SIMON Denevan
A travaillé sur le projet en collaboration avec CHAPMAN Matt

Projet NSI

Sommaire :

I – La répartition des tâches

Pour commencer, nous nous sommes réparties les tâches de façon équitable. Nous avions tout les deux autant de choses à faire de notre côté.
Nous avons choisi les fonctions à coder au hasard, j'ai personnellement eu à faire les fonctions suivantes :

- nombre_aleatoire(n)
- couleur_aleatoire()
- tracer_carre(x, y, longueur)
- tracer_point(x, y, longueur)
- afficher_diagonale_1(x, y, longueur)
- afficher_diagonale_2(x, y, longueur)
- et enfin, lancer_jeu() que j'ai fais avec mon camarade

Je n'ai eu aucune difficulté à travailler avec Matt, nous avions un Google Drive où nous rangions chaque fichier de fonctions afin de les retrouver sans problèmes. Nous avons échangés nos Discord afin de pouvoir communiquer ensemble.

II – L'explication des fonctions codées

1. nombre_aleatoire(n) :

Cette fonction est très simple à comprendre et coder. Elle sert à donner un nombre aléatoire entre 1 et n, un nombre défini lors de l'appel de la fonction.
J'ai utilisé le module randint() afin de faire la fonction.

```
return randint(1,n)
```

2. couleur_aleatoire() :

Cette fonction sert à donner trois valeurs entre 1 et 255 afin de former une couleur. J'ai utilisé la fonction nombre_aleatoire() codée précédemment afin de faire fonctionner le code.

```
a = nombre_aleatoire(255)
b = nombre_aleatoire(255)
c = nombre_aleatoire(255)
return(a,b,c)
```

3. tracer_carre(x, y, longueur) :

Cette fonction était un peu plus difficile à faire car elle nécessite d'utiliser turtle, une bibliothèque permettant de dessiner et encore inconnue pour moi avant le projet. Elle trace un carré d'une longueur donnée et à un endroit choisi qui est ensuite rempli d'une couleur choisie avec couleur_aleatoire.

```
penup()
goto(x,y)
pendown()
colormode(255)
fillcolor(couleur_aleatoire())
begin_fill()
for i in range(4):
    forward(longueur)
    left(90)
end_fill()
```

4. tracer_point(x, y, longueur) :

Cette fonction permet de tracer un point via turtle à un endroit et d'une longueur tout deux donnés. Elle sert donc à faire les points à l'intérieur du dé.

```
goto(x, y)
dot(longueur/5, "black")
```

5. afficher_diagonale_1/_2(x, y, longueur) :

Ces deux fonctions ont un comportement pratiquement identique, leur seule différence est leur sens. Elles permettent donc de tracer les trois points en diagonale d'un dé de droite à gauche ou de gauche à droite. Elles utilisent tracer_point .

diagonale 1 :

```
tracer_point(x+(3/4*longueur), y+(3/4*longueur), longueur)
tracer_point(x+(1/4*longueur), y+(1/4*longueur), longueur)
```

diagonale 2 :

```
penup()
tracer_point(x+(1/4*longueur), y+(3/4*longueur), longueur)
tracer_point(x+(3/4*longueur), y+(1/4*longueur), longueur)
```

6. lancer_jeu() :

La dernière fonction, elle synthétise toutes les fonctions codées afin de lancer le jeu. Elle demande le nom des joueurs, puis lance trois dés pour chacun des deux. Les chiffres tirés sont ensuite rangés dans l'ordre décroissant et le joueur ayant le score le plus élevé remporte la partie.

```
def lancer_jeu(): # Ensemble
    """ Programme prinicipal de la gestion du jeu"""
    ## Ecrivez ici le code de la fonction
    nom_j1 = textinput("Nom joueur 1", "Veuillez saisir le nom du joueur 1.")
    nom_j2 = textinput("Nom joueur 2", "Veuillez saisir le nom du joueur 2.")

    afficher_message(-300, 150, nom_j1)
    afficher_message(300, 150, nom_j2)

    lancej1_1=nombre_aleatoire(6)
    choisir_face_a_afficher(-450,50,lancej1_1, 80)
    lancej1_2=nombre_aleatoire(6)
    choisir_face_a_afficher(-350,50,lancej1_2, 80)
    lancej1_3=nombre_aleatoire(6)
    choisir_face_a_afficher(-250,50,lancej1_3, 80)
    scorej1 = comparer_chiffre(lancej1_1, lancej1_2, lancej1_3)
    afficher_message(-300, -10,(f"Score : {scorej1}"))

    lancej2_1=nombre_aleatoire(1)
    choisir_face_a_afficher(350,50,lancej2_1, 80)
    lancej2_2=nombre_aleatoire(1)
    choisir_face_a_afficher(250,50,lancej2_2, 80)
    lancej2_3=nombre_aleatoire(1)
    choisir_face_a_afficher(150,50,lancej2_3, 80)
    scorej2 = comparer_chiffre(lancej2_1, lancej2_2, lancej2_3)
    afficher_message(300, -10,(f"Score : {scorej2}"))

    if scorej1 > scorej2 :
        afficher_message(0, -60 ,nom_j1+" a gagné !")
    elif scorej1 < scorej2 :
        afficher_message(0 ,-60 ,nom_j2+" a gagné !")
    else:
        afficher_message(0, -60, "égalité !")
```

III – Les difficultés rencontrées

J’ai rencontré plusieurs difficultés, notamment pour prendre en main la bibliothèque turtle. Par exemple, pour la fonction tracer_carre, je n’arrivait pas à trouver comment remplir l’intérieur du carré, ou encore je ne pensais pas à lever le stylo lorsque j’utilisais le module goto(x, y) ce qui traçait donc une droite.
De même pour la fonction tracer_point où j’avais du mal à comprendre comment l’organiser, ce qui au final s’est avéré simple et faisable en deux petites lignes.
Cependant, le reste du travail était accessible pour ma part. J’étais même étonné de la facilité à construire la fonction lancer_jeu() après avoir bien travaillé les fonctions antérieures.

IV – Bibliographie

site utilisé pour comprendre turtle : https://docs.python.org/fr/3/library/turtle.html