**Bài thực hành 1**

**Lập trình giao diện Python với PyQt5 và Qt Designer**

**\*\*\*\*\***

**A. Mục tiêu**

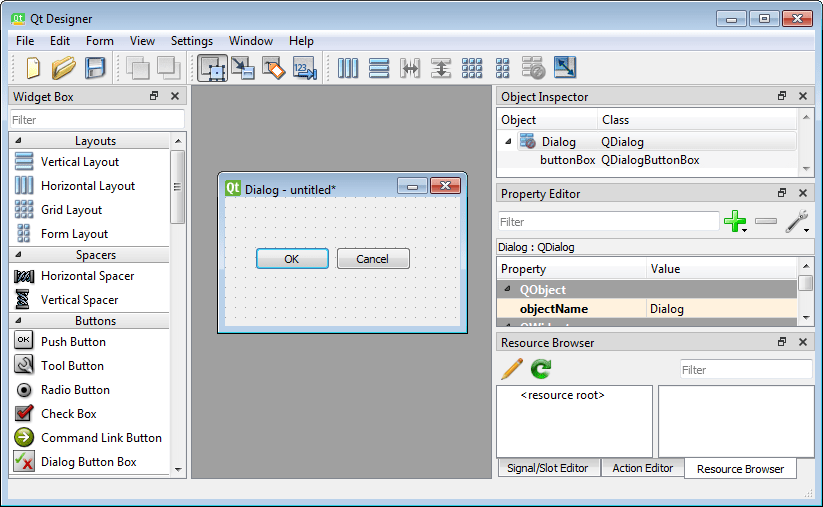
- SV làm quen với phần mềm thiết kế GUI trong Python.

- Thiết kế GUI đẹp.

**B. Nội dung**

**1. Qt Designer là gì?**

Qt Designer là một công cụ để nhanh chóng xây dựng giao diện người dùng đồ họa với các widget từ khung Qt GUI. Nó cung cấp cho bạn một giao diện kéo và thả đơn giản để bố trí các thành phần như nút, trường văn bản, hộp tổ hợp và hơn thế nữa. Đây là ảnh chụp màn hình của Qt Designer trên Windows:



**2. Qt Designer và Python**

Nhiều người thích sử dụng Qt Designer cùng với Python vì nó là một ngôn ngữ động có lợi cho việc thiết kế nhanh. Cách dễ nhất để kết hợp Qt Designer và Python là thông qua liên kết PyQt. Để cài đặt PyQt, chỉ cần nhập nội dung sau vào dòng lệnh:

python3 -m venv venv

source venv/bin/activate # or "call venv\Scripts\activate.bat" on Windows

python3 -m pip install PyQt5

**3. PyQt5 là gì?**

- Qt là một Application framework đa nền tảng viết trên ngôn ngữ C++, được dùng để phát triển các ứng dụng trên desktop, hệ thống nhúng và mobile. Hỗ trợ cho các platform bao gồm: Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS và một số platform khác. PyQt là Python interface của Qt, kết hợp của ngôn ngữ lập trình Python và thư viện Qt, là một thư viện bao gồm các thành phần giao diện điều khiển (widgets, graphical control elements).

– PyQt API bao gồm các module bao gồm số lượng lớn với các classes và functions hỗ trợ cho việc thiết kế ra các giao diện giao tiếp với người dùng của các phần mềm chức năng. Hỗ trợ với Python 2.x và 3.x.

* ***Các class của PyQt5 được chia thành các module, bao gồm:***

+ QtCore: là module bao gồm phần lõi không thuộc chức năng GUI, ví dụ dùng để làm việc với thời gian, file và thư mục, các loại dữ liệu, streams, URLs, mime type, threads hoặc processes.

+ QtGui: bao gồm các class dùng cho việc lập trình giao diện (windowing system integration), event handling, 2D graphics, basic imaging, fonts và text.

+ QtWidgets: bao gồm các class cho widget, ví dụ: button, hộp thoại, … được sử dụng để tạo nên giao diện người dùng cơ bản nhất.

+ QtMultimedia: thư viện cho việc sử dụng âm thanh, hình ảnh, camera,…

+ QtBluetooth: bao gồm các class giúp tìm kiếm và kết nối với các thiết bị có giao tiếp với phần mềm.

+ QtNetwork: bao gồm các class dùng cho việc lập trình mạng, hỗ trợ lập trình TCP/IP và UDP client, server hỗ trợ việc lập trình mạng.

+ QtPositioning: bao gồm các class giúp việc hỗ trợ xác định vị.

+ Enginio: module giúp các client truy cập các Cloud Services của Qt.

+ QtWebSockets: cung cấp các công cụ cho WebSocket protocol.

+ QtWebKit: cung cấp các class dùng cho làm việc với các trình duyệt Web , dựa trên thư viện WebKit2.

+ QtWebKitWidgets: các widget cho WebKit.

+ QtXml: các class dùng cho làm việc với XML file.

+ QtSvg: dùng cho hiển thị các thành phần của SVG file.

+ QtSql: cung cấp các class dùng cho việc làm việc với dữ liệu.

+ QtTest: cung cấp các công cụ cho phép test các đơn vị của ứng dụng với PyQt5.

* Giả sử bạn đã lưu tệp của mình từ Qt Designer dưới dạng dialog.ui. Sau đó, bạn có thể tạo một tệp khác, chẳng hạn như main.py, với nội dung sau:

**Cách 1:**

from PyQt5 import uic

from PyQt5.QtWidgets import QApplication

Form, Window = uic.loadUiType("dialog.ui")

app = QApplication([])

window = Window()

form = Form()

form.setupUi(window)

window.show()

app.exec()

**Cách 2**

from PyQt5 import QtWidgets, uic

import sys

class Ui(QtWidgets.QMainWindow):

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

super(Ui, self).\_\_init\_\_()

uic.loadUi('dialog.ui', self)

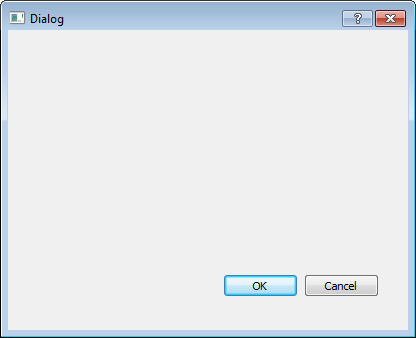
self.show()

app = QtWidgets.QApplication(sys.argv)

window = Ui()

app.exec()

Sau đó, khi bạn gọi python main.py trên dòng lệnh, hộp thoại của bạn sẽ mở ra:



Ngoài ra các bạn cũng có thể chuyển đổi từ tệp .ui (tệp giao diện người dùng) sang tệp .py (tệp Python) bằng câu lệnh trên Terminal như sau:

# To convert .ui file to .py using pyqt5 module

>> pyuic5 -x main.ui -o main.py # pyQt5 version

>> pyuic4 -x main.ui -o main.py # pyQt4 version

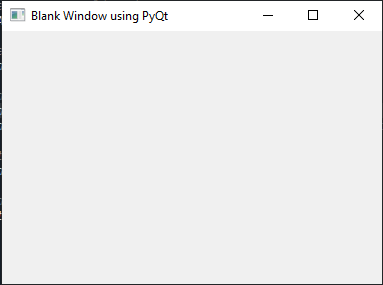
Tải về phần mềm Qt Designer tại đường dẫn sau:

<https://build-system.fman.io/qt-designer-download>

**C. Thực hiện các bài tập**

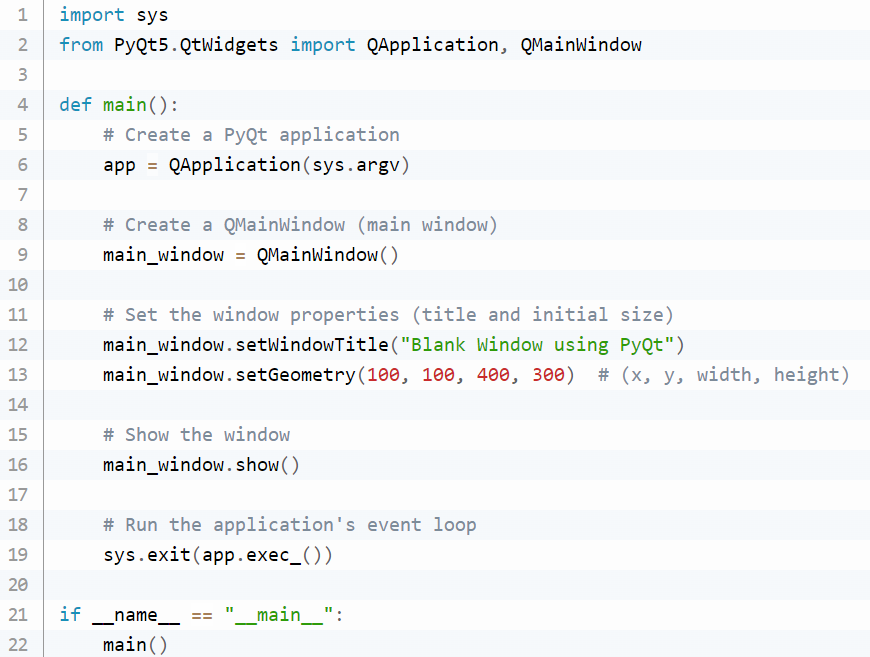
**Bài 1:** Viết chương trình Python để tạo một cửa sổ trống bằng PyQt.

**Kết quả:**

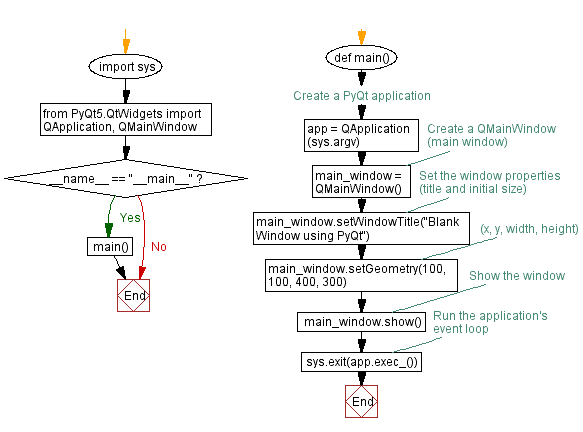


**Hướng dẫn:**

* Nhập các mô-đun cần thiết từ PyQt5.
* Tạo đối tượng 'QApplication' để quản lý cài đặt và luồng điều khiển của ứng dụng.
* Tạo đối tượng 'QMainWindow', đối tượng này đóng vai trò là cửa sổ ứng dụng chính của chúng tôi.
* Đặt tiêu đề và kích thước ban đầu của cửa sổ bằng cách sử dụng các phương thức setWindowTitle và setGeometry.
* Hiển thị cửa sổ bằng "main\_window.show()".
* Cuối cùng, bắt đầu vòng lặp sự kiện của ứng dụng với app.exec() để xử lý các sự kiện và tương tác của người dùng.

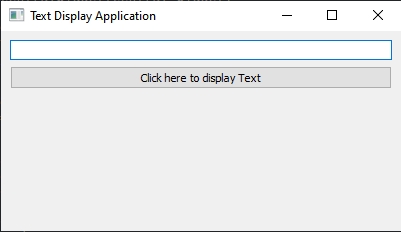
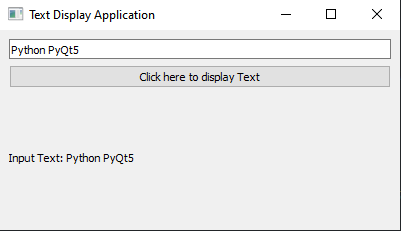


* **Lưu đồ**



**Bài 2:** Viết chương trình Python tạo ứng dụng PyQt với tiện ích QLineEdit và QPushButton. Khi nhấp vào nút, nó sẽ hiển thị văn bản đã nhập trong QlineEdit.

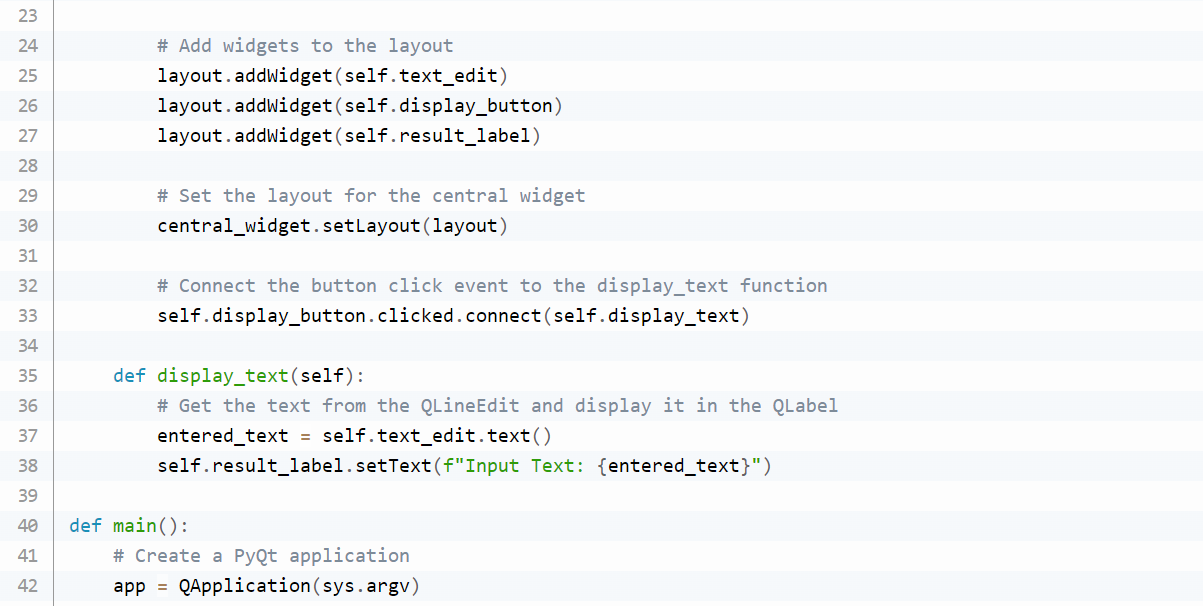
Kết quả:

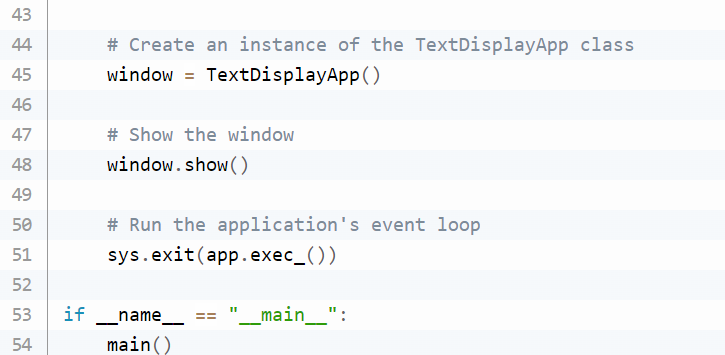
 🡪 

**Hướng dẫn:**

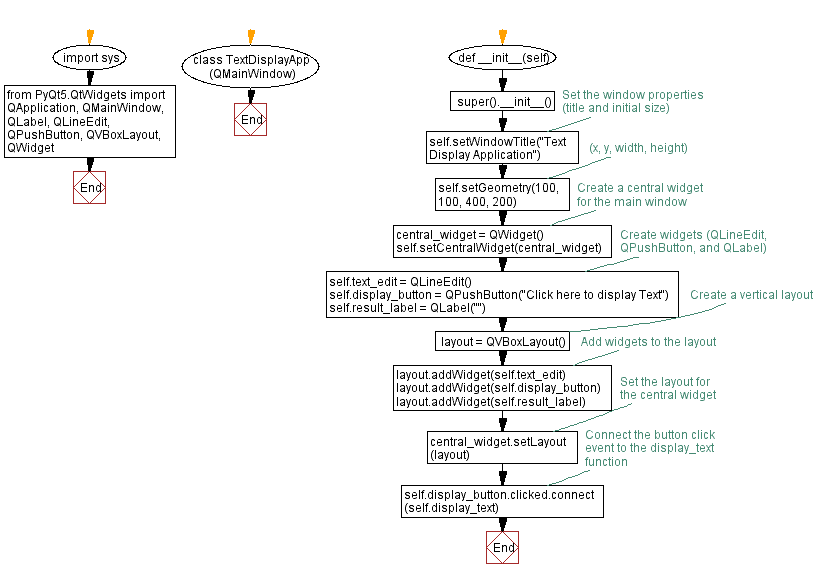
* Nhập các mô-đun cần thiết.
* Tạo lớp "TextDisplayApp" tùy chỉnh kế thừa từ "QMainWindow".
* Bên trong lớp, chúng tôi tạo cửa sổ và các tiện ích (QLineEdit, QPushButton và QLabel) cũng như đặt thuộc tính và bố cục của chúng.
* Kết nối sự kiện nhấp chuột của nút với phương thức display\_text để truy xuất văn bản từ 'QLineEdit' và hiển thị nó trong 'QLabel'.
* Tạo một phiên bản của 'TextDisplayApp', hiển thị cửa sổ và bắt đầu vòng lặp sự kiện của ứng dụng trong chức năng chính

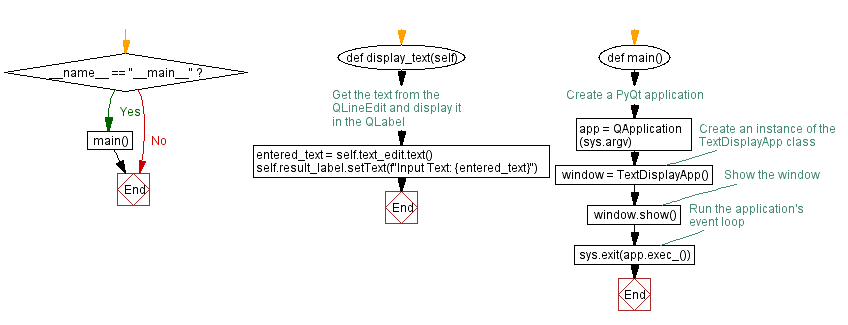






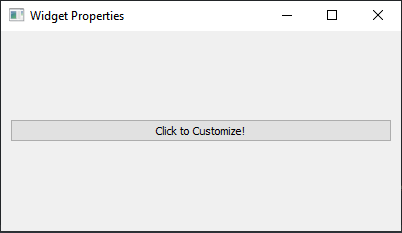
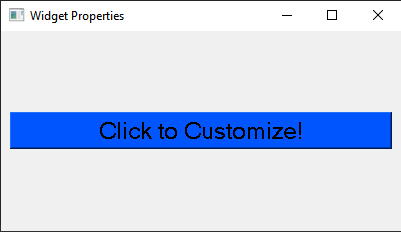
* **Lưu đồ**





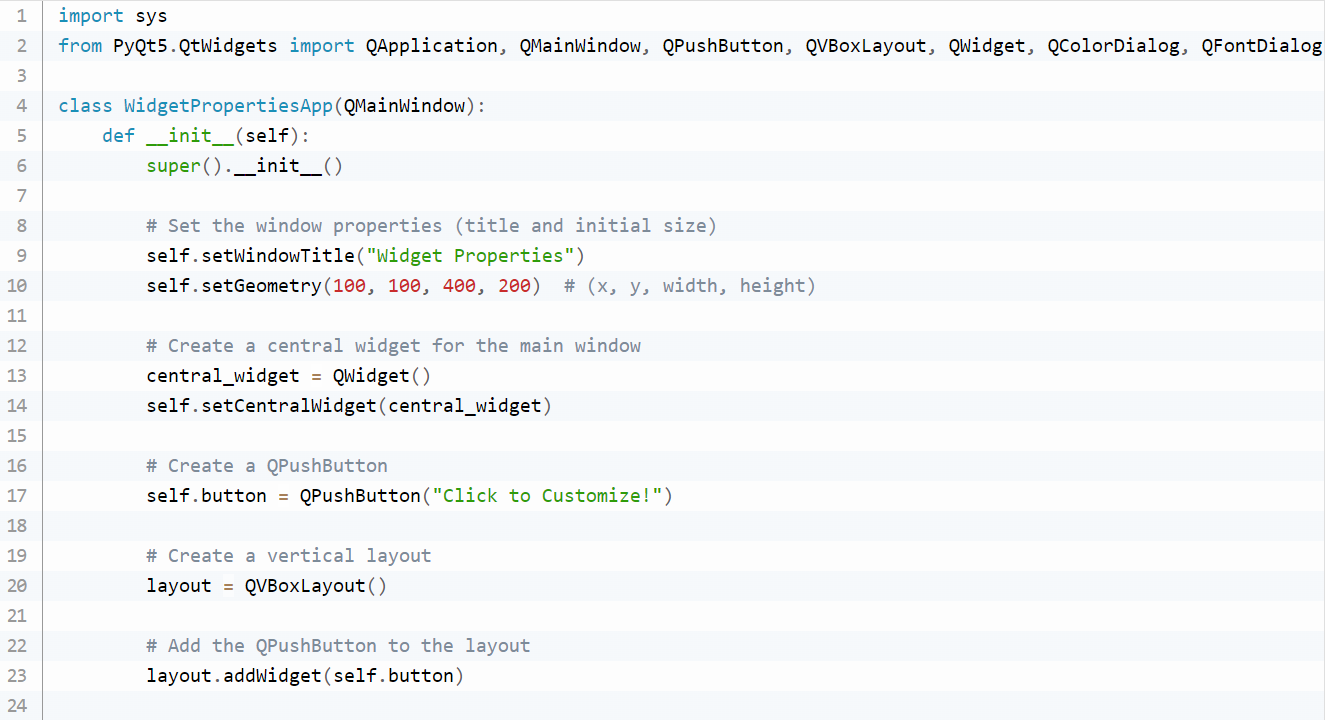
**Bài 3:** Viết chương trình Python bằng PyQt5 yêu cầu người dùng thay đổi các thuộc tính của tiện ích, chẳng hạn như màu sắc, phông chữ hoặc kích thước. Quan sát xem nó ảnh hưởng như thế nào đến diện mạo của tiện ích.

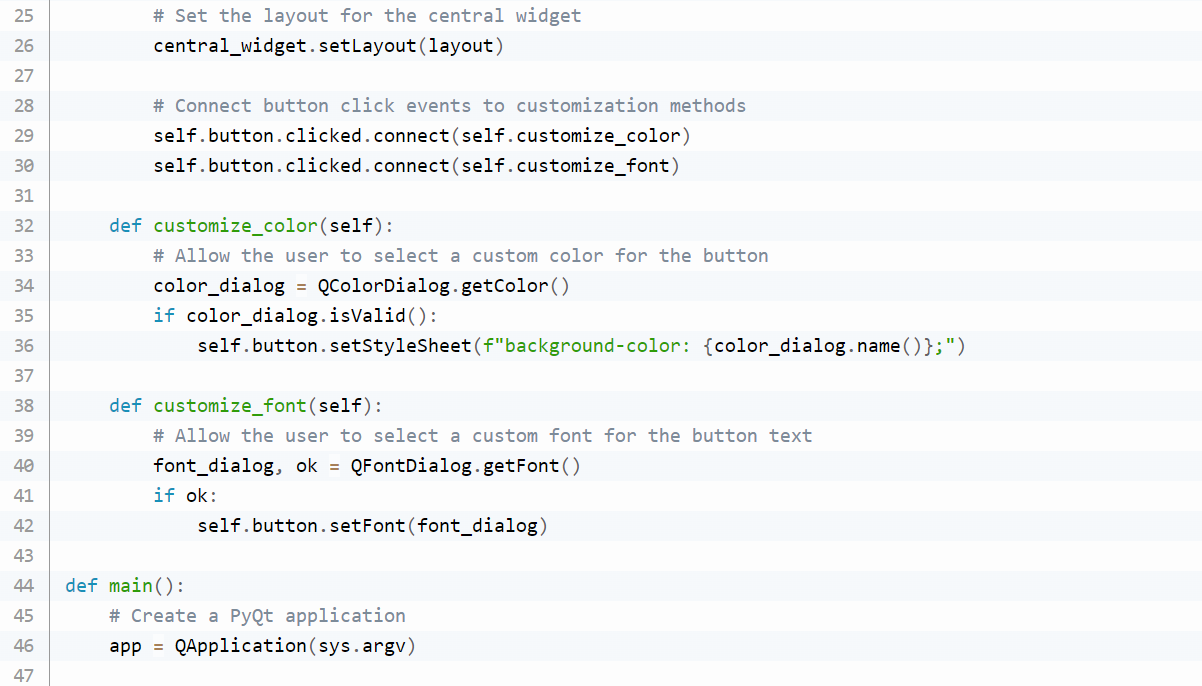
**Kết quả:**

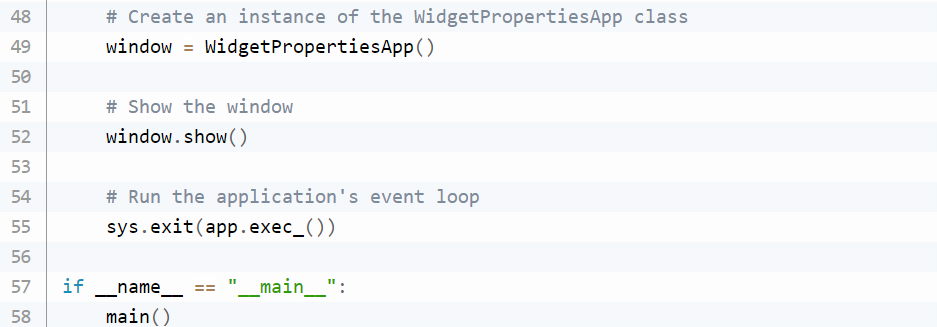
 🡪 

Hướng dẫn:

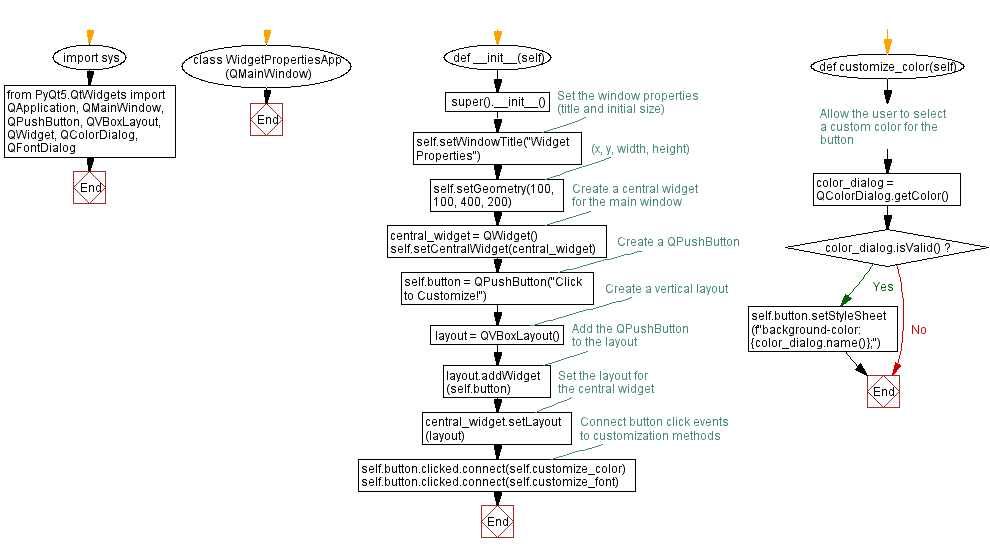
* Nhập các mô-đun cần thiết.
* Tạo lớp "WidgetPropertiesApp" tùy chỉnh kế thừa từ 'QMainWindow'.
* Bên trong lớp, chúng tôi tạo cửa sổ chính và 'QPushButton' mà người dùng có thể tùy chỉnh.
* Tạo các phương thức tùy chỉnh "customize\_color()" và "customize\_font()" cho phép người dùng chọn màu sắc và phông chữ tùy chỉnh cho nút.
* Kết nối các sự kiện nhấp chuột của nút với các phương pháp tùy chỉnh này bằng cách sử dụng 'clicked.connect'.
* Hàm chính tạo một phiên bản của WidgetPropertiesApp, hiển thị cửa sổ và bắt đầu vòng lặp sự kiện của ứng dụng

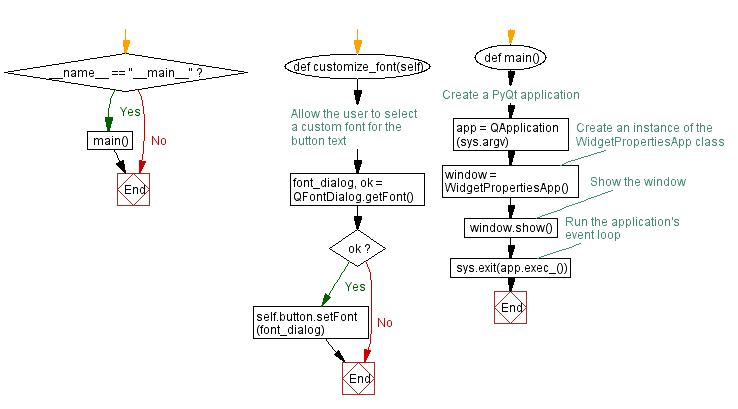






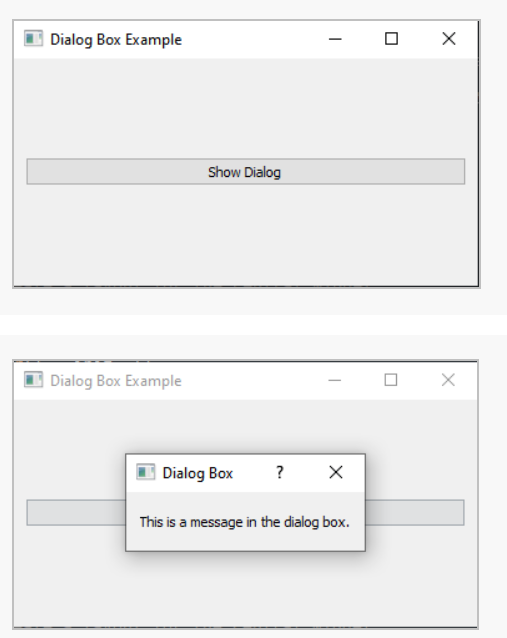
* **Lưu đồ**





**Bài 4:** Viết chương trình Python tạo một hộp thoại đơn giản hiển thị thông báo hoặc biểu mẫu nhập khi nhấp vào nút bằng PyQt.

**Kết quả:**



**Hướng dẫn:**

Nhập các mô-đun cần thiết.

Tạo "QMainWindow" (MyWindow) bằng tiện ích trung tâm.

Bên trong tiện ích trung tâm, chúng tôi tạo một 'QPushButton' (dialog\_button) có nhãn "Hiển thị hộp thoại" và kết nối tín hiệu được nhấp vào của nó với phương thức show\_dialog.

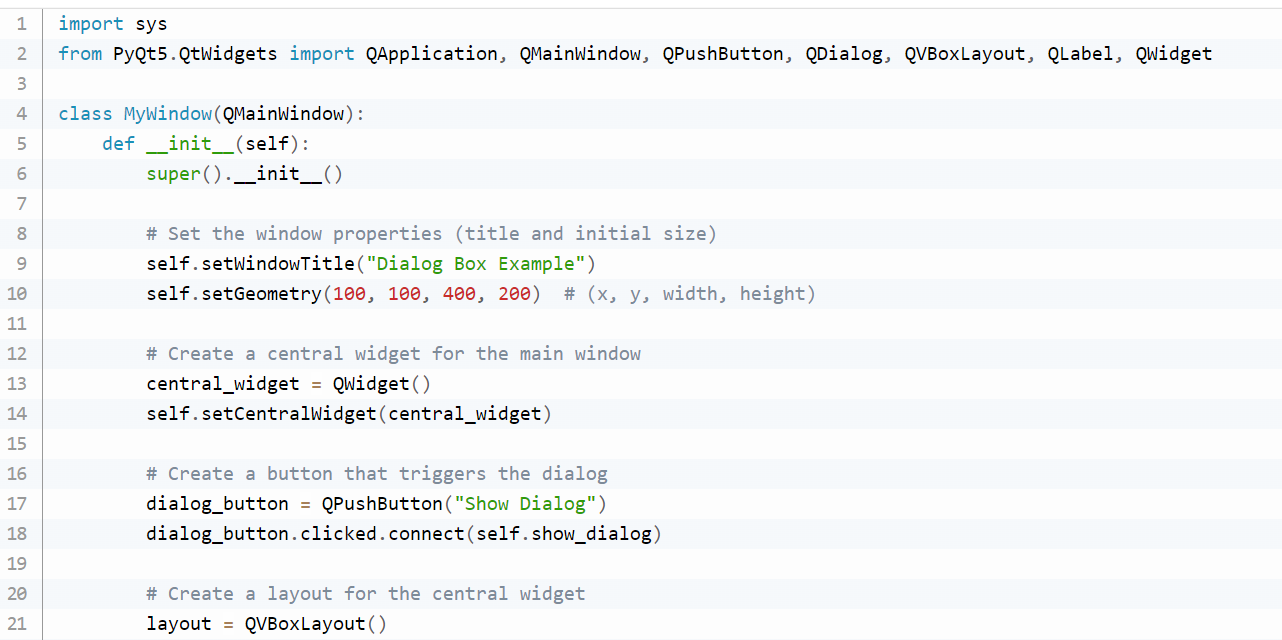
Trong phương thức "show\_dialog()", chúng tôi tạo một phiên bản QDialog (hộp thoại) và đặt tiêu đề cho nó.

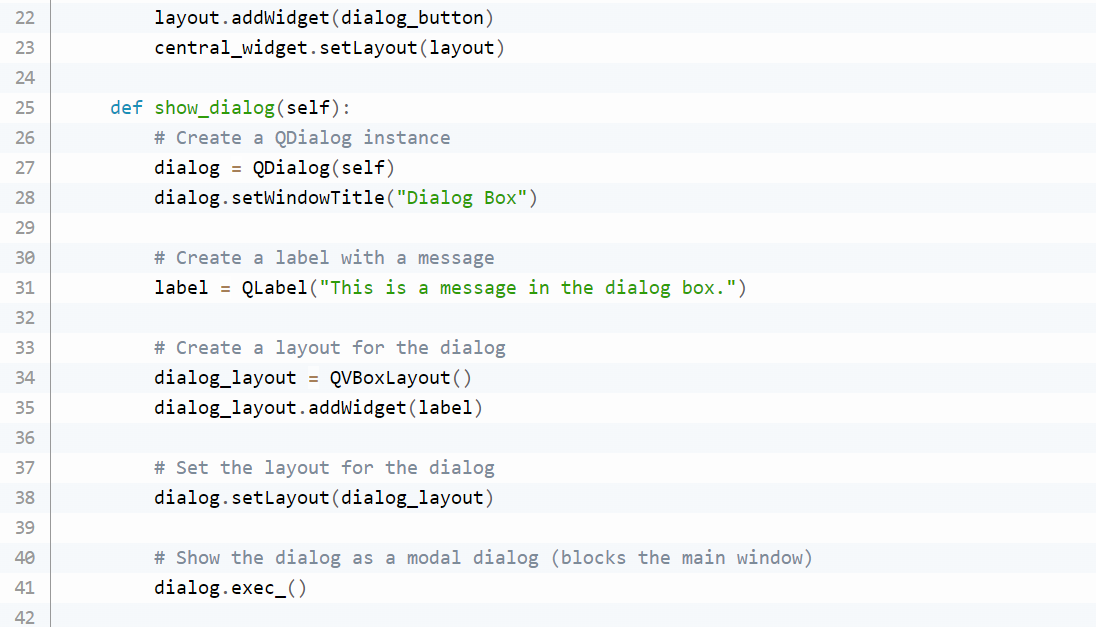
Tạo một "QLabel" (nhãn) với thông báo chúng tôi muốn hiển thị trong hộp thoại.

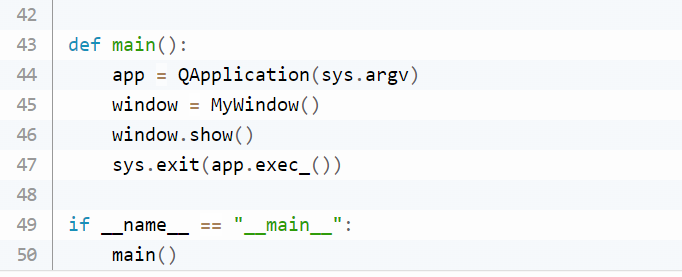
Tạo QVBoxLayout cho hộp thoại và thêm nhãn vào đó.

Đặt bố cục hộp thoại bằng cách sử dụng "dialog.setLayout".

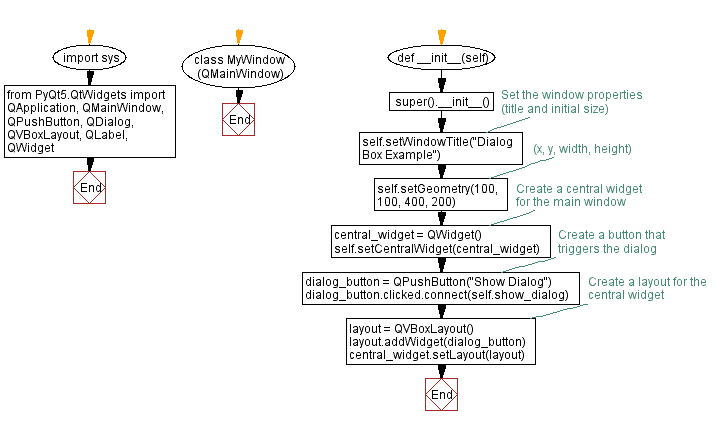
Khi nhấp vào nút "Hiển thị hộp thoại", phương thức show\_dialog sẽ được gọi, hiển thị hộp thoại dưới dạng hộp thoại phương thức sử dụng “dialog.exec()”. Để tương tác lại với cửa sổ chính, hộp thoại phải được đóng lại.

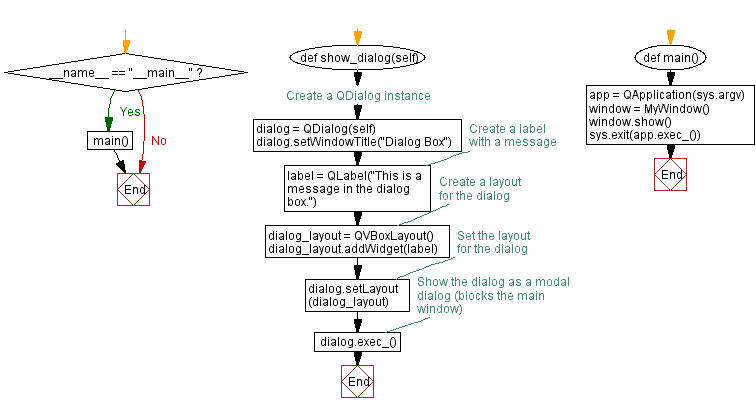






* **Lưu đồ**





-------------Hết------------