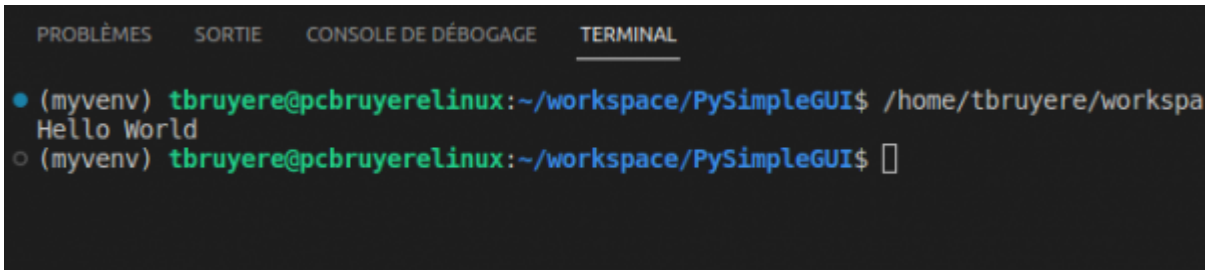
Interface graphique (GUI) sous python.

A la découverte des interfaces graphiques (GUI) pour mon application python.

La version console

```
print('Hello World')
```



La version graphique

```
import tkinter as tk

root=tk.Tk()

a = tk.Label(root, text="Hello, world!")

a.pack()

root.mainloop()
```

Ajout d'une librairie graphique

Pour créer une interface graphique, nous avons dû ajouter une librairie.

```
import tkinter as tk
```

Le choix de la librairie va définir les possibilités et le rendu de mon application.

Les librairies graphique pour python

- tkinter
- wxPython et wxWidgets
- GTK
- QT
- Kivy
- PySimpleGUI

tkinter

Tkinter (de l'anglais Tool kit interface) est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. Elle vient d'une adaptation de la bibliothèque graphique Tk écrite pour Tcl. wikipedia

wxPython

wxPython est une implémentation libre en Python de l'interface de programmation wxWidgets. Cette bibliothèque Python est utilisée pour la création d'interfaces graphiques, et est l'alternative de Tkinter la plus utilisée. wikipedia

GTK

GTK (The GIMP Toolkit, anciennement GTK+2) est un ensemble de bibliothèques logicielles, c'est-à-dire un ensemble de fonctions permettant de réaliser des interfaces graphiques. Cette bibliothèque a été développée originellement pour les besoins du logiciel de traitement d'images GIMP. wikipedia

QT

une API orientée objet et développée en C++, conjointement par The Qt Company et Qt Project. Qt offre des composants d'interface graphique wikipedia

Kivy

Kivy est une bibliothèque libre et open source pour Python, utile pour créer des applications tactiles pourvues d'une interface utilisateur naturelle. Cette bibliothèque fonctionne sur Android, iOS, GNU/Linux, OS X et Windows. Elle est distribuée gratuitement et sous licence MIT. wikipedia

Cas particulier de PySimpleGUI

PySimpleGUI vise à simplifier le développement d'applications GUI pour Python. Il ne réinvente pas la roue mais fournit une enveloppe autour d'autres frameworks existants tels que Tkinter, Qt (PySide 2), WxPython

Utilisation de Tkinter

Création d'un environnement virtuel python

Un environnement virtuel python permet d'isoler les librairies pour mon projet.

```
apt install virtualenv
virtualenv myvenv
source ./myvenv/bin/activate
```

Installation de la librairie tkinter

Vérifier que l'environnement virtuel est actif en regardant le prompt de ma console.

```
(myvenv) tbruyere@pcbruyerelinux:~/workspace/PySimpleGUI$
```

Installer la librairie

```
pip install tkinter
```

Création d'un projet python

Importer le module

```
import tkinter as tk
```

Créer un objet de base via la méthode Tk()

```
root=tk.Tk()
```

Créer un "label"

```
a = tk.Label(root, text="Hello, world!")
```

Générer l'interface

```
a.pack()
```

Démarrer l'interface

```
root.mainloop()
```

Exemple avec wxPython

et si on demandait à chatgpt : créée moi un hello world en python avec interface graphique wxpython

Résultat

```
import wx

```

```
class MyFrame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, None, wx.ID_ANY, "Hello World")
        panel = wx.Panel(self, wx.ID_ANY)
        text = wx.StaticText(panel, wx.ID_ANY, "Hello World!")
        sizer = wx.BoxSizer(wx.VERTICAL)
        sizer.Add(text, 0, wx.ALL | wx.CENTER, 5)
        panel.SetSizer(sizer)

if __name__ == "__main__":
    app = wx.App(False)
    frame = MyFrame()
    frame.Show(True)
    app.MainLoop()
```

exemple en gtk

```
import gi
gi.require_version('Gtk', '3.0')
from gi.repository import Gtk

class MyWindow(Gtk.Window):
    def __init__(self):
        Gtk.Window.__init__(self, title="Hello World")
        label = Gtk.Label("Hello World!")
        self.add(label)

if __name__ == "__main__":
    win = MyWindow()
    win.connect("destroy", Gtk.main_quit)
    win.show_all()
    Gtk.main()
```

exemple en QT

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Hello World')
        self.setGeometry(100, 100, 250, 100)
        label = QLabel('Hello World!', self)
        label.move(90, 40)
        self.show()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
```

```
    sys.exit(app.exec_())
```

exemple en Kivy

```
import kivy
kivy.require('1.11.1')
from kivy.app import App
from kivy.uix.label import Label

class HelloWorldApp(App):
    def build(self):
        return Label(text='Hello World')

if __name__ == '__main__':
    HelloWorldApp().run()
```

La gestion de la grille (grid)

Cette stratégie permet de positionner chacun de vos widgets dans une ou plusieurs cellules d'une grille. La grille est organisée en lignes et en colonnes : vous pouvez, bien entendu, contrôler le nombre de lignes et de colonnes. Il est aussi à noter qu'il est possible qu'un widget occupe plusieurs cellules de la grille. Voici une capture d'écran montrant une grille un peu sophistiquée.

GTK et les conteneurs

GTK+ organise les widgets de manière hiérarchique, en utilisant des conteneurs. Ils sont invisibles pour l'utilisateur final et sont insérés dans une fenêtre ou placés les uns dans les autres pour mettre en page les composants. Il existe deux types de conteneurs : les conteneurs à enfant unique, qui sont tous des descendants de Gtk.Bin, et les conteneurs à enfants multiples, qui sont des descendants de Gtk.Container. Les plus couramment utilisés sont les boîtes verticales ou horizontales (Gtk.Box) et les grilles (Gtk.Grid).

https://python-gtk-3-tutorial.readthedocs.io/en/latest/layout.html

Exemple GTK

```
import gi

gi.require_version("Gtk", "3.0")
from gi.repository import Gtk


class MyWindow(Gtk.Window):
    def __init__(self):
        super().__init__(title="Hello World")

        self.box = Gtk.Box(spacing=6)
        self.add(self.box)
```

```
        self.button1 = Gtk.Button(label="Hello")
        self.button1.connect("clicked", self.on_button1_clicked)
        self.box.pack_start(self.button1, True, True, 0)

        self.button2 = Gtk.Button(label="Goodbye")
        self.button2.connect("clicked", self.on_button2_clicked)
        self.box.pack_start(self.button2, True, True, 0)

    def on_button1_clicked(self, widget):
        print("Hello")

    def on_button2_clicked(self, widget):
        print("Goodbye")


win = MyWindow()
win.connect("destroy", Gtk.main_quit)
win.show_all()
Gtk.main()
```

Lien

- https://gayerie.dev/docs/python/python3/tkinter.html#un-premier-programme
- https://kivy.org/
- https://github.com/kivymd/KivyMD
- https://github.com/ParthJadhav/Tkinter-Designer
- http://pascal.ortiz.free.fr/contents/tkinter/tkinter/tkinter.pdf
- https://morioh.com/p/b03ffc6e35f6
- http://livre21.com/LIVREF/F6/F006059.pdf
- https://koor.fr/Python/Tutoriel_Tkinter/tkinter_layout_grid.wp
- https://www.pythonguis.com/tutorials/create-ui-with-tkinter-grid-layout-manager/
- https://zestedesavoir.com/tutoriels/870/des-interfaces-graphiques-en-python-et-gtk/1446_decouverte/5775_le-positionnement-grace-aux-layouts/
- http://hmalherbe.fr/thalesm/gestclasse/documents/Premiere_NSI/Projets/Calculatrice_programmeur/calculatrice_programmeur.html

From:
https://www.loligrub.be/wiki/ - **LoLiGrUB**

Permanent link:
https://www.loligrub.be/wiki/atelier20230316-python-gui?rev=1679078627

Last update: **2023/03/17 18:43**