hauptfenster, einschließlich Menüleisten, Werkzeugleisten, Statusleisten und Dock-Widgets.

Widgets:

Widgets sind die Bausteine der Benutzeroberfläche. Das Wissen, wie man Widgets verwendet und anordnet, ist grundlegend für die Erstellung von GUI-Anwendungen. Jedes UI-Element, wie Buttons, Labels und Textfelder, ist ein Widget.

Layout-Management:

Wichtig für die Organisation und Anpassung der Benutzeroberfläche. Layouts helfen dabei, Widgets in einem Fenster zu organisieren und die Benutzeroberfläche an verschiedene Fenstergrößen anzupassen. Ohne Layouts kann die Benutzeroberfläche unübersichtlich und schwer zu verwalten sein. Layout-Manager wie QVBoxLayout, QHBoxLayout und QGridLayout arrangieren Widgets automatisch und dynamisch.

Events:

Basis für das Verständnis der Ereignisverarbeitung und Benutzerinteraktionen (z.B. wie Mausklicks und Tastendrücke). Die QApplication-Klasse startet die Ereignisschleife, die die Verarbeitung von Ereignissen ermöglicht. In PyQt5 gibt es zwei Hauptmethoden, um auf Ereignisse zu reagieren: Direkte Ereignisbehandlung und Signal- und Slot-Mechanismus.

Wichtigste Konzepte in PyQT (für Beginner!)

Signal- und Slot-Mechanismus:

Essentiell für die Handhabung von Ereignissen und die Interaktion zwischen verschiedenen Teilen der Anwendung. Kann als eine Unterkategorie der Ereignisbehandlung in PyQt5 betrachtet werden. Signale werden von Objekten gesendet, wenn bestimmte Ereignisse eintreten, und Slots sind Funktionen, die als Reaktion auf diese Signale aufgerufen werden.

Multi-Document Interface (MDI):

Ein Multi-Document Interface (MDI) ist eine Art von Benutzeroberfläche, bei der mehrere Dokumente innerhalb eines einzigen Hauptfensters geöffnet und bearbeitet werden können. Jedes Dokument wird in einem eigenen Unterfenster innerhalb des Hauptfensters geöffnet. n PyQt5 wird dies durch die Verwendung von QMdiArea und QMdiSubWindow realisiert.

Einführung - PyQT in Python

Wichtigste Widgets in PyQT5

Widget	Beschreibung	Klasse
Label	Textanzeige	QLabel
Button	Klickbares Element	QPushButton
Textfeld	Einzeilige Eingabe	QLineEdit
Textbereich	Mehrzeilige Eingabe	QTextEdit
Checkbox	Auswahloption	QCheckBox
Radiobutton	Einzelauswahl	QRadioButton
Kombinationsfeld	Dropdown-Auswahl	QComboBox
Liste	Elementliste	QListWidget
Tabelle	Tabellarische Anzeige	QTableWidget
Fortschrittsbalken	Visualisiert Fortschritt	QProgressBar

Eine vollständige Übersicht aller Qt-Klassen findest du unter: https://doc.qt.io/archives/qt-5.15/classes.html

Einführung - PyQT in Python

Mehr erkunden :

https://www.pythonguis.com/pyqt5-tutorial/

https://www.tutorialspoint.com/pyqt/pyqt_basic_widgets.htm

https://www.tutorialspoint.com/pyqt5/index.htm

https://www.riverbankcomputing.com/static/Docs/PyQt5/