1 Branchement conditionnel `if [else]`

Il permet de conditionner, l'exécution d'un bloc d'instruction, à la valeur `True` ou `False` d'une condition.

`if` *condition* **:**

..... Bloc d'Instructions

else :

..... Bloc d'instructions

meme espace — bloc d'instructions

– Si *condition* a valeur `True` le premier bloc d'instruction est exécuté, le deuxième ne l'est pas, puis l'exécution du programme se poursuit.

– Si *condition* a valeur `False` le deuxième bloc d'instruction est exécuté (le premier ne l'est pas), puis l'exécution du programme se poursuit.ß
L'instruction **`else`** est optionnelle.

• **Exemple.**

```
a = float(input('Saisissez un nombre'))
if  a >= 0 :
    print(a, 'est positif')
else :
    print(a, 'est négatif')
```

Exercice 1. Calcul de sommes doubles
Dans chaque cas, écrire une fonction `somme(n)` prenant en paramètre un entier positif n et qui calcule la somme double :

1. $\sum_{1\leqslant i,j\leqslant n} \min(i,j)$.
2. $\sum_{1\leqslant i,j\leqslant n} |i-j|$.

Exercice 2. Evolution de population

1. Une population constituée initialement de 1000 individus s'accroit de 5% chaque jour. Le milieu étant limité en ressource, dès que la population atteint le seuil de 10000 individus, la moitié d'entre eux périssent.
   Combien d'individus constitueront la population au bout d'un an (écrire un script).
2. Une population constituée initialement de 1000 individus s'accroît de 3% chaque jour. Après chaque mois (= 30 jours) on retire de la population 100 individus.
   Déterminer au bout de combien de jours la population aura (au moins) doublé.

Exercice 3 (Complement). La suite de Syracuse, et la conjecture 3X+1

Soit la suite définie par :

$$u_0 \in \mathbb{N}^*, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3\times u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

On l'appelle la **suite de Syracuse**. Il y a en fait une suite bien définie pour chaque valeur du premier $u_0 \in \mathbb{N}^*$.

1. Écrire une fonction `syracuse(u0,n)` prenant en paramètre un entier strictement positif u_0 et un entier n, et qui renvoie le terme de rang n de la suite de Syracuse définit pour cette valeur de u_0.

Il s'avère que quelle que soit la valeur du premier terme $u_0 \in \mathbb{N}^*$, la suite finit toujours tôt ou tard, par prendre les valeurs $1, 4, 2, 1, 4, 2, \ldots$. Même si on le vérifie pour tous les nombres testés, personne ne sait démontrer ce résultat, et encore moins expliquer pourquoi. On l'appelle la conjecture $3X+1$, ouverte depuis près d'un siècle.

1. Écrire une fonction `tempDeVol(u0)` prenant en paramètre un entier strictement positif, et qui renvoie le premier rang pour lequel la suite de Syracuse correspondante prend la valeur 1.
2. Écrire un script qui permette de tester la conjecture pour tous les entiers entre 1 et 1000.
3. Pour tous les entiers u_0 entre 1 et 10000, quel est celui pour lequel le "temps de vol" est le plus grand, et quel est ce temps de vol ?

2 Branchements multiples `if [elif] [else]`

L'écriture de tests imbriqués :

`if` (condition1) :
    Bloc d'instructions 1
`else` :
    `if` (condition2) :
        Bloc d'instructions 2
    `else` :
        Bloc d'instructions 3

peut s'écrire à l'aide d'une seule structure de test :

if (condition1) :
    Bloc d'instructions 1
elif (condition2) :
    Bloc d'instructions 2
else :
    Bloc d'instructions 3

elif et **else** sont optionnels. **elif** peut être utilisé plusieurs fois.

- **Exemple.** Dans le calendrier grégorien (actuel), une année est bissextile si :
  - Elle est divisible par 4 sans être divisible par 100, ou
  - Elle est divisible par 400.

La fonction suivante renvoie `True` ou `False` selon si l'année passée en paramètre est ou non bissextile :

```
def bissextile(n):
    if n % 400 == 0:
        return True
    elif n % 4 == 0 and n % 100 != 0:
        return True
    else:
        return False
```

Exercice 4.
Écrire le script du jeu suivant :

- L'ordinateur tire au sort un nombre entier entre 1 et 100. C'est le nombre mystère que doit trouver le joueur.
- L'utilisateur a 6 chances de deviner ce nombre.
- A chaque essai infructueux, l'ordinateur indique au joueur si le nombre mystère est plus grand ou plus petit que celui saisi par l'utilisateur.

Pour tirer au sort un entier entre 1 et 100, on utilisera la fonction :
`randint(1,100)`,
que l'on aura importé du module `random` par la commande :
`from random import randint`.