TRAVAUX PRATIQUES n° 4
Boucle while

Pour bien démarrer ce TP, commencez par lire la fiche méthode suivante :

⋆ **I6** - Utiliser une boucle `while`

Une boucle `while` répète un bloc **tant qu'une condition est vraie**. Contrairement à la boucle `for`, on ne connaît pas à l'avance le nombre de tours : on s'arrête dès que la condition devient fausse. Il faut donc **modifier** à l'intérieur de la boucle une variable qui intervient dans la condition, sinon la boucle ne s'arrête jamais.

Exercice 1 (*Échauffement*) :

1) Exécuter un par un les scripts suivants :

```
## Script 1
i = 0
while i < 10:
    print(i)
    i = i + 1

## Script 2
i = 0
while i < 10:
    i = i + 1
    print(i)

## Script 3
x = 0
while x**2 < 100:
    print(x)
    x = x + 1
```

Pour chaque script : quelles sont les valeurs affichées ? Combien de tours de boucle ?

2) Que se passera-t-il si on exécute les scripts suivants ? (On pourra les tester, puis les interrompre avec le bouton d'interruption de Pyzo si besoin.)

```
## Script 1
i = 0
while i < 10:
    print(i)

## Script 2
i = 0
while i > 10:
    print(i)
```

3) Exécuter le script suivant, le tester et expliquer à quoi il sert :

```
from math import pi

theta = float(input("Entrer un angle : "))
while not 0 <= theta < 2*pi:
    theta = theta - 2*pi
print(theta)
```

Ce script n'est pas parfait : pourquoi ? (Penser à un angle **négatif**, par exemple -10.)

4) **Saisie contrôlée.** Écrire un script qui demande à l'utilisateur un entier strictement positif. Tant que la valeur entrée n'est pas strictement positive, on redemande un entier. Une fois qu'un entier strictement positif a été saisi, on l'affiche.

Exercice 2 (*Le plus petit entier*) : Écrire un script qui calcule et affiche le plus petit entier n tel que

$$1 \times 2 \times 3 \times \cdots \times n > 10^6.$$

Exercice 3 (*Origami*) : Une feuille de papier a une épaisseur de $0,1$ mm. La distance Terre-Lune est d'environ 384 400 km.

1) Estimer, sans calcul, combien de fois il faut plier la feuille sur elle-même pour que l'épaisseur dépasse la distance Terre-Lune.

   Attention : quand on plie la feuille 2 fois sur elle-même, son épaisseur n'est pas de $0,3$ *mm (chaque pli* ***double*** *l'épaisseur). On travaillera en millimètres.*

   Le résultat attendu est de l'ordre de quelques dizaines de plis.

2) Déterminer la valeur exacte de ce nombre à l'aide d'une boucle `while`.

Exercice 4 (*Suite de Syracuse*) : On considère la suite définie par $u_0 = n$ (un entier $n \geqslant 1$ choisi par l'utilisateur) et

$$u_{k+1} = \begin{cases} u_k/2 & \text{si } u_k \text{ est pair} \\ 3u_k + 1 & \text{si } u_k \text{ est impair} \end{cases}$$

On s'arrête au premier terme égal à 1.

1) Écrire un script qui demande n à l'utilisateur, affiche tous les termes jusqu'à 1 (inclus), puis affiche le nombre de termes calculés (le temps de vol).

   Vérification : pour $n = 7$*, on obtient* 17 *termes.*

2) Tester avec $n = 7$, $n = 27$ et $n = 27\,000$.

   Pour $n = 27\,000$*, on pourra commenter les* `print` *des termes si l'affichage est trop long, et ne garder que le temps de vol.*

Exercice 5 (*Nombre de zéros finaux*) : Écrire un script qui demande un entier n et affiche le nombre de zéros à la fin de l'écriture décimale de n.

Idée : tant que n *est divisible par* 10*, on le remplace par* $n//10$ *et on compte. Si* $n = 0$*, le résultat est* 1*.*

Exemples : 1200 a 2 zéros finaux ; 123 a 0 zéro final ; 0 a 1 zéro final.

Exercice 6 (*PGCD*) : Le PGCD (Plus Grand Commun Diviseur) de deux entiers naturels non nuls a et b est le plus grand entier qui divise à la fois a et b.

Par exemple, les diviseurs positifs de 12 sont 1, 2, 3, 4, 6, 12 et ceux de 18 sont 1, 2, 3, 6, 9, 18 : leur plus grand commun diviseur est donc 6. On note $\text{PGCD}(12, 18) = 6$.

Pour le calculer efficacement, on utilise l'algorithme d'Euclide :

- ⋆ tant que b n'est pas nul, on remplace le couple (a, b) par $(b, a \bmod b)$ (où $a \bmod b$ est le reste de la division euclidienne de a par b, soit `a % b` en Python) ;
- ⋆ lorsque b devient nul, le PGCD est la valeur actuelle de a.

Illustration avec $a = 252$ et $b = 105$:

a	b	$a \bmod b$
252	105	42
105	42	21
42	21	0
21	0	—

On s'arrête : $\mathrm{PGCD}(252, 105) = 21$.

Écrire un script qui demande deux entiers a et b strictement positifs et affiche leur PGCD à l'aide d'une boucle `while`.

Vérification : le PGCD de 252 *et* 105 *vaut* 21 *; celui de* 12 *et* 18 *vaut* 6.