Chap. 1 Introduction à la programmation Python Thème 1

Interaction graphique

Sommaire du chapitre 1

L'interaction purement textuelle avec `input` et `print` est rapidement limitée si l'on veut créer des programmes plus interactifs. L'**interaction graphique** consiste à afficher des informations sous forme graphique, comme avec la bibliothèque `turtle`, et à permettre à l'utilisateur d'entrer des données et de lancer des actions à l'aide d'**interacteurs**. Les interacteurs sont par exemple des boutons, des champs de saisie de texte, ou des « tirettes » (« *sliders* » en anglais) pour entrer des nombres.

I Fonctions de rappel

Il est possible en Python de passer une fonction en paramètres à une autre fonction :

```
def coucou() :
    print("Coucou")
def repete(f) : # f est une fonction
    f()         # on appelle f ...
    f()         # ... deux fois

repete(coucou)  # affiche "Coucou" deux fois
```

La plupart des interacteurs sont associés à des fonctions, appelées **fonctions de rappel**. Par exemple, un bouton est associé à une fonction qu'il appelle lorsque l'utilisateur clique le bouton. Pour cela la fonction de rappel est passée en paramètre à la fonction qui crée le bouton :

```
button("Effacer", clear) # appelle clear() lorsqu'on clique le
    bouton
```

II Interacteurs de base

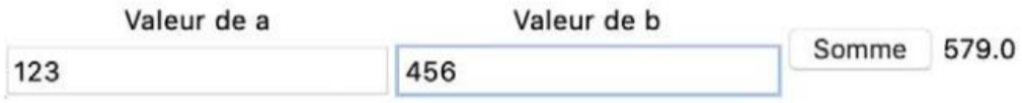
L'exemple suivant calcule la somme de deux nombres. On a besoin de deux champs de texte (fonction `entry`) pour entrer les nombres `a` et `b`, d'un bouton (`button`) pour lancer le calcul et d'une zone de texte (`label`) pour afficher le résultat :

```
1  from nsi_ui import *
2
3  # champs de saisie de texte pour entrer les valeurs de a et b
4  champA = entry("Valeur de a")
5  champB = entry("Valeur de b")
6
7  def ajouter():
8      """ Afficher la somme des valeurs des champs A et B """
9      a = get_float(champA)      # valeur entrée dans le champ A
10     b = get_float(champB)      # valeur entrée dans la champ B
11     set_text(resultat, a + b) # afficher le résultat
12
13 button("Somme", ajouter)       # bouton pour lancer le calcul
14 resultat = label("")           # zone d'affichage du résultat
15 main_loop()                    # lancer l'interface
```

— La fonction `entry` (lignes 4 et 5) crée un champ de texte avec un titre. Elle retourne une valeur qui peut être passée à la fonction `get_float` (lignes 9 et 10) pour obtenir la valeur entrée par l'utilisateur.

— La fonction `button` (ligne 13) crée un bouton qui permet de déclencher une action lorsque l'on clique dessus. Ici, on associe la fonction `ajouter` au bouton intitulé `Somme`. Celle-ci va chercher les valeurs de `a` et `b` depuis les interacteurs `champA` et `champB` (via `get_float`) et calcule le résultat, qu'elle affiche dans l'interacteur `label` grâce à la fonction `set_text`.

— La fonction `label` (ligne 14) crée une zone non interactive pour afficher un texte. Elle retourne une valeur qui peut être passée à la fonction `set_text` (ligne 11) pour afficher de texte. L'interface résultante est la suivante :

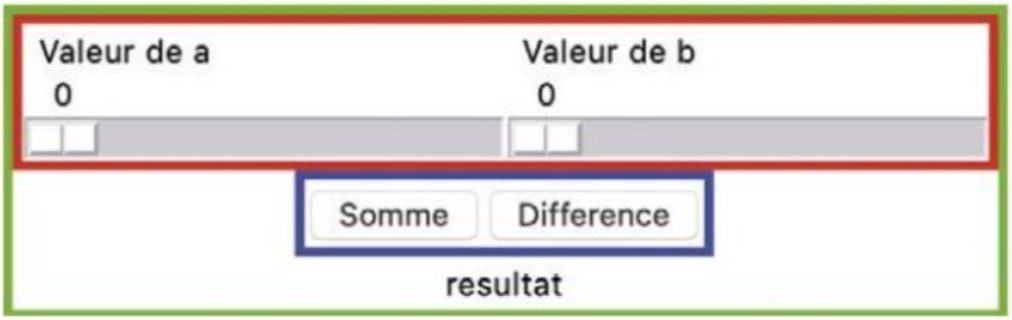


Contrairement à l'interaction textuelle, cette interface permet d'entrer les valeurs de `a` et `b` dans un ordre quelconque et de modifier leurs valeurs. Ce style de programmation est dit **réactif** : au lieu d'imposer à l'utilisateur un ordre strict, le programme réagit à ses actions. Ainsi, la fonction `ajouter` n'est pas appelée directement par le programme, mais lorsque l'utilisateur clique sur le bouton `Somme`.

III Mise en page d'interfaces : les conteneurs

Dans l'exemple précédent, les interacteurs s'affichent les uns à la suite des autres. Pour mieux contrôler leur placement, on utilise des **conteneurs** permettant de grouper leur contenu horizontalement ou verticalement :

L'interface résultante est la suivante. On a encadré chaque conteneur pour mettre en évidence leur placement.



IV Les écouteurs d'évènements

Les interacteurs tels que le bouton ou le champ de saisie de texte réagissent aux actions de l'utilisateur telles que cliquer ou déplacer la souris ou taper au clavier. Il est possible de réagir directement à ces actions, par exemple lorsque l'on clique dans la fenêtre de la tortue.
On appelle **évènement** toute action de l'utilisateur sur un périphérique d'entrée d'un ordinateur : appuyer sur une touche du clavier, la relâcher, déplacer la souris, cliquer l'un de ses boutons, toucher un écran tactile, etc. Pour réagir à ces événement on utilise des fonctions appelées **écouteurs d'événements** qui prennent en paramètre une fonction de rappel, de façon similaire à la fonction `ajouter` pour le bouton `Somme` de l'exemple précédent.
Par exemple, la bibliothèque `turtle` permet de définir trois écouteurs d'évènements pour les actions sur la souris, et un autre pour les actions sur le clavier :

```
1 ttiii
```

Dans l'exemple suivant, l'appui sur la touche `a` fait avancer la tortue, et cliquer-tirer la tortue lui fait suivre le curseur de la souris : la fonction `goto(x, y)` de la bibliothèque `turtle` déplace la tortue à la position `(x, y)`.

```
1 ttiii
```

Interaction graphique : la bibliothèque nsi_ui

Interacteurs

```
button(nom, f)
entry(nom)
slider(nom, min, max)
label(nom)
```

Fonction de rappel

```
def f():
    print("Bonjour")
button("Hello", f)
```

Accès aux valeurs
i est un interacteur

```
get_text(i)
get_int(i)
get_float(i)

set_text(i, chaine)
set_int(i, entier)
set_float(i, flottant)

set_width(i, w)
```

Fonctions générales

```
animate(f, ms)
stop_animate()

main_loop()
```

Conteneurs

```
begin_horizontal()
end_horizontal()
begin_vertical()
end_vertical()
```

Evénements (module turtle)

actionXY est une fonction avec deux paramètres (x, y)

```
onclick(actionXY)
onscreenclick(actionXY)
ondrag(actionXY)
onkey(f, touche)
```