2ème PARTIE

Création d'interfaces graphiques en Python

Mon nom n'est pas Bond

Créez une application graphique interactive avec la bibliothèque guizero.

MAKER

Laura Sach

Laura dirige l'équipe *A Level* de la Fondation RPi chargée des ressources pédagogiques en informatique à destination des étudiants.

@CodeBoom

MAKER

Martin O'Hanlon

Martin crée des cours, des projets et des ressources en ligne au sein de l'équipe *Learning* de la Fondation RPi.

@martinohanlon

Dans la première partie, nous avons appris à créer une fenêtre et à y positionner texte et image. Nous allons ici ajouter à cette fenêtre ce qui fait tout l'intérêt d'une interface graphique : sa capacité à interagir avec l'utilisateur. Pour cela nous créerons un gros bouton rouge générant des noms d'agents secrets.

Nous partirons de ce que vous savez déjà faire, à savoir créer une fenêtre principale (un widget App) contenant du texte (un widget Text). Voici notre code de départ (les lignes commençant par # sont des commentaires, spy = espion) :

```
# Imports ---------------
from guizero import App, Text

# Functions -------------

# App -------------------
app = App("TOP SECRET")

# Widgets ---------------
title = Text(app, "Push the red button to find out your spy name")

# Display ---------------
app.display()
```

Si vous l'exécutez, vous devriez voir la fenêtre reproduite sur la **figure 1**.

Ajout d'un bouton

Ajoutez aux classes App et Text importées en début de code la classe **PushButton** (notez les deux majuscules) qui est la classe permettant de créer des objets de type bouton. Pour en créer un, ajoutez la ligne suivante juste après la ligne instanciant le

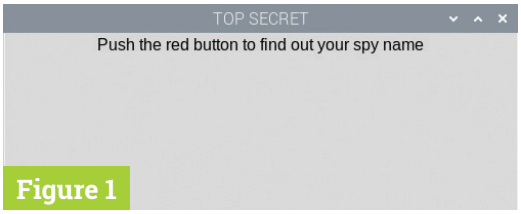
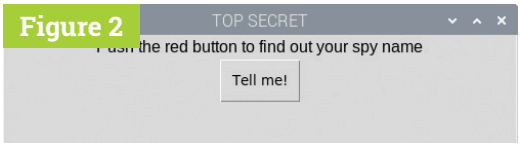


Figure 1

▶ **Figure 1** Affichage du widget **title** dans la fenêtre.

Figure 2



▲ **Figure 2** Création du bouton.

widget Text appelé **title** (listage **spy1.py**) :

```
button = PushButton(app, choose_name, text="Tell me!")
```

L'exécution de ce nouveau code ne crée pas de bouton, mais renvoie une erreur :

```
NameError: name 'choose_name' is not defined
```

De quoi se plaint l'interpréteur Python ? La plupart des éléments graphiques d'une interface peuvent être associés à une commande qui sera lancée lorsque l'utilisateur cliquera sur l'élément en question, un bouton p. ex. Ce que nous dit l'interpréteur est justement : vous m'avez donné un nom de commande (`choose_name`), mais je n'en trouve la définition nulle part ; revenez me voir quand vous aurez corrigé cette erreur. Cette commande est presque toujours le nom d'une fonction à exécuter, et c'est le cas ici : **choose-name** est le nom de la fonction qui générera des noms d'espion.

Définition de la fonction

Écrivons donc le code de la fonction `choose_name` qui sera lancée lorsque nous cliquerons sur le bouton.

Placer dans une même section toutes les fonctions associées aux widgets du programme améliore la lisibilité du code. C'est la raison pour laquelle notre squelette de programme contient une section *Functions*. Ajoutez-y ces deux lignes (listage **spy2.py**) :

```
def choose_name():
    print("Button was pressed")
```

Cette fois-ci le bouton apparaît (**fig. 2**). S'il ne se passe apparemment rien lorsque vous cliquez dessus, c'est que la commande `print()` de la

fonction affiche sa sortie dans la fenêtre de l'interpréteur de commandes, ou shell (**fig. 3**).

Utiliser `print()` dans la fonction représentant la commande associée au bouton est un moyen pratique et rapide de s'assurer que cette fonction est bien appelée. Cette vérification faite, on peut alors remplacer `print()` par le véritable code à déclencher.

Placez le caractère `#` au début de la ligne **print("Button was pressed")**. Lorsque l'interpréteur Python verra ce # au début d'une ligne, il l'ignorera entièrement et passera à la ligne suivante. C'est ce qu'on appelle « commenter » une ligne de code. Pourquoi laisser une ligne qui sera ignorée aussi sûrement que si elle n'existait pas ? Tout simplement pour éviter d'avoir à la retaper si elle devait servir à nouveau, puisque dans ce cas il suffira de supprimer le `#`.

spy1.py

TÉLÉCHARGEZ LE CODE COMPLET :
magpi.cc/guizerocode

Langage : **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Text, PushButton
003.
004.  # Functions -------------
005.
006.  # App -------------------
007.  app = App("TOP SECRET")
008.
009.  # Widgets ---------------
010.  title = Text(app, "Push the red button to find out your spy name")
011.  button = PushButton(app, choose_name, text="Tell me!")
012.
013.  # Display ---------------
014.  app.display()
```

spy2.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.  from guizero import App, Text, PushButton
003.
004.  # Functions -------------
005.
006.  # App -------------------
007.  app = App("TOP SECRET")
008.
009.  # Widgets ---------------
010.  title = Text(app, "Push the red button to find out your spy name")
011.  button = PushButton(app, choose_name, text="Tell me!")
012.
013.  # Display ---------------
014.  app.display()
```

Un gros bouton rouge, disiez-vous ?

Pour l'instant le bouton n'est ni gros ni rouge ! Pour qu'il le devienne il faut utiliser les propriétés définissant la couleur de fond et la taille du texte d'un widget, ici le widget **PushButton**. Référez-vous pour cela au tutoriel précédent ou au code final (lignes 24 et 25).

Notez que certaines versions du système d'exploitation macOS interdisent de modifier la couleur d'un bouton. Vous devriez cependant être à même de modifier la taille du texte.

Figure 3

▲ **Figure 3** Sortie de `print()` dans le shell.

Un vrai nid d'espions

Créez une liste de prénoms sur une nouvelle ligne. Ajoutez autant de prénoms que souhaité en veillant à ce que chacun soit entouré de guillemets droits et séparé du suivant par une virgule. Une suite de caractères composée de lettres, chiffres et/ou signes de ponctuation entourés de guillemets droits est appelée une chaîne. Autrement dit, chaque prénom doit être une chaîne.

```
first_names = ["Barbara", "Woody",
"Tiberius", "Smokey", "Jennifer", "Ruby"]
```

Ajoutez de même une liste de noms de famille :

```
last_names = ["Spindleshanks", "Mysterioso",
"Dungeon", "Catseye", "Darkmeyer",
"Flamingobreath"]
```

Il nous faut maintenant une fonction capable d'extraire de façon aléatoire un prénom et un nom à partir de nos deux listes. Ce genre de fonction est si utile qu'elle fait partie du langage Python, plus précisément du module **random** (aléatoire). Son nom est **choice**, et nous l'importons donc en début de code avec :

```
from random import choice
```

La syntaxe de la fonction `choice()` est simple : on lui fournit le nom d'une liste, et elle renvoie de façon aléatoire un élément de cette liste. Nous pouvons dès lors joindre (les programmeurs disent « concaténer ») le prénom et le nom retournés par `choice()` à l'aide du symbole plus (`+`), sans oublier de les séparer par une espace :

```
spy_name = choice(first_names) + " " +
choice(last_names)
print(spy_name)
```

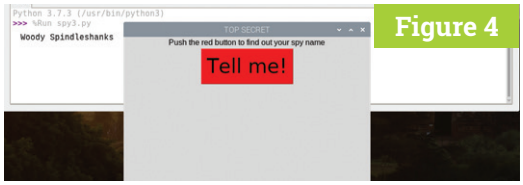


Figure 4

▲ **Figure 4** Affichage du nom de l'espion.

Enregistrez votre code (**spy3.py**) et lancez-le. Appuyez sur le bouton : vous devriez voir s'afficher un nom d'espion dans la fenêtre de l'interpréteur de commandes, là où s'était affichée la sortie de la commande **print("Button was pressed")** (**fig. 4**).

Un espion à la fenêtre

Il serait bien sûr plus agréable de voir le nom de l'espion s'afficher dans la fenêtre principale. Pour cela nous avons besoin d'un autre widget **Text**. Appelons-le **name** et ajoutons-le dans la section *Widgets* :

```
name = Text(app, text="")
```

Ce widget ne doit rien afficher pour l'instant puisque le nom de l'espion n'apparaîtra qu'après un clic sur le bouton. C'est pour cela que nous avons passé au paramètre **text** une chaîne vide (`text=""`).

Commentez la ligne **print(spy_name)** de la fonction `choose_name()` pour qu'elle ne s'affiche plus, puis ajoutez la ligne suivante à la fin de la fonction :

```
name.value = spy_name
```

Lorsque la fonction `choose_name` sera appelée, cette instruction affectera le contenu de la variable `spy_name` (le prénom et le nom de notre espion) à la valeur du widget **name**, c.-à-d. à son paramètre **text**.

Lancez le code final (**03-spy-name-chooser.py**) et cliquez sur le bouton pour voir un nom d'espion aléatoire s'afficher dans la fenêtre (**fig. 5**). Cliquez à nouveau sur le bouton si vous refusez de partir en mission affublé d'un nom pareil. **(VF : Hervé Moreau)**

Figure 5

▲ **Figure 5** La fenêtre créée par le code final.

spy3.py

Langage : **Python 3**

```
# Imports ---------------
from guizero import App, Text, PushButton
from random import choice

# Functions -------------
def choose_name():
    #print("Button was pressed")
    first_names = ["Barbara", "Woody", "Tiberius", "Smokey", "Jennifer", "Ruby"]
    last_names = ["Spindleshanks", "Mysterioso", "Dungeon", "Catseye", "Darkmeyer", "Flamingobreath"]
    spy_name = choice(first_names) + " " + choice(last_names)
    print(spy_name)

# App -------------------
app = App("TOP SECRET")

# Widgets ---------------
title = Text(app, "Push the red button to find out your spy name")
button = PushButton(app, choose_name, text="Tell me!")
button.bg = "red"
button.text_size = 30

# Display ---------------
app.display()
```

03-spy-name-chooser.py

Langage : **Python 3**

```
# Imports ---------------

from guizero import App, Text, PushButton
from random import choice

# Functions -------------

def choose_name():
    #print("Button was pressed")
    first_names = ["Barbara", "Woody", "Tiberius", "Smokey", "Jennifer", "Ruby"]
    last_names = ["Spindleshanks", "Mysterioso", "Dungeon", "Catseye", "Darkmeyer", "Flamingobreath"]
    spy_name = choice(first_names) + " " + choice(last_names)
    #print(spy_name)
    name.value = spy_name

# App -------------------

app = App("TOP SECRET")

# Widgets ---------------

title = Text(app, "Push the red button to find out your spy name")
button = PushButton(app, choose_name, text="Tell me!")
button.bg = "red"
button.text_size = 30
name = Text(app, text="")

# Display ---------------

app.display()
```

Note de l'éditeur : cet article est paru initialement dans le magazine MagPi (le magazine officiel du Raspberry Pi).
La maison d'édition Elektor publie ce magazine en néerlandais, allemand et français (www.magpi.fr).