

Programmation en langage Python.

Utilisation de la S.D.L.

Inspiré de travaux réalisés en C à l'UVSQ – site de Versailles

MISE EN SITUATION.

Jusqu'à maintenant nous avons uniquement **programmé en console**, nous allons maintenant utiliser une **bibliothèque tierce nommée : SDL**.



```

Console Python
*** Console de processus distant Réinitialisée ***

Le fichier de sauvegarde des scores à été modifié

# rang # Joueur # Score
# 1 # JCD # 1 256 634
# 2 # JMB # 121 708
# 3 # DS # 25 695
# 4 # FL # 7
>>>
  
```

Programmation en console



Programmation avec la SDL

La SDL est dite **bibliothèque tierce** car elle n'est pas installée par défaut contrairement aux **bibliothèques standard**.

Dans EduPython, la bibliothèque PyGame est installée par défaut.

Pour vérifier, importer la bibliothèque en tapant l'instruction suivante dans l'interpréteur Python :

```
import pygame
```

Voici la réponse dans la console :

```

Console Python
>>>
*** Console de processus distant Réinitialisée ***
pygame 1.9.4
Hello from the pygame community. https://www.pygame.org/contribute.html
  
```

A SAVOIR AVANT DE COMMENCER

Création d'une fenêtre :

```
fenetre = pygame.display.set_mode((640, 480)) # Déclaration d'une variable fenêtre
                                             # et création d'une fenêtre vide
```

L'objet retourné par la fonction est enregistré dans la variable "fenetre"

Appel de la fonction set_mode() contenue dans le module display de la bibliothèque pygame

La fonction prend en paramètre un Tuple (donc entre parenthèses) contenant la largeur et la hauteur.

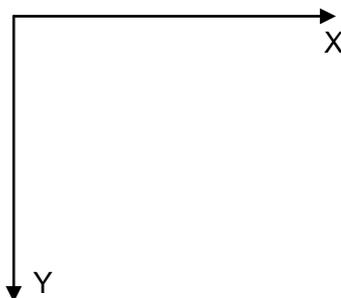
Astuce :

Afin d'optimiser votre code, vous pouvez déclarer, et donc utiliser, 2 variables pour les dimensions de la fenêtre. Ainsi vous pourrez modifier les dimensions de votre fenêtre (modification des variables), en fonction de votre écran par exemple, sans devoir changer tout votre code.

```
WIDTH = 680 ; HEIGHT = 480 # Déclaration des variables
fenetre = pygame.display.set_mode((WIDTH, HEIGHT)) #Ouverture de La fenêtre Pygame
```

Note :

Lors de la création d'une fenêtre les axes sont repérés de la façon suivante :



Couleurs :

Les noms des couleurs sont définis dans le fichier graphique.py

```
#####
# variables couleur 140 en Anglais
#####
aliceblue = (240, 248, 255)
antiquewhite = (250, 235, 215)
aqua = (0, 255, 255)
aquamarine = (127, 255, 212)
azure = (240, 255, 255)
beige = (245, 245, 220)
bisque = (255, 228, 196)
black = (0, 0, 0)
blanchedalmond = (255, 235, 205)
blue = (0, 0, 255)
blueviolet = (138, 43, 226)
brown = (165, 42, 42)
burlywood = (222, 184, 135)
cadetblue = (95, 158, 160)
chartreuse = (127, 255, 0)
chocolate = (210, 105, 30)
coral = (255, 127, 80)
cornflowerblue = (65, 105, 225)
```

```
#####
# variables couleur 18 principales
#####
vert = (0, 255, 0)
rouge = (255, 0, 0)
bleu = (0, 0, 255)
cyan = (0, 255, 255)
blanc = (255, 255, 255)
jaune = (255, 255, 0)
orange = (255, 165, 0)
noir = (0, 0, 0)
argent = (192, 192, 192)
bleumarine = (0, 0, 128)
citronvert = (0, 255, 0)
magenta = (255, 0, 255)
gris = (128, 128, 128)
marron = (128, 0, 0)
sarcelle = (0, 128, 128)
vertclair = (0, 128, 0)
vertolive = (128, 128, 0)
violet = (128, 0, 128)
```

Note sur les TUPLES :

Il est tout à fait possible, **et même préférable**, de définir les tuples dans des constantes. Exemple :

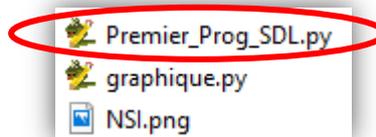
```
p1 = (40, 10)
p2 = (340, 310)

pygame.draw.line(fenetre, black, p1, p2, 3)
```

TRAVAIL DEMANDE

1. 100

Ouvrir le fichier : "Premier_Prog_SDL.py"



Exécuter ce programme et analyser le lien entre le code et l'affichage.

2. PROGRAMMATION.

En utilisant les fonctions disponibles dans la [bibliothèque pygame](#), modifier le fichier afin de réaliser successivement les dessins demandés.

Vous pouvez insérer une pause entre chaque exercice avec la fonction `attendre(ms)` de la bibliothèque `graphique.py`. Le paramètre `ms` détermine la durée de la pause en millisecondes.

Note : Il est impératif d'utiliser, dès que possible, les variables `WIDTH` et `HEIGHT` (voir : [page 3 § A savoir avant de Astuce](#)) afin de définir les différents points.

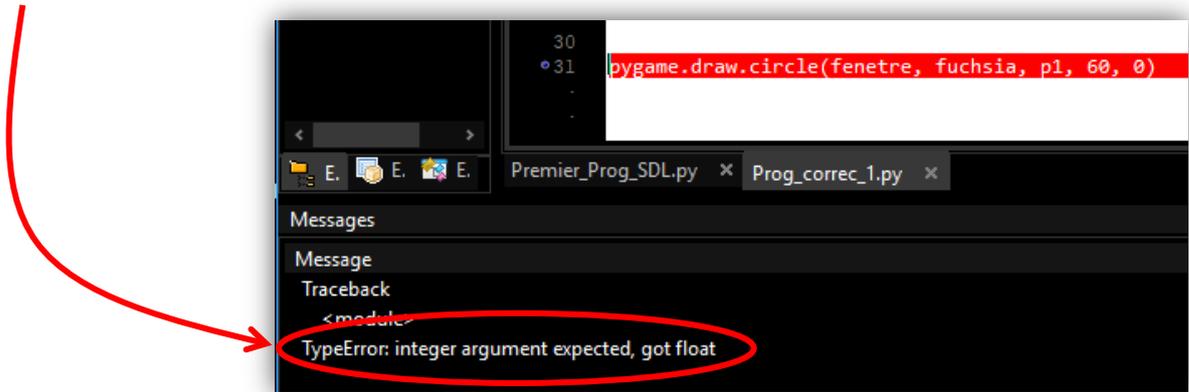
Ces variables sont déclarées et initialisées ligne 20 du fichier `1_Premier_Prog_SDL.py`

```
# Déclaration, initialisation des variables et ouverture de la fenêtre Pygame
WIDTH = 680 ; HEIGHT = 480
fenetre = pygame.display.set_mode((WIDTH, HEIGHT))
```

Pour faciliter la lecture du code, vous pouvez également créer des tuples pour les différents points utilisés (voir : [page 2 § A savoir avant de programmer / Note sur les tuples](#)).



Attention : la méthode `draw.circle` de la bibliothèque `pygame` prend des tuples d'entiers en argument et non pas des flottants.



Il faut donc faire du transtypage, c'est à dire une conversion de type (de `float` à `int`). Pour cela on utilise la fonction `int()` qui donne la partie entière d'un nombre. Par exemple :

```
p1 = (int(WIDTH / 6), int(HEIGHT / 2))
```

- 2.1. Ecrire un script permettant de dessiner un carré de 100 pixels de côté dont le coin supérieur gauche se situe :
 - ✓ A 50 pixels du bord gauche.
 - ✓ A 100 pixels du haut.
- 2.2. Modifier votre script afin de dessiner un cercle de 50 pixels de diamètre, centré sur votre fenêtre (attention les points doivent-êtres des entiers voir ci-dessus).
- 2.3. Modifier la taille de la fenêtre (variables `WIDTH` et `HEIGHT` ligne 20). Si votre cercle n'est plus centré, modifier votre script afin qu'il le reste quelque soit la taille de la fenêtre (utilisation des variables `WIDTH` et `HEIGHT` dans le calcul des points).
- 2.4. Modifier de nouveau votre script afin d'afficher un carré de 100 pixels de côté soit centré sur la fenêtre, quelque soit la taille de cette fenêtre.

Faites valider votre programmation par le professeur.

- 2.5. Dessiner une ligne horizontale d'une longueur égale à $\frac{4}{6}$ ^{ème} de la largeur de la fenêtre, centrée sur la page, et placer sur cette droite 3 cercles non remplis :
 - ✓ un centré sur l'extrémité gauche de la ligne,
 - ✓ le deuxième centré au milieu de la ligne,
 - ✓ le troisième situé sur l'extrémité droite de la ligne.
- 2.6. Dessiner un carré de 100 points de côté, dont les quatre côtés sont de couleurs différentes, le carré se situera en bas à droite de la fenêtre à 50 des bords (quelque soit la taille de la fenêtre).
- 2.7. Dessiner une croix (x) formée de deux segments, insérée dans un carré de côté 150 et dont le centre est (le centre de la fenêtre).

Faites valider votre programmation par le professeur.

3. GESTION DE LA SOURIS.

L'instruction bloquante `wait_clic()` permet de stopper le programme en attendant que l'utilisateur clique dans la fenêtre SDL.

Les fonctions :

- ✓ `wait_left_clic()`,
- ✓ `wait_center_clic()`,
- ✓ `wait_right_clic()`

implémentées dans la bibliothèque `graphique.py`, fonctionnent de la même manière que la fonction `wait_clic()`.

Ces fonctions renvoies un TUPLES contenant les coordonnées du point cliqué (x, y) .

Exemple : le tuple renvoyé par la fonction `wait_clic()` est enregistré dans la variable `p1`.

```
p1 = wait_clic()
```

- 3.1. En utilisant la fonction `wait_clic()` (ou une des fonctions citées ci-dessus), écrire un programme qui attend un clic et dessine un cercle de rayon 100 là où l'utilisateur a cliqué.
- 3.2. Reprendre votre programme en ajoutant l'effacement du cercle après un deuxième clic dans la fenêtre (redessiner un cercle au même endroit et de la couleur du fond d'écran).

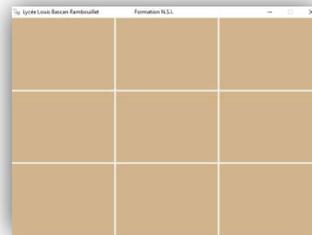
Note : la couleur du fond d'écran est définie à la ligne 24.

```
# Remplissage du fond de la fenêtre
fenetre.fill((darkkhaki))
```

Faites valider votre programmation par le professeur.

4. POUR ALLER PLUS LOIN

- 4.1. Réaliser un quadrillage de (3×3) .



- 4.2. Compléter votre programme de quadrillage qui attend un clic de l'utilisateur et tracer un cercle **centré** dans la case cliquée.
 Indice : Utiliser la division entière `//` pour connaître le numéro de la case cliquée (0, 1 ou 2).

Pour récupérer les données d'un tuple (coordonnées du point cliqué) indépendamment et pouvoir les modifier, il y a deux possibilités :

```
p1 = wait_clic()
p1_x = p1[0]
p1_y = p1[1]
```

```
p1_x, p1_y = wait_clic()
```

Dans les cas précédents, on enregistre le premier élément du tuple dans la variable `p1_x` et le deuxième dans la variable `p1_y`.

Note : Pour ce type de problème, il est impératif d'utiliser une feuille et un crayon pour réfléchir.

Sans cette démarche, aucune aide ne vous sera apportée !