Petits introduction à l'utilisation du module P5 Python

Différents P5

Il existe différentes implémentations du module `P5` pour lesquelles les méthodes sont légèrement différentes dans leur écriture. Par exemple, pour créer une zone de dessin de dimension 300×200 :

- dans l'implémentation disponible sur Capytale, il faut utiliser `createCanvas(300,200)` ;
- dans l'implémentation importée par Python 3.10, il faut utiliser `size(300,200)`.

Dans tous les cas, pour connaître la liste exacte des méthodes disponibles :

```
import p5
help(p5)
```

1. Le principe

1.1. Un premier exemple

Un exemple d'utilisation de `p5` est donné dans la documentation (en anglais). Je le donne et l'explicite ici :

```
1  from p5 import *
2
3  def setup():
4      size(640, 360)
5      no_stroke()
6      background(204)
7
8  def draw():
9      if mouse_is_pressed:
10         fill(random_uniform(255), random_uniform(127), random_uniform(51), 127)
11     else:
12         fill(255, 15)
13
14     circle_size = random_uniform(low=10, high=80)
15
16     circle((mouse_x, mouse_y), circle_size)
17
18 def key_pressed(event):
19     background(204)
20
21 run()
```

1.2. Description des fonctions

Dans la première ligne, on importe la totalité du module `p5`.

Le module `p5` utilise obligatoirement deux fonctions :

- la fonction `setup`, exécutée une seule fois, et permettant de régler un certain nombre de paramètres graphiques ;
- la fonction `draw`, exécutée à intervalle réguliers (par défaut tous les 60èmes de secondes), et qui à chaque appel trace sur le précédent canevas

Les fonctions `setup` et `draw` sont exécutées à l'appel de la fonction `run` (ligne 21). Il est possible de changer le paramètre `frame_rate` (par défaut 60), qui correspond au nombre d'appels de la fonction `draw` par secondes.

La fonction `key_pressed` est une fonction spcifique traitant les évènements liés au clavier, et est appelée dès qu'un évènement (touche pressée ou relâchée) est détecté.

1.3. La fonction `setup`

```
3  def setup():
4      size(640, 360)
5      no_stroke()
6      background(204)
```

Cette fonction **n'est exécutée qu'une seule fois** au déclenchement du programme.

- la fonction `size(640, 360)` fixe la taille de la fenêtre graphique (du canevas, ou du *sketch* en anglais) à une largeur de 640 pixels et une hauteur de 360 pixels.
- la fonction `no_stroke()` désactive le tracé des bordures de tous les objets suivants.
- la fonction `background` fixe la couleur de fond, ici en *nuances de gris*. Il est possible de passer une couleur `RGB` ou `RGBA`, simplement par une commande du type `background(120,12,204,255)`.

1.4. La fonction `draw`

```
8  def draw():
9      if mouse_is_pressed:
10         fill(random_uniform(255), random_uniform(127), random_uniform(51), 127)
11     else:
12         fill(255, 15)
13
14     circle_size = random_uniform(low=10, high=80)
15
16     circle((mouse_x, mouse_y), circle_size)
```

Cette fonction est exécutée 60 fois par seconde par défaut. Tous les dessins effectués dans cette fonctions vont être tracés sur le canevas précédent. Dans cette fonction :

- ligne 9 : on teste la valeur de la variable booléenne `mouse_is_pressed`, qui vaut `True` si n'importe quel bouton de la souris est pressé :
  - dans le cas idoine (ligne 10), la couleur de remplissage des formes `fill` est fixée à une valeur aléatoire, grâce à la loi de probabilité uniforme discrète.
  - dans le cas contraire (ligne 12, cette couleur de remplissage est fixée au blanc.
- ligne 14 : une valeur aléatoire comprise entre 10 et 80 est calculée avec la *loi de probabilité uniforme discrète*, et affectée à la variable `circle_size`.
- ligne 16 : un cercle est tracé grâce à la fonction `circle`, prenant en argument un objet de type `tuple` représentant les coordonnées du centre, obtenues grâce aux variables `mouse_x` et `mouse_y`, et un objet `int` représentant le rayon du cercle.

1.5. La fonction `key_pressed`

```
18 def key_pressed(event):
19     background(204)
```

Cette fonction est une fonction déjà définie dans le module `P5`, et qui est ici **re-définie**. Elle prend en argument un **évènement** clavier, et dans ce cas redéfini la couleur de fond - ce qui a pour effet d'effacer les tracés antérieurs.

2. Quelques Fonctions et méthodes utiles

2.1. Formes basiques

- `line(x1, y1, x2, y2)` ou `line((x1, y1), (x2, y2))` : trace une ligne entre les points de coordonnées $(x1; y1)$ et $(x2; y2)$.
- `rect(x, y, width, height)` ou `rect((x, y), width, height)` : trace un rectangle en partant du **coin supérieur gauche** de coordonnées $(x; y)$, et de dimension `width` (largeur) et `height` (hauteur). Il est aussi possible de modifier le comportement de la fonction `rect` en utilisant la fonction `rect_mode` :
  - `rect_mode(CENTER)` suivi de `rect(x, y, width, height)` tracera un rectangle de centre $(x; y)$ et de largeur et de hauteur donnée.

  - `rect_mode(CORNERS)` suivi de `rect(x1, y1, x2, y2)` tracera un rectangle de **cooin supérieur gauche** de coordonnées $(x1;y1)$ et de **coin inférieur droit** de coordonnées $(x2;y2)$.
- `ellipse(x,y, width, height)` ou `ellipse((x,y), width, height)` : fonctionne de la même manière que la fonction `rect`, en traçant une ellipse de centre $(x;y)$ à l'intérieur du rectangle de centre (x,y) et de largeur `width` et de hauteur `height`. Par défaut le mode de l'ellipse est `CENTER`, mais il est possible de le changer grâce à `ellipse_mode(CORNERS)`.
- `triangle(x1, y1, x2, y2, x3, y3)` ou `triangle((x1, y1), (x2, y2), (x3, y3))` : trace un triangle entre les points de coordonnées $(x1;y1)$, $(x2;y2)$ et $(x3;y3)$.
- `circle(x, y, r)` ou `circle((x, y) ,r)` : trace un cercle de centre $(x;y)$ et de rayon r.

2.2. Evènements

La gestion des évènements clavier et souris peut-être finement configurée dans `P5`, mais je laisse le soin à ceux intéressés de voir la documentation correspondante. Nous ne donnerons ici qu'une description sommaire des évéènements

Evènements souris

Le module `P5` donne l'accès à un certain nombre de variables permettant d'obtenir l'état de la souris :

- `mouse_x` et `mouse_y` : coordonnées du pointeur de la souris, de type `float`, valable dans l'**appel actuel** de la fonction `draw()`.
- `pmouse_x` et `pmouse_y` : coordonnées du pointeur de la souris, de type `float`, valable dans l'**appel précédent** de la fonction `draw()`
- `mouse_is_pressed` : booléen valant `True` si un quelconque bouton de la souris est pressé.
- `mouse_button` : objet global pouvant valoir :
  - `None` si aucun bouton n'est pressé ;
  - `"LEFT"` si le bouton gauche est pressé ;
  - `"CENTER"` si le bouton central est pressé ;
  - `"RIGHT"` si le bouton droit est pressé.
- `mouse_is_dragging` : booléen valant `True` si la souris est en *glissement* (c'est-à-dire le bouton gauche maintenu et la souris en mouvement).

Il est aussi possible de définir un certain nombre de fonctions :

- `mouse_moved()` : permet de définir les actions à faire quand la souris est en mouvement ;
- `mouse_pressed()` : permet de définir les actions à faire quand un bouton est pressé ;
- `mouse_released()` : permet de définir les actions à faire quand un bouton est relâché ;
- `mouse_clicked()` : permet de définir les actions à faire quand un bouton est cliqué, c'est-à-dire immédiatement relâché après avoir été pressé ;
- `mouse_draged()` : permet de définir les actions à faire quand la souris est en glissement ;
- `mouse_wheel()` : permet de définir les actions à faire quand la molette de la souris est activée.

Evènements clavier

Les deux principales variables sont :

- `key_is_pressed` : booléen valant `True` si une touche quelconque est pressée ;
- `key` : variable globale traquant les touches appuyées. Elle vaut `None` si aucune touche n'est pressée, sinon elle peut être comparée à des chaînes de caractères **en majuscule** comme `"A"`, `"T"`, `"2"`, `"ESC"`, `"SPACE"`, `"ENTER"`, etc...

On peut aussi utiliser les trois fonctions suivantes :

- `key_pressed()` : permet de définir les actions à faire quand une touche est pressée ;
- `key_released()` : permet de définir les actions à faire quand une touche est relâchée ;
- `key_typed()` : permet de définir les actions à faire quand une touche est pressée puis immédiatement relachée.

2.3. Gestion des images

Voir la documentation