

# **Atelier Python GUI & MVC**

Introduction à Tkinter, PySide6 et l'architecture MVC

Killian PAVY

# Plan

## 1. Initiation aux interfaces graphiques en Python

- a. Bibliothèques graphiques Tkinter & PySide6**
- b. Étapes de création d'une GUI**
- c. Widgets et Layouts**
- d. Exemples**
- e. Utilisation QTDesigner (optionnel)**
- f. Exercices**

## 2. Introduction MVC

- a. Explication du concept**
- b. Exemples**

# Lien du repo GitHub

[bit.ly/4f3jjm7](https://bit.ly/4f3jjm7)



Crackvignoule / python-gui-mvc-tutorial (Public)

<> Code Issues Pull requests Actions Projects Security Insights

main 1 Branch 0 Tags

Go to file

<> Code

About

A short tutorial on Python GUI and MVC using Tkinter & PySide6

Readme View license Activity 0 stars 1 watching 0 forks Report repository

Releases

No releases published

Packages

No packages published

Languages

Python 100.0%

Crackvignoule Add French translation to README and create English version 21c3bf1 · 4 minutes ago 15 Commits

examples	add ttk example	last month
exercises	exercise 3	last month
presentation	exercise 1 solutions	last month
.gitignore	examples	last month
LICENSE	Update license	last month
README.md	Add French translation to README and create English version	4 minutes ago
README_EN.md	Add French translation to README and create English version	4 minutes ago

README License

[README in English](#)

## Tutoriel Python GUI et MVC



# **1 ■ Initiation aux interfaces graphiques en Python**

# Initiation aux interfaces graphiques en Python

Bibliothèques graphiques Tkinter & PySide6



Bibliothèque native	Interface moderne et responsive
Bibliothèque très légère	Large collection de widgets
Simple à utiliser pour projets légers	Documentation complète
Exemples abondants sur internet	Fonctions supplémentaires (drag'n drop, signaux...)
	Ecosystème Qt, ex: Concepteur Graphique « QtDesigner »

Critère	Tkinter	PySide6
Facilité d'apprentissage	★★★★★	★★★
Puissance et flexibilité	★★★	★★★★★
Apparence native et cohérence multi-plateformes	★★	★★★★★

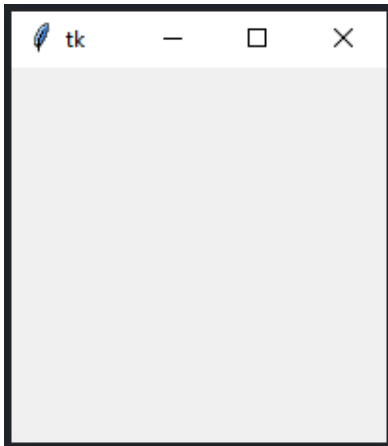
# Initiation aux interfaces graphiques en Python

Étapes de création d'une GUI

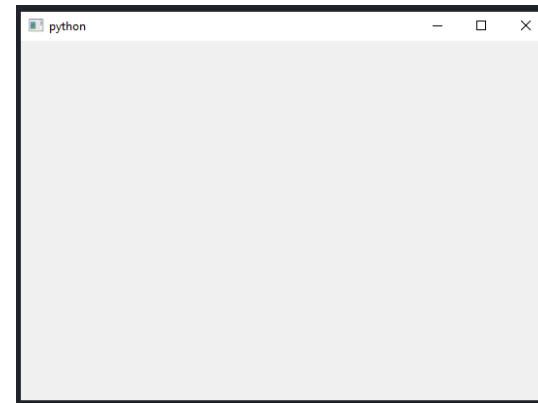
1. Importer la bibliothèque
2. Créer la fenêtre principale (Main Window)
3. Ajouter des widgets (boutons, textes...)
4. Entrer dans la boucle d'évènements (Event Loop)



```
import tkinter as tk # Importer tkinter
root = tk.Tk()        # Créer Main Window
root.mainloop()       # Lancer la boucle principale
```



```
from PySide6.QtWidgets import QApplication, QWidget
app = QApplication([]) # Créer une application
window = QWidget()     # Créer une fenêtre
window.show()          # Afficher la fenêtre
app.exec()             # Lancer la boucle principale
```



# Initiation aux interfaces graphiques en Python

Étapes de création d'une GUI

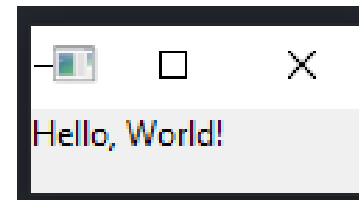
1. Importer la bibliothèque
2. Créer la fenêtre principale (Main Window)
3. Ajouter des widgets (boutons, textes...)
4. Entrer dans la boucle d'évènements (Event Loop)



```
import tkinter as tk # Importer tkinter
root = tk.Tk()        # Créer Main Window
label = tk.Label(root, text="Hello, World!") # Créer un label
label.pack()          # Ajouter le label
root.mainloop()       # Lancer la boucle principale
```



```
from PySide6.QtWidgets import QApplication, QWidget, QLabel
app = QApplication([]) # Créer une application
window = QWidget()     # Créer une fenêtre
label = QLabel("Hello, World!", parent=window) # Créer un label
window.show()          # Afficher la fenêtre
app.exec()             # Lancer la boucle principale
```



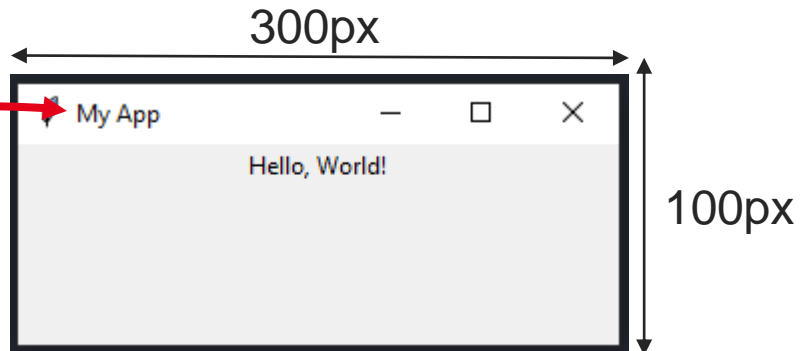
# Initiation aux interfaces graphiques en Python

Étapes de création d'une GUI

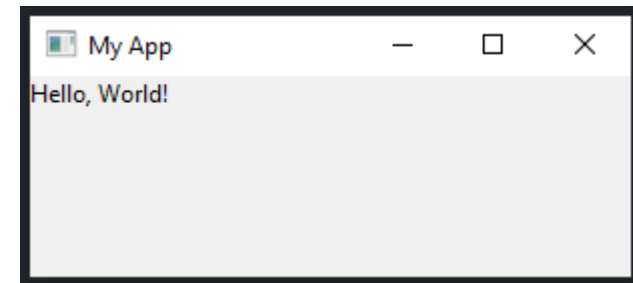
(optionnel) Changer le titre et la taille de la fenêtre



```
import tkinter as tk # Importer tkinter
root = tk.Tk()        # Créer Main Window
root.title("My App")  # Changer le titre de la fenêtre
root.geometry("300x100") # Définir la taille de la fenêtre
label = tk.Label(root, text="Hello, World!") # Créer un label
label.pack()          # Ajouter le label
root.mainloop()       # Lancer la boucle principale
```



```
from PySide6.QtWidgets import QApplication, QWidget, QLabel
app = QApplication([]) # Créer une application
window = QWidget()    # Créer une fenêtre
window.setWindowTitle("My App") # Changer le titre de la fenêtre
window.setGeometry(0, 0, 300, 100) # Définir la taille de la fenêtre
label = QLabel("Hello, World!", parent=window) # Créer un label
window.show()         # Afficher la fenêtre
app.exec()             # Lancer la boucle principale
```



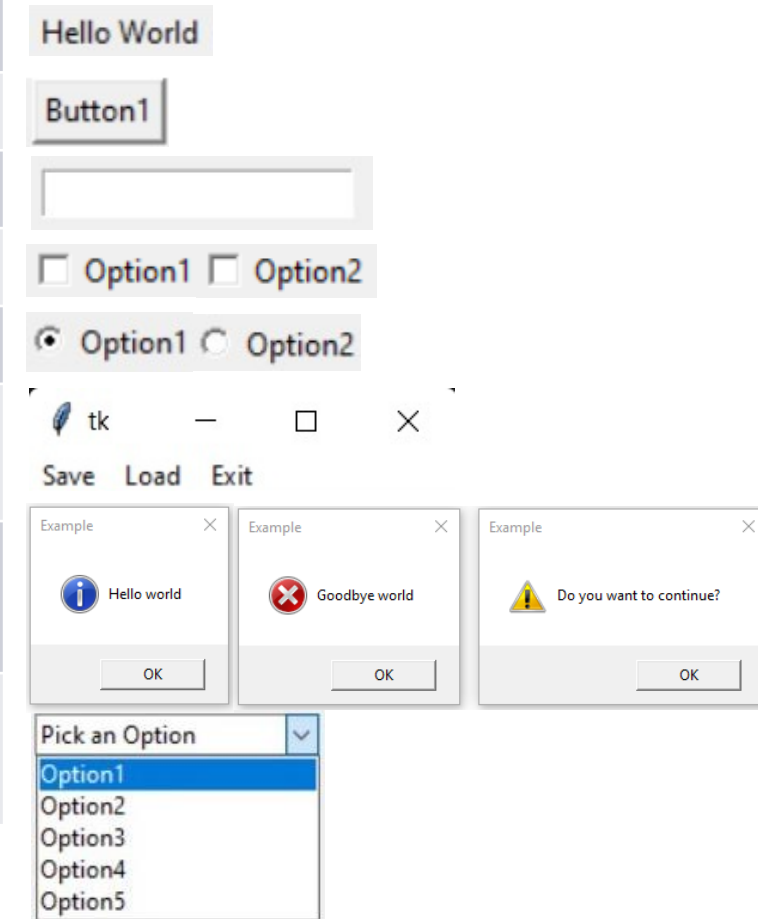


# Initiation aux interfaces graphiques en Python

Widgets et Layouts

Une liste non exhaustive de widgets:

Widget	Tkinter	PySide6
Label	<code>tk.Label(text="...")</code>	<code>QLabel("...")</code>
Bouton	<code>tk.Button(text="...")</code>	<code>QPushButton("...")</code>
Entry	<code>tk.Entry</code>	<code>QLineEdit</code>
Checkbox	<code>tk.Checkbutton</code>	<code>QCheckBox</code>
Radiobutton	<code>tk.Radiobutton</code>	<code>QRadioButton</code>
Menu	<code>tk.Menu</code>	<code>QMenuBar</code>
MessageBox	<code>tk.messagebox</code>	<code>QMessageBox</code>
Combobox	<code>from tkinter import ttk</code> <code>ttk.Combobox</code>	<code>QComboBox</code>

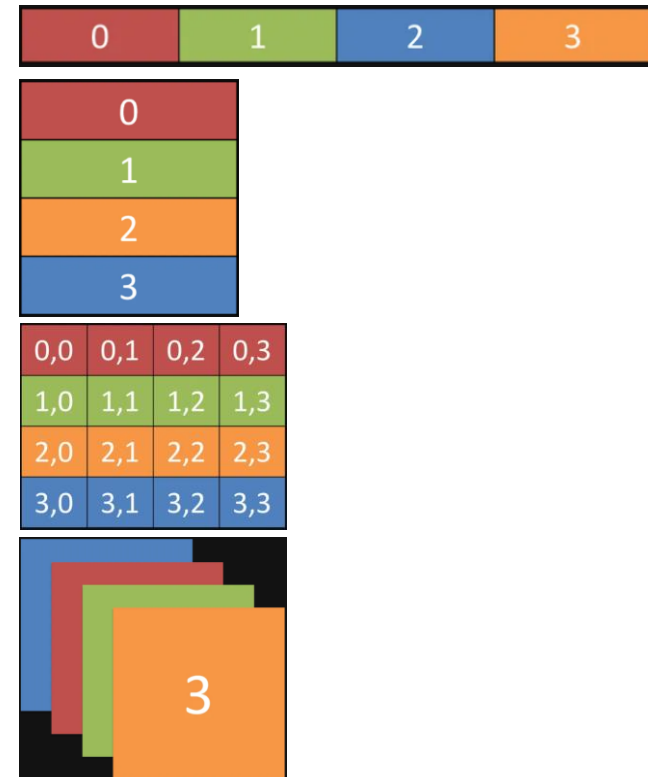


# Initiation aux interfaces graphiques en Python

## Widgets et Layouts

**Layout:** Gestionnaire de disposition qui organise les widgets dans une interface utilisateur.

Layout	Tkinter	Description
QHBoxLayout	<code>widget.pack(side="left")</code>	Ligne Horizontale
QVBoxLayout	<code>widget.pack(side="top")</code>	Ligne Verticale
QGridLayout	<code>widget.grid(row=0, column=1)</code>	Grille X,Y
QStackedLayout	Pas d'équivalent direct	Stack empilé Z



source: <https://www.pythonguis.com/tutorials/pyside6-layouts/>

# Initiation aux interfaces graphiques en Python

## Exemples

[bit.ly/4f3jjm7](https://bit.ly/4f3jjm7)



1. Ouvrir l'invite de commande (cmd) sur Windows:

- Appuyez sur **Win + R** pour ouvrir la boîte de dialogue Exécuter.
- Tapez **cmd** et appuyez sur **Entrée**.

2. Cloner le dépôt :

```
git clone https://github.com/Crackvignoule/python-gui-mvc-tutorial.git  
cd python-gui-mvc-tutorial
```

3. Créer un environnement virtuel :

```
python -m venv .venv
```

4. Activer l'environnement virtuel :

- Sur Windows :

```
.\.venv\Scripts\activate
```

- Sur macOS et Linux :

```
source .venv/bin/activate
```

5. Installer les dépendances :

```
pip install pyside6
```

# Initiation aux interfaces graphiques en Python

Exemples

## Utilisation QMainWindow + Layout



```
app = QApplication([])
window = QMainWindow()

central_widget = QWidget()
layout = QVBoxLayout()
```

⋮

```
layout.addWidget(email_label)
layout.addWidget(email_input)
layout.addWidget(submit_button)

central_widget.setLayout(layout)
window.setCentralWidget(central_widget)
```



```
label.pack()
entry.pack()
button.pack()
```

```
label.grid(row=0, column=0)
entry.grid(row=0, column=1)
button.grid(row=1, column=0)
```

# Initiation aux interfaces graphiques en Python

Exemples

## Utilisation Button:



```
tk.Button(root, text="Submit", command=submit)
```



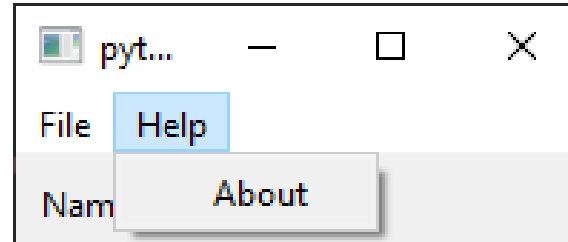
```
submit_button = QPushButton("Submit")  
submit_button.clicked.connect(submit)
```

```
def submit():  
    print("...")
```

# Initiation aux interfaces graphiques en Python

Exemples

Utilisation MenuBar:



```
menu_bar = tk.Menu(root)
root.config(menu=menu_bar)
```

```
help_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="Help", menu=help_menu)
help_menu.add_command(label="About", command=show_about)
```



```
menu_bar = QMenuBar()
window.setMenuBar(menu_bar)
```

```
help_menu = menu_bar.addMenu("Help")
about_action = QAction("About", window)
about_action.triggered.connect(show_about)
help_menu.addAction(about_action)
```

# Initiation aux interfaces graphiques en Python

Exemples

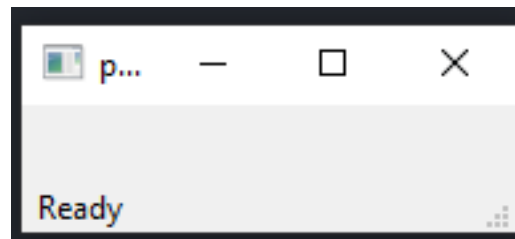
## Utilisation StatusBar



```
status_var = tk.StringVar()
status_var.set("Ready")
status_bar = tk.Label(root, textvariable=status_var, bd=1, anchor=tk.W)
status_bar.grid(row=4, columnspan=2, sticky=tk.W+tk.E)
```



```
status_bar = QStatusBar()
window.setStatusBar(status_bar)
status_bar.showMessage("Ready")
```

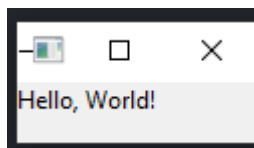


# Initiation aux interfaces graphiques en Python

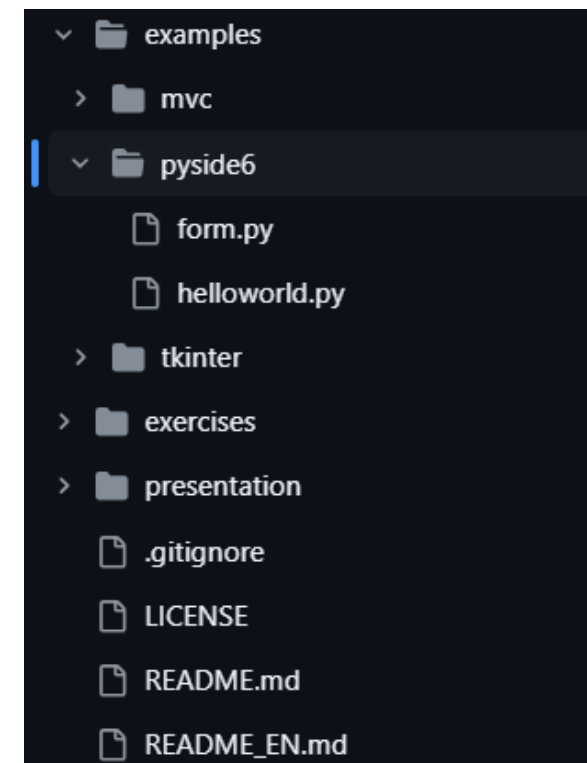
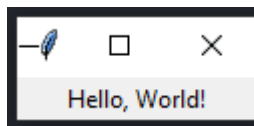
## Exemples



```
python ./examples/tkinter/helloworld.py
```



```
python ./examples/pyside6/helloworld.py
```



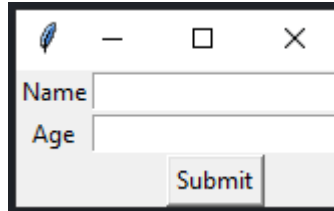


# Initiation aux interfaces graphiques en Python

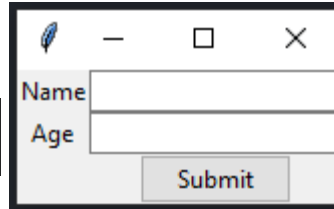
## Exemples



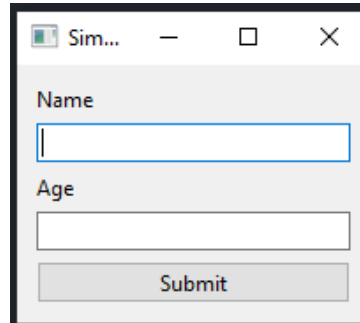
```
python ./examples/tkinter/form.py
```



```
python ./examples/tkinter/ttk_form.py
```



```
python ./examples/pyside6/form.py
```



```
import tkinter as tk
from tkinter import ttk

def submit():
    print(f"Name: {name_entry.get()}, Age: {age_entry.get()}")

root = tk.Tk()
root.title("Simple Form with ttk")

ttk.Label(root, text="Name").grid(row=0)
ttk.Label(root, text="Age").grid(row=1)

name_entry = ttk.Entry(root)
age_entry = ttk.Entry(root)

name_entry.grid(row=0, column=1)
age_entry.grid(row=1, column=1)

ttk.Button(root, text="Submit", command=submit).grid(row=2, column=1)

root.mainloop()
```

# Initiation aux interfaces graphiques en Python

## Exemples



```
python ./examples/tkinter/form.py
```



```
python ./examples/tkinter/ttk_form.py
```



```
python ./examples/pyside6/form.py
```

```
import tkinter as tk
from tkinter import ttk
```

```
def submit():
    print(f"Name: {name_entry.get()}, Age: {age_entry.get()}")
```

```
root = tk.Tk()
root.title("Simple Form with ttk")
```

```
ttk.Label(root, text="Name").grid(row=0)
ttk.Label(root, text="Age").grid(row=1)
```

```
name_entry = ttk.Entry(root)
age_entry = ttk.Entry(root)
```


```
name_entry.grid(row=0, column=1)
age_entry.grid(row=1, column=1)
```

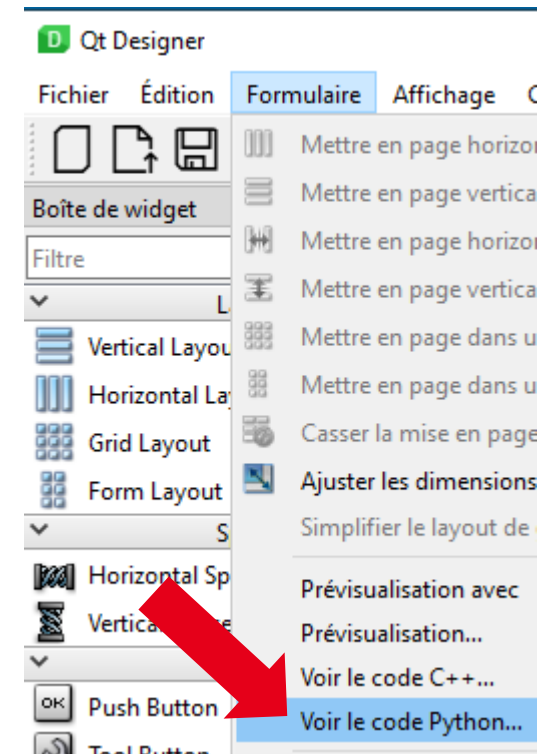
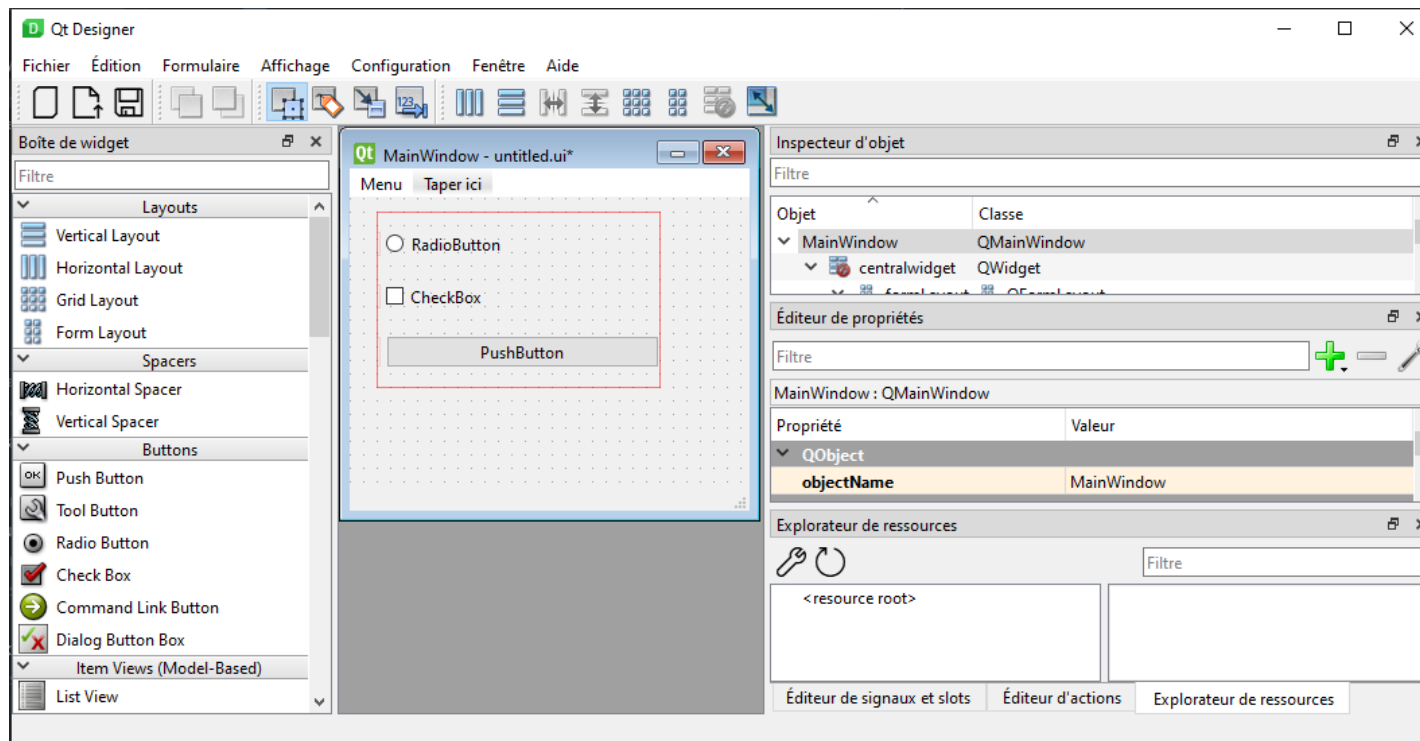
```
ttk.Button(root, text="Submit", command=submit).grid(row=2, column=1)
```

```
root.mainloop()
```

# Initiation aux interfaces graphiques en Python

Utilisation QTDesigner (optionnel)

1. `pip install pyside6` 
2. `python -c "import os, site, subprocess; subprocess.run([os.path.join(site.getsitepackages()[1], 'PySide6', 'designer'])]"`



# Initiation aux interfaces graphiques en Python

## Exercices

### Exercice 1 : Ajouter un champ Email

#### Objectif

Ajouter un champ de saisie d'email au formulaire.

#### Instructions

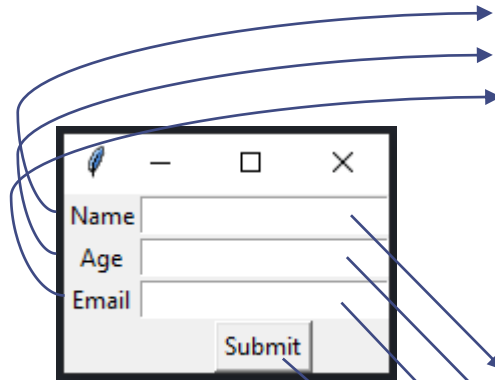
1. Ouvrez le fichier de formulaire pour le framework que vous utilisez :
  - Tkinter: [form.py](#)
  - PySide6: [form.py](#)
2. Ajoutez le champ email.
3. Mettez à jour la fonction `submit` pour afficher l'email dans la console.

#### Solutions

- [Solution Tkinter](#)
- [Solution PySide6](#)

# Initiation aux interfaces graphiques en Python

## Exercices



```
import tkinter as tk

def submit():
    print(f"Name: {name_entry.get()}, Age: {age_entry.get()}, Email: {email_entry.get()}")

root = tk.Tk()
root.title("Simple Form")

tk.Label(root, text="Name").grid(row=0)
tk.Label(root, text="Age").grid(row=1)
tk.Label(root, text="Email").grid(row=2)

name_entry = tk.Entry(root)
age_entry = tk.Entry(root)
email_entry = tk.Entry(root)

name_entry.grid(row=0, column=1)
age_entry.grid(row=1, column=1)
email_entry.grid(row=2, column=1)

tk.Button(root, text="Submit", command=submit).grid(row=3, column=1)

root.mainloop()
```

# Initiation aux interfaces graphiques en Python

## Exercices

### Exercice 2 : Ajouter une validation de formulaire

#### Objectif

Ajouter une validation aux champs du formulaire pour s'assurer que les champs nom, âge et email ne sont pas vides.

#### Instructions

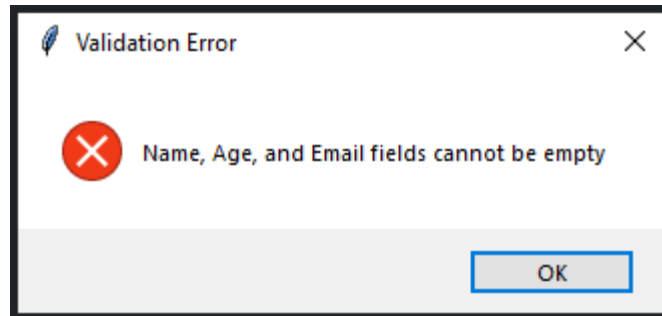
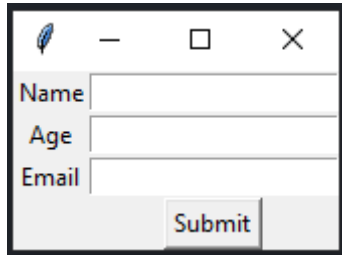
1. Ouvrez le fichier de solution pour l'Exercice 1 pour le framework que vous utilisez :
  - Tkinter : [tkinter-solution.py](#)
  - PySide6 : [pyside6-solution.py](#)
2. Ajoutez une validation pour s'assurer que les champs nom, âge et email ne sont pas vides.
3. Affichez un message d'erreur si la validation échoue.

#### Solutions

- [Solution Tkinter](#)
- [Solution PySide6](#)

# Initiation aux interfaces graphiques en Python

## Exercices



```
def submit():  
    name = name_entry.get()  
    age = age_entry.get()  
    email = email_entry.get()  
  
    if not name or not age or not email:  
        messagebox.showerror("Validation Error", "Name, Age, and Email fields cannot be empty")  
    else:  
        print(f"Name: {name}, Age: {age}, Email: {email}")
```

# Initiation aux interfaces graphiques en Python

## Exercices

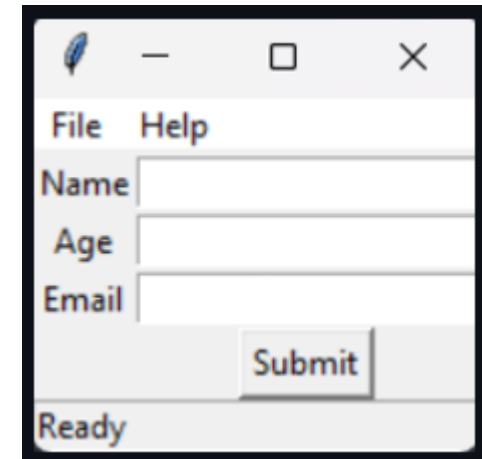
### Exercice 3 : Ajouter un menu et une barre d'état

#### Objectif

Améliorer l'application en ajoutant un menu et une barre d'état.

#### Instructions

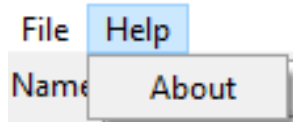
1. Ouvrez le fichier corrigé de l'Exercice 2 correspondant au framework que vous utilisez :
  - Tkinter: [tkinter-solution.py](#)
  - PySide6: [pyside6-solution.py](#)
2. Ajoutez un menu avec les options suivantes :
  - **File** contenant une option **Quit** pour fermer l'application.
  - **Help** contenant une option **About** pour afficher un message d'information.
3. Ajoutez une barre d'état pour afficher un message par défaut.





# Initiation aux interfaces graphiques en Python

## Exercices



```
menu_bar = tk.Menu(root)
root.config(menu=menu_bar)

file_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="File", menu=file_menu)
file_menu.add_command(label="Quit", command=quit_app)

help_menu = tk.Menu(menu_bar, tearoff=0)
menu_bar.add_cascade(label="Help", menu=help_menu)
help_menu.add_command(label="About", command=show_about)
```

```
def submit():
```

```
...
```

```
    print(f"Name: {name}, Age: {age}, Email: {email}")
    status_var.set("Form submitted successfully")
```

```
def quit_app():
    root.quit()
```

```
def show_about():
    messagebox.showinfo("About", "This is a simple form app")
```



```
status_var = tk.StringVar()
status_var.set("Ready")
status_bar = tk.Label(root, textvariable=status_var, bd=1, anchor=tk.W)
status_bar.grid(row=4, columnspan=2, sticky=tk.W+tk.E)
```

# Sondage

[bit.ly/3AMMMM](https://bit.ly/3AMMMM)





# 2 ■ Introduction MVC

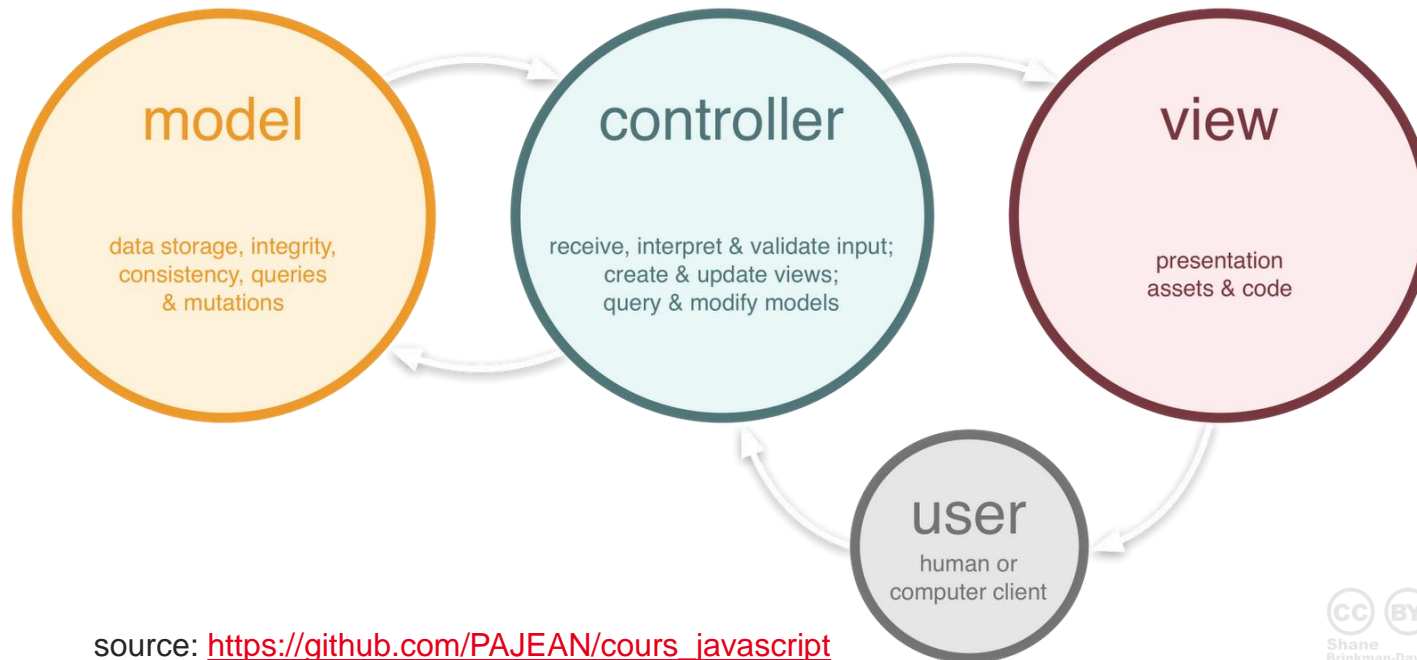
# Introduction MVC

## Explication du concept

**Définition :** Une méthode de structuration du code pour séparer la logique métier, l'affichage, et les interactions utilisateur.

### Principe de base :

- **Modèle :** Gère les données et le contenu algorithmique
- **Vue :** Interface Graphique, permet à l'utilisateur d'interagir avec le(s) modèle(s)
- **Contrôleur :** Relie l'utilisateur (la vue) et le système (le modèle).



### Avantages :

- Code plus propre et maintenable.
- Facilite le travail en équipe.
- Simplifie les modifications futures.

source: [https://github.com/PAJEAN/cours\\_javascript](https://github.com/PAJEAN/cours_javascript)



# Introduction MVC

## Exemples

[python-gui-mvc-tutorial](#) / [examples](#) / [mvc](#) / [simple](#) / [helloworld.py](#)



```
class Model:
    def __init__(self):
        self.data = "Hello MVC"

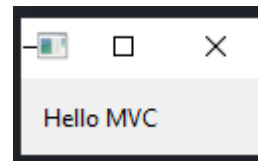
class View(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Hello MVC App")

        self.layout = QVBoxLayout()
        self.label = QLabel("")
        self.layout.addWidget(self.label)
        self.setLayout(self.layout)

    def display(self, data):
        self.label.setText(data)
```

```
class Controller:
    def __init__(self, model, view):
        self.model = model
        self.view = view
        self.update_view()

    def update_view(self):
        data = self.model.data
        self.view.display(data)
```



```
class Model
    func __init__

class View
    func __init__
    func display

class Controller
    func __init__
    func update_view
```

# Introduction MVC

## Exemples

[python-gui-mvc-tutorial / examples / mvc / simple / count.py](#)



```
class Model:
    def __init__(self):
        self.count = 0

    def increment_count(self):
        self.count += 1
```

```
class Controller:
    def __init__(self, model, view):
        self.model = model
        self.view = view
        self.view.button.clicked.connect(self.increment_count)
        self.update_view()

    def increment_count(self):
        self.model.increment_count()
        self.update_view()

    def update_view(self):
        count = self.model.count
        self.view.update_label(count)
```

```
class View(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Counter App")

        self.layout = QVBoxLayout()

        self.label = QLabel("Count: 0")
        self.layout.addWidget(self.label)

        self.button = QPushButton("Increment")
        self.layout.addWidget(self.button)

        self.setLayout(self.layout)

    def update_label(self, count):
        self.label.setText(f"Count: {count}")
```

```
class Model
  func __init__
  func increment_count
class View
  func __init__
  func update_label
class Controller
  func __init__
  func increment_count
  func update_view
```

# Introduction MVC

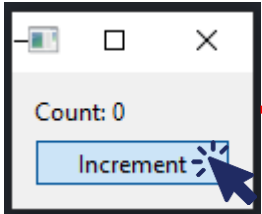
## Exemples

[python-gui-mvc-tutorial / examples / mvc / simple / count.py](#)



```
class View
```

```
self.button = QPushButton("Increment")
```



```
class Controller
```

```
self.view.button.clicked.connect(self.increment_count)
```

2

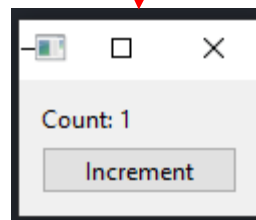
```
def increment_count(self):  
    self.model.increment_count()  
    self.update_view()  
  
def update_view(self):  
    count = self.model.count  
    self.view.update_label(count)
```

3

```
class Model
```

```
def increment_count(self):  
    self.count += 1
```

4

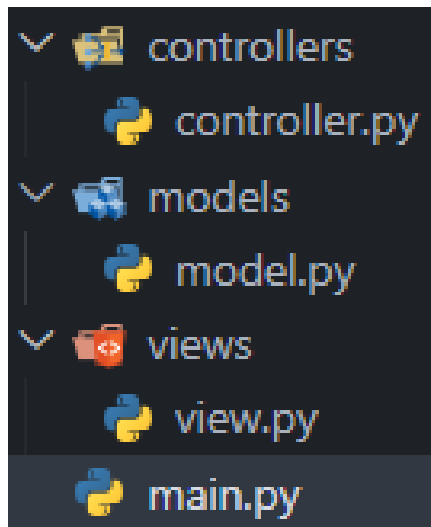


```
class Model  
    func __init__  
    func increment_count  
class View  
    func __init__  
    func update_label  
class Controller  
    func __init__  
    func increment_count  
    func update_view
```

# Introduction MVC

## Exemples

[python-gui-mvc-tutorial](#) / [examples](#) / [mvc](#) / [full](#) /



main.py

```
from PySide6.QtWidgets import QApplication
from models.model import Model
from views.view import View
from controllers.controller import Controller

if __name__ == "__main__":
    app = QApplication([])

    model = Model()
    view = View()
    controller = Controller(model, view)

    view.show()
    app.exec()
```



# Liens Utiles

**Lien repo GitHub:** <https://github.com/Crackvignoule/python-gui-mvc-tutorial>

**Documentation PySide6:** <https://doc.qt.io/qtforpython-6#documentation>

**Documentation Tkinter:** <https://docs.python.org/3/library/tkinter.html>

**PythonGUIs: Tkinter** <https://www.pythonguis.com/tkinter/>

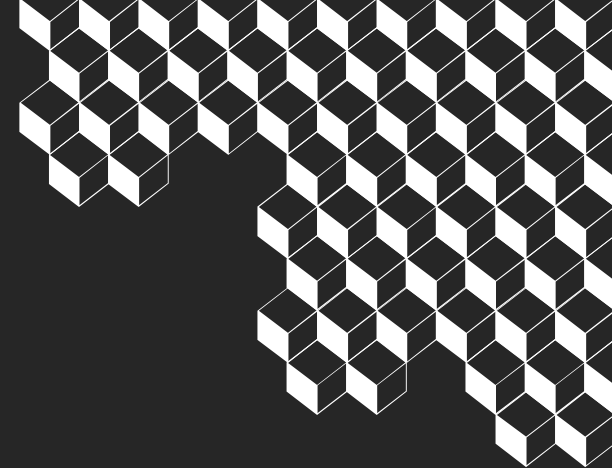
**PythonGUIs: PySide6** <https://www.pythonguis.com/pyside6/>

**PythonGUIs Examples** <https://github.com/pythonguis/pythonguis-examples>

**Which Python GUI Library Should You Choose?** <https://www.pythonguis.com/faq/which-python-gui-library/>

**superqt - A collection of custom widgets for PySide6** <https://pyapp-kit.github.io/superqt/>

**PySide6 Examples** <https://doc.qt.io/qtforpython-6/examples/index.html>



# Merci

Des questions/remarques ?

[killian.pavy@cea.fr](mailto:killian.pavy@cea.fr)