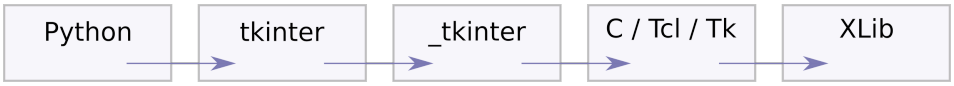
Interface graphiques

Contexte

- `Tcl` : langage de programmation [41]
- `Tk` : toolkit d'IHM de `Tcl` [42]
- `Tkinter` : binding python pour `Tcl` / `Tk`



Etapes de traduction du code

41. https://fr.wikipedia.org/wiki/Tool_Command_Language
42. https://fr.wikipedia.org/wiki/Tk_(informatique)

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Principe de fonctionnement des IHM

Par définition : on intéragit avec une interface graphique

Problématiques :

- organisation de l'information (UX)
  - non traité ici
- réaction aux actions de l'utilisateur (informatique)
  - programmation événementielle
- rafraîchissement de l'interface (performance / ingénierie)
  - géré par le framework (normalement...) / optimisation
- garantir la simplicité du code (informatique / ingénierie)
  - patron de construction MVC

Programmation événementielle

> En informatique, la programmation événementielle est un paradigme de programmation fondé sur les événements. Elle s'oppose à la programmation séquentielle. Le programme sera principalement défini par ses réactions aux différents événements qui peuvent se produire, c'est-à-dire des changements d'état de variable, par exemple l'incrémentation d'une liste, un mouvement de souris ou de clavier.
>
> https://fr.wikipedia.org/wiki/Programmation_événementielle

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Programmation événementielle

La programmation événementielle peut également être définie comme une technique d'architecture logicielle où l'application a une boucle principale divisée en deux sections : la première section détecte les événements, la seconde les gère. Elle est particulièrement mise en œuvre dans le domaine des interfaces graphiques.

---

https://fr.wikipedia.org/wiki/Programmation_événementielle

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

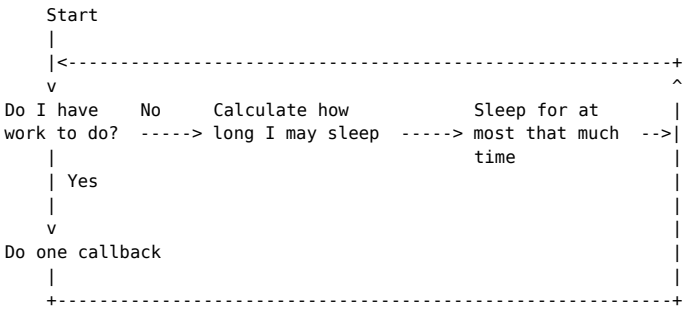
Programmation événementielle

- déclenchement d'événements suite à une interaction
- déclenchement d'événements programmés périodiques
- déclenchement d'événements programmés ponctuels
- du code va réagir à ces événements

Matthieu Falce

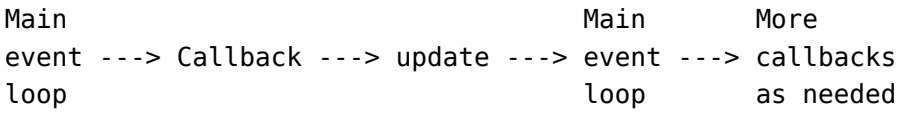
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Programmation événementielle



Source: https://wiki.tcl.tk/1527

Programmation événementielle



Source: https://wiki.tcl.tk/1527

Programmation événementielle

Les callbacks doivent s'exécuter rapidement.
Sinon blocage de la boucle d'événement

Programmation événementielle

Bonus : boucle d'événement minimale (en tkinter)

```
import tkinter

while True:
    tkinter.update_idletasks()
    tkinter.update()

## équivalent à
# tkinter.mainloop()
```

Permet de rajouter sa propre boucle d'événements
(modélisation physique par exemple)

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Patron de conception : Modèle - Vue - Contrôleur

Modèle-vue-contrôleur ou MVC est un motif d'architecture logicielle destiné aux interfaces graphiques lancé en 1978 et très populaire pour les applications web. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur

MVC est également très utilisé pour l'architecture d'interfaces graphiques

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Patron de conception : Modèle - Vue - Contrôleur

- le modèle (Model) : contient les données à afficher
  - base de données
  - liste de nom en mémoire
  - API
- le vue (View) : contient la présentation de l'interface graphique
  - tableau
  - HTML
- le contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur
  - supprimer une ligne des données
  - mettre à jour une information

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Patron de conception : Modèle - Vue - Contrôleur

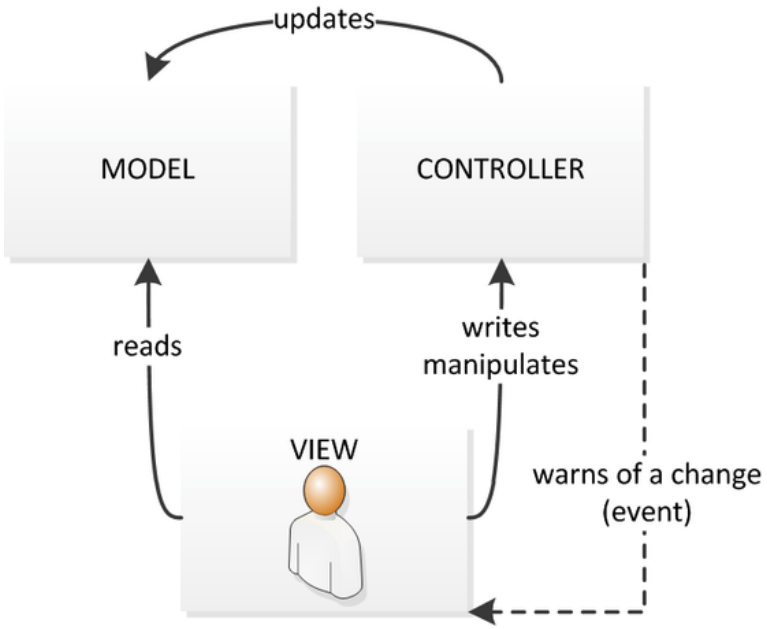


Schéma du modèle MVC

Source: https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur#/media/File:ModeleMVC.png

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Patron de conception : Modèle - Vue - Contrôleur

Et `Tcl / Tk` dans tout ça ?

In Tkinter, the standard widgets all use tight coupling between the model and the view; the model data is managed by the actual widget instance. Unfortunately, this means that you cannot display data from the same model in two different widgets (for example, two independent views into a text editor buffer). It also means that you have to convert your data to a form suitable for Tk.

---

http://effbot.org/zone/model-view-controller.htm

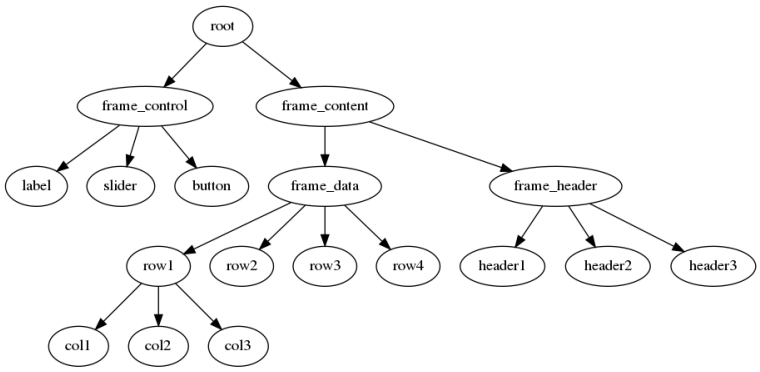
Inspiration du MVC pour découpler et éviter le code spaghetti.

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Conteneurs

Le conteneur principal est un cadre (Frame).

- la fenêtre principale est un cadre
- chaque cadre possède son propre système de positionnement
- permet de créer des applications modulaires



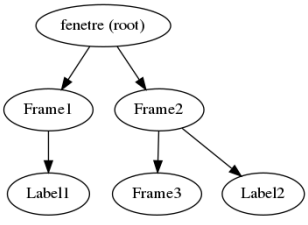
Example de hiérarchie de widgets

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Conteneurs [43]

Frames

```
from tkinter import *

fenetre = Tk(); fenetre['bg']='white'

# frame 1
Frame1 = Frame(fenetre, borderwidth=2, relief=GROOVE)
Frame1.pack(side=LEFT, padx=30, pady=30)
# frame 2
Frame2 = Frame(fenetre, borderwidth=2, relief=GROOVE)
Frame2.pack(side=LEFT, padx=10, pady=10)
# frame 3 dans frame 2
Frame3 = Frame(Frame2, bg="white", borderwidth=2, relief=GROOVE)
Frame3.pack(side=RIGHT, padx=5, pady=5)
# Ajout de labels
Label(Frame1, text="Frame 1").pack(padx=10, pady=10)
Label(Frame2, text="Frame 2").pack(padx=10, pady=10)
Label(Frame3, text="Frame 3",bg="white").pack(padx=10, pady=10)

fenetre.mainloop()
```



43.Exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Conteneurs [43]

Frames

```
from tkinter import *

fenetre = Tk(); fenetre['bg']='white'

# frame 1
Frame1 = Frame(fenetre, borderwidth=2, relief=GROOVE)
Frame1.pack(side=LEFT, padx=30, pady=30)
# frame 2
Frame2 = Frame(fenetre, borderwidth=2, relief=GROOVE)
Frame2.pack(side=LEFT, padx=10, pady=10)
# frame 3 dans frame 2
Frame3 = Frame(Frame2, bg="white", borderwidth=2, relief=GROOVE)
Frame3.pack(side=RIGHT, padx=5, pady=5)
# Ajout de labels
Label(Frame1, text="Frame 1").pack(padx=10, pady=10)
Label(Frame2, text="Frame 2").pack(padx=10, pady=10)
Label(Frame3, text="Frame 3",bg="white").pack(padx=10, pady=10)

fenetre.mainloop()
```

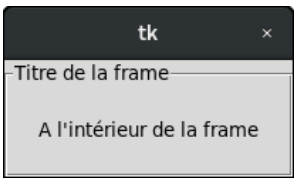
43.Exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Conteneurs [43]

LabelFrame

```
from tkinter import *


fenetre = Tk()

l = LabelFrame(fenetre, text="Titre de la frame", padx=20, pady=20)
l.pack(fill="both", expand="yes")

Label(l, text="A l'intérieur de la frame").pack()

fenetre.mainloop()
```



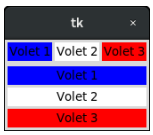
43.Exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Conteneurs [43]

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Paned window (peuvent se redimensionner)

```
from tkinter import *

fenetre = Tk()

p = PanedWindow(fenetre, orient=HORIZONTAL)
p.pack(side=TOP, expand=Y, fill=BOTH, pady=2, padx=2)
p.add(Label(p, text='Volet 1', background='blue', anchor=CENTER))
p.add(Label(p, text='Volet 2', background='white', anchor=CENTER) )
p.add(Label(p, text='Volet 3', background='red', anchor=CENTER) )
p.pack()

p2 = PanedWindow(fenetre, orient=VERTICAL)
p2.pack(side=BOTTOM, expand=Y, fill=BOTH, pady=2, padx=2)
p2.add(Label(p2, text='Volet 1', background='blue', anchor=CENTER))
p2.add(Label(p2, text='Volet 2', background='white', anchor=CENTER) )
p2.add(Label(p2, text='Volet 3', background='red', anchor=CENTER) )
p2.pack()

fenetre.mainloop()
```



43.Exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Widgets [45]

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Composant d'interface graphiques avec lequel on peut interagir [44]

- Label
- Button
- Text
- RadioButton
- ListBox
- Menu
- ...

44.https://fr.wikipedia.org/wiki/Composant_d'interface_graphique

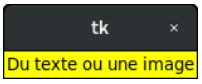
45.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Widgets [44]

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Label

```
from tkinter import *


fenetre = Tk()

label = Label(fenetre, text="Du texte ou une image", bg="yellow")
label.pack()

fenetre.mainloop()
```



44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

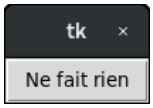
Widgets [44]

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Button

```
from tkinter import *


fenetre = Tk()

bouton = Button(fenetre, text="Ne fait rien")
bouton.pack()


fenetre.mainloop()
```



44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Widgets [44]

List

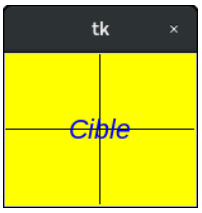
```
from tkinter import *


fenetre = Tk()

liste = Listbox(fenetre)
liste.insert(1, "Python")
liste.insert(2, "PHP")
liste.insert(3, "CSS")
liste.insert(4, "Javascript")

liste.pack()

# pour savoir ce qui est selectionné
index_selectionnes = liste.curselection()
if index_selectionnes:
    # index est un tuple avec les indexs sélectionnés
    valeur_selectionnee = liste.get(index_selectionnes[0])

fenetre.mainloop()
```

44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Widgets [44]

Canvas

```
from tkinter import *


fenetre = Tk()

canvas = Canvas(fenetre, width=150, height=120, background='yellow')
ligne1 = canvas.create_line(75, 0, 75, 120)
ligne2 = canvas.create_line(0, 60, 150, 60)
txt = canvas.create_text(75, 60, text="Cible", font="Arial 16 italic", fill="blue")
canvas.pack()

fenetre.mainloop()
```



44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

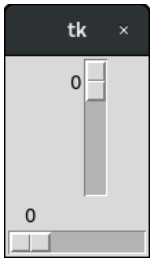
Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Widgets [44]

Scale

```
from tkinter import *


fenetre = Tk()

scale_ver = Scale(fenetre)
scale_ver.pack()

scale_hor = Scale(fenetre, orient="horizontal")
scale_hor.pack()

# TODO : get value

fenetre.mainloop()
```



44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Widgets [44]

Scrollbar

```
from tkinter import *

fenetre = Tk()

scrollbar = Scrollbar(fenetre)
scrollbar.pack(side=RIGHT, fill=Y)

# double connection :
#  * on scroll dans le widget => met à jour scrollbar
#  * on bouge l'ascenseur => met à jour le widget
listbox = Listbox(fenetre, yscrollcommand=scrollbar.set)
for i in range(1000):
    listbox.insert(END, "ligne : " + str(i))
listbox.pack(side=LEFT, fill=BOTH)

scrollbar.config(command=listbox.yview)

fenetre.mainloop()
```

- permet d'afficher des widgets plus gros que la fenêtre
- modifie le scroll en X ou Y
- s'utilise avec :
  - ListBox
  - Text
  - Canvas

44.exemples inspirés de
https://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Variables de contrôle

Les variables modifiées en Tk (dans des widgets par exemple) ne sont pas modifiées en Python

Les classes `Variables`

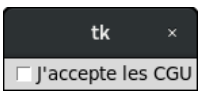
- `BooleanVar`
- `DoubleVar`
- `IntVar`
- `StringVar`

Certains *widgets* en ont besoin pour fonctionner

Variables de contrôle

CheckBox

```
from tkinter import *

fenetre = Tk()
var = IntVar()

bouton = Checkbutton(fenetre, text="J'accepte les CGU", variable=var)
bouton.pack()

# récupération de la valeur
print(var.get())

fenetre.mainloop()
```



Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Variables de contrôle

RadioButton

```
from tkinter import *


fenetre = Tk()

value = IntVar()
bouton1 = Radiobutton(fenetre, text="H", variable=value, value=1)
bouton2 = Radiobutton(fenetre, text="F", variable=value, value=2)
bouton3 = Radiobutton(fenetre, text="Autre", variable=value, value=3)
bouton1.pack()
bouton2.pack()
bouton3.pack()

valeur = value.get(); print(type(valeur), valeur)

fenetre.mainloop()
```

Variables de contrôle

Scale – la suite

```
from tkinter import *


fenetre = Tk()

value = DoubleVar()
scale = Scale(fenetre, variable=value)
scale.pack()

valeur = value.get()
print(type(valeur), valeur)

fenetre.mainloop()
```

Variables de contrôle

Entry

```python
from tkinter import *


fenetre = Tk()


value = StringVar()
value.set("Valeur")
entree = Entry(fenetre, textvariable=value, width=30)
entree.pack()

# label est mis à jour tout automatiquement
label = Label(fenetre, textvariable=value)
label.pack()

valeur = value.get()
print(type(valeur), valeur)

fenetre.mainloop()
```


Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Variables de contrôle

Entry – validation

```python
# plus de détails ici
# https://stackoverflow.com/questions/4140437/
# ou ici : http://tkinter.fdex.eu/doc/entw.html

from tkinter import *

fenetre = Tk()

def validate(valeur_dans_entry):
    print("passée:", valeur_dans_entry)
    if valeur_dans_entry == "a":
        return True
    fenetre.bell()
    return False

# validation desactivée avec les StringVar
# on peut enregistrer la valeur dans une globale
# ou utiliser les callbacks pour la modifiction de la Variable sinon...
# key : appelle la validation à chaque appuie de touche
# %P : la valeur que l'on aurait eue si c'était valide
tcl_function_validate = (fenetre.register(validate), "%P")
entree = Entry(
    fenetre, width=30, validate="key",
    validatecommand=tcl_function_validate
)
entree.pack()

fenetre.mainloop()
```


Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Widget quizz

Quels types de widgets pour quelle interaction ?

- entrer un numéro de téléphone [45]
- sélectionner un volume [46]
- créer un mot de passe [47]
- choisir dans une liste d'actions [48]
- choisir un login / mot de passe [49]

45. https://qz.com/679782/programmers-imagine-the-most-ridiculous-ways-to-input-a-phone-number/
46. https://uxdesign.cc/the-worst-volume-control-ui-in-the-world-60713dc86950
47. https://www.reddit.com/r/ProgrammerHumor/comments/904mko/password_input_with_extra_security/
48. https://www.extremetech.com/extreme/262166-hawaiis-missile-scare-driven-terrible-ui-fcc-launches-investigation
49. https://www.reddit.com/r/ProgrammerHumor/comments/8r9xua/so_ive_heard_we_are_now_making_logins_right/

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Barre de menu

```python
from tkinter import *

def ma_fonction():
    print('coucou', bv.get(), rv.get())

fenetre = Tk()
menubar = Menu(fenetre)

bv = BooleanVar(fenetre)
rv = StringVar(fenetre)

menu1 = Menu(menubar, tearoff=0)
menu1.add_command(label="Nouveau", command=ma_fonction)
menu1.add_checkbutton(
    label="Autosave", variable=bv, command=ma_fonction)
menubar.add_cascade(label="Fichier", menu=menu1)

menu2 = Menu(menubar, tearoff=0)
menu2.add_radiobutton(label='rouge', variable=rv, value="(1, 0, 0)")
menu2.add_radiobutton(label='vert', variable=rv, value="(0, 1, 0)")
menubar.add_cascade(label="Couleurs", menu=menu2)
menu1.add_cascade(label="Couleurs", menu=menu2) # sous menu

fenetre.config(menu=menubar)
fenetre.mainloop()
```

Matthieu Falce
Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Contexte
IHM
Programmation événementielle
MVC
Conteneurs
Widgets
Variables de contrôle
Menu
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Barre de menu

- `add_command` : ajoute un élément cliquable à une colonne de menu
- `add_checkbutton` : ajoute une case à cocher à une colonne de menu
- `add_radiobutton` : ajoute un radio à une colonne de menu
- `add_cascade` : ajoute une colonne au menu global

Structure du code [52]

- gros codes → encapsulation dans des classes [50]
- soit classe normale / soit widget custom
  - pour une classe normale on passe le widget parent
  - si on hérite de `Tk.frame` / de `Tk` on créée un widget [51]
  - permet une réutilisation facile dans d'autres projets

50. https://softwareengineering.stackexchange.com/questions/213935/why-use-classes-when-programming-a-tkinter-gui-in-python
51. https://stackoverflow.com/questions/7300072/inheriting-from-frame-or-not-in-a-tkinter-application
52. https://stackoverflow.com/questions/17466561/best-way-to-structure-a-tkinter-application/17470842

Approche orientée objet

```python
# source
# https://www.pythontutorial.net/tkinter/tkinter-object-oriented-window/

import tkinter as tk
from tkinter import ttk
from tkinter.messagebox import showinfo


class App(tk.Tk):
    def __init__(self):
        super().__init__()

        # configure the root window
        self.title("My Awesome App")
        self.geometry("300x50")

        # label
        self.label = ttk.Label(self, text="Hello, Tkinter!")
        self.label.pack()

        # button
        self.button = ttk.Button(self, text="Click Me")
        self.button["command"] = self.button_clicked
        self.button.pack()

    def button_clicked(self):
        showinfo(title="Information", message="Hello, Tkinter!")


if __name__ == "__main__":
    app = App()
    app.mainloop()
```


Approche orientée objet

```python
# source
# https://stackoverflow.com/questions/17466561/

import tkinter as tk

class MainApplication(tk.Frame):
    def __init__(self, parent, *args, **kwargs):
        tk.Frame.__init__(self, parent, *args, **kwargs)
        self.parent = parent

        <create the rest of your GUI here>

if __name__ == "__main__":
    root = tk.Tk()
    MainApplication(root).pack(side="top", fill="both", expand=True)
    root.mainloop()
```

Layout Managers

2 algorithmes de layout :

- pack
  - placement des éléments en fonction des autres
  - le plus simple
- grid
  - placement des éléments sur une grille
  - le plus puissant

Options :

- expand
- fill
- padding : ipadx / ipady / padx / pady

Packing

```
"""
Placement du widget Listbox utilisant toute la fenêtre.
"""

from tkinter import *

root = Tk()

listbox = Listbox(root)
listbox.pack(fill=BOTH, expand=1)

for i in range(20):
    listbox.insert(END, str(i))

mainloop()
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
QT
Conclusion
Code natif

Packing

```
"""
Les widgets sont placés les uns sous les autres
et occupent toute la largeur (en X).
"""

from tkinter import *

root = Tk()

w = Label(root, text="Red", bg="red", fg="white")
w.pack(fill=X)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(fill=X)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(fill=X)

mainloop()
```


Packing

```
"""
Placement des widgets les uns à la gauche des autres
"""

from tkinter import *

root = Tk()

w = Label(root, text="Bleu", bg="blue", fg="white")
w.pack(side=LEFT)
w = Label(root, text="Blanc", bg="white", fg="black")
w.pack(side=LEFT)
w = Label(root, text="Rouge", bg="red", fg="white")
w.pack(side=LEFT)

mainloop()
```

Grid layout

```
""" Utilisation du grid layout pour construire une
interface plus complexe.
"""

from tkinter import *

fen1 = Tk()

# création de widgets 'Label' et 'Entry' :
txt1 = Label(fen1, text="Premier champ :")
txt2 = Label(fen1, text="Second :")
txt3 = Label(fen1, text="Troisième :")
entr1 = Entry(fen1)
entr2 = Entry(fen1)
entr3 = Entry(fen1)

# création d'un widget 'Canvas' contenant une image bitmap :
can1 = Canvas(fen1, width=160, height=160, bg="white")
photo = PhotoImage(file="ptichat.png")
item = can1.create_image(80, 80, image=photo)

# Mise en page à l'aide de la méthode 'grid' :
txt1.grid(row=1, sticky=E)
txt2.grid(row=2, sticky=E)
txt3.grid(row=3, sticky=E)
entr1.grid(row=1, column=2)
entr2.grid(row=2, column=2)
entr3.grid(row=3, column=2)
can1.grid(row=1, column=3, rowspan=3, padx=10, pady=5)

# démarrage :
fen1.mainloop()
```


Gestion des événéments

Plusieurs façons de réagir aux événements

- `command` : appelle un fonction quand on clic / interagit sur un widget
- `bind` : relie une fonction à un événement particulier
- `trace` : appelle une fonction quand on change une `*Var`
- `after` : exécute une fonction après N millisecondes

Gestion des événements

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

command

La plupart des widgets ont une méthode command

```
from tkinter import *

def on_click():
    print("clic")

fenetre = Tk()

bouton = Button(fenetre, text="clic", command=on_click)
bouton.pack()


fenetre.mainloop()
```

Gestion des événements

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

command

Comment passer des paramètres à la fonction ?

```
from tkinter import *

def on_click(bouton_id):
    print("clic", bouton_id)

fenetre = Tk()

bouton1 = Button(fenetre, text="clic", command=lambda: on_click(1))
bouton1.pack()

bouton2 = Button(fenetre, text="clic 2", command=lambda: on_click(2))
bouton2.pack()


fenetre.mainloop()
```

Gestion des événements

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

bind

```
from tkinter import *

fenetre = Tk()

def clavier(event):
    touche = event.keysym
    print(touche)

def mouvement(event):
    pos = event.x, event.y
    print(pos, event.widget)

canvas = Canvas(fenetre, width=500, height=500)
label = Label(fenetre, text="Survolez moi", height=10)

canvas.bind("<B1-Motion>", mouvement)
label.bind("<Motion>", mouvement)
fenetre.bind("<Key>", clavier)

canvas.pack()
label.pack()

fenetre.mainloop()
```

Gestion des événements

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

bind

L'objet event [53]

- passé aux fonctions bindées
- toujours les même champs, quelque soit l'événement
- contient les informations sur l'événement
  - le widget d'appel
  - la position de l'événement
  - la touche pressée
  - ...

53. http://tkinter.fdex.eu/doc/event.html

Gestion des événements

`bind`

Liste des événements que l'on peut binder :

- `<Button-1>` : Click gauche
- `<Button-2>` : Click milieu
- `<Button-3>` : Click droit
- `<Double-Button-1>` : Double click droit
- `<Double-Button-2>` : Double click gauche
- `<KeyPress>` : Pression sur une touche
- `<KeyPress-a>` : Pression sur la touche A (minuscule)
- `<Return>` : Pression sur la touche entrée
- `<Escape>` : Touche Echap

Gestion des événements

`bind`

- `<Up>` : Pression sur la flèche directionnelle haut
- `<Down>` : Pression sur la flèche directionnelle bas
- `<ButtonRelease>` : Lorsque qu'on relâche le click
- `<Motion>` : Mouvement de la souris
- `<B1-Motion>` : Mouvement de la souris avec click gauche
- `<Enter>` : Entrée du curseur dans un widget
- `<Leave>` : Sortie du curseur dans un widget
- `<Configure>` : Redimensionnement de la fenêtre
- `<Map>` `<Unmap>` : Ouverture et iconification de la fenêtre
- `<MouseWheel>` : Utilisation de la roulette

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

Gestion des événéments

trace

```
from tkinter import *


def mise_a_jour_valeur(*args):
    print(value.get())

fenetre = Tk()

value = StringVar()
value.set("Valeur")
entree = Entry(fenetre, textvariable=value)
entree.pack()

# on peut choisir d'avoir des infos
# quand la variable est lue ("r") / écrite ("w")
value.trace("w", mise_a_jour_valeur)


fenetre.mainloop()
```

wm [53]

Permet de modifier le comportement et l'apparence de la fenêtre. Dépend du gestionnaire de fenêtre (Window Manager) de l'OS ⇒ options non multiplateforme

```
# source : https://stackoverflow.com/questions/33286544/
from tkinter import *
frame = Tk()

# Remove shadow & drag bar. Note: Must be used before
#  wm calls otherwise these will be removed.
frame.overrideredirect(1)

# Always keep window on top of others
# appel aux attibuts en Tk
frame.call("wm", "attributes", ".", "-topmost", "true")
# appel à l'attribut  objet
frame.topmost = True

# Set offset from top-left corner of screen as well as size
frame.geometry("100x100+500+500")

# Fullscreen mode
frame.call("wm", "attributes", ".", "-fullscreen", "true")

# Window Opacity 0.0-1.0
frame.call("wm", "attributes", ".", "-alpha", "0.9")

frame.mainloop()
```

53. https://wiki.tcl.tk/9457

Applications Multifenêtre

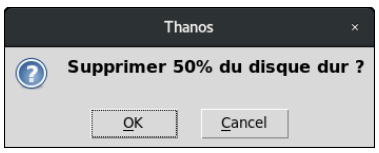
- choix d'un fichier / dossier
- réponse à une question
- formulaire supplémentaire pour finir une action
- "simplifier" la présentation

Message / dialogues / popup

Interaction ponctuelle avec l'utilisateur.
Poser une question / informer...

- `showinfo`, `showwarning`, `showerror`
- `askquestion`, `askokcancel`, `askyesno`
- `askretrycancel`

```python
from tkinter import messagebox

# la fenêtre principale Tk est crée
# automatiquement si elle n'existe
# pas déjà

val = messagebox.askokcancel(
    "Thanos",
    "Supprimer 50% du disque dur ?"
)
print(val)
```



Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

Message / dialogues / popup

Interaction ponctuelle avec l'utilisateur.

Choisir d'un fichier / dossier [54]

- askopenfilename et askopenfilenames
- asksaveasfile et asksaveasfilename
- askopenfile et askopenfiles
- askdirectory

54.http://tkinter.fdex.eu/doc/popdial.html

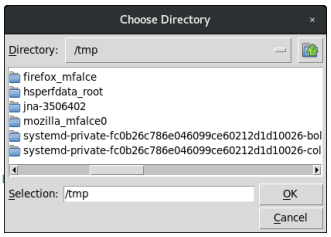
Message / dialogues / popup

Interaction ponctuelle avec l'utilisateur.

Choisir d'un fichier / dossier [54]

- askopenfilename et askopenfilenames
- asksaveasfile et asksaveasfilename
- askopenfile et askopenfiles
- askdirectory

```
from tkinter import filedialog

val = filedialog.askdirectory()
print(type(val), val)  # <class
```



54.http://tkinter.fdex.eu/doc/popdial.html

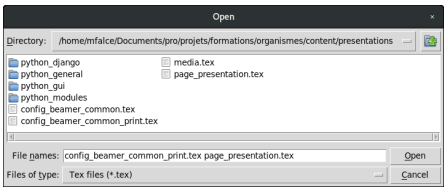
Message / dialogues / popup

Interaction ponctuelle avec l'utilisateur.

Choisir d'un fichier / dossier [54]

- askopenfilename et askopenfilenames
- asksaveasfile et asksaveasfilename
- askopenfile et askopenfiles
- askdirectory

```
from tkinter import filedialog

val = filedialog.askopenfiles(
    filetypes=[
        ("Tex files", "*.tex"),
        ("png files", "*.png"),
        ("All files", "*")
    ]
)
print(type(val), val)
# <class 'list'>
# [
#   <_io.TextIOWrapper name='../config.tex' mode='r' encoding='UTF-8'>,
#   <_io.TextIOWrapper name='../pres.tex' mode='r' encoding='UTF-8'>
# ]
```



54.http://tkinter.fdex.eu/doc/popdial.html

Fenetres secondaires

On utilise `TopLevel` [55] :

```
from tkinter import *

top_levels = []
def on_click():
    n = Toplevel(fenetre)
    t = str(len(top_levels)
    Button(
        master=n, text=t)
    ).pack()
    top_levels.append(n)

fenetre = Tk()

bouton = Button(
    fenetre,
    command=on_click,
    text="Ouvre une fenetre",
)
bouton.pack()

fenetre.mainloop()
```

55.http://effbot.org/tkinterbook/toplevel.htm

Style

TTK (themed Tk) : des widgets avec des styles pour ressembler à des applications natives

Bibliographie / Aller plus loin I

Méthodes communes aux widgets :
`http://tkinter.fdex.eu/doc/uwm.html`
Event loop :

- `https://wiki.tcl.tk/17363`
- `https://stackoverflow.com/questions/29158220/tkinter-understanding-mainloop/29158947`

MVC :

- Article fondateur (smalltalk)
  `http://www.math.sfedu.ru/smalltalk/gui/mvc.pdf`
- `https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur`
- tutoriels MVC en Qt
  - `https://doc.qt.io/archives/qt-4.8/model-view-programming.html`

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
Gestionnaire de fenêtre
Multifenêtre
QT
Conclusion
Code natif

Bibliographie / Aller plus loin II

- https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1902176-larchitecture-mvc-avec-les-widgets-complexes
- https://www.codeguru.com/cpp/cpp/implementing-an-mvc-model-with-the-qt-c-framework.html

- MVC en Tkinter https://codereview.stackexchange.com/questions/163342/applying-model-view-controller-to-tkinter-matplotlib-application

RAD : https://github.com/alejandroautalan/pygubu
Organisation d'un code `Tkinter` :

- https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html

Contexte

> Qt (prononcé officiellement en anglais cute mais couramment prononcé Q.T.) est une API orientée objet et développée en C++, conjointement par The Qt Company et Qt Project. Qt offre des composants d'interface graphique (widgets), d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc. Par certains aspects, elle ressemble à un framework lorsqu'on l'utilise pour concevoir des interfaces graphiques ou que l'on conçoit l'architecture de son application en utilisant les mécanismes des signaux et slots par exemple.
>
> https://fr.wikipedia.org/wiki/Qt

Contexte

- développé en C++ avec des bindings dans de nombreux langages
- utilise fortement l'orienté objet pour décrire une arborescence (entre autres) de widgets
- Qt a un système de licence assez particulier (à considérer pour des applications propriétaires)
- a 2 bindings python : pyside (maintenue par RiverBank Commputing) et pyqt (maintenu par Nokia), la différence tient principalement à la licence des bibliothèques (autres différences ici : https://www.pythonguis.com/faq/pyqt5-vs-pyside2/)
- Qt utilise un mécanisme particulier pour faire communiquer ses éléments : les signaux et les slots
- Qt permet d'avoir des outils de prototypage rapide pour construire facilement des interfaces graphiques visuellement

Qt5 / Qt6

Une nouvelle version majeure de Qt est sortie en 2021 : Qt6. Il y a des différences entre Qt5 et Qt6 et donc également dans les versions Python. Cette page liste les modifications à effectuer : https://www.pythonguis.com/faq/pyqt5-vs-pyqt6/.

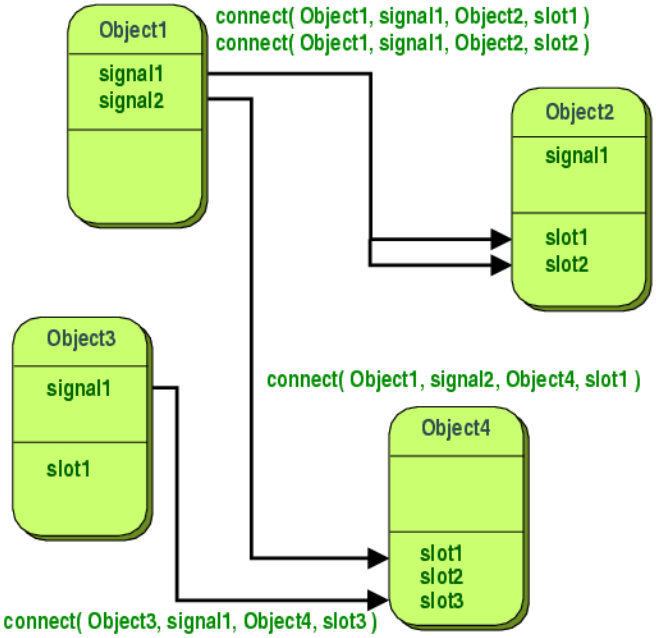
Par quoi commencer ?

- Les ressources sont plus nombreuses avec Qt5 pour l'instant.
- je recommande de commencer avec la version Qt5, puis, une fois habitué, passer à Qt6 en faisant les changements.

Signaux et slots

- mécanisme central de QT et absent des autres frameworks graphiques
- système de communication entre les objets
- permet d'organiser proprement un ensemble de `callbacks`
- un `signal` est émis pour signaler un événement, un `slot` est la fonction qui est appelée lors de cet événement (il peut y en avoir plusieurs), le mécanisme de lien entre les 2 est la `connexion`
- les objets Qt viennent avec leurs propres signaux / slots, mais on peut en rajouter

Signaux et slots



Mécanisme de communication entre objets (source : https://doc.qt.io/qt-5/signalsandslots.html)

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
QT
Contexte
Exemples
Conclusion
Code natif

Exemples de code

Des ressources peuvent se trouver ici :

- https://github.com/pyqt/examples
- https://www.pythonguis.com/tutorials/pyqt-signals-slots-events/

Exemples de code

```python
# Source : https://www.pythonguis.com/tutorials/pyqt-signals-slots-events/

import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton


class MainWindow(QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()

        self.setWindowTitle("My App")


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```

Matthieu Falce

Vue d'ensemble
Langage Python
Programmation Orientée objet (POO)
Bonnes pratiques
Bibliothèque standard
Interface graphiques
Tkinter
Structure du code
Placement widgets
Evénements
QT
Contexte
Exemples
Conclusion
Code natif

Exemples de code

```python
# source: https://www.pythonguis.com/tutorials/pyqt-signals-slots-events/
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QPushButton


class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()

        self.button_is_checked = True

        self.setWindowTitle("My App")

        button = QPushButton("Press Me!")
        button.setCheckable(True)
        button.clicked.connect(self.the_button_was_toggled)
        button.setChecked(self.button_is_checked)

        self.setCentralWidget(button)

    def the_button_was_toggled(self, checked):
        self.button_is_checked = checked

        print(self.button_is_checked)


app = QApplication(sys.argv)

window = MainWindow()
window.show()

app.exec()
```


Elements à considérer

- il n'y a pas de `framework` qui soit systématiquement à privilégier
- cela dépend des conditions d'utilisation / complexité de l'application
- est-il pertinent de réaliser une application
  - lourde (accessible depuis une application) / web (accessible depuis un navigateur)
  - native (spécifique à un OS) ou multi-plateforme (généraliste mais peut être moins adapté)

Comparaison

	Qt	Tkinter
Avantages	* Multi-plateforme / widgets spécifiques * Flexible / permet d'organiser le code * Qt creator (création d'interfaces en glissé déposé) * Fourni un écosystème d'outils (connexion aux bases de données, threads, fichiers...) * Nombreux widgets * Beaucoup de ressources en ligne	* Disponible de base en python sans rien installer * Facile à prendre en main
Inconvénients	* Complexe (POO, il faut chercher la documentation pour le C++) * Mécanisme de licence compliqué quand on ne fait pas de l'open source * Doit être installé	* Pas de widgets avancés (un tableau par exemple) * Intégration au style de l'OS compliqué * Gestion de la complexité compliquée

Avantage / inconvénients des solutions (source : https://dev.to/amigosmaker/python-gui-pyqt-vs-tkinter-5hdd)

Listing

Il existe d'autre framework d'interfaces graphiques

- GTK
- wxPython
- Kivy

Il existe également des bibliothèques permettant d'abstraire le choix du framework qui peuvent être intéressantes : `https://pysimplegui.readthedocs.io/en/latest/` (tk, qt, wxpython et web)