Module 6 : Structure itérative à condition

Situations

Le lièvre et la tortue

Règles du jeu

- On lance un dé :
  - Si on obtient 6, le lièvre gagne
  - Si on obtient une autre valeur, la tortue avance d'un pas
- Si la tortue a avancé de six cases elle gagne, sinon on rejoue une autre fois.

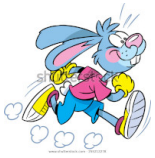

Figure 1, Lièvre

Figure 2, Tortue

Figure 3, Dé

Simulation

Figure 4, Simulation du jeu lièvre et tortue

Travail demandé

On demande de :

1. Copier/coller le code suivant.
2. Tester ce code, puis dire qu'est ce qu'il fait ?
   Permet de lancer un dé et puis de modifier la position de l'un des deux joueurs en fonction de la valeur obtenue.
3. Combien faut-t-il jouer de fois pour que l'un des joueurs gagne ?
   Tout dépend de la valeur du dé. Normalement, dès que l'un des joueurs atteint la position d'arrivée le jeu se termine.
4. Est-ce qu'on peut utiliser la boucle **for** pour compléter ce programme ?
   La boucle *for* est utilisable dans les cas où le nombre de répétitions est connu à l'avance. Dans ce jeu on ne sait pas combien de fois il faudra rejouer avant que l'un des deux joueurs gagne.
5. Quelle est la structure itérative à utiliser dans ce jeu ?
   Il convient d'utiliser la boucle *while* dans ce jeu.

Code Python

```
from random import randint
# (1) Placer les joueurs en position de départ
pos_lievre = 0
pos_tortue = 0
# (2) Lancer un dé
de = randint(1, 6)
print("Dé =", de)
# (3) Déplacer les joueurs
if de != 6:
    pos_tortue = pos_tortue + 1
    print("La tortue avance à la case", pos_tortue)
else:
    pos_lievre = 6
    print("Le lièvre avance à la position", pos_lievre)
```

Structure itérative à condition

(1) Tester le code suivant pour les valeurs suivantes de a : -2, 11, 5, 1, 10, 9, 29. Quelles sont les valeurs acceptées ?
5, 1, 10, 9 sont acceptées car elles sont comprises dans l'intervalle [1, 10].

```
a = 0
while not 1 <= a <= 10:
    a = int(input("Donner un entier ? "))
print(a)
```

(2) Exploiter le code précédent pour écrire un programme qui permet de **saisir deux entiers positifs a et b avec 0 < a ≤ b ≤ 100**. Puis d'afficher la moyenne des deux nombres.

```
a = -1
while not 0 < a <= 100:
    a = int(input("Donner 0 < a <= 100 ? "))
b = -1
while not a <= b <= 100:
    b = int(input(f"Donner {a} <= b <= 100 ? "))
print((a + b) / 2)
```

(3) Compléter le jeu du **lièvre et de la tortue** pour que l'un des joueurs gagne lorsqu'il atteint la case d'arrivée (position 6) ?

```
from random import randint
# (1) Placer les joueurs en position de départ
pos_lievre = 0
pos_tortue = 0
while pos_lievre != 6 and pos_tortue != 6:
    # (2) Lancer un dé
    de = randint(1, 6)
    print("Dé =", de)
    # (3) Déplacer les joueurs
    if de != 6:
        pos_tortue = pos_tortue + 1
        print("La tortue avance à la case", pos_tortue)
    else:
        pos_lievre = 6
        print("Le lièvre avance à la case", pos_lievre)
print("Le lièvre est en position", pos_lievre)
print("La tortue est en position", pos_tortue)
# (4) Afficher le gagnant
if pos_lievre == 6:
```

```
    print("Le lièvre a gagné.")
else:
    print("La tortue a gagné.")
```

Jeu de l'oie

Principe de jeu

Le jeu de l'oie est un jeu classique qui se joue à 2, 3 ou 4 joueurs.
Chaque joueur, à son tour, lance un dé à six faces et avance son pion du nombre tiré.
Le gagnant est celui qui atteint la case d'arrivée (case 63).
On désire implémenter une version simplifiée, sans pièges, de ce jeu où un joueur humain joue tout seul. **La case d'arrivée est la case 32.**
Remarque : Si la nouvelle position du joueur dépasse la case finale il recule au lieu d'avancer.
Exemple : Je suis dans la position 27 et j'obtiens 6, je recule à la position 21.

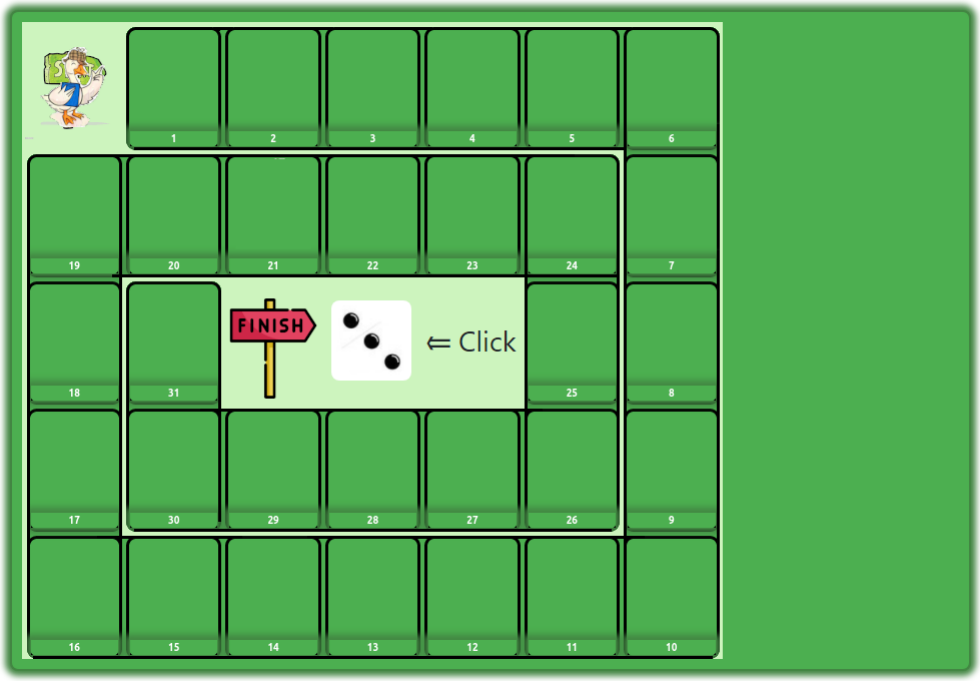

Figure 5, Simulation du jeu de l'oie

Code départ

```
from random import randint
case_finale = 32 # 1. n° de la case d'arrivée
pos_j = 0        # 2. position du joueur
gagne = False    # 3. gagne = True lorsqu'il atteint la case d'arrivée
nbre_tours = 0   # 4. nombre de lancées du dé
# 4. incrémenter le nombre de lancées de dé
nbre_tours = nbre_tours + 1
de = randint(1, 6)     # 5. lancée du dé
nouv_pos = pos_j + de  # 6. nouvelle position du joueur s'il avance de (de) pas
action = "Av"          # 7. "Av" : le joueur avance - "Re" : il recule
# 8. Afficher la nouvelle position du joueur
print(f"Tour {nbre_tours:2} - Dé : {de} - {action} {pos_j:2} → {nouv_pos:2}")
pos_j = nouv_pos       # 9. Après son avancement le joueur atteint la position (nouv_pos)
# 10. gagne = True lorsqu'il atteint la case d'arrivée, sinon False
gagne = pos_j == case_finale
```

Travail demandé

- Copier/Coller le code précédent.
- Commenter ce code pour expliquer la fonction de chaque partie du code.
- Est-ce que le jeu est complet ?
  Le jeu est incomplet.
  Le joueur effectue uniquement une seule lancée de dé. Il est impossible qu'il atteigne la position d'arrivée avec une seule lancée.
- Quelles sont les étapes qui devront être répétées ? Combien de fois ?
  Les étapes 4 à 10 devront être répétées tant que le joueur n'a pas encore gagné.
- Compléter le code pour que le joueur continue de jouer jusqu'à ce qu'il gagne.

```
from random import randint

case_finale = 32
pos_j = 0
gagne = False
nbre_tours = 0
while not gagne:
    nbre_tours = nbre_tours + 1
    de = randint(1, 6)
    if pos_j + de <= case_finale:
        nouv_pos = pos_j + de
        action = "Av"
    else:
        nouv_pos = pos_j - de
        action = "Re"
    print(f"Tour {nbre_tours:2} - Dé : {de} - {action} {pos_j:2} → {nouv_pos:2}")
    pos_j = nouv_pos
    gagne = pos_j == case_finale
```