Programmer avec Python

Cours

Séance n° 6	**Les interfaces graphiques**

Vous savez à présent utiliser les concepts de la programmation orientée objet ainsi que les modules. Nous allons réutiliser ces connaissances pour les appliquer à la conception d'interfaces graphiques pour nos programmes. Jusqu'ici, chaque programme que vous réalisiez était exécuté dans la console. Bien sûr en 2022, ce mode de fonctionnement n'est plus très répandu ! Python propose de nombreux modules pour créer des interfaces graphiques qui permettront de développer pour vos programmes des fenêtres, boutons, images. . .. Celui que nous verrons lors des deux dernières séances se nomme `PySimpleGUI`.

1 Première fenêtre avec PySimpleGUI

1.1 L'objet Window

L'objet correspondant à une fenêtre dans PySimpleGui se nomme `Window`. Lorsqu'on crée une fenêtre, on crée ainsi un objet Window qui comportera obligatoirement un titre (celui qui s'affichera dans la barre de la fenêtre) et le plus souvent une liste des éléments à inclure dans la fenêtre.

1.2 Les éléments

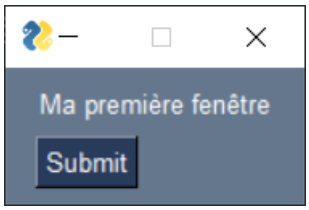
Les différents composants graphiques d'une fenêtre comme une zone de texte, un bouton, un menu, une liste ou encore un curseur sont appelés éléments dans PySimpleGUI. Vous pouvez trouver la liste complète ce qui est possible ici : https://www.pysimplegui.org/en/latest/call%20reference/#the-elements.

Tous ces éléments peuvent être utilisé par un utilisateur. L'action d'un utilisateur sur un élément d'interface est appelé un évènement. A chaque évènement est attaché une valeur. Par exemple, lorsque l'utilisateur saisit un texte, il se produit un évènement et la (ou les !) valeur qui y est attachée est le texte saisi par l'utilisateur. On récupère ses évènements et leurs valeurs grâce à la méthode `read` de l'objet `window`.

Voici le code d'une toute première fenêtre avec PySimpleGui :

```
import PySimpleGUI as sg      # On importe PySimpleGui, qu'on
   nommera sg

# On crée un élément de type zone de texte (sg.Text),
# et un bouton submit.
# Tous les éléments sont rangés dans une liste !
layout = [[sg.Text('Ma première fenêtre')], [sg.Submit()]]

# On crée une fenêtre avec un titre et notre liste d'éléments à
   placer dans la fenêtre
window = sg.Window('Premier essai', layout)

# Les évènements et leurs valeurs correspondent aux actions de l'
   utilisateur
# On les récupère grâce à la méthode read
event, values = window.read()

# On ferme toujours la fenêtre à la fin
window.close()
```

On obtient ceci :



Cette première fenêtre est encore simple. Voyons comment utiliser les évènements, par exemple en lisant un texte saisi par l'utilisateur.

1.3 Les évènements

Un évènement peut être produit lors d'un clic sur un bouton, un texte saisi, un item choisi dans une liste ou si la fenêtre est fermée par exemple. La méthode `read` de l'objet `window` retourne alors un tuple constitué de l'évènement et d'un dictionnaire contenant les valeurs attachées. Ces valeurs, comme toute valeur de dictionnaire, sont définies par des clés. Si vous n'avez pas défini de clé particulière, une clé par défaut sera utilisée.

1.3.1 Gérer un seul évènement dans une fenêtre

Voici un exemple pour récupérer un texte saisi par l'utilisateur et l'afficher dans un popup.

```
import PySimpleGUI as sg      # On importe PySimpleGui, qu'on
    nommera sg

# On crée un élément de type zone de texte (sg.Text),
# et un bouton submit.
# Tous les éléments sont rangés dans une liste !
layout = [[sg.Text('Ma première fenêtre')], [sg.Submit()],
          [sg.InputText(key='-INPUT-')]]

# On crée une fenêtre avec un titre et notre liste d'éléments à
    placer dans la fenêtre
window = sg.Window('Premier essai', layout)

# Les évènements et leurs valeurs correspondent aux actions de l'
    utilisateur
# On les récupère grâce à la méthode read
# La fenêtre restera ouverte en attente d'un évènement
event, values = window.read()

# On ferme toujours la fenêtre à la fin
window.close()

# On récupère le texte saisi dans l'élément Text défini par la clé
    INPUT
text_input = values['-INPUT-']
# On l'affiche dans un popup
sg.popup('Tu as écrit : ', text_input)
```

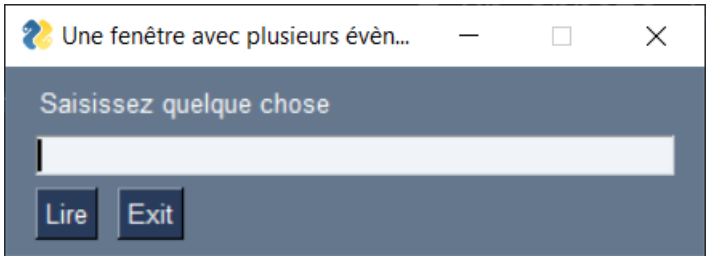
Notez bien que vous ne pouvez interagir avec les éléments de la fenêtre qu'une fois que la méthode read a été appelée. Si d'aventure vous aviez besoin de le faire avant, vous devez impérativement ajouter l'option `finalize=true` à la fenêtre lors de sa création.

1.3.2 Gérer plusieurs évènements dans une fenêtre

L'exemple précédent donne un cas simple où un seul évènement se produit dans notre fenêtre. C'est en réalité rarement le cas. La plupart du temps plusieurs évènements seront produits dans une même fenêtre qui restera ouverte : on clique un bouton, puis un autre, on saisit un texte, on affiche un menu etc. Dans ce cas, nous devons passer par une boucle pour gérer ses multiples évènements, comme ceci :

```
import PySimpleGUI as sg      # On importe PySimpleGUI


layout = [[sg.Text('Saisissez quelque chose')],      # Un champ
   texte
          [sg.Input(key='-INPUT-')],                 # Un champ de
             saisie
          [sg.Button('Lire'), sg.Exit()]]            # 2 boutons,
             un pour lire le champ et pour quitter

window = sg.Window('Une fenêtre avec plusieurs évènements', layout)

# La boucle d'évènements
while True:     # Tant qu'il se produit des évènements
    # On lit les évènements
    event, values = window.read()
    # On les affiche
    print(event, values)
    # Si la fenêtre est fermée ou le bouton Quitter cliqué

    if event == sg.WIN_CLOSED or event == 'Exit':
        break  # On arrête la boucle

# Fermeture de la fenêtre toujours à la fin
window.close()
```

Si vous testez ce code, vous obtiendrez cette fenêtre :



Si vous tapez "Bonjour" dans le champ de saisie, puis cliquez sur le bouton Lire puis Exit, la console vous indiquera au fur et à mesure la valeur des évènements. Avec ces opérations, au moment où on clique sur le bouton Lire et par la suite sur le bouton Exit, le champ de texte avec la clé INPUT contient le texte "bonjour", vous verrez ainsi ceci dans la console s'afficher :

```
Lire {'-INPUT-': 'bonjour'}
Exit {'-INPUT-': 'bonjour'}
```

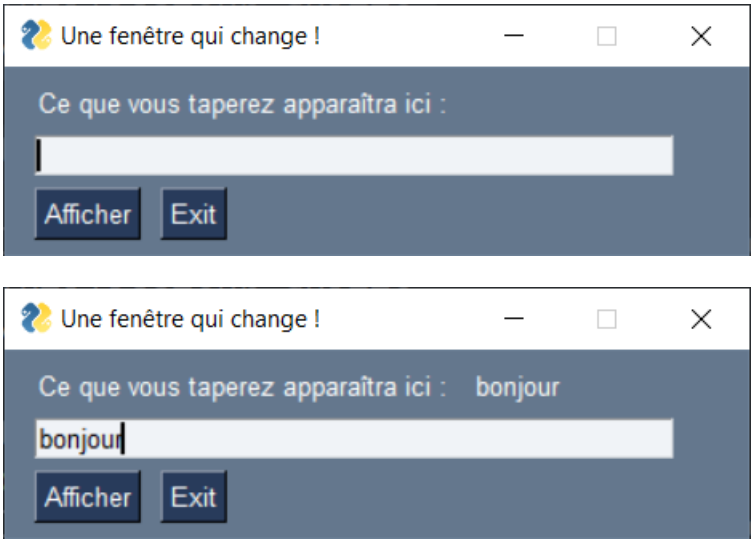
N'hésitez pas à lancer le programme vous-même et à le tester pour bien comprendre.

1.3.3 Modifier une fenêtre en fonction des évènements

Maintenant que vous savez interagir avec votre interface graphique, lire les évènements qui s'y produisent et récupérer des valeurs données par l'utilisateur, vous voudrez mettre à jour votre fenêtre pour en tenir compte. Par exemple, il s'agira de mettre à jour un texte, une image, afficher une nouvelle information... Vous pouvez modifier n'importe quel élément graphique avec la méthode `update`, à la condition d'avoir déjà appelé au moins une fois la méthode `window.read()` de votre objet window. Ensuite, la fenêtre sera mise à jour au prochain appel à `window.read()`, ou plus tôt si vous le souhaitez en utilisant `window.refresh()`. Un exemple :

```
import PySimpleGUI as sg # On importe PySimpleSUI

layout = [[sg.Text('Ce que vous taperez apparaîtra ici :'), sg.Text
   (size=(15,1), key='-OUTPUT-')],
          [sg.Input(key='-INPUT-')],
          [sg.Button('Afficher'), sg.Button('Exit')]]

window = sg.Window('Une fenêtre qui change !', layout)

while True:  # La boucle d'évènements
    # On récupère les évènements dans la fenêtre
    event, values = window.read()
    # On les affiche dans la console
    print(event, values)
    # On vérifie si l'utilisateur quitte
    if event == sg.WIN_CLOSED or event == 'Exit':
        break
    # Si c'est le bouton Afficher qui a été cliqué
    if event == 'Afficher':
        # On affiche le texte saisi dans INPUT dans le champ OUTPUT
        window['-OUTPUT-'].update(values['-INPUT-'])

window.close()
```

Lorsqu'on saisit bonjour dans le champ -INPUT- puisqu'on clique sur le bouton Afficher, la fenêtre met à jour le texte OUTPUT avec la valeur saisie.



2 Et pour aller plus loin

2.1 Changer le thème de la fenêtre

PySimpleGUI propose de multiples thèmes de couleurs pour les fenêtres. Vous pouvez les visualiser en créant un programme Python contenant uniquement ces 2 lignes et le lancer :

```
import PySimpleGUI as sg

sg.preview_all_look_and_feel_themes()
```

Vous obtenez ceci un aperçu comme ceci :

Et pour modifier le thème de votre fenêtre, vous ajoutez cette ligne au tout début de votre programme :

```
sg.theme('Dark Teal 12') # Avec le nom de votre thème préféré si
   vous n'aimez pas le bleu :)
```

2.2 Ordonner les éléments dans la fenêtre

Pour organiser vos éléments, vous utiliserez le layout que nous avons déjà vu. Des éléments qui doivent être placés sur le même axe horizontal seront placés dans la même liste. Chaque nouvel axe sera contenu dans une nouvelle liste. Reprenons l'exemple précédent :

```
import PySimpleGUI as sg # On importe PySimpleSUI

layout = [[sg.Text('Ce que vous taperez apparaîtra ici :'), sg.Text
   (size=(15,1), key='-OUTPUT-')], # Ces deux éléments sont sur la
   même ligne
          [sg.Input(key='-INPUT-')], # Une nouvelle ligne
          [sg.Button('Afficher'), sg.Button('Exit')]] # Une
             nouvelle ligne avec les deux boutons

window = sg.Window('Une fenêtre qui change !', layout)
```

```
while True:  # La boucle d'évènements
    # On récupère les évènements dans la fenêtre
    event, values = window.read()
    # On les affiche dans la console
    # On vérifie si l'utilisateur quitte
    if event == sg.WIN_CLOSED or event == 'Exit':
        break
    # Si c'est le bouton Afficher qui a été cliqué
    if event == 'Afficher':
        # On affiche le texte saisi dans INPUT dans le champ OUTPUT
        window['-OUTPUT-'].update(values['-INPUT-'])

window.close()
```

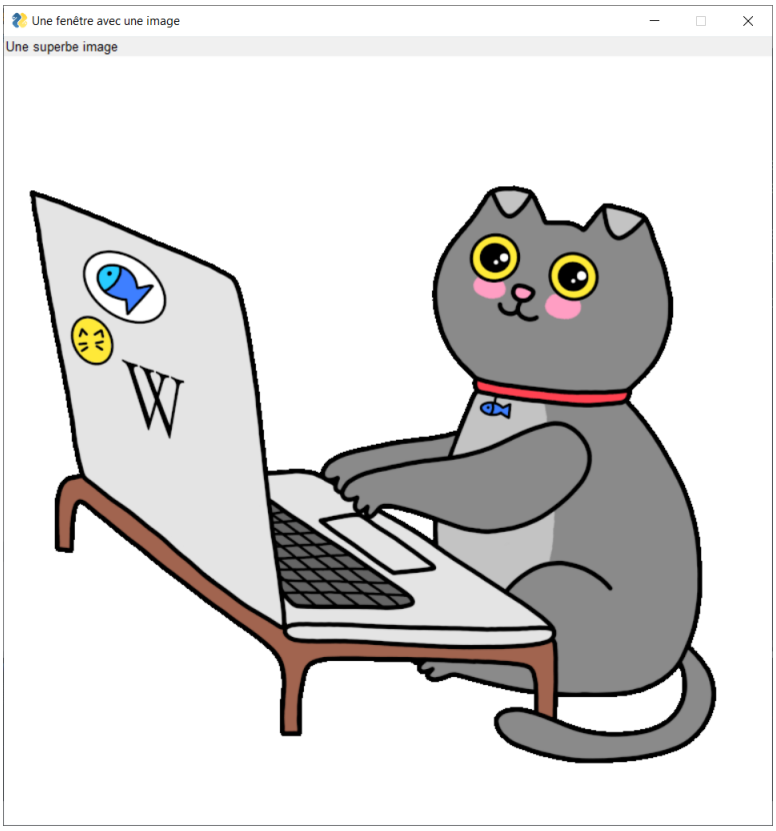
Si on veut que les boutons Afficher et Quitter soient l'un en dessous de l'autre, on modifie le layout comme ceci :

```
import PySimpleGUI as sg # On importe PySimpleSUI

layout = [[sg.Text('Ce que vous taperez apparaîtra ici :'), sg.Text
   (size=(15,1), key='-OUTPUT-')], # Ces deux éléments sont sur la
   même ligne
          [sg.Input(key='-INPUT-')], # Une nouvelle ligne
          [sg.Button('Afficher')], # Une nouvelle ligne avec le
             bouton
          [sg.Button('Exit')]] # Une nouvelle ligne encore

### ...
```

2.3 Afficher une image dans une fenêtre

L'élément à utiliser est l'élément `Image`. Vous indiquez le nom de l'image si l'image est placée dans le même dossier que le programme, le chemin complet sinon.

```
import PySimpleGUI as sg

layout = [ [sg.Text('Une superbe image')],
           [sg.Image(filename="gifcat.gif")]] # L'image est placée
              dans le même dossier que le programme

# On crée la fenêtre en précisant la localisation du layout et les
   marges autour (padding et margins)
window = sg.Window('Une fenêtre avec une image', layout, location
   =(0,0), element_padding=(0,0), margins=(0,0))

while True:             # Event Loop
    event, values = window.read()
    if event == sg.WIN_CLOSED:
        break

window.close()
```



Pour ajouter un évènement à une image, le principe est le même.

```
import PySimpleGUI as sg      # On importe PySimpleGui, qu'on
    nommera sg

# Le layout
layout = [[sg.Image(filename="gifcat.gif", enable_events = True,key
    ="Image")]] # L'image est placée dans le même dossier que le
    programme

# On crée la fenêtre en précisant la localisation du layout et les
    marges autour (padding et margins)
window = sg.Window('Evènement sur une image', layout, location
    =(100,100), element_padding=(0,0), margins=(0,0))

while True:             # Event Loop
    event, values = window.read()
    if event == "Image":
        window['Image'].update(filename="cat.gif") #L'image dispara
            îtra quand on la clique
    if event == sg.WIN_CLOSED:
        break

window.close()
```