**Loading and Displaying Images in a Window using PyQt6 and PySide6**

Both **PyQt6** and **PySide6** provide the QLabel widget, which can display images using QPixmap.

**1. Using PyQt6**

Create a simple GUI that loads and displays an image.

**Example: display\_image\_pyqt.py**

import sys

from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout

from PyQt6.QtGui import QPixmap

class ImageWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Image Viewer - PyQt6")

self.setGeometry(100, 100, 600, 400)

layout = QVBoxLayout()

# QLabel to display image

self.label = QLabel(self)

pixmap = QPixmap("assets/logo.png") # Path to the image file

self.label.setPixmap(pixmap)

self.label.setScaledContents(True) # Scale image to fit the label

layout.addWidget(self.label)

self.setLayout(layout)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())

**2. Using PySide6**

The code is nearly identical to PyQt6, with just the import changed.

**Example: display\_image\_pyside.py**

import sys

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout

from PySide6.QtGui import QPixmap

class ImageWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Image Viewer - PySide6")

self.setGeometry(100, 100, 600, 400)

layout = QVBoxLayout()

# QLabel to display image

self.label = QLabel(self)

pixmap = QPixmap("assets/logo.png") # Path to the image file

self.label.setPixmap(pixmap)

self.label.setScaledContents(True) # Scale image to fit the label

layout.addWidget(self.label)

self.setLayout(layout)

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())

**Explanation**

1. **QPixmap** loads the image from the given path.
2. **QLabel.setPixmap(pixmap)** sets the image inside the label.
3. **setScaledContents(True)** ensures the image resizes to fit the label.
4. **Window Size Handling**: Ensure the image file exists in the correct path (assets/logo.png in this case).

**How to Run**

1. Install dependencies:

pip install PyQt6 # or pip install PySide6

1. Place an image inside an assets/ folder.
2. Run the script:

python display\_image\_pyqt.py

or

python display\_image\_pyside.py

**Understanding PyQt6 and PySide6 Code for Displaying Images**

Both **PyQt6** and **PySide6** use Qt's QPixmap and QLabel to display images in a GUI. Since their syntax is nearly identical, the explanation applies to both.

**1. Importing Required Modules**

import sys

from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout

from PyQt6.QtGui import QPixmap

* sys: Required for system-level operations like exiting the application properly.
* QApplication: The application instance, which handles events and the main loop.
* QWidget: The base class for all UI windows.
* QLabel: A widget that can display text or images.
* QVBoxLayout: A vertical layout manager to arrange widgets.
* QPixmap: A Qt class used for handling images.

For PySide6, the imports are exactly the same, except replacing PyQt6 with PySide6:

from PySide6.QtWidgets import QApplication, QWidget, QLabel, QVBoxLayout

from PySide6.QtGui import QPixmap

**2. Creating a Main Window**

class ImageWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setWindowTitle("Image Viewer - PyQt6")

self.setGeometry(100, 100, 600, 400)

* ImageWindow(QWidget): Defines a new window that inherits from QWidget.
* super().\_\_init\_\_(): Calls the constructor of QWidget to set up the window.
* self.setWindowTitle("Image Viewer - PyQt6"): Sets the window title.
* self.setGeometry(100, 100, 600, 400): Defines the **position (100, 100)** and **size (600x400)** of the window.

**3. Creating a Layout**

layout = QVBoxLayout()

* QVBoxLayout(): A layout that arranges widgets vertically.

**4. Creating a QLabel for Displaying the Image**

self.label = QLabel(self)

pixmap = QPixmap("assets/logo.png") # Path to the image file

self.label.setPixmap(pixmap)

self.label.setScaledContents(True) # Scale image to fit the label

* QLabel(self): Creates a label widget inside the window.
* QPixmap("assets/logo.png"): Loads an image file from the assets/ directory.
* self.label.setPixmap(pixmap): Sets the loaded image into the label.
* self.label.setScaledContents(True): Ensures the image resizes to fit the label dimensions.

**5. Adding the Label to the Layout**

layout.addWidget(self.label)

self.setLayout(layout)

* layout.addWidget(self.label): Adds the image label to the layout.
* self.setLayout(layout): Applies the layout to the window.

**6. Running the Application**

if \_\_name\_\_ == "\_\_main\_\_":

app = QApplication(sys.argv)

window = ImageWindow()

window.show()

sys.exit(app.exec())

* if \_\_name\_\_ == "\_\_main\_\_": Ensures the script runs only when executed directly.
* QApplication(sys.argv): Initializes the Qt application.
* window = ImageWindow(): Creates an instance of the ImageWindow class.
* window.show(): Displays the window.
* sys.exit(app.exec()): Starts the event loop and ensures a clean exit.

**How This Works**

1. **The script initializes the application** (QApplication).
2. **A QWidget window is created** (ImageWindow).
3. **A QLabel loads an image** using QPixmap.
4. **The image is displayed** inside the window.
5. **The layout manager ensures proper positioning**.
6. **The event loop starts, allowing interaction** (app.exec()).

**Common Modifications**

**1. Selecting an Image Dynamically**

Instead of hardcoding "assets/logo.png", allow users to select an image using QFileDialog:

from PyQt6.QtWidgets import QFileDialog

image\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")

if image\_path:

pixmap = QPixmap(image\_path)

self.label.setPixmap(pixmap)

This opens a file dialog and updates the QLabel dynamically.

**2. Resizing the Window to Fit the Image**

To resize the window based on the image dimensions:

self.resize(pixmap.width(), pixmap.height())

This adjusts the window size automatically.

**Conclusion**

* **PyQt6 and PySide6** provide the same functionality for displaying images.
* QPixmap loads images efficiently.
* QLabel acts as a container for images.
* Layouts like QVBoxLayout organize widgets properly.

**Advanced PyQt6/PySide6 example**

**Advanced PyQt6/PySide6 example** that allows users to **load an image dynamically** using a button and displays it in the window.

**Project Structure**

**A close-up of a white background

AI-generated content may be incorrect.**

**Complete Code (main.py)**

import sys  
  
from PyQt6.QtCore import Qt  
from PyQt6.QtWidgets import QApplication, QWidget, QLabel, QPushButton, QVBoxLayout, QFileDialog  
from PyQt6.QtGui import QPixmap  
  
class ImageViewer(QWidget):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.setWindowTitle("Dynamic Image Viewer - PyQt6")  
 self.setGeometry(200, 200, 600, 500)  
  
 *# Layout* self.layout = QVBoxLayout()  
  
 *# QLabel to display image* self.label = QLabel(self)  
 self.label.setText("No image loaded")  
 self.label.setStyleSheet("QLabel { font-size: 16px; color: gray; }")  
 self.label.setAlignment(Qt.AlignmentFlag.AlignCenter) *# Center alignment label.setAlignment(Qt.AlignmentFlag.AlignCenter)  
  
 # QPushButton to load image* self.load\_button = QPushButton("Load Image")  
 self.load\_button.clicked.connect(self.load\_image)  
  
 *# Add widgets to layout* self.layout.addWidget(self.label)  
 self.layout.addWidget(self.load\_button)  
  
 self.setLayout(self.layout)  
  
 def load\_image(self):  
 *""" Opens a file dialog to select an image and displays it in QLabel """* image\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")  
 if image\_path:  
 pixmap = QPixmap(image\_path)  
 self.label.setPixmap(pixmap)  
 self.label.setScaledContents(True) *# Scale image to fit label size* self.label.setFixedSize(500, 400) *# Set a fixed size for better appearance* else:  
 self.label.setText("No image selected")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = ImageViewer()  
 window.show()  
 sys.exit(app.exec())

**Code Explanation**

**1. Initializing the Window (ImageViewer class)**

self.setWindowTitle("Dynamic Image Viewer - PyQt6")

self.setGeometry(200, 200, 600, 500)

* Sets the title and window size.

**2. QLabel (For Displaying Image)**

self.label = QLabel(self)

self.label.setText("No image loaded")

self.label.setStyleSheet("QLabel { font-size: 16px; color: gray; }")

self.label.setAlignment(4) # Center alignment

* Displays text when no image is loaded.
* Uses **CSS styling** for better appearance.

**3. QPushButton (For Loading an Image)**

self.load\_button = QPushButton("Load Image")

self.load\_button.clicked.connect(self.load\_image)

* Clicking the button **calls load\_image()**.

**4. File Dialog to Select Image**

image\_path, \_ = QFileDialog.getOpenFileName(self, "Open Image File", "", "Images (\*.png \*.jpg \*.jpeg \*.bmp)")

* Opens a dialog for users to select an image.
* Filters file types to **PNG, JPG, JPEG, BMP**.

**5. Displaying the Image**

if image\_path:

pixmap = QPixmap(image\_path)

self.label.setPixmap(pixmap)

self.label.setScaledContents(True) # Scale image to fit label size

self.label.setFixedSize(500, 400) # Set a fixed size for better appearance

else:

self.label.setText("No image selected")

* Loads the selected image using QPixmap.
* **Scales the image** to fit within the QLabel.
* **Handles cases when no image is selected**.

**How to Run the Application**

1. **Install dependencies**:

pip install PyQt6

or for PySide6:

pip install PySide6

1. **Run the script**:

python main.py

**Expected Behavior**

* Initially, the window displays **"No image loaded"**.
* Clicking **"Load Image"** opens a file dialog.
* Selecting an image **displays it in the window**.
* If no image is selected, it shows **"No image selected"**.