

Création d'interfaces graphiques en Python

Mon nom n'est pas Bond

Créez une application graphique interactive avec la bibliothèque `guizero`.



Laura Sach

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

@CodeBoom



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 partons 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

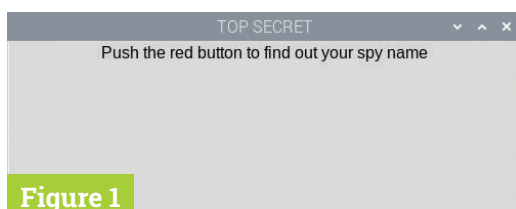
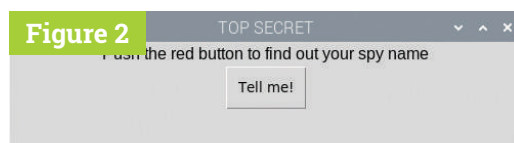


Figure 1

► **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

spy1.py

> Langage : Python 3

TÉLÉCHARGEZ
LE CODE COMPLET :



magpi.cc/guizero

```
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.

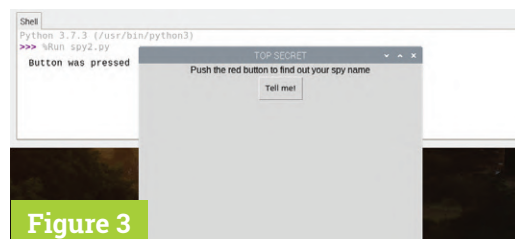


Figure 3

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

« 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 `#`.

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 :

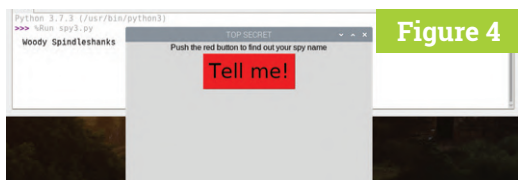
```
last_names = ["Spindleshanks", "Mysterioso",
              "Dungeon", "Catseye", "Darkmeyer",
              "Flamingobreath"]
```

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 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

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003. from random import choice
004.
005. # Functions -----
006. def choose_name():
007.     #print("Button was pressed")
008.     first_names = ["Barbara", "Woody", "Tiberius", "Smokey",
009.     "Jennifer", "Ruby"]
010.     last_names = ["Spindleshanks", "Mysterioso", "Dungeon",
011.     "Catseye", "Darkmeyer", "Flamingobreath"]
012.     spy_name = choice(first_names) + " " + choice(last_names)
013.     print(spy_name)
014.
015. # App -----
016. app = App("TOP SECRET")
017.
018. # Widgets -----
019. title = Text(app, "Push the red button to find out your spy name")
020. button = PushButton(app, choose_name, text="Tell me!")
021. button.bg = "red"
022. button.text_size = 30
023.
024. # Display -----
025. app.display()
```

03-spy-name-chooser.py

► Langage : Python 3

```
001. # Imports -----
002. from guizero import App, Text, PushButton
003. from random import choice
004.
005. # Functions -----
006. def choose_name():
007.     #print("Button was pressed")
008.     first_names = ["Barbara", "Woody", "Tiberius", "Smokey",
009.     "Jennifer", "Ruby"]
010.     last_names = ["Spindleshanks", "Mysterioso", "Dungeon",
011.     "Catseye", "Darkmeyer", "Flamingobreath"]
012.     spy_name = choice(first_names) + " " + choice(last_names)
013.     #print(spy_name)
014.     name.value = spy_name
015.
016. # App -----
017. app = App("TOP SECRET")
018.
019. # Widgets -----
020. title = Text(app, "Push the red button to find out your spy name")
021. button = PushButton(app, choose_name, text="Tell me!")
022. button.bg = "red"
023. button.text_size = 30
024. name = Text(app, text="")
025.
026. # Display -----
027. app.display()
```



productronica
fast forward
the start-up platform
powered by Elektor

gagnants du concours Fast Forward 2021 du salon productronica

Des technologies passionnantes et des solutions d'ingénierie créatives

Clemens Valens (Elektor)

productronica fast forward 2021, la plateforme pour les jeunes pousses soutenue par Elektor, a offert aux entreprises novatrices une occasion unique de présenter leurs technologies à la communauté électronique mondiale. Voici les gagnants !



Il y a eu longtemps une incertitude sur les conditions de déroulement du salon productronica 2021. Serions-nous tous devant nos écrans, ou bien présents sur notre stand ? La seule façon d'avancer était donc de travailler en ligne. Par conséquent, au lieu de se présenter devant un jury et un public en direct, les candidats au Fast Forward Award du salon productronica 2021, originaires de pays tels que la Hongrie, l'Allemagne, Israël et la France, ont dû enregistrer leurs présentations et les soumettre en ligne. Chaque jeune pousse a également fourni des informations supplémentaires sous la forme de plans d'affaires et de dossiers de présentation.

Après avoir regardé les « elevator pitches » pré-enregistrés et étudié toute la documentation, le jury du Fast Forward Award du salon productronica 2021 – qui comprenait Benjamin Klingenberg (NCAB Group, sponsor « platine » de l'événement) et des représentants du labo et de la rédaction d'Elektor – a sélectionné les trois jeunes pousses gagnantes.

Regardez la cérémonie de
remise des prix !



50.000 € de budget marketing chez Elektor

ioTech Group – Basée à Jérusalem (Israël), l'entreprise ioTech développe de nouvelles technologies pour la fabrication additive de masse et l'impression multi-matériaux.

www.elektormagazine.com/news/iotech-group-productronica-fast-forward-2021



25.000 € de budget marketing chez Elektor

Reedu GmbH & Co. KG – L'entreprise Reengineering Education alias Reedu ou re:edu de Münster (Allemagne) est connue pour la gamme senseBox (station de mesures environnementales) et la plateforme openSenseMap.

www.elektormagazine.com/news/reedu-gmbh-co-kg-productronica-fast-forward-2021



15.000 € de budget marketing chez Elektor

Pozi Development Ltd – L'entreprise Pozi de Budapest (Hongrie) propose un système de surveillance de la fabrication et de la logistique, et identifie et élimine les gaspillages tout au long de la chaîne d'approvisionnement.

www.elektormagazine.com/news/pozi-development-ltd-productronica-fast-forward-2021

Les trois jeunes pousses gagnantes ont été choisies par le jury en raison du fort potentiel commercial de leur innovation et de la qualité de leurs équipes, deux critères importants pour qu'une jeune pousse devienne une entreprise prospère. Félicitations aux gagnants !

On se voit au salon electronica 2022 ?

Le Fast Forward Award devrait être remis en jeu à l'occasion du salon electronica 2022 (<https://electronica.de/de/>), à Munich, en novembre de cette année. Espérons que tout sera rentré dans l'ordre d'ici là et que nous vous y verrons en personne !

210658-04