1ère partie

création d'interfaces graphiques en Python **avec guizero**

Installez la bibliothèque Python guizero et créez vos propres interfaces graphiques.

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

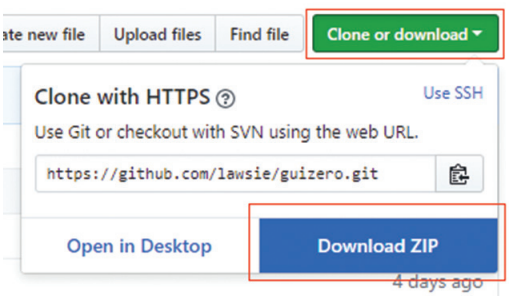
Martin crée des cours, des projets et des ressources en ligne au sein de l'équipe *Learning* de la Fondation RPi.

@ martinohanlon

Une interface utilisateur (GUI en anglais, pour *Graphical User Interface*, prononcez goo-i) facilite l'utilisation d'un programme en le rendant plus maniable. Une interface graphique est composée d'éléments graphiques (« *widgets* ») servant d'interfaces entre l'utilisateur et le programme. Vous les connaissez tous : liste déroulante d'un menu, bouton déclenchant une certaine action, etc. Nous utiliserons ici la bibliothèque *guizero*, écrite pour faciliter la création d'interfaces graphiques et donc idéale pour les débutants. La bibliothèque standard pour la création d'interfaces graphiques en Python est *tkinter*, et est installée par défaut avec Python sur la plupart des plateformes. La bibliothèque *guizero* est une adaptation des classes de *tkinter*, dont elle fournit un accès simplifié.

01 Installation de guizero

La première étape consiste bien sûr à installer la bibliothèque Python *guizero* (**lawsie.github.io/guizero**). L'installation proprement dite dépend de votre système d'exploitation et des droits dont vous disposez sur votre ordinateur (autrement dit si vous en êtes l'administrateur ou non).
Si vous avez accès à la ligne de commande (à un terminal), entrez la commande suivante :



Vous pouvez aussi installer guizero en téléchargeant son archive depuis GitHub.

```
pip3 install guizero
```

Si le terminal renvoie un message d'erreur, référez-vous aux instructions d'installation de **lawsie.github.io/guizero**, tous les systèmes d'exploitation et tous les cas sont couverts. Vous y trouverez notamment un installeur pour Windows.

02 Hello World

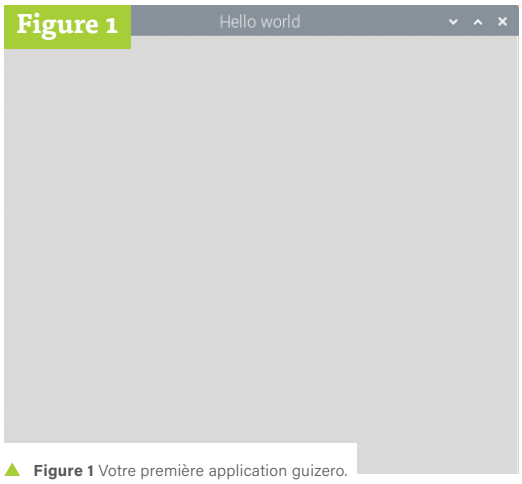
Assurons-nous de la bonne installation de *guizero* en écrivant une petite application appelée « *Hello World* », respectant en cela la tradition consistant à afficher le message Hello World lorsqu'on écrit son premier programme avec un nouveau langage.
Ouvrez votre éditeur de texte ou un EDI Python. Tout programme guizero commence par l'importation des widgets qui composeront l'interface graphique. Un widget ne s'importe qu'une seule fois et peut ensuite être utilisé autant de fois que souhaité dans le programme. Pour importer le widget `App`, ajoutez à la première ligne de votre fichier :

```
from guizero import App
```

Le widget `App` est le conteneur graphique de votre programme, autrement dit sa fenêtre principale. Un programme guizero se termine toujours par l'instruction lançant l'affichage du widget `App`. Ajoutez les deux lignes suivantes sous la première ligne d'importation :

```
app = App(title="Hello world")
app.display()
```

Enregistrez et exécutez le code : une fenêtre appelée *Hello World* devrait s'ouvrir (**fig. 1**). Bravo, vous venez de créer votre première application guizero !



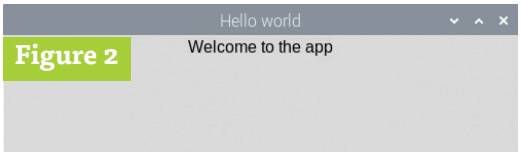
▲ **Figure 1** Votre première application guizero.

01-helloworld.py

TÉLÉCHARGEZ LE CODE COMPLET :
magpi.cc/guizerocode

> Langage : **Python 3**

```
001.  <from guizero import App, Text
002.  app = App(title="Hello world")
003.  message = Text(app, text="Welcome to the app")
004.  app.display()
```



◀ Figure 2 Ajout d'un widget Text.

03 Ajout de widgets

Un widget est un élément de l'interface tel qu'un bouton ou une barre de défilement, mais peut aussi être du texte brut.
Tous les widgets se placent entre la ligne créant la fenêtre `App` et l'instruction `app.display()`. Voici comment ajouter un widget Text à notre fenêtre précédente :

```
from guizero import App, Text
app = App(title="Hello world")
message = Text(app, text="Welcome to the app")
app.display()
```

La première ligne demande l'importation de la classe de widget `Text` en plus du widget `App`, tandis que la troisième ajoute le widget `Text` à l'interface. La **figure 2** montre notre nouvelle fenêtre.

Détaillons la syntaxe de cette troisième ligne ::

```
message = Text(app, text="Welcome to the app")
```

Un widget est une variable, et donc doit avoir un nom. Nous avons choisi `message` ici. Nous définissons ensuite `message` en tant que widget de type `Text`, et indiquons entre parenthèses ses propriétés. Le premier paramètre spécifie le conteneur dans lequel doit être placé `message`, soit, ici, `app`. Tout widget doit être contenu dans un widget conteneur. Il s'agira la plupart du temps de la fenêtre principale (le conteneur `App`), mais vous découvrirez plus tard qu'un widget peut être placé dans d'autres types de conteneurs. Le second paramètre demande au widget de contenir le texte *Welcome to the app*.

Affichette *Wanted*

01 Embellissement

Notre fenêtre est pour le moins dépouillée. Une façon de l'égayer serait de lui ajouter un fond coloré, des images et du texte utilisant différentes polices de tailles et couleurs variées. Voyons comment procéder en reproduisant l'affichette « Wanted » de la **figure 9**.
Commençons par créer une fenêtre appelée *Wanted* avec le code suivant :

```
from guizero import App

app = App("Wanted!")

app.display()
```

Enregistrez et lancez le code. La **figure 3** montre ce que vous devriez obtenir, à savoir une fenêtre grise nommée *Wanted*.

02 Couleur d'arrière-plan

Pour rendre notre affichette plus réaliste, nous allons substituer au fond gris un jaune pâle, la couleur parcheminée traditionnelle des avis de recherche placardés dans les westerns.
La fenêtre (le widget) `app` possède une propriété `bg` qui définit sa couleur de fond et celle de ses widgets. `bg` (abréviation de *background*, arrière-plan) a pour valeur par défaut *None*, d'où le gris. Nous lui affectons la valeur `yellow` juste après la création de l'objet `App`, comme ceci :

Ingrédients

- Ordinateur (RPi ou PC sous Linux, Windows ou macOS)
- Connexion internet
- Python 3 (**python.org**)
- Éditeur de texte ou EDI, p. ex. IDLE (installé avec Python 3), Thonny (**thonny.org**), Mu (**codewith.mu**), PyCharm (**jetbrains.com/pycharms**).
- Bibliothèque Python guizero (**lawsie.github.io/guizero**)

Figure 3
La fenêtre de base.

```
from guizero import App

app = App("Wanted!")
app.bg = "yellow"

app.display()
```

Pour modifier une propriété, nous spécifions d'abord le widget auquel nous nous référons (`app`), le faisons suivre de la propriété à modifier (`bg`), puis affectons la valeur souhaitée (`"yellow"`).
Même si notre affichette concerne un chat, vous trouvez peut-être ce jaune un peu trop canari (**fig. 4**). La couleur pouvant être exprimée en valeur RVB ou hexadécimale, nous pouvons facilement l'adoucir en nous rendant sur l'un des nombreux sites de sélection de couleurs, p. ex. **htmlcolorcodes.com** (**fig. 5**).
Au jaune pâle que nous avons sélectionné correspondent les valeurs RGB (251, 251, 208) et hexadécimale #FBFBD0. Nous pouvons utiliser l'une ou l'autre de ces représentations dans le code, *guizero* les interprétera correctement. Autrement dit nous avons le choix entre les deux écritures suivantes :

```
app.bg = "#FBFBD0"
app.bg = (251, 251, 208)
```

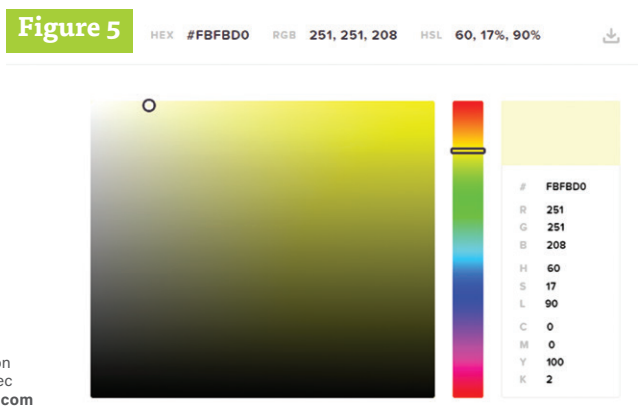


Figure 5 Sélection d'une couleur avec **htmlcolorcodes.com**

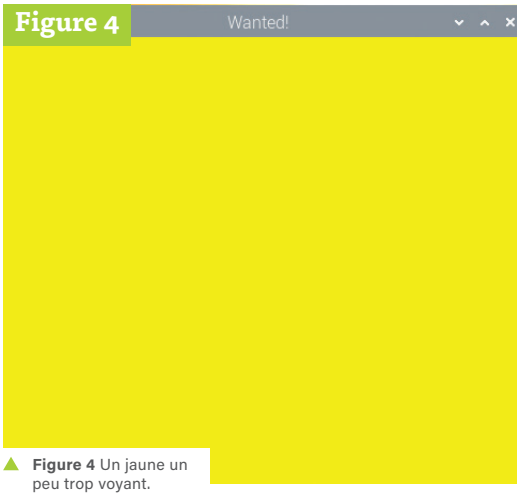


Figure 4 Un jaune un peu trop voyant.

03 Ajout de texte

Votre fenêtre devrait ressembler à celle de la **figure 6**. Ajoutons-lui le classique *Wanted* des avis de recherche, celui que même le plus analphabète des chasseurs de prime a toujours su traduire mentalement en dollars.
Nous voulons ajouter du texte, donc vous savez que c'est sur la première ligne d'importation, à savoir :

```
from guizero import App
```

que nous devons ajouter la classe **Text** :

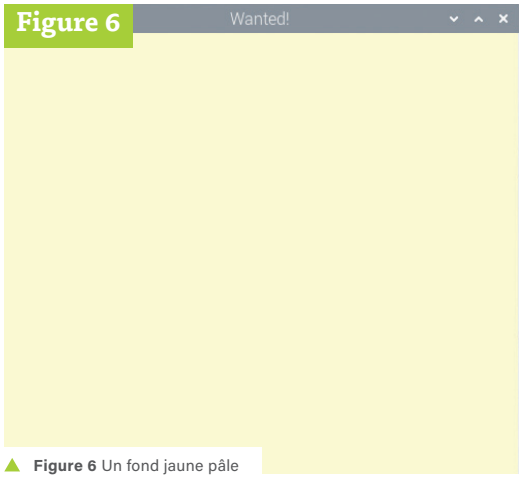
```
from guizero import App, Text
```

Vous l'avez compris, il est inutile d'encombrer le code avec une ligne d'importation par type de widget utilisé,

« La fenêtre **App** possède une propriété bg qui définit sa couleur de fond. »

Lisez la doc

Peut-être vous demandez-vous comment connaître les méthodes (fonctions), propriétés et paramètres de tel ou tel widget. Plus qu'une évidence, la réponse devrait être un réflexe : consulter la documentation. Qu'il s'agisse de *guizero* ou d'une autre bibliothèque, c'est elle qui sera votre meilleure guide. Celle de guizero se trouve sur **lawsie.github.io/guizero**. Le menu déroulant Widgets liste tous les widgets de la bibliothèque. Si par exemple vous sélectionnez le widget `Text`, vous découvrirez qu'il offre bien plus de paramètres que ceux que nous avons exploités ici. Dans la plupart des cas, la documentation illustre également l'utilisation d'une propriété ou d'une méthode particulière avec un extrait de code. Ne craignez pas de la parcourir - on n'est jamais à l'abri d'apprendre quelque chose d'utile !



▲ **Figure 6** Un fond jaune pâle

◀ **Figure 7** Le texte est trop petit.

nous les regroupons tous sur la même ligne en les séparant par une virgule.
Ajoutons maintenant un widget `Text`. Vous vous en souvenez, tous les widgets de l'interface doivent être ajoutés entre la ligne de création de la fenêtre `App` et la méthode `display()` l'instanciant. Ce qui donne

```
from guizero import App, Text

app = App("Wanted!")
app.bg = "#FBFBD0"

wanted_text = Text(app, "WANTED")

app.display()
```

Sa syntaxe ne vous est plus inconnue, mais détaillons à nouveau la ligne :

```
wanted_text = Text(app, "WANTED")
```

`wanted_text` est le nom de la variable identifiant le widget `Text`. Nous aurions tout aussi bien pu l'appeler *Grosminet*, l'ordinateur s'en bat les puces. Le point ici est qu'affecter un nom à un objet permet de s'y référer dans le code.
Ici encore nous retrouvons les deux paramètres utilisés précédemment. Le second est le texte que nous souhaitons voir afficher à l'emplacement du widget `Text`. Vous souvenez-vous du premier ? Oui, il s'agit du conteneur qui commandera le widget `wanted_text`, et ici aussi il s'agit de la fenêtre principale `app`. Nous ne l'avions pas précisé en début de tutoriel, mais ce conteneur est appelé le conteneur maître (*master*). Nous en verrons d'autres ultérieurement..

04 Taille et couleur du texte

Problème, le texte est trop petit (fig. 7) pour aider la mère Michel (qui a perdu son chat). Pour l'agrandir, nous donnons la valeur `50` à la propriété définissant la taille du texte. Rappelez-vous, nous devons spécifier trois choses :

1. Le nom du widget
2. La propriété à modifier
3. La valeur à affecter

Dans notre cas, le nom du widget est `wanted_text`, la propriété s'appelle `text_size`, et la valeur souhaitée est `50`. Nous complétons donc le code définissant notre widget avec la deuxième des lignes ci-dessous :

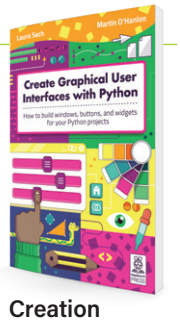
```
wanted_text = Text(app, "WANTED")
wanted_text.text_size = 50
```

Le texte se distingue maintenant beaucoup mieux (**fig. 8**). À titre d'entraînement, essayez d'en changer la police. Tous les systèmes d'exploitation n'offrent pas le même jeu de polices, mais voici quelques suggestions :

- **Times New Roman**
- **Verdana**
- **Courier**
- **Impact**

Un avis de recherche ne serait pas ce qu'il est sans une photo, donc ajoutons une image à notre interface. J'ai utilisé une photo de mon chat, car il aime à disparaître de ma vue pour aller gratter des trucs qui n'aiment pas être grattés.
Enregistrez votre image dans le dossier contenant votre programme. Vous pouvez utiliser les images d'un autre dossier, mais dans ce cas il vous faudra spécifier leur chemin d'accès. Un débutant trouve donc souvent plus simple de mettre les images dans le dossier du programme.

▶ **Figure 8** Un texte plus lisible.



Creation d'interfaces graphiques avec Python

Pour aller plus loin dans la création d'interfaces graphiques avec guizero, lisez notre livre (en anglais) *Create Graphical User Interfaces with Python*. Ses 156 pages abordent aussi la programmation évènementielle et comprennent dix projets stimulants. **magpi.cc/pythongui**

Figure 9 Wanted!

WANTED

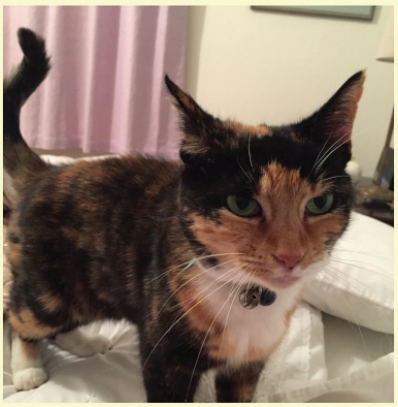


Figure 9 Aidons la mère Michel.

À ce stade l'ajout de widgets n'a normalement plus de secret pour vous. Rappelons tout de même qu'ils doivent toujours être importés en début de code et être définis entre la ligne créant la fenêtre principale et la ligne finale `app.display()`. Pensez aussi à leur affecter un nom parlant.

Le widget pour les images s'appelle Picture. Pour l'importer, nous écrivons donc :

```
from guizero import App, Text, Picture
```

Nous allons créer notre widget avec deux paramètres. Le premier spécifie son conteneur (`app`), le second le nom du fichier image. Dans mon cas (fichier image **tabitha.jpg**), l'instruction s'écrit :.

```
cat = Picture(app, image="tabitha.png")
```

Lancez votre programme (qui devrait ressembler à **02-wanted.py**) pour voir votre image s'afficher sous le texte (**fig. 9**).

À vous maintenant de compléter ou modifier ce code pour découvrir les nombreuses possibilités qu'offre *guizero* !

210419-04 (VF : Hervé Moreau)

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).

Formats d'images

Notre objectif en écrivant la bibliothèque *guizero* était de la rendre facile à installer et à utiliser. Elle n'offre donc pas de fonctions complexes de manipulation d'image, ce qui aurait nécessité l'ajout de la bibliothèque *pillow*. Vous pouvez toutefois utiliser les formats GIF non-animés et PNG sur toutes les plateformes, à l'exception de PNG sur macOS. Tenez-vous-en à ces deux formats si vous n'êtes pas sûr d'avoir *pillow* sur votre système.

02-wanted.py

TÉLÉCHARGEZ LE CODE COMPLET :
magpi.cc/guizerocode

> Langage : **Python 3**

```
001. from guizero import App, Text, Picture
002.
003. app = App("Wanted!")
004. app.bg = "#FBFBD0"
005.
006. wanted_text = Text(app, "WANTED")
007. wanted_text.text_size = 50
008. wanted_text.font = "Times New Roman"

009. cat = Picture(app, image="tabitha.png")
010.
011. app.display()
```