wxPython : Création de la fenêtre de l'application

Importation du module

```
#!/usr/bin/python3
# -*- coding: iso-8859-1 -*-
# verification de la disponibilite de wxPython
try:
    import wx
except ImportError:
    print("Le module wxPython est nécessaire pour cette application.")
    raise
```

Créer la classe

wx.Frame est la classe de base pour les fenêtres standards. La classe de l'application va hériter de cette classe mère WX.Frame.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
```

- Ajout du constructeur

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   # le constructeur de la classe fenetre hérite de wx.Frame,
   # il faut appeler le constructeur de la classe mère wx.Frame en passant
l'instance en paramètre :
   # super(wx.Frame, self).__init__().
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      # ancienne manière de procéder
      # wx.Frame.__init__(self, parent, id, title,pos, size)
```

Une interface graphique est une **hiérarchie d'objets** :

- Un bouton peut être **contenu** dans un panneau qui est contenu dans un onglet qui est contenu dans une fenêtre, etc.
- chaque élément de l'interface graphique (widget) possède un **parent** (le widget qui le contient, généralement).
- donc chaque **constructeur a un paramètre parent**.

La **référence** du parent est utile quand il faut **montrer/masquer** des groupes de widgets, les redessiner à l'écran ou tout simplement les détruire quand la fenêtre est fermée. Une bonne habitude consiste à mémoriser la référence du parent.

Le paramètre **id** est un **identifiant** unique du widget.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      self.parent = parent
```

Définir les composants de l'interface

Pour plus de clarté dans le code, on peut mettre dans une fonction spécifique initialise() la création des widgets et appeler cette fonction dans le constructeur.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      self.parent = parent
      self.initialise()

   def initialise(self):
      # forcer l'apparition de la fenetre
      self.Show(True)
```

Création du programme principal qui va lancer la création de la fenêtre

- il faut **instancier** un objet wx.App() avant de créer des éléments graphiques.
- il faut ensuite créer une **instance de la classe Fenetre** ; on ne précise pas de parent (None) car c'est le premier élément graphique que nous créons.
- identifiant sera wx.ID_ANY pour laisser wxPython le définir,
- on précise le titre la position et la dimension de la fenêtre.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      self.parent = parent
      self.initialise()

   def initialise(self):
```

```
        # forcer l'apparition de la fenetre
        self.Show(True)

if __name__ == "__main__":
    app = wx.App()
    fenetre_1 = Fenetre(None,wx.ID_ANY,'Première application', (25,25),
(400,100))
```

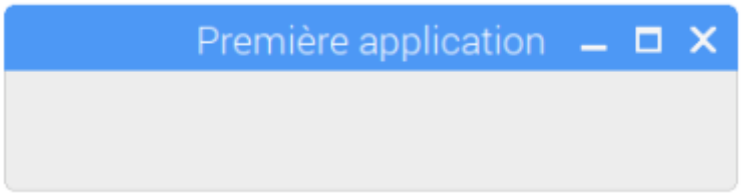
- création de la boucle d'événement sans fin qui va attendre les événements utilisateur (clic sur un bouton par exemple). C'est le principe de la **programmation événementielle**.

```
import wx
# création de la classe de l'application
class Fenetre(wx.Frame):
   def __init__(self, parent, id, title, pos, size) :
      super(wx.Frame, self).__init__(parent, id, title,pos, size)
      self.parent = parent
      self.initialise()

   def initialise(self):
      # forcer l'apparition de la fenetre
      self.Show(True)

if __name__ == "__main__":
    app = wx.App()
    fenetre_1 = Fenetre(None,wx.ID_ANY,'Première application', (25,25),
(400,100))
    app.MainLoop()
```

Au lancement du programme on obtient une fenêtre vide :



Retour au menu Interface graphique avec wxPython

- Interface graphique avec WxPython