# Exemples de programmes Python pour les suites

Ce document ne prétend pas faire un cours complet de Python, mais les exemples donnés permettent de comprendre la majorité des questions de programmation au bac (épreuves terminales ou contrôle continu).

**Énoncé:** (Valable pour tout ce document):

"Un biologiste étudie une population de bactéries dans une cuve métallique. Au début de l'expérience, la cuve contient un million de bactéries par millilitre. La cuve a un effet bactéricide: chaque heure, la population de bactérie est réduite de 5%. Le niveau de dangerosité pour ce type de bactérie est de 1000 bactéries par millilitre."

1. Premier type de question possible (Interpréter):
   **Question:** Interpréter dans le contexte de l'énoncé la valeur de la variable U à la fin de l'exécution du programme suivant:

```
U= 1000000
for i in range(50):
    U= U * 0.95
```

   **Réponse:** U contiendra le nombre de bactéries par millilitre après 50 heures.
   **Explications:** La première ligne, U= 1000000, place la valeur 1000000 dans la **variable** U. Mais attention, à la différence des maths, les variables en Python peuvent varier: leur valeur peut changer si une autre ligne commençant par U=... est exécutée. Par exemple, la dernière ligne:

$$\text{U= U} * 0.95$$

   Cette ligne peut être lue comme ceci: "faire le calcul U $*$ 0.95, puis placer le résultat dans U.". Ou encore: "La nouvelle valeur de U est égale à l'ancienne valeur de U fois 0.95". En maths, cela correspond à la relation de récurrence $\boldsymbol{u_{n+1} = u_n * 0,95}$. Et la première ligne correspond à $\boldsymbol{u_0 = 1000000}$. Les numéros de termes en maths (0, n, n+1, etc) correspondent ici aux étapes de l'exécution en Python. On remarque aussi que les nombres à virgule (0,95) sont en notation anglo-saxonne: 0.95.
   Enfin, la ligne la plus compliquée:

$$\textbf{for i in range}(50):$$

   Signifie que les lignes suivantes doivent être répétée 50 fois. Ceci s'appelle une "boucle pour" ("for" signifie "pour" en anglais). L'information la plus importante est donc le nombre entre parenthèses.

**Remarque:** Noter l'importance du contexte de l'énoncé dans la rédaction: ici par exemple, U n'est pas le nombre total de bactéries dans la cuve. Inversement, il n'est pas nécessaire d'avoir compris tous les termes de l'énoncé (bactéricide, etc) pour faire l'exercice.
**Remarque:** L'exercice contiendra aussi des questions de maths sans Python: type de la suite, terme général, etc. Ici l'énoncé pourrait aussi demander de donner une formule pour la valeur de U à la fin de l'exécution; la réponse serait $\boldsymbol{u_n = u_0 * q^n = 1000000 * 0,95^{50}}$. La raison est égale à 0,95 car quand on enlève 5% il reste 95%.

2. Deuxième type de question possible (Compléter):
**Question:** Compléter le programme suivant pour que, à la fin de l'exécution, la variable U contienne le nombre de bactéries par millilitre au bout de 50 heures.

```
U= ...
for i in range(...):
    U= ...
```

**Réponse:** (Programme de la question précédente.)
**Explications:** Il s'agit de la question inverse. En général, on ne vous demandera pas d'écrire un programme sans être guidé. Par contre il faut avoir suffisamment compris le principe pour pouvoir reconstituer en complétant. Et en général, ce que vous devez écrire correspond à la partie la plus mathématique du programme. Ici "1000000", "50", et "U∗0.95"; c'est un peu comme de taper à la calculatrice.

3. Un autre type de boucle (while):
**Question:** Interpréter dans le contexte de l'énoncé la valeur de la variable N à la fin de l'exécution du programme suivant:

```
N= 0
U= 1000000
while U > 1000:
    U= U * 0.95
    N= N+1
```

**Réponse:** N contiendra le nombre d'heures nécessaires pour descendre en dessous du seuil de dangerosité.
**Explications:** Si on oublie la variable N, on a presque le même programme qu'avant, mais la boucle est différente:

**while** U > 1000:

Ceci signifie que les lignes suivantes seront répétée **tant que** ("while" en anglais) la condition "U > 1000" est vraie. Donc d'après l'énoncé, tant que l'eau est dangereuse. On remarque ensuite que ce n'est pas la valeur de U qu'on nous demande d'interpréter, mais celle de N. Cette nouvelle variable part de 0 et augmente de 1 à chaque répétition (à cause de la ligne N= N+1). Elle compte donc le nombre d'étapes, c'est à dire le nombre d'heures.

4. Programme nommé (fonction):
**Question:** (Adapté d'une question de bac.) Le biologiste a programmé la fonction **seuil()** ci-dessous. Interpréter dans le contexte de l'exercice la valeur renvoyée lorsqu'on appelle la fonction **seuil()**.

```
def seuil():
    n= 0
    U= 1000000
    while U > 1000:
        U= U * 0.95
        n= n+1
    return n
```

**Réponse:** La valeur retourné est le nombre d'heures nécessaires pour descendre en dessous du seuil de dangerosité.
**Explications:** Il s'agit en fait du même programme qu'avant, auquel on a donné un nom: **seuil**; on a ainsi créé une fonction, qui peut être lancée plus tard. Les parenthèses vides à la première ligne sont surprenantes, mais nécessaires en Python (elle pourraient contenir des paramètres; ici il n'y a pas, mais il faut quand même les mettre pour que Python le sache).
La ligne **return** n ("retourner", ou "renvoyer" en Python) indique que le résultat de cette fonction est la valeur de n au moment où cette ligne est exécutée. On aurait aussi pu écrire **return** U, mais le programme serait différent (il nous donnerait un nombre de bactéries au lieu du nombre d'heures).
**Remarque:** La ligne **return** n est exécutée une fois la boucle terminée (elle n'est pas répétée à chaque étape). Ici c'est logique (pour un humain), mais Python le sait en voyant que cette ligne n'est pas au même niveau de décalage que les deux lignes au dessus; la dernière ligne ne fait donc pas partie de la boucle.