3e partie

Création d'interfaces graphiques en Python
Générateur de mèmes

Créez des mèmes internet à la volée avec la bibliothèque guizero.

MAKER
Laura Sach

Laura dirige l'équipe *A Level* de la Fondation Raspberry Pi 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 Raspberry Pi.

@martinohanlon

Partant des bases acquises précédemment, nous allons écrire un programme de création de mèmes. L'idée est simple : nous fournissons en entrée du texte et une image, et le code les combinera à la façon des mèmes internet.

Commençons par créer la fenêtre principale. Elle contiendra deux widgets **TextBox** pour la saisie du texte à afficher en haut et en bas de l'image. Importez d'abord les widgets nécessaires :

```
from guizero import App, TextBox, Drawing
```

Créez ensuite les deux zones de saisie :

```
app = App("meme")

top_text = TextBox(app, "top text")
bottom_text = TextBox(app, "bottom text")

app.display()
```

Ceci étant posé, passons au widget **Drawing** qui contiendra l'image et le texte du mème.

Création du mème

Ajoutez la ligne ci-dessous juste avant `app.display()`. La valeur `fill` donnée à la largeur et à la hauteur du widget **Drawing** le fera occuper toute la fenêtre.

```
meme = Drawing(app, width="fill",
height="fill")
```

Le mème sera créé et affiché dès que l'utilisateur entrera du texte dans une zone de saisie. Confions cette tâche à une fonction appelée `draw_meme()`. Son rôle sera d'effacer le contenu précédent, d'afficher une image (ici la photo d'un pic noir), et d'insérer en haut et en bas de cette image le texte entré.

Comme nous l'avions fait lors de la deuxième partie, nous utilisons la propriété **value** pour récupérer la valeur du widget **Text**. Autrement dit `top_text.value` signifie : « STP Python, retourne la chaîne de texte contenue dans l'objet `top_text` ».

```
def draw_meme():
    meme.clear()
    meme.image(0, 0, "woodpecker.png")
    meme.text(20, 20, top_text.value)
    meme.text(20, 320, bottom_text.value)
```

Dans `meme.image()` et `meme.text()`, les deux premiers nombres passés en argument sont les coordonnées `x` et `y` de l'objet à afficher. L'objet `image` sera ainsi placé au point `(0,0)`, soit au coin supérieur gauche de la fenêtre.

Ajoutez la définition de la fonction `draw_meme()` à votre code, et appelez-la juste avant la ligne `app.display()` :

```
draw_meme()
```

Votre code devrait maintenant être celui du listage **meme1.py**.

Exécutez-le, et modifiez le texte des deux zones

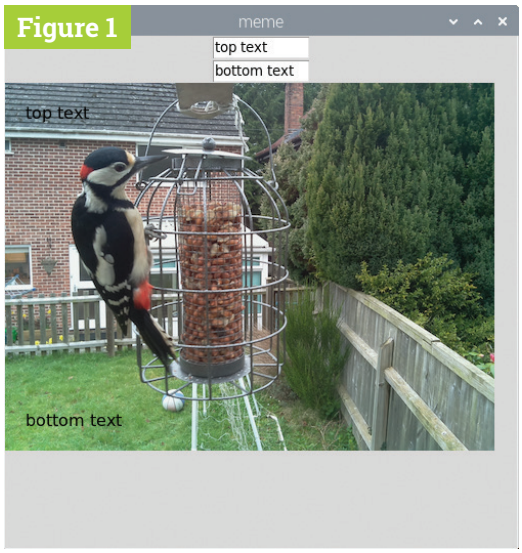


Figure 1 Notre mème avec un texte non stylisé.

de saisie : le mème n'est pas mis à jour (**fig. 1**). Pour qu'il le soit, il faudrait que la fonction `draw_meme()` soit appelée dès que du texte est saisi. C'est justement ce que permet l'instruction `command`, que nous utilisons donc dans nos deux widgets **TextBox**, comme ceci :

```
  top_text = TextBox(app, "top text",
command=draw_meme)
  bottom_text = TextBox(app, "bottom text",
command=draw_meme)
```

> « Le texte affiché est plutôt triste. Modifions son style. »

Le nouveau code est celui du listage **meme2.py**. Lancez-le pour vérifier qu'il fonctionne comme attendu. Le texte affiché est plutôt triste. Modifions sa couleur, sa taille et sa police :

```
  meme.text(
      20, 20, top_text.value,
      color="orange",
      size=40,
      font="courier")
  meme.text(
```

meme1.py

TÉLÉCHARGEZ LE CODE COMPLET :
magpi.cc/guizerocode

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text")
020.  bottom_text = TextBox(app, "bottom text")
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

meme2.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(20, 20, top_text.value)
012.      meme.text(20, 320, bottom_text.value)
013.
014.
015.  # App -------------------
016.
017.  app = App("meme")
018.
019.  top_text = TextBox(app, "top text", command=draw_meme)
020.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
021.
022.  meme = Drawing(app, width="fill", height="fill")
023.
024.  draw_meme()
025.
026.  app.display()
```

Astuce

Nous avons coupé au niveau des virgules certaines instructions pour faciliter leur lecture, mais pour le compilateur il s'agira bel et bien de la même instruction !

```
        20, 320, bottom_text.value,
        color="blue",
        size=28,
        font="times new roman",
        )
```

Jouez avec les paramètres de votre nouveau code (**meme3.py**) pour trouver un style de texte qui vous plaise (**fig. 2**).

Personnalisation du mème

Profitons maintenant d'autres widgets de la bibliothèque *guizero* pour permettre à l'utilisateur de modifier à sa guise la police, la taille et la couleur du texte. Pour la couleur et la police, nous utiliserons le widget **Combo** de liste déroulante. Pour la taille, le widget **Slider** (glissière, ou curseur) sera parfait. Commençons donc par les importer :

```
  from guizero import App, TextBox, Drawing,
Combo, Slider
```

Passons ensuite à la création d'un widget **Combo** pour la couleur. Nous l'appelons `color`, et plaçons son code juste après la définition des widgets **TextBox** :

```
  bottom_text = TextBox(app, "bottom text",
command=draw_meme)
  color = Combo(app,
      options=["black", "white", "red",
"green", "blue", "orange"],
      command=draw_meme)
```

> « L'ordre d'apparition à l'écran sera le même que celui de la liste. »

Le paramètre `options` fournit les couleurs possibles sous forme de liste. Vous pouvez en ajouter d'autres (cf. la documentation pour leurs noms anglais). Leur ordre d'apparition à l'écran sera le même que celui de la liste. La valeur par défaut sera celle du premier élément, mais vous pouvez aussi la spécifier avec le paramètre `selected` :

```
  color = Combo(app,
      options=["black", "white", "red",
"green", "blue", "orange"],
      command=draw_meme,
      selected="blue")
```

Une fois que l'utilisateur a sélectionné une couleur, il faut la récupérer et la transmettre au paramètre `color` des deux instructions `meme.text()`. Pour cela nous recourons à nouveau à la méthode `value` :

```
  meme.text(
      20, 20, top_text.value,
      color=color.value,
      size=40,
```

meme3.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color="orange",
014.          size=40,
015.          font="courier")
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color="blue",
019.          size=28,
020.          font="times new roman",
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  meme = Drawing(app, width="fill", height="fill")
032.
033.  draw_meme()
034.
035.  app.display()
```

Figure 2

▲ **Figure 2** Stylisation du texte.

```
    font="courier")
```

Modifiez aussi l'instruction **meme.text** du widget **bottom_text** (code **meme4.py**), puis procédez de la même façon pour ajouter un deuxième widget **Combo** de sélection de polices. Voici leur liste : `["times new roman", "verdana", "courier", "impact"]`. Utilisez ensuite la méthode **font.value** dans la fonction `draw_meme()` pour que la police affichée soit celle sélectionnée.

Créez de même un widget **Slider** afin que l'utilisateur puisse choisir la taille du texte :

```
size = Slider(app, start=20, end=40,
command=draw_meme)
```

Les paramètres `start` et `end` définissent les bornes de l'intervalle des valeurs possibles (ici `20` et `40`). Là encore, appliquez la méthode **value** à l'objet **size** pour récupérer la valeur de la taille sélectionnée :

```
meme.text(
    20, 20, top_text.value,
    color=color.value,
    size=size.value,
    font=font.value)
```

Votre code devrait ressembler au listage **04-meme-generator.py** et à la **figure 3** une fois lancé.

L'interactivité serait encore plus grande si

meme4.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=40,
015.          font="courier")
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=28,
020.          font="times new roman",
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  meme = Drawing(app, width="fill", height="fill")
036.
037.  draw_meme()
038.
039.  app.display()
```

Widget Drawing

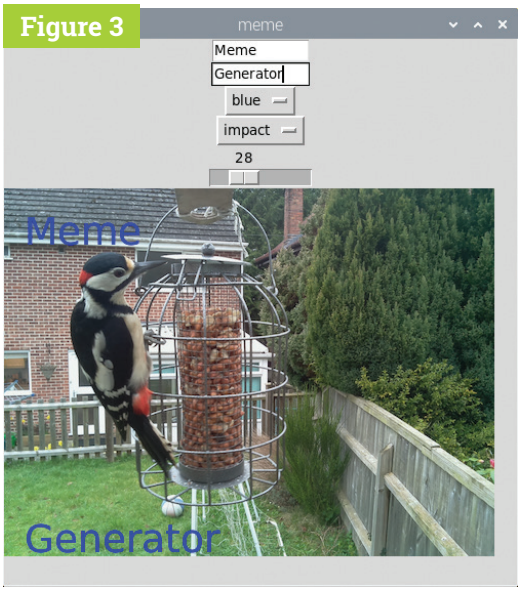
Outre l'affichage d'images et de textes, le widget **Drawing** permet de créer différentes formes géométriques et de les colorer. Pour en savoir plus, consultez l'appendice C du livre **magpi.cc/pythongui** ou la documentation en ligne : **awsie.github.io/guizero/drawing**.

04-meme-generator.py

Langage : **Python 3**

```
001.  # Imports ---------------
002.
003.  from guizero import App, TextBox, Drawing, Combo, Slider
004.
005.
006.  # Functions -------------
007.
008.  def draw_meme():
009.      meme.clear()
010.      meme.image(0, 0, "woodpecker.png")
011.      meme.text(
012.          20, 20, top_text.value,
013.          color=color.value,
014.          size=size.value,
015.          font=font.value)
016.      meme.text(
017.          20, 320, bottom_text.value,
018.          color=color.value,
019.          size=size.value,
020.          font=font.value,
021.          )
022.
023.
024.  # App -------------------
025.
026.  app = App("meme")
027.
028.  top_text = TextBox(app, "top text", command=draw_meme)
029.  bottom_text = TextBox(app, "bottom text", command=draw_meme)
030.
031.  color = Combo(app,
032.                options=["black", "white", "red", "green",
      "blue", "orange"],
033.                command=draw_meme, selected="blue")
034.
035.  font = Combo(app,
036.               options=["times new roman", "verdana", "courier",
      "impact"],
037.               command=draw_meme)
038.
039.  size = Slider(app, start=20, end=50, command=draw_meme)
040.
041.  meme = Drawing(app, width="fill", height="fill")
042.
043.  draw_meme()
044.
045.  app.display()
```

Figure 3



▲ **Figure 3** Le mème et ses widgets de sélection.

l'utilisateur pouvait choisir une image en la sélectionnant dans une liste (widget **Combo**) ou en saisissant son nom (widget **TextBox**). Sauriez-vous modifier le code en conséquence ? **(VF : Hervé Moreau)**

Python 3 Programming and GUIs

Destiné aux ingénieurs, scientifiques et amateurs, ce livre (en anglais) explique comment interfacer un PC et des projets matériels au moyen d'interfaces graphiques. L'écriture d'applications pour environnements de bureau et web est également abordée. Python 3 est un langage éminemment lisible, une qualité essentielle pour écrire rapidement des programmes. Guide simple et pratique, cette seconde édition révisée et mise à jour entend mettre le pied à l'étrier aux débutants.
www.elektor.fr/python-3-programming-and-guis

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).