

# Formation Python

Interface graphique : Qt Designer + PyQt5

**Formateur**

**Yassine Ben Salah**



# Formation Python

## Interface graphique en Python

Qt Designer + PyQt5



# Interface graphique

## Qt Designer + PyQt5

### Présentation de l'interface graphique (GUI)

Les **interfaces graphiques** (ou interfaces homme-machine) sont appelées **GUI** (Graphical User Interface).

Elles permettent à l'utilisateur d'**interagir** avec un **programme informatique**, grâce aux différents **objets graphiques** (zone de texte, case à cocher, bouton radio, bouton poussoir, menu, ...).

Ces **objets graphiques** sont généralement **actionnés** avec un dispositif de pointage, le plus souvent la souris.

# Interface graphique

## Qt Designer + PyQt5

La création d'une **GUI** peut se réaliser par **deux méthodes** :

- Ecrire **le code** qui lui correspond en utilisant un langage de programmation.
- Créer les **objets graphiques** (zone de texte, case à cocher, bouton radio, bouton poussoir, menu, ...) directement sur **une fenêtre** en utilisant un **outil de conception et de création d'interfaces graphiques** (un éditeur graphique tel que **Qt Designer** ou **Qt Creator**) qui contient un éditeur d'interface utilisateur **glisser et déposer (drag and drop)**



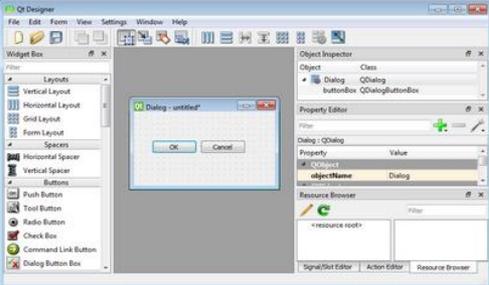
# Interface graphique Qt Designer + PyQt5

## Environnement

1<sup>ère</sup> étape : télécharger et installer le logiciel Qt designer :

Lien de téléchargement

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



### Qt Designer Download

Install Qt Designer on Windows or Mac.  
Tiny download: Only 40MB!



Many people want to use Qt Designer without having to download gigabytes of other software. Here are small, standalone installers of Qt Designer for Windows and Mac:

 Windows (31 MB)

 Mac (40 MB)



# Interface graphique

## Qt Designer + PyQt5

**2<sup>ème</sup> étape : télécharger et installer Python :**  
**lien de téléchargement**

**<https://www.python.org/downloads/>**

**Choisir la version compatible avec votre système 32-bits ou bien 64-bits**

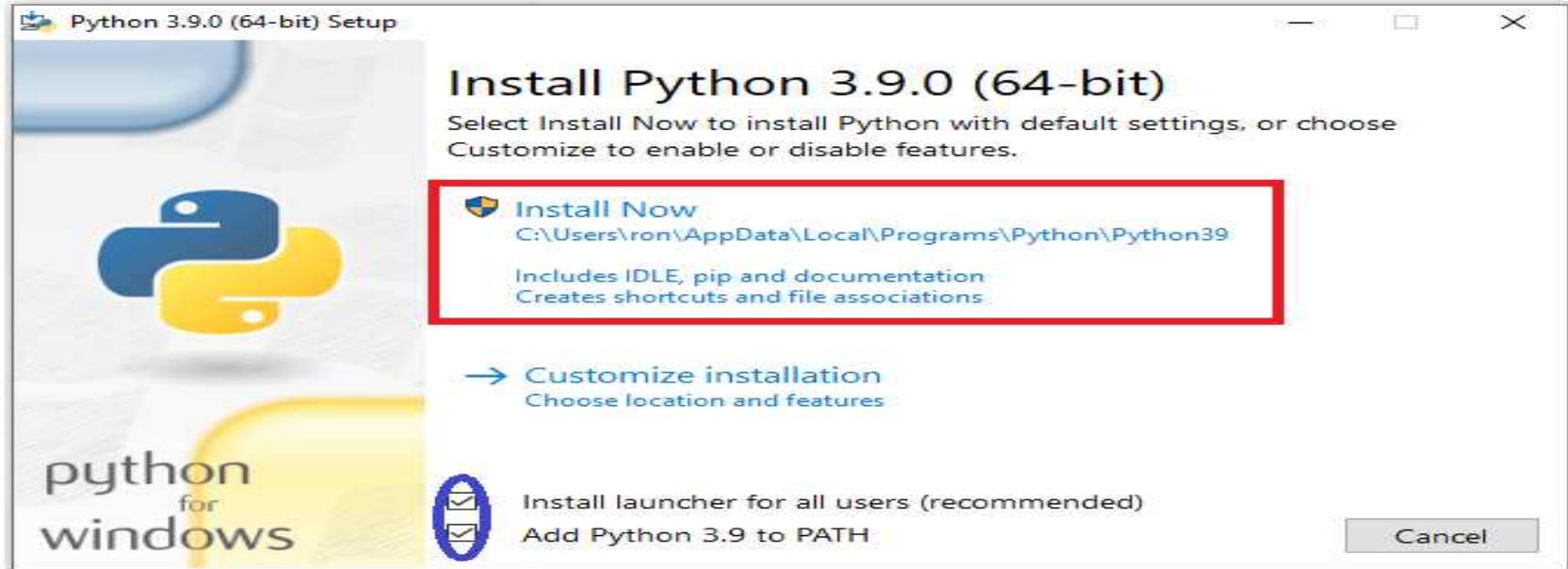
The screenshot shows the Python.org website with a dark blue header and navigation menu. The main content area features a yellow button labeled "Download Python 3.9.0" under the heading "Download the latest version for Windows". Below this, there are links for "Windows", "Linux/UNIX", "Mac OS X", and "Other". There is also a link for "Prereleases" and "Docker images". The background of the main content area features an illustration of two yellow and white striped parachutes with cardboard boxes hanging from them, set against a blue sky with clouds.



# Interface graphique

## Qt Designer + PyQt5

### Installer ensuite python



#### N.B. :

- Cocher le lanceur d'installation pour tous les utilisateurs
- Cocher aussi **Ajouter Python 3.9 à PATH** (Ajouter le chemin Python aux **variables d'environnement**) pour placer l'interpréteur dans le chemin d'exécution et pouvoir utiliser les commandes de python n'importe où dans votre système (même avec cmd), et dans n'importe quel dossier



# Interface graphique

## Qt Designer + PyQt5

### N.B. :

- Une fois que vous avez fini d'installer Python sur windows, on peut accéder au répertoire dans lequel Python a été installé sur le système.

`C:\Users\Username\AppData\Local\Programs\Python\PythonVersion`

Dans notre cas, il s'agit de :

`C:\Users\LENOVO\AppData\Local\Programs\Python\Python39`

- Pour toutes les versions récentes de Python, les options d'installation recommandées incluent **IDLE** (Integreted DeveLopement Environement) et **Pip** (système de gestion de packages)



# Interface graphique

## Qt Designer + PyQt5

**N.B.** : On peut Vérifier l'installation de Python avec l'invite de commande (Windows)

- Ouvrir **l'invite de commande** (cliquer sur la fenêtre windows et taper cmd)
- Taper la commande **python --version**
- Valider avec le bouton **Entrée**

```
C:\> Invite de commandes

Microsoft Windows [version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\LENOVO>python --version
Python 3.9.0

C:\Users\LENOVO>
```



# Interface graphique

## Qt Designer + PyQt5

### 3<sup>ème</sup> étape : Installer les packages PyQt5 et pyqt5-tools

On peut installer le package **PyQt5** avec l'invite de commande (Windows) :

- Ouvrir l'invite de commande (cliquer sur la fenêtre windows et taper **cmd**)
- Taper la commande **pip3 install PyQt5**

```
C:\> Invite de commandes
Microsoft Windows [version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\LENOVO>pip3 install PyQt5_
```



# Interface graphique

## Qt Designer + PyQt5

Après avoir installé la bibliothèque **PyQt5**, on doit installer le package **pyqt5-tools** (les outils auxiliaires) à l'aide de la commande **pip3 install pyqt5-tools**

```
Microsoft Windows [version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Tous droits réservés.

C:\Users\LENOVO>pip3 install pyqt5-tools
Collecting pyqt5-tools
  Downloading pyqt5_tools-5.15.2.3.0.2-py3-none-any.whl (28 kB)
Collecting pyqt5-plugins<5.15.2.3,>=5.15.2.2
  Downloading pyqt5_plugins-5.15.2.2.0.1-cp39-cp39-win_amd64.whl (65 kB)
  | ██████████ | 10 kB 640 kB/s eta 0:
```



# Interface graphique

## Qt Designer + PyQt5

**N.B.** : On peut aussi utiliser un éditeur Python (par exemple **Thonny**), on doit installer les packages **PyQt5**, **pyqt5-tools** ou bien **PyQt5Designer**.

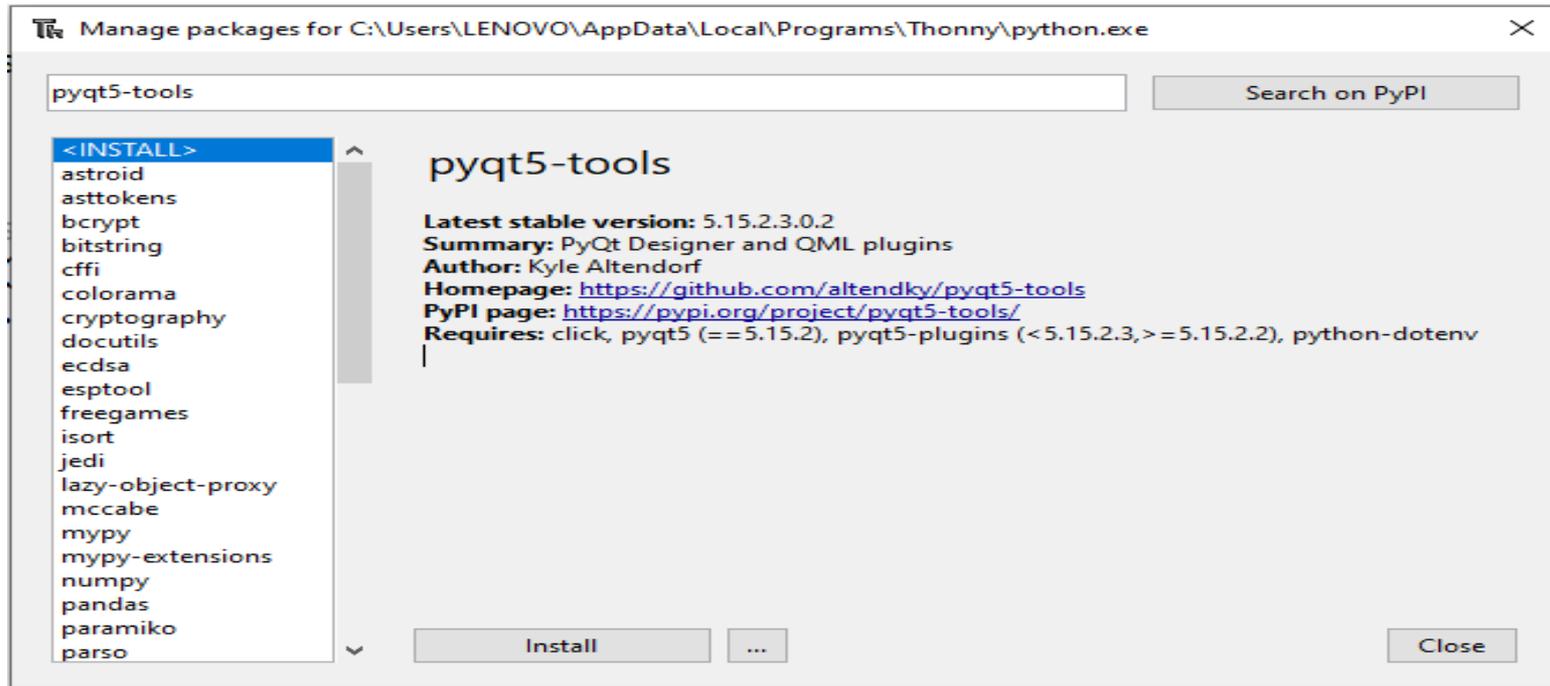
Pour ajouter un package avec **Thonny** en ligne :

- Ouvrir **thonny** / **Menu tools** / **Manage packages**
- Taper le **nom du package** à installer (**PyQt5**, **pyqt5-tools** ou bien **PyQt5Designer**)
- Cliquer sur **install**



# Interface graphique Qt Designer + PyQt5

## Exemple : installation du package pyqt5-tools



**N.B.** : Après l'installation, les outils se trouvent dans le dossier suivant :

**C:\Users\...\AppData\Roaming\Python\Python37\site-packages**



# Interface graphique

## Qt Designer + PyQt5

### Présentation de la bibliothèque PyQt

**PyQt** est une **bibliothèque** qui permet de créer des interfaces graphiques avec Python

Le module **PyQt** est considéré comme une **liaison** du langage Python avec la **boîte à outils GUI toolkit Qt**

**PyQt** prend en charge Microsoft Windows ainsi que diverses versions d'UNIX, et MacOS

**PyQt** implémente environ 440 classes et plus de 6000 fonctions et méthodes.

**PyQt6** est la version utilisée actuellement par Python.



# Interface graphique

## Qt Designer + PyQt5

### Présentation du module PyQt5

Dans cette formation, on va s'intéresser à la version **PyQt5**

**PyQt5** offre trois méthodes de création d'interface graphique :

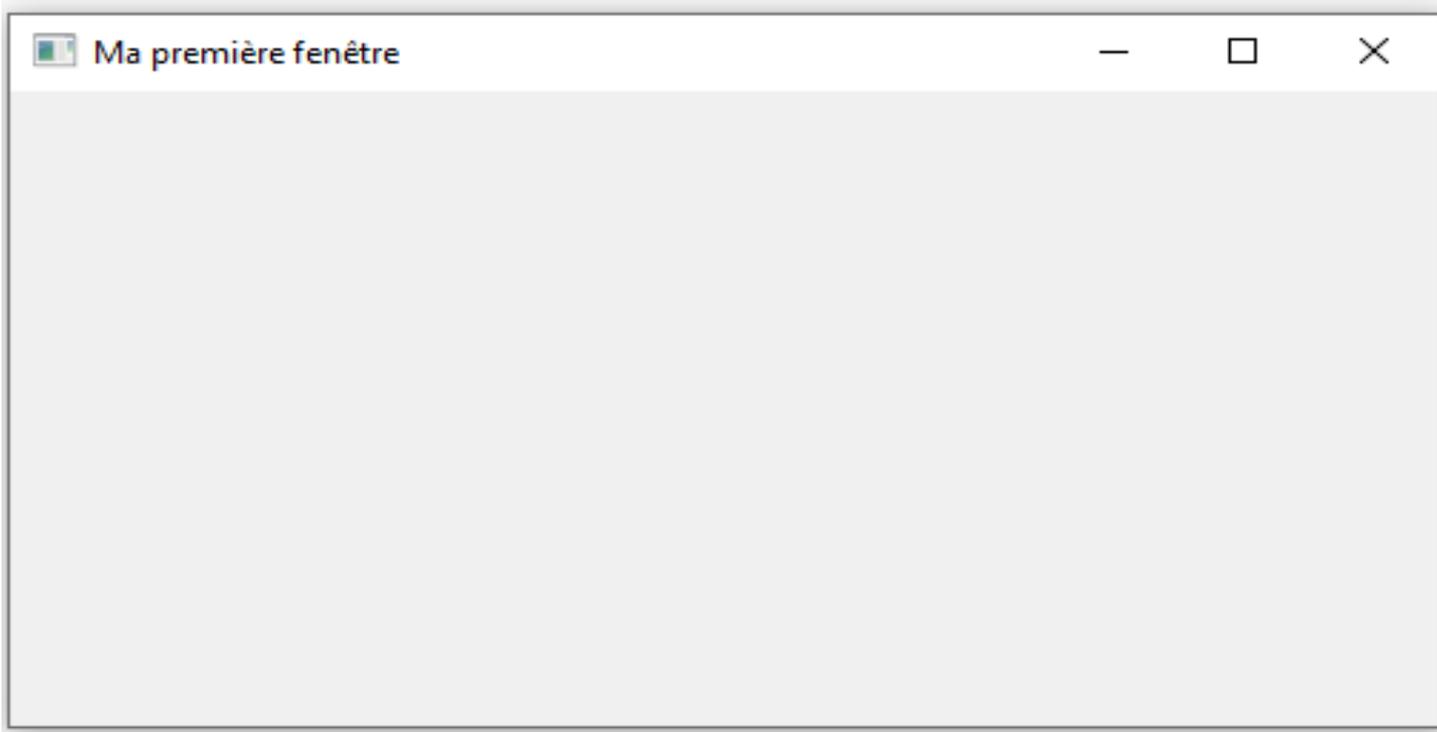
- L'interface est programmé puis exécuté (**fichier.py**)
- L'interface peut être créé par un outil de création d'interface graphique GUI (**fichier.ui**) puis converti en un code Python (**fichier.py**) pour qu'il soit exécuté.
- L'interface peut être créé par un outil de création d'interface graphique GUI (**fichier.ui**) puis appelé ce fichier dans le code python (**fichier.py**). Cette méthode est l'avantage majeur de ce module



# Interface graphique

## Qt Designer + PyQt5

**Création d'une fenêtre avec un code en utilisant la bibliothèque PyQt5 du langage Python**





# Interface graphique

## Qt Designer + PyQt5

Pour créer une **fenêtre graphique en PyQt5**, on doit :

- Importer le module système : **sys**
- Importer la classe qui génère l'application : **QApplications** depuis le package **PyQt5.QtWidgets**
- Importer la classe **QWidget** depuis le package **PyQt5.QtWidgets**
- Création d'une instance de Qapplication avec **QApplication(sys.argv)**
- Créer une fenêtre avec la méthode **QWidget()**:  
`fen = QWidget()`
- Visualiser la fenêtre à l'aide de la méthode **show()** :  
`fen.show()`
- Exécuter l'application à l'aide de la méthode **exec\_()** :  
`app.exec_()`



# Interface graphique

## Qt Designer + PyQt5

```
# Création d'une fenêtre avec PyQt5  
# importations nécessaire à la réalisation d'une interface graphique  
import sys  
from PyQt5.QtWidgets import QApplication, QWidget  
# Création d'une instance de QApplication  
app = QApplication(sys.argv)  
# On crée une fenêtre à l'aide de l'objet QWidget  
fen = QWidget()  
# On donne un titre à la fenêtre  
fen.setWindowTitle("Ma première fenêtre")  
# On fixe la taille de la fenêtre  
fen.resize(500,250)  
# On déplace le widget (fenêtre fen) à une position sur l'écran  
fen.move(300, 200)  
# On visualise la fenêtre  
fen.show()  
# Exécution de l'application  
app.exec_()
```



# Interface graphique

## Qt Designer + PyQt5

### Exécution





# Interface graphique

## Qt Designer + PyQt5

**N.B.** : Si on va créer des grandes applications où les interfaces sont plus compliquées, le travail peut devenir un peu lourd et pénible si on va créer ces interfaces par écriture d'un code (par programmation).

Pour cela il vaut mieux utiliser un outil de conception et de création d'interfaces graphiques GUI pour faciliter le travail.



# Interface graphique

## Qt Designer + PyQt5

### **N.B. :**

Dans cette formation on va s'intéresser à la troisième méthode de création d'interface graphique avec **PyQt5** c'est-à-dire, on va créer l'interface par l'outil de création d'interface graphique **GUI (Qt Designer)** puis on va appeler le fichier (**Nom\_fichier.ui**) dans le **code python (Nom\_fichier.py)** crée par l'éditeur Python (**Thonny**)



# Interface graphique

## Qt Designer + PyQt5

### Présentation de Qt Designer

**Qt Designer** est un **générateur d'interface graphique**, elle permet de créer vos **fenêtres visuellement**, elle vous permet de créer **des objets graphiques (widgets)** et modifier ses **propriétés** (Nom, Taille, couleur, ...), d'utiliser des **layouts**, etc..

Il permet d'effectuer **la connexion** entre **signaux** et **slots**. Le mécanisme **signal/slot** est la base de la **programmation événementielle** des applications basées sur **Qt**.

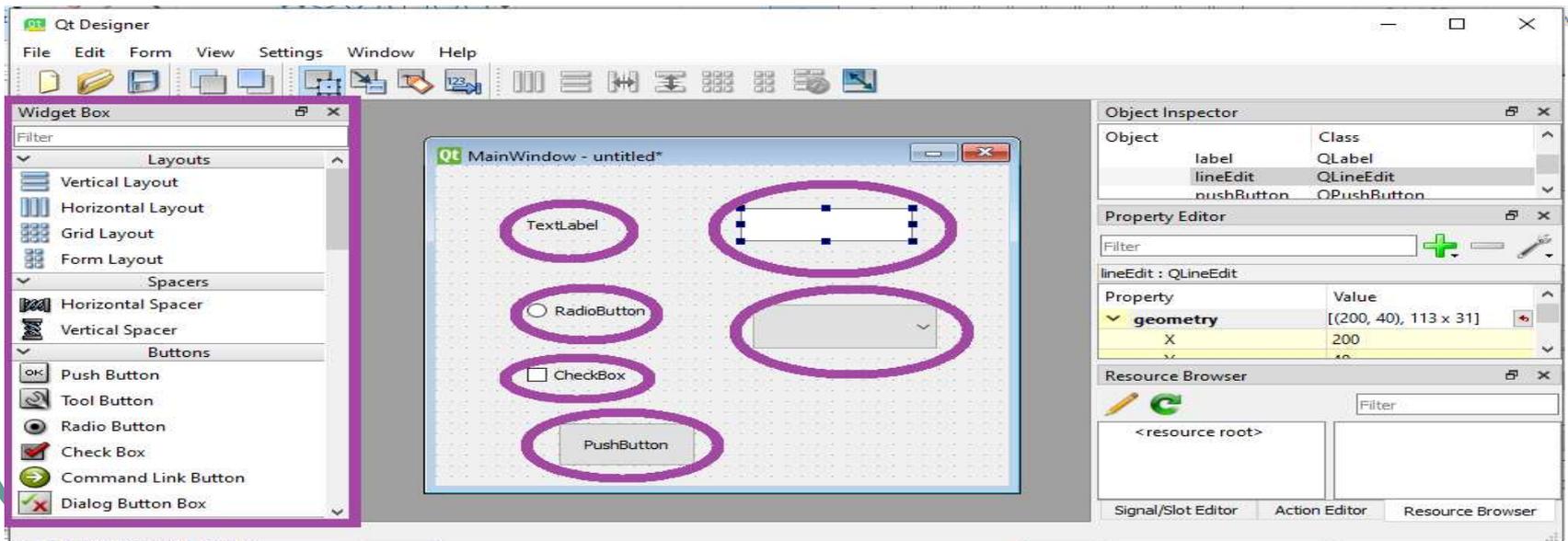


# Interface graphique

## Qt Designer + PyQt5

### Notion de Widget

Le **Widget** est un composant d'interface graphique (Label, zone de texte, bouton radio, case à cocher, bouton poussoir, menu, liste déroulante, barre de défilement, ...)





# Interface graphique

## Qt Designer + PyQt5

### La programmation événementielle

La **programmation événementielle** est une programmation basée sur les événements.

Les **modules (Fonctions)** définies par un **code**, seront appelés et exécutés suite à un événement déclenché par l'utilisateur.

#### Exemples d'évènements :

- Le click sur un bouton
- La modification du contenu d'une zone
- l'incrémentement d'une liste
- Le passage du pointeur de la souris sur une zone.
- Etc.



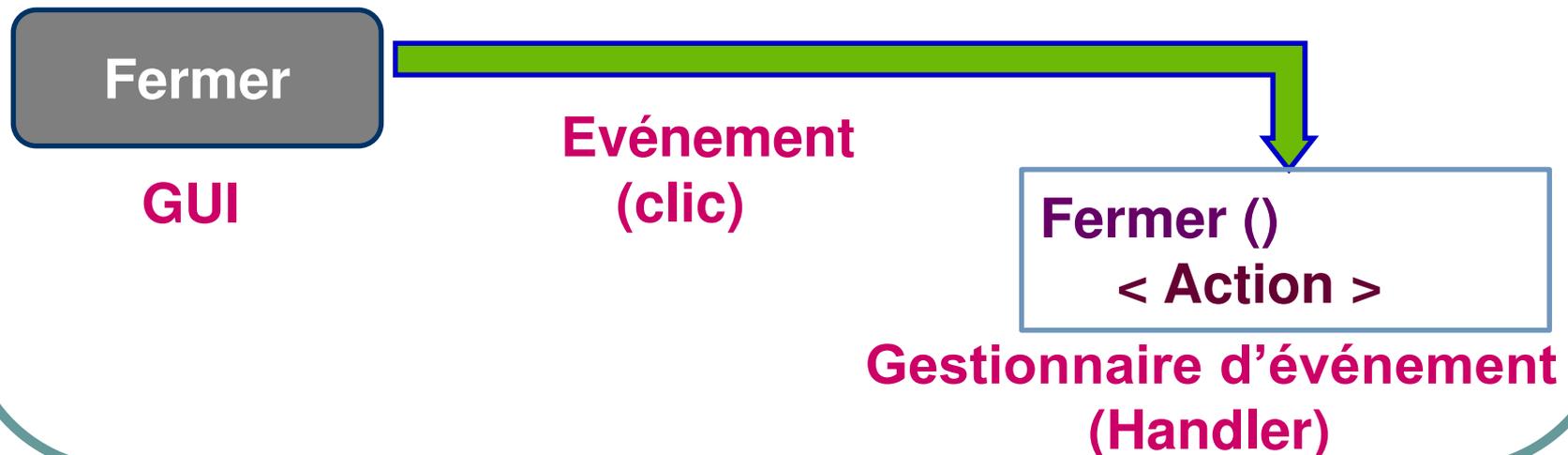
# Interface graphique

## Qt Designer + PyQt5

### Le principe de la programmation événementielle

Pour chaque **évènement** à gérer, il faut lui associer une **action** à réaliser : c'est le **gestionnaire d'évènement**

Ensuite, à chaque fois que l'évènement sera détecté par la boucle d'évènement, le **gestionnaire d'évènement** sera alors **exécuté**.





# Interface graphique

## Qt Designer + PyQt5

### La programmation événementielle des applications Qt

La **programmation événementielle** des applications Qt est basée sur un mécanisme appelé **signal / slot** :

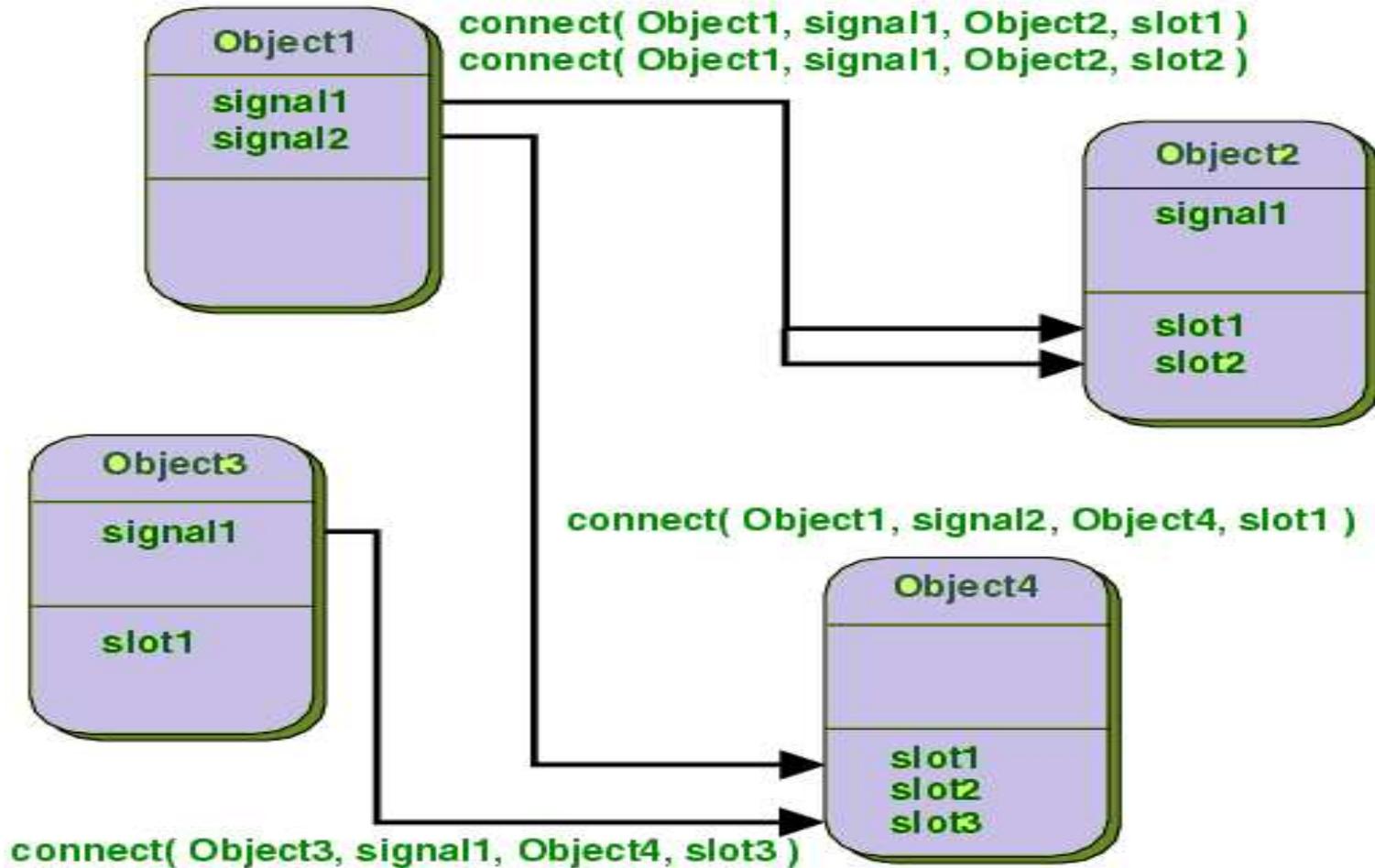
- Un **signal** est l'événement réalisé sur un **Widget** (un clic, un appui de touche, ...)
- Un **slot** est une **fonction** qui va être définie et appelée en réponse à un **signal** particulier

L'association d'un **signal** à un **slot** est réalisée par une **connexion** avec l'appel **connect (nom\_fonction)**



# Interface graphique

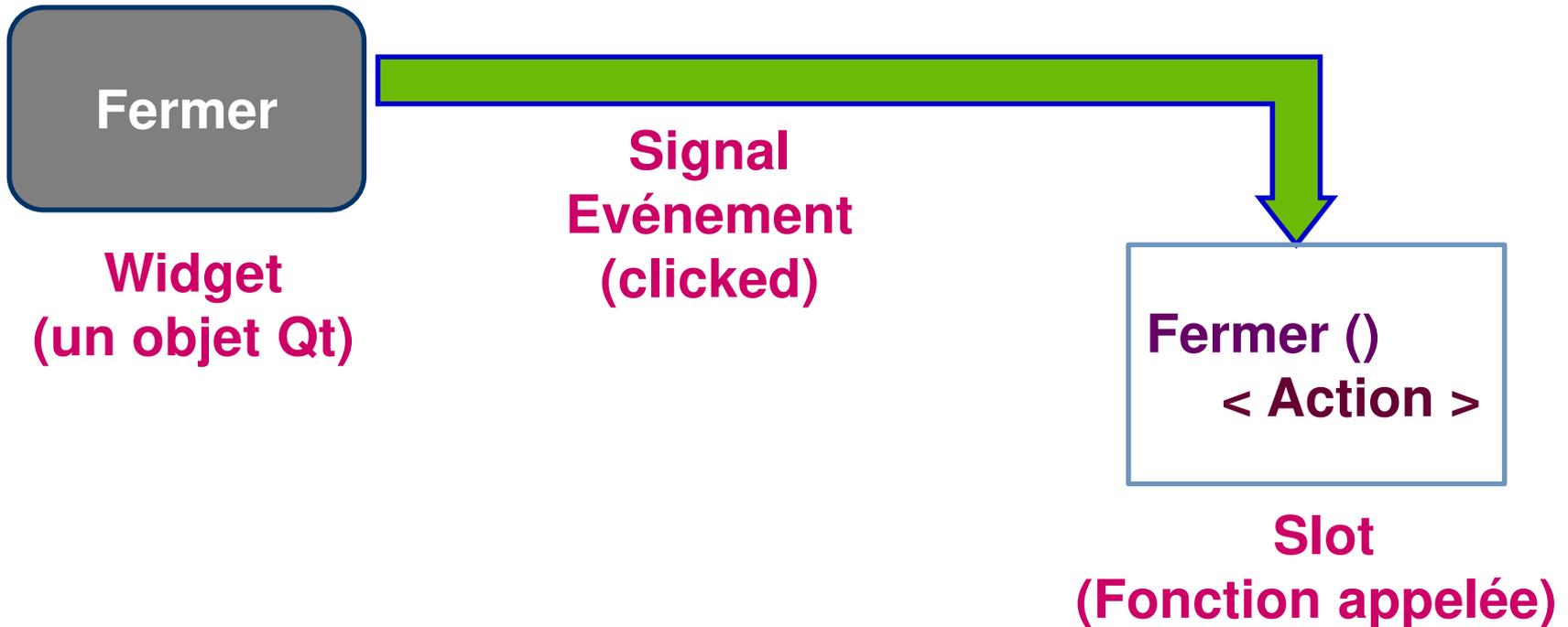
## Qt Designer + PyQt5





# Interface graphique

## Qt Designer + PyQt5





# Interface graphique

## Qt Designer + PyQt5

### Présentation de l'interface de Qt Designer

La **fenêtre principale** de **Qt Designer** fournit une **barre de menus (menu bar)**

La fenêtre principale fournit également une **barre d'outils (toolbar)** qui affiche les options couramment utilisées.

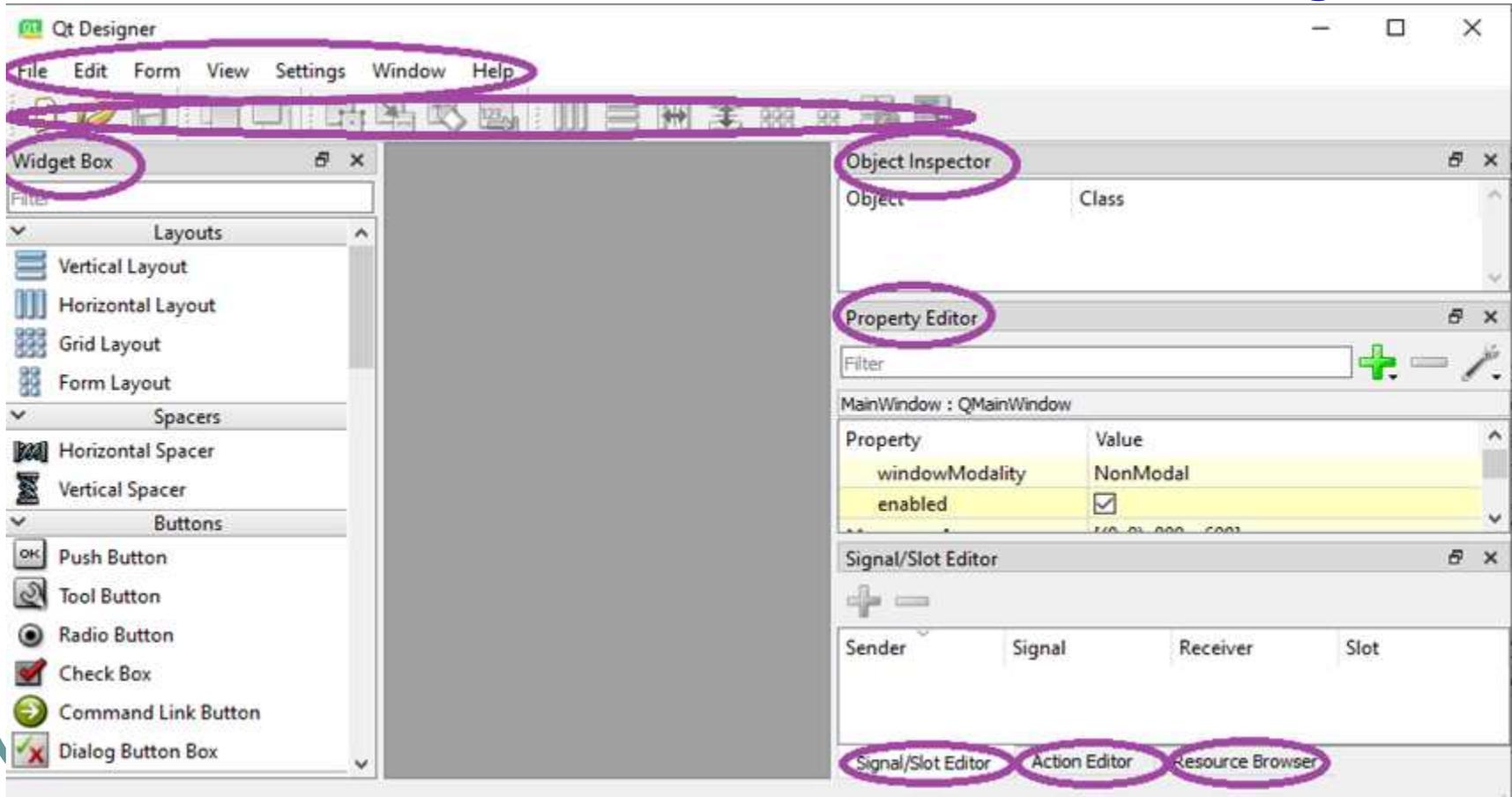
Elle comprend également d'autres **composantes** qui fournissent un ensemble de fonctionnalités et d'outils :

- **Widget Box** : Boîte de widgets
- **Object Inspector** : Inspecteur d'objets
- **Property Editor** : Éditeur de propriétés
- **Resource Browser** : Navigateur de ressources
- **Action Editor** : Éditeur d'actions
- **Signal/Slot Editor** : Éditeur de signal / slot



# Interface graphique Qt Designer + PyQt5

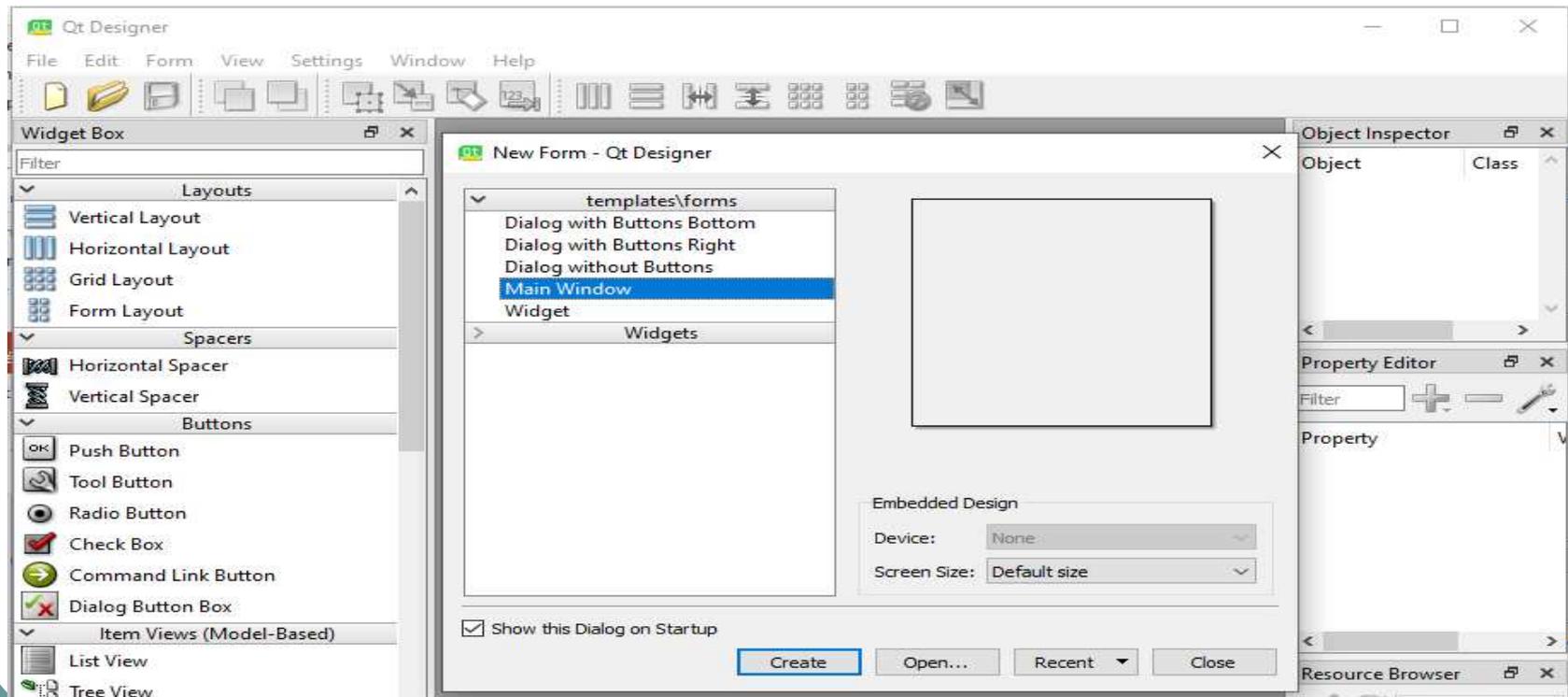
## Présentation de l'interface de Qt Designer





# Interface graphique Qt Designer + PyQt5

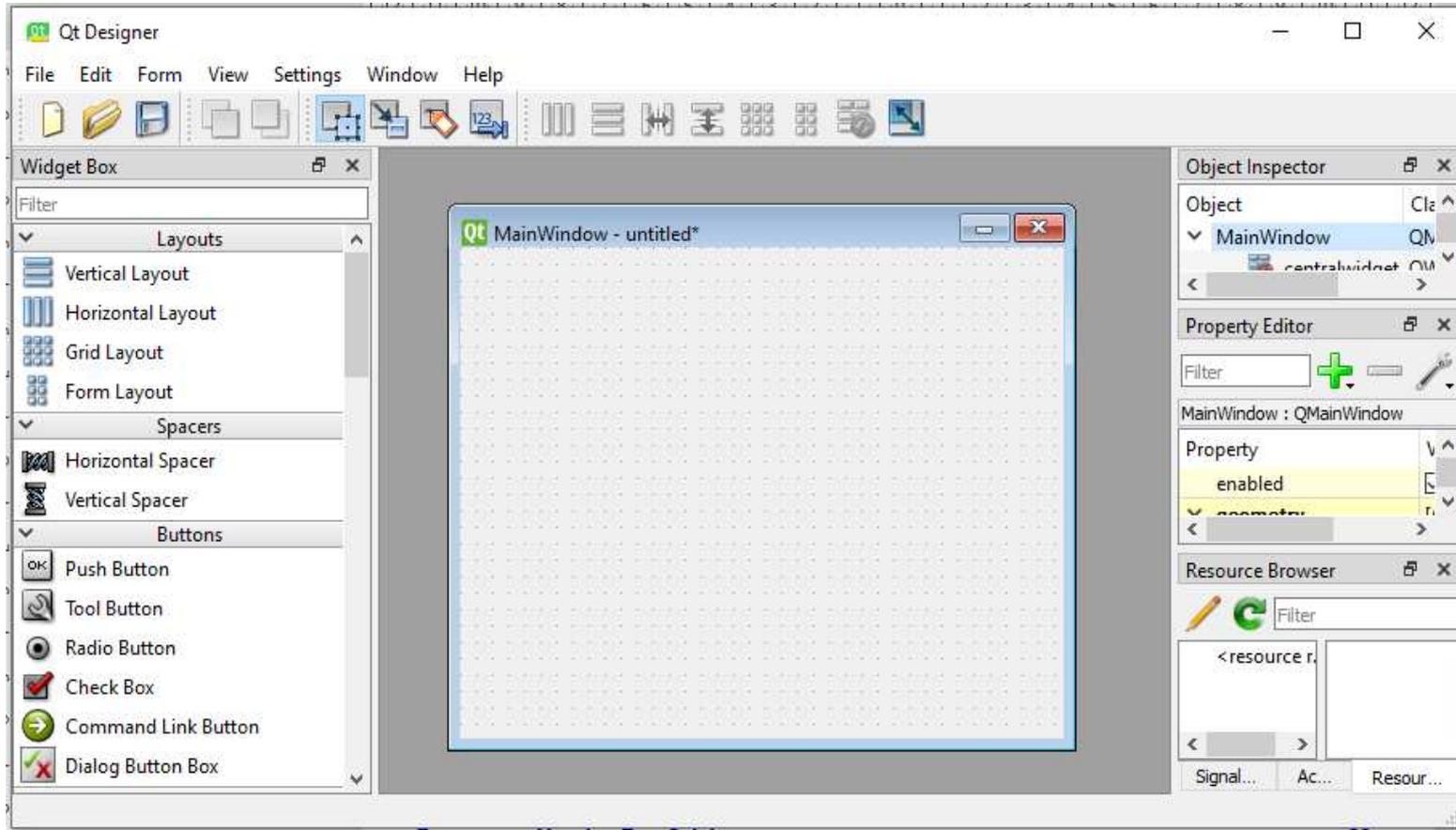
**Création d'une fenêtre principale avec Qt Designer**  
**Pour créer une **fenêtre principale** pour une application :**  
**Menu File / New / Main Window / Create**





# Interface graphique Qt Designer + PyQt5

## Création d'une fenêtre principale avec Qt Designer





# Interface graphique

## Qt Designer + PyQt5

### Modification des propriétés du fenêtre principale avec Qt Designer

On peut modifier les propriétés du fenêtre principale à partir de l'éditeur de propriétés (Property Editor) comme exemple :

- Changer le nom de l'objet (Window)
- Ajouter un titre pour la fenêtre
- Changer la taille du fenêtre (largeur et hauteur)
- Changer le font et la couleur
- Ajouter une icône pour la fenêtre
- Etc ...



# Interface graphique Qt Designer + PyQt5

## Modification des propriétés du fenêtre principale avec Qt Designer

The screenshot displays the Qt Designer application window. The central canvas shows a blank window titled "Qt MainWindow - untitled\*". On the left, the "Widget Box" contains various UI components like layouts, spacers, and buttons. On the right, the "Object Inspector" shows the hierarchy: MainWindow (QMainWindow) containing a centralwidget (QWidget). The "Property Editor" is highlighted with a purple box and shows the following properties for the selected object:

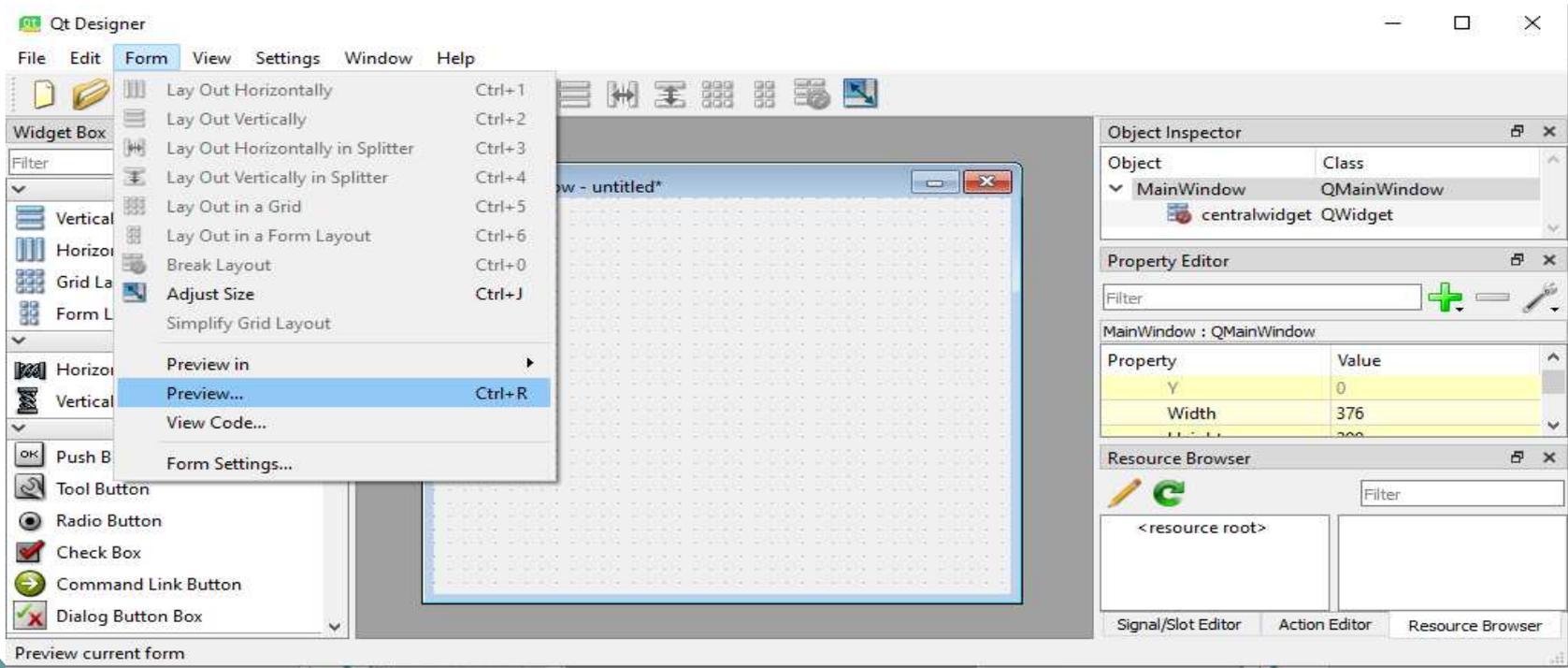
Property	Value
MainWindow : QMainWindow	
▼ QObject	
objectName	MainWindow
▼ QWidget	
enabled	<input checked="" type="checkbox"/>
▼ geometry	
X	0
Y	0

Below the Property Editor is the "Resource Browser" showing the resource root.



# Interface graphique Qt Designer + PyQt5

**Prévisualisation du fenêtre avec Qt Designer**  
**Pour pré-visualiser une fenêtre pour une application :**  
**Menu Form / Preview**





# Interface graphique

## Qt Designer + PyQt5

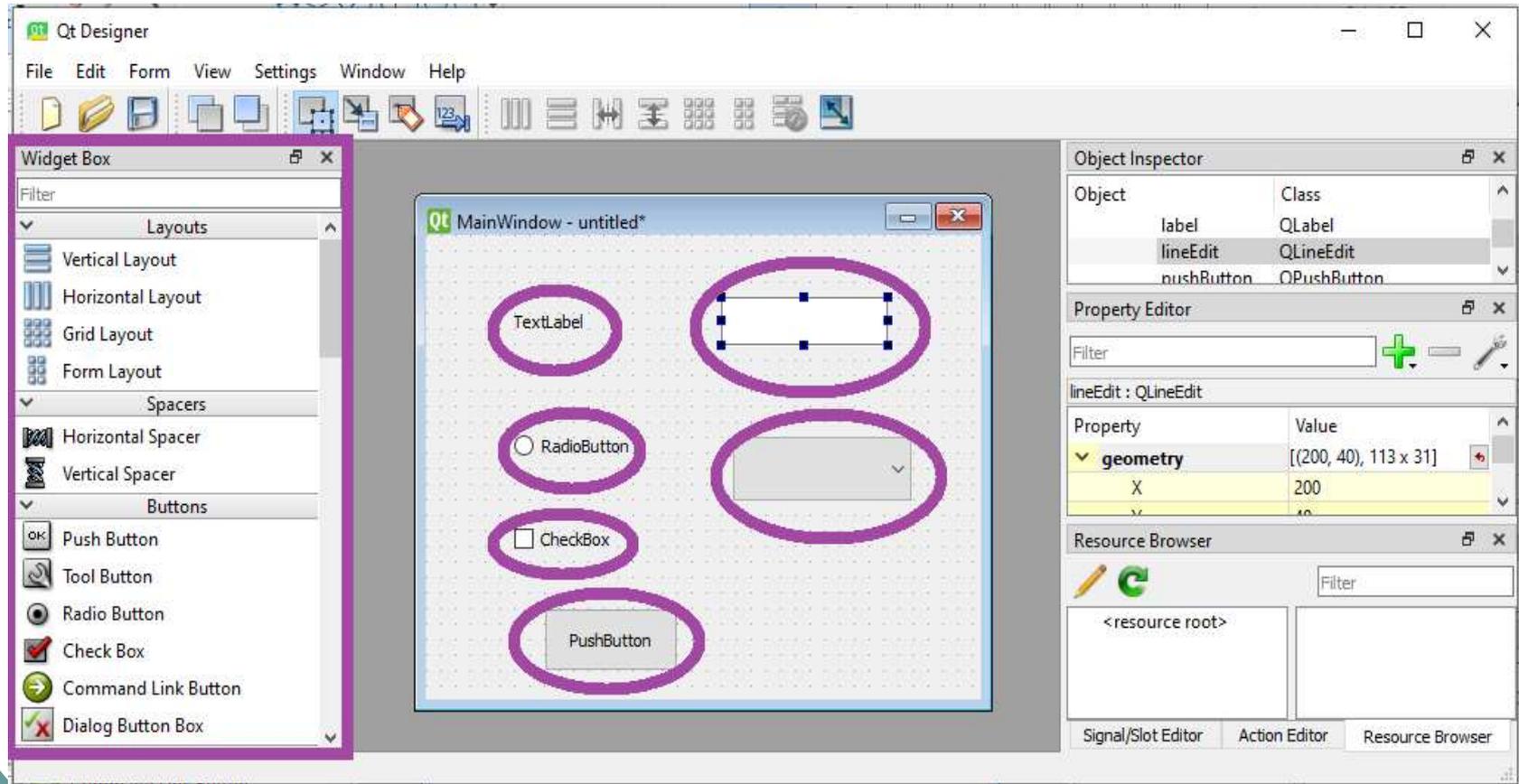
### Ajout des Widgets dans la fenêtre avec Qt Designer

On peut ajouter **des Widgets** (Label, Line Edit, Push Button, Radio Button, Check Box, Combo Box, List Widget, Table Widget, etc...) directement sur **une fenêtre** avec le mécanisme **glisser et déposer** (drag and drop) à partir de la **boîte de widgets** (**Widget Box**)



# Interface graphique Qt Designer + PyQt5

## Ajout des Widgets dans la fenêtre avec Qt Designer





# Interface graphique

## Qt Designer + PyQt5

### Modification des propriétés des widgets avec Qt Designer

On peut modifier **les propriétés** des **widgets** à partir de l'éditeur de propriétés (Property Editor) comme exemple :

- Changer le **nom de l'objet** (widget)
- Changer le **texte de l'objet**
- Changer la **taille de l'objet** (largeur et hauteur)
- Changer **le font, la couleur, ...**
- Etc ...



# Interface graphique Qt Designer + PyQt5

## Modification des propriétés des widgets avec Qt Designer

The screenshot displays the Qt Designer application window. The main canvas shows a window titled "Qt MainWindow - untitled\*" with a grid background. A purple oval highlights a widget labeled "TextLabel". To the left is the "Widget Box" with categories like Layouts, Spacers, Buttons, and Item Views. To the right is the "Object Inspector" showing a list of objects: comboBox (QComboBox), label (QLabel), lineEdit (QLineEdit), and pushButton (QPushButton). Below it is the "Property Editor" for the selected "label : QLabel" widget, showing properties like "objectName" (value: "label"), "enabled" (checked), and "geometry" (value: "[(60, 40), 101 x 31]"). At the bottom right is the "Resource Browser".



# Interface graphique

## Qt Designer + PyQt5

### Signaux et slots avec Qt Designer

Les **signaux** et les **slots** sont utilisés pour la communication entre **objets (Widgets)**. Ce mécanisme **signaux/slots** est une fonctionnalité centrale de **Qt**

Un **signal** est émis lorsqu'un **événement particulier se produit**

Un **slot** est une **fonction** qui va être **appelée en réponse à un signal particulier**



# Interface graphique

## Qt Designer + PyQt5

### Signal / Slot Editor (Éditeur de signal / slot)

Avec **Qt Designer** on peut attribuer un **Signal/Slot** à un widget (*Exemple* : Bouton poussoir, ...) en utilisant l'éditeur de signal/slot (**Signal / Slot Editor**)

Par exemple, si l'utilisateur clique sur le bouton de fermeture, la **fonction de fermeture de la fenêtre** (`close()`) sera appelée



# Interface graphique Qt Designer + PyQt5

## Signal / Slot Editor

The screenshot displays the Qt Designer interface for a window titled "Facture - Interface\_graphique\_1.ui". The central canvas shows a form with three input fields labeled "Prénom :", "Nom :", and "Montant :", and two buttons labeled "Envoyer" and "Fermer". The "Fermer" button is highlighted with a red border. To the left is the "Widget Box" containing various Qt widgets. To the right is the "Object Inspector" showing the object hierarchy: MainWindow (QMainWindow) -> centralwidget (QWidget) -> BFermer (QPushButton). Below it is the "Property Editor" for the selected BFermer widget, showing properties like enabled, geometry, X, Y, Width, and Height. At the bottom right is the "Signal/Slot Editor" showing a connection: BFermer clicked() to MainWindow close().



# Interface graphique

## Qt Designer + PyQt5

### Bouton Edit Signals / Slots

Avec **Qt Designer** on peut aussi attribuer un **Signal/Slot** à un **widget** en utilisant le bouton **Edit signals/slots** de la barre d'outils (**toolbar**)

**Exemple** : Attribuer un **Signal/Slot** à un **widget** (bouton de **fermeture d'une fenêtre**) en utilisant le bouton **Edit signals/slots**



# Interface graphique Qt Designer + PyQt5

The screenshot shows the Qt Designer interface with a window titled "Facture - Interface\_graphique\_1.ui". The window contains three labels: "Prénom :", "Nom :", and "Montant :", and a button labeled "Envoyer". A red box labeled "Fermer" is connected to the "Envoyer" button. A "Configure Connection - Qt Designer" dialog is open, showing the connection between the "clicked()" signal of the "BFermer (QPushButton)" widget and the "close()" slot of the "MainWindow (QMainWindow)" widget. The "Show signals and slots inherited from QWidget" checkbox is checked. The "OK" button is highlighted.

**Widget Box**

- Filter
- Line Edit
- Text Edit
- Plain Text Edit
- Spin Box
- Double Spin Box
- Time Edit
- Date Edit
- Date/Time Edit
- Dial
- Horizontal Scroll Bar
- Vertical Scroll Bar
- Horizontal Slider
- Vertical Slider
- Key Sequence Edit
- Display Widgets
- Label
- Text Browser
- Graphics View
- Calendar Widget

**Edit Signals / Slots**

**MainWindow (QMainWindow)**

- close()
- deleteLater()
- hide()
- lower()
- raise()
- repaint()
- setFocus()
- show()
- showFullScreen()
- showMaximized()
- showMinimized()
- showNormal()
- update()

**BFermer (QPushButton)**

- clicked()
- clicked(bool)
- customContextMenuRequested(QPoint)
- destroyed()
- destroyed(QObject\*)
- objectNameChanged(QString)
- pressed()
- released()
- toggled(bool)
- windowIconChanged(QIcon)
- windowIconTextChanged(QString)
- windowTitleChanged(QString)

**Signal/Slot Editor**

Sender	Signal	Receiver



# Interface graphique Qt Designer + PyQt5

**Widget Box**

- Font Combo Box
- Line Edit
- Text Edit
- Plain Text Edit
- Spin Box
- Double Spin Box
- Time Edit
- Date Edit
- Date/Time Edit
- Dial
- Horizontal Scroll Bar
- Vertical Scroll Bar
- Horizontal Slider
- Vertical Slider
- Key Sequence Edit
- Display Widgets
- Label
- Text Browser
- Graphics View
- Calendar Widget
- LCD Number

**Object Inspector**

Object	Class
MainWindow	QMainWindow
centralwidget	QWidget
BFerre	QPushButton
Bppcm	QPushButton
Res	QLineEdit

**Property Editor**

Filter

MainWindow : QMainWindow

Property	Value
QObject	
objectName	MainWindow
QWidget	
enabled	<input checked="" type="checkbox"/>
geometry	[(0, 0), 500 x 400]
X	0
Y	0
Width	500
Height	400

**Signal/Slot Editor**

Sender	Signal	Receiver	Slot
BFerre	clicked()	MainWindow	close()



# Interface graphique

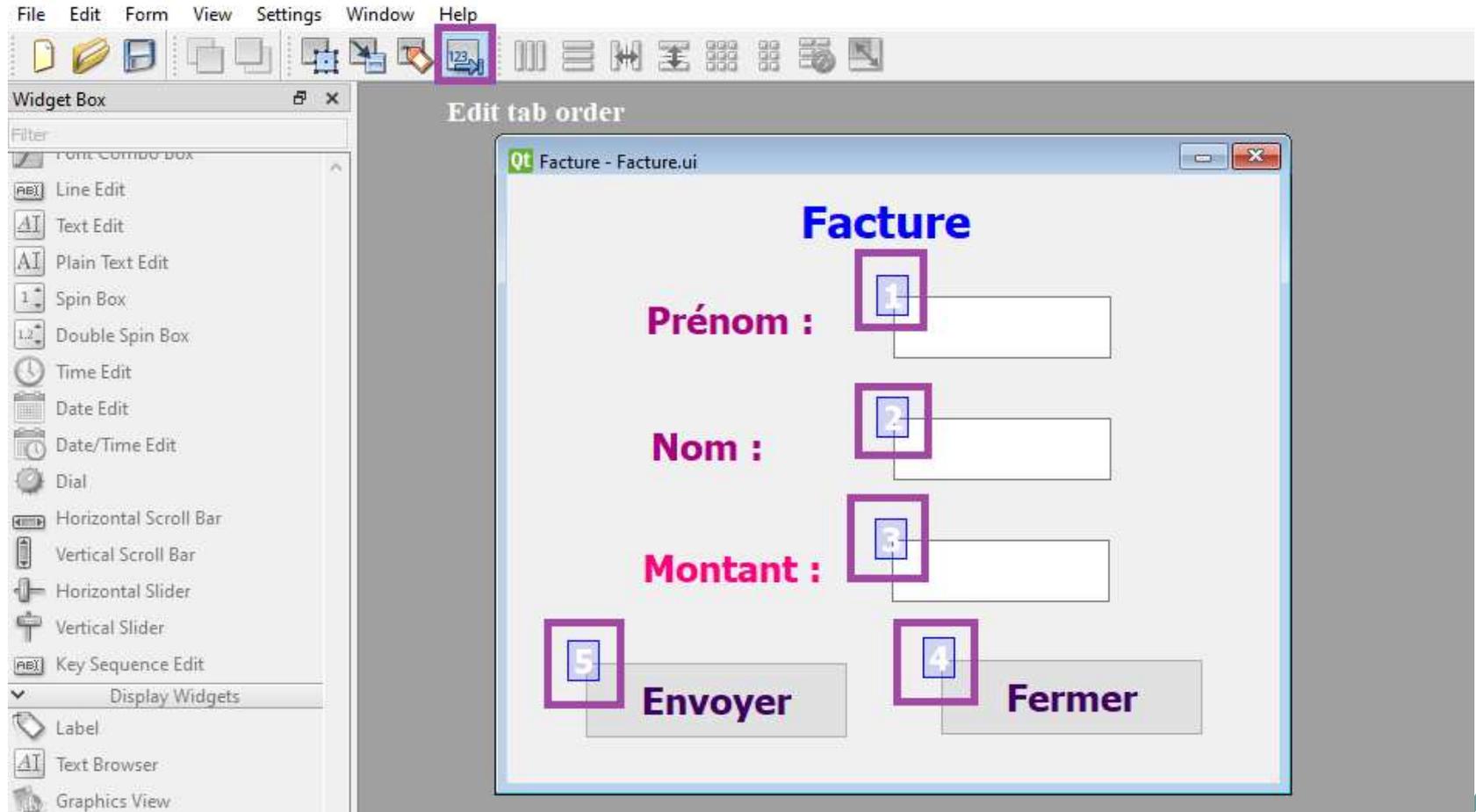
## Qt Designer + PyQt5

### Bouton Edit tab order

Avec **Qt Designer** on peut changer l'ordre des tabulations des widgets en utilisant le bouton **Edit tab ordre** de la barre d'outils (**toolbar**)

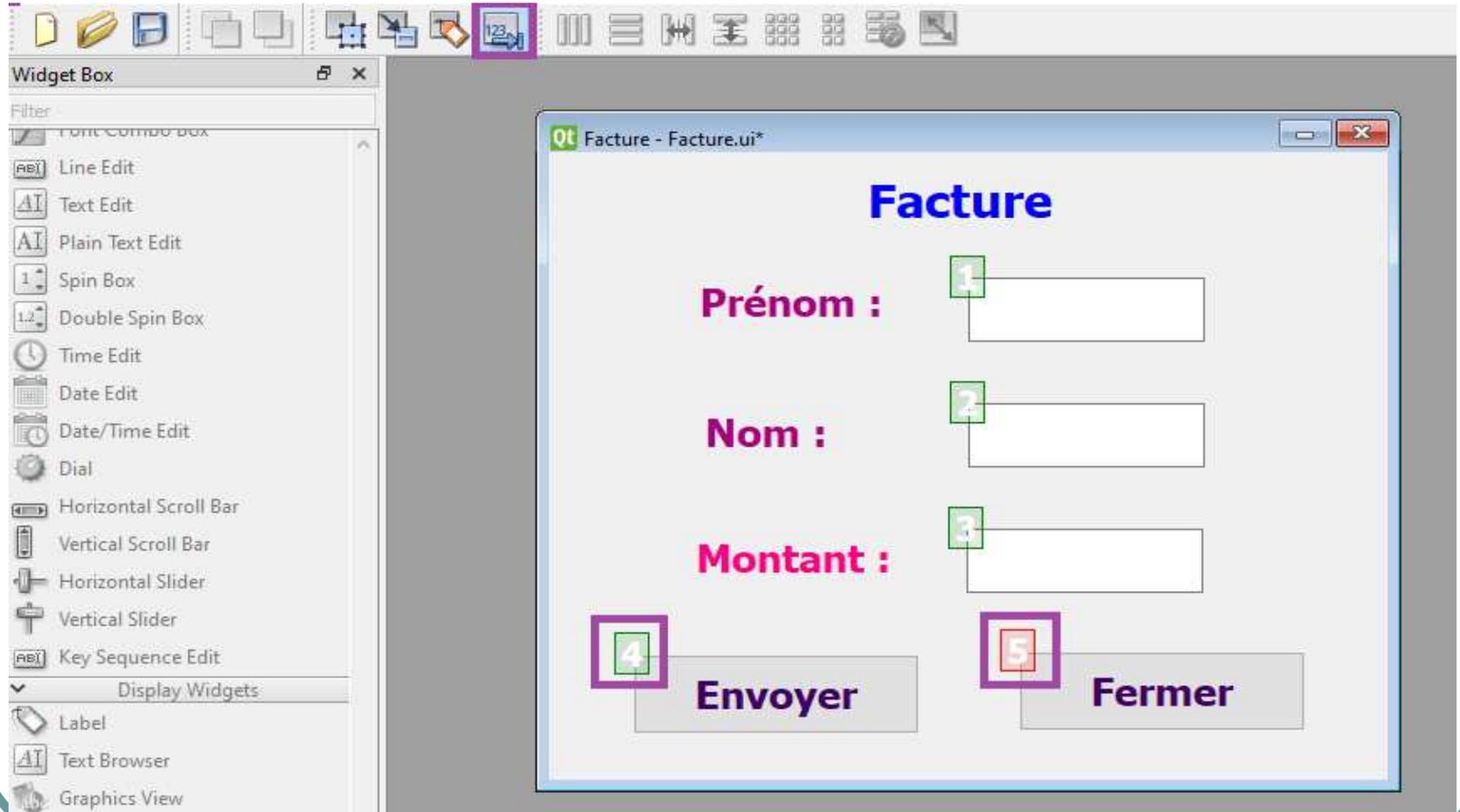


# Interface graphique Qt Designer + PyQt5





# Interface graphique Qt Designer + PyQt5





# Interface graphique

## Qt Designer + PyQt5

### N.B. :

- Le fichier créé par **Qt Designer** (l'interface graphique) possède l'extension **".ui"** (User Interface),
- Après avoir réalisé l'interface graphique de votre application (**Nom\_Fichier.ui**), on doit passer à la partie **programmation** (écriture du **code**) (**Nom\_Fichier.py**), pour cela il faut lancer un **éditeur Python** (Environnement de développement) et écrire le **code** correspondant à votre application.
- De préférence, il faut mettre les **deux fichiers** **Nom\_Fichier.ui** et **Nom\_Fichier.py** dans le **même répertoire**



# Interface graphique

## Qt Designer + PyQt5

### N.B. :

Pour écrire le **code** correspondant à notre application (**Nom\_Fichier.py**) on va utiliser l'éditeur **Thonny** et bien sûr il faut installer les **packages PyQt5** et **PyQt5-tools** qu'on a déjà vu dans la partie **Environnement**



# Interface graphique

## Qt Designer + PyQt5

### Écriture du code avec PyQt5 pour votre application

Généralement le fichier **Nom\_fichier.py** qui contient de **code de votre application** doit :

- Importer la bibliothèque **PyQt5** et les **modules** pour réaliser votre interface graphique

```
from PyQt5 import QtWidgets, uic
```

- Importer la classe qui permet de créer l'application :

```
Nom_objet_App = QtWidgets.QApplication([])
```

- Charger le fichier **Nom\_fichier.ui** crée par **Qt**

**Designer**

```
Nom_objet_Fen = uic.loadUi( "Nom_fichier.ui" )
```



# Interface graphique

## Qt Designer + PyQt5

- Visualiser la fenêtre à l'aide de la méthode `show()` :  
`Nom_objet_Fen.show()`
- Générer un **Signal** (événement) : Exemple `clicked` : Appel du fonction  
`Nom_objet_Fen.Nom_Widget.clicked.connect(nom_fonction)`
- Exécuter l'application à l'aide de la méthode `exec_()` :  
`Nom_objet_App.exec_()`
- Créer le **slot** : la **fonction** qui va être appelée en réponse d'un signal



# Interface graphique

## Qt Designer + PyQt5

### Nom\_Fichier.py

```
# importations à faire pour réaliser une interface graphique  
from PyQt5 import QtWidgets , uic
```

```
# Slot : Définir une fonction
```

```
def nom_fonction( ) :  
    < Actions >
```

```
# Création d'une application
```

```
Nom_objet_App = QtWidgets.QApplication([])
```

```
# charger le fichier crée par Qt Designer
```

```
Nom_objet_Fen = uic.loadUi( "nom_fichier.ui" )
```

```
# Visualiser la fenêtre
```

```
Nom_objet_Fen.show()
```

```
# Signal (événement) : Exemple clicked : Appel du fonction
```

```
Nom_objet_Fen.Nom_Widget.clicked.connect(nom_fonction)
```

```
# exécution de l'application
```

```
app.exec_()
```



# Interface graphique

## Qt Designer + PyQt5

### Signal / Slot avec PyQt5

**Exemple : Programmer avec PyQt5 le bouton Fermer de l'interface créée par Qt Designer (Facture.ui)**



# Interface graphique

## Qt Designer + PyQt5

### Fichier Facture.py

```
# Importations à faire pour réaliser une interface graphique
from PyQt5 import QtWidgets , uic
# Slot : Définir une fonction
def Fermer ( ) :
    f.close()                # Fermer la fenêtre
App = QtWidgets.QApplication([]) # Création d'une instance de QApplication
f = uic.loadUi("Facture.ui" ) # Charger le fichier crée par Qt Designer
f.show()                    # Visualiser la fenêtre
# Signal (clicked) clic sur le bouton de fermeture BFermer
f.BFermer.clicked.connect(Fermer) # Appel de la fonction Fermer
App.exec_()                 # Exécution de l'application
```



# Interface graphique Qt Designer + PyQt5

## Exécution : Fichier Facture.py

The screenshot shows a window titled "Facture" with a light gray background. At the top center, the word "Facture" is written in a bold blue font. Below this, there are three input fields, each preceded by a label in a bold magenta font: "Prénom :", "Nom :", and "Montant :". Each label is followed by a white rectangular input box with a thin blue border. At the bottom of the window, there are two buttons with a light gray background and a thin blue border. The left button is labeled "Envoyer" in a bold blue font, and the right button is labeled "Fermer" in a bold blue font. The window has standard window control buttons (minimize, maximize, close) in the top right corner.

Le clic sur le bouton **Fermer** permet la fermeture de la fenêtre

# Interface graphique

## Qt Designer + PyQt5

### Mini Projet

Calcul arithmétique

Plus Grand Commun Diviseur

avec interface graphique

Qt Designer + PyQt5



# Interface graphique

## Qt Designer + PyQt5

### PGCD avec interface graphique

Notre **mini projet** consiste à créer une **interface graphique avec** l'outil de création **Qt Designer** qui permet de saisir deux entiers **M** et **N** positifs non **nuls** et afficher le **Plus Grand Commun Diviseur (PGCD)** en utilisant la **méthode des différences**

# Interface graphique

## Qt Designer + PyQt5

### *Exemples :*

- **PGCD (24,18) = PGCD (6,18)**  
**= PGCD (6,12)**  
**= PGCD (6,6) = 6**
- **PGCD (27,90) = PGCD (27, 63)**  
**= PGCD (27, 36)**  
**= PGCD (27, 9)**  
**= PGCD (18,9)**  
**= PGCD (9,9) = 9**

# Interface graphique

## Qt Designer + PyQt5

**Création de l'interface graphique  
avec Qt Designer**

**Le fichier**

**PGCD\_interface\_graphique.ui**

# Interface graphique

## Qt Designer + PyQt5

### Interface graphique



The image shows a screenshot of a Python GUI application window titled "Calcul PGCD". The window has a standard title bar with minimize, maximize, and close buttons. The main content area has a light gray background and contains the following elements:

- A title "Plus Grand Commun Diviseur" in bold blue text.
- Three input fields with labels in red text:
  - "Taper M" followed by a text input box.
  - "Taper N" followed by a text input box.
  - "Résultat" followed by a text input box.
- Two buttons at the bottom:
  - "Calcul PGCD" on the left.
  - "Fermer" on the right.

# Interface graphique

## Qt Designer + PyQt5

### Le fichier

**PGCD\_interface\_graphique.py**

# Interface graphique Qt Designer + PyQt5

Exécution :



# Interface graphique

## Qt Designer + PyQt5

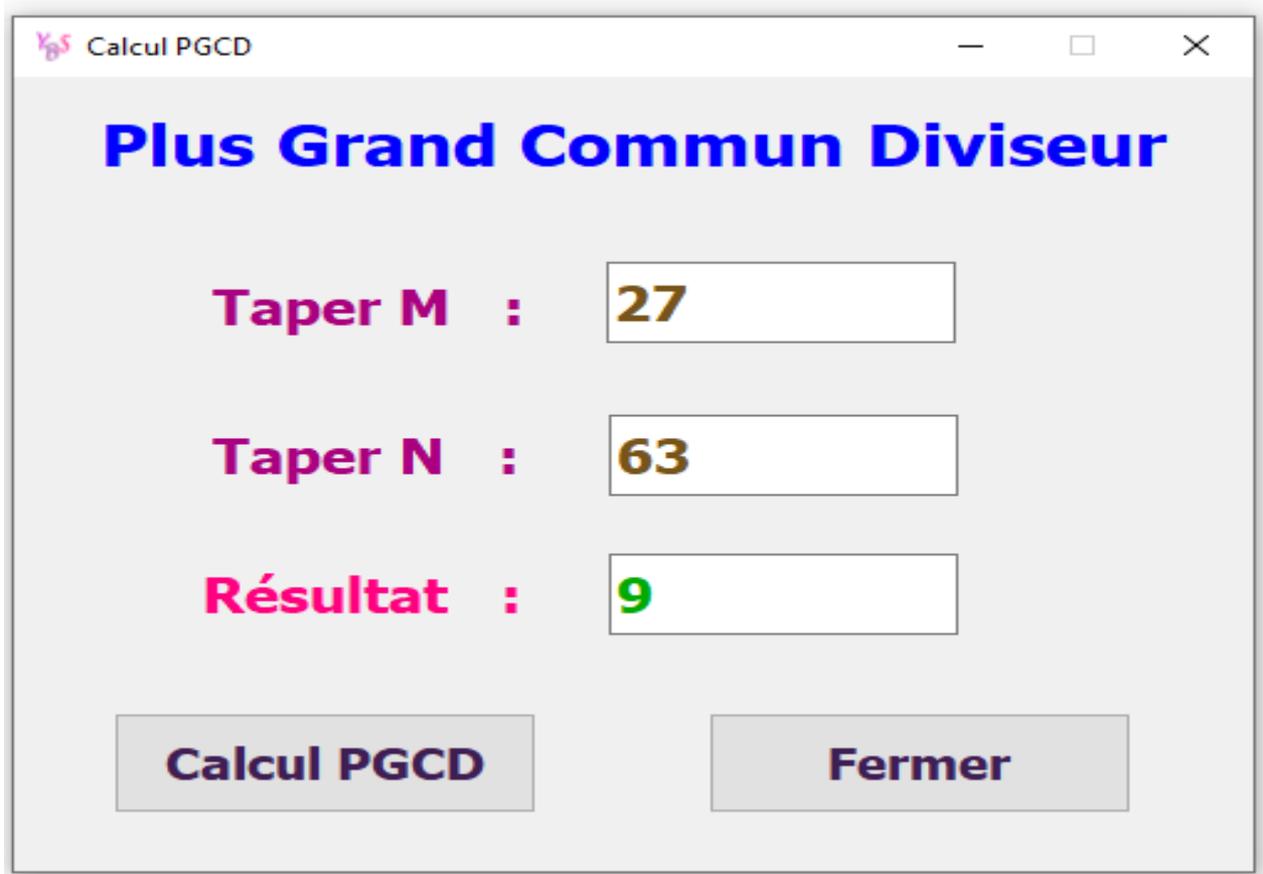
Exécution :



# Interface graphique

## Qt Designer + PyQt5

**Exécution :**



Calcul PGCD

### Plus Grand Commun Diviseur

Taper M :

Taper N :

Résultat :

Calcul PGCD Fermer

# Interface graphique

## Qt Designer + PyQt5

### Mini Projet

Calcul arithmétique

Nombres premiers

avec interface graphique

Qt Designer + PyQt5



# Interface graphique

## Qt Designer + PyQt5

### Nombres premiers avec interface graphique

Notre **mini projet** consiste à créer une **interface graphique** avec l'outil de création **Qt Designer** qui permet de saisir un nombre entier **NB** (avec  **$NB > 1$** ) et afficher s'il est un nombre premier ou non

**N.B.** : Un **nombre est dit premier** s'il est divisible seulement par 1 et par lui même

**Exemples de nombres premiers :**

2, 3, 5, 7, 11, 13, .....

# Interface graphique

## Qt Designer + PyQt5

### Création de l'interface graphique avec Qt Designer

#### Le fichier

**Nombres\_Premiers\_interface\_graphique.ui**

# Interface graphique

## Qt Designer + PyQt5

### Interface graphique



# Interface graphique

## Qt Designer + PyQt5

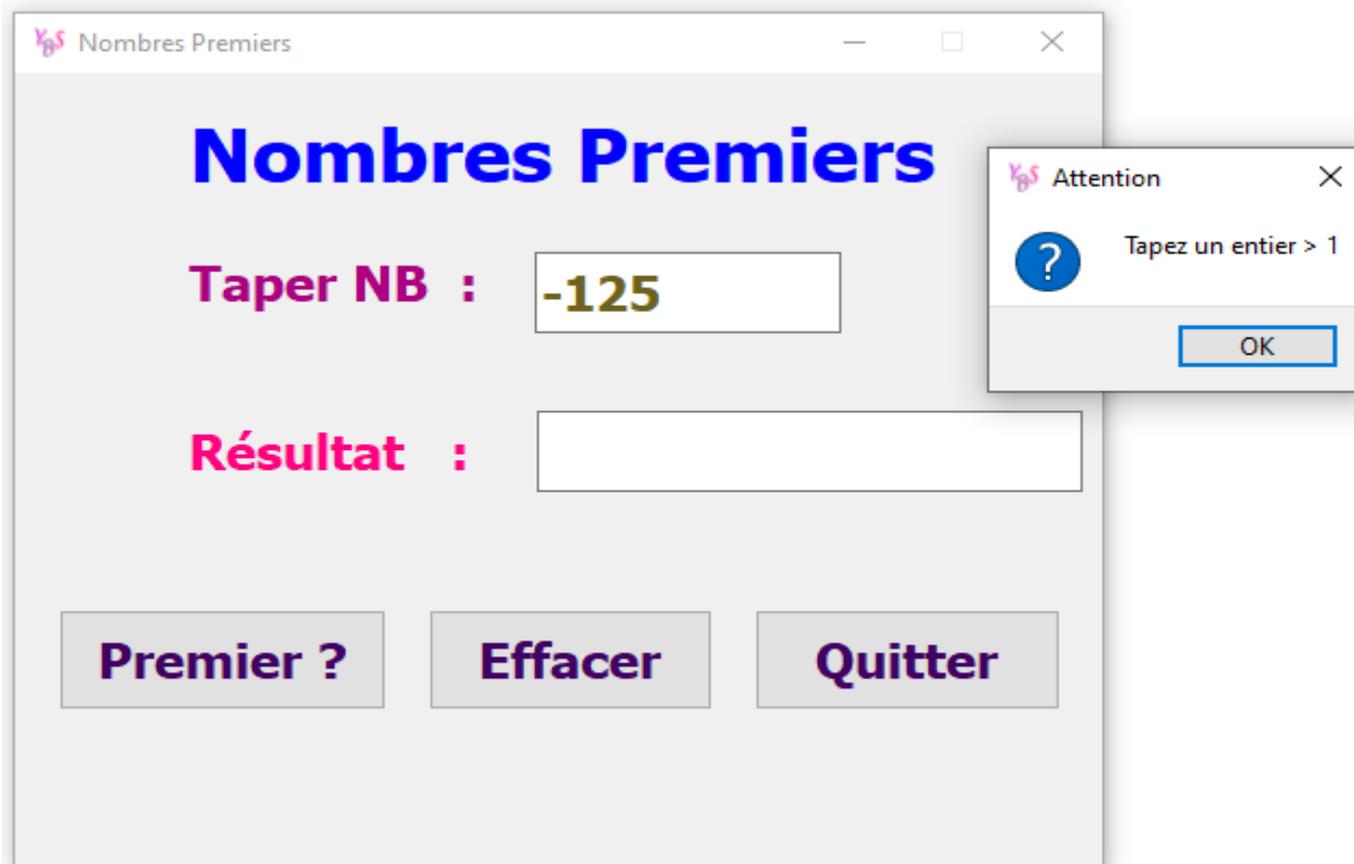
### Le fichier

**Nombres\_Premiers\_interface\_graphique.py**

# Interface graphique

## Qt Designer + PyQt5

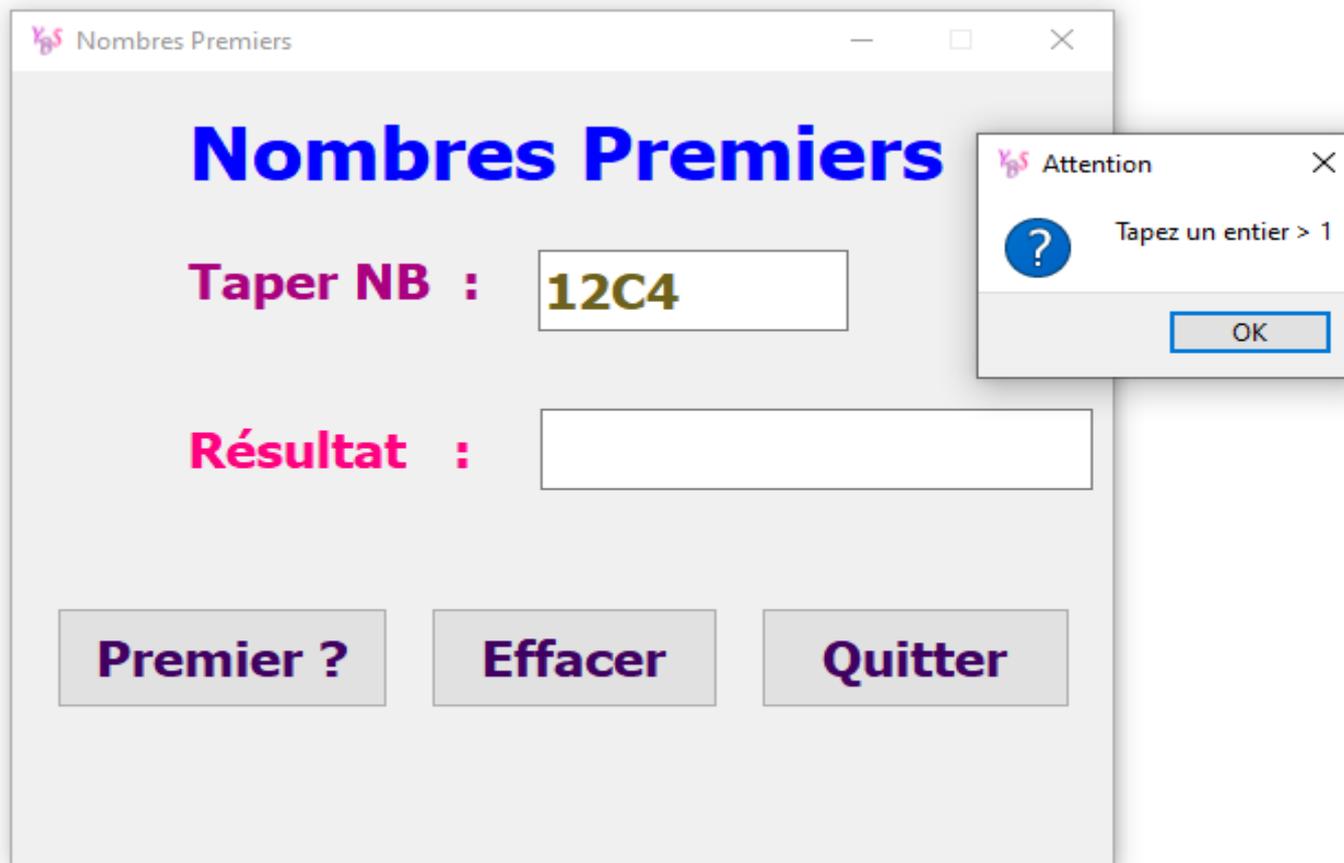
Exécution :



# Interface graphique

## Qt Designer + PyQt5

### Exécution :



# Interface graphique

## Qt Designer + PyQt5

**Exécution :**



# Interface graphique

## Qt Designer + PyQt5

**Exécution :**

