N14 Algorithmique : suites, seuils et simulations

OBJECTIFS

- comprendre le rôle des variables et des affectations ;
- écrire une boucle bornée ‘for‘ et une boucle conditionnelle ‘while‘ ;
- programmer le calcul des termes d'une suite ;
- déterminer un seuil par un algorithme ;
- écrire une fonction Python simple ;
- utiliser une simulation pour conjecturer ou estimer.

ACTIVITÉ DE DÉPART

Une population vaut 500 au départ. Chaque année, elle augmente de 12 %. On souhaite savoir au bout de combien d'années elle dépasse 1000.

1. Peut-on répondre rapidement à la main ?
2. Quelle quantité faut-il recalculer à chaque étape ?
3. Quel test permet de savoir quand s'arrêter ?

1. Variables, affectation et trace d'exécution

Un programme manipule des valeurs stockées dans des variables. En Python, l'instruction ‘=‘ ne signifie pas « est égal à » comme en mathématiques : elle signifie que l'on affecte une valeur à une variable.

Définition (Variable) Une variable est un nom qui désigne une valeur stockée en mémoire. Cette valeur peut changer au cours de l'exécution du programme.

Définition (Affectation) Une affectation est une instruction qui donne une valeur à une variable. Par exemple, ‘u = 500‘ affecte la valeur 500 à la variable ‘u‘.

Méthode Pour comprendre un programme, on peut faire une trace d'exécution : on suit ligne par ligne l'évolution des variables importantes dans un tableau.

Exemple Le programme ci-dessous effectue trois mises à jour d'une suite définie par $u_{n+1} = 1{,}12u_n$, avec $u_0 = 500$.

```
u = 500
u = 1.12*u
u = 1.12*u
u = 1.12*u
print(u)
```

Après trois étapes, la variable ‘u‘ contient $500 \times 1{,}12^3$, soit environ 702,46.

Vigilance. Dans ‘u = 1.12*u‘, la valeur de droite est calculée avec l'ancienne valeur de ‘u‘, puis le résultat remplace l'ancienne valeur de ‘u‘.

⇒ **TD N14 – Partie A : lire un programme et compléter une trace d'exécution.**

2. Boucle bornée : répéter un nombre connu de fois

Lorsqu'on sait à l'avance combien de répétitions sont nécessaires, on utilise une boucle bornée.

Définition (Boucle bornée) Une boucle bornée répète un bloc d'instructions un nombre connu de fois. En Python, on utilise souvent ‘for‘.

Méthode Pour calculer le terme u_n d'une suite définie par récurrence, on initialise le premier terme, puis on répète la relation de récurrence le nombre de fois nécessaire.

Exemple On considère la suite définie par $u_0 = 500$ et $u_{n+1} = 1{,}12u_n$. Le programme suivant calcule u_8.

```
u = 500
for k in range(8):
    u = 1.12*u
print(u)
```

La boucle est exécutée 8 fois : on passe de u_0 à u_8.

Propriété (Instruction ‘range‘) En Python, ‘range(n)‘ produit n répétitions : les valeurs prises sont $0, 1, \ldots, n-1$. Ainsi, ‘for k in range(8) :‘ exécute la boucle 8 fois.

Vigilance. Dans ‘range(8)‘, la dernière valeur de ‘k‘ est 7, mais la boucle est bien répétée 8 fois.

⇒ **TD N14 – Partie A : boucles bornées et suites définies par récurrence.**

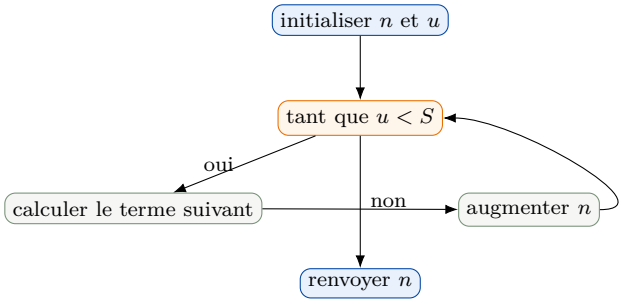
3. Boucle conditionnelle : chercher un seuil

Lorsqu’on ne sait pas à l’avance combien de répétitions sont nécessaires, on utilise une boucle conditionnelle. C’est le cas lorsqu’on cherche un seuil.

Définition (Boucle conditionnelle) Une boucle conditionnelle répète un bloc d’instructions tant qu’une condition est vraie. En Python, on utilise ‘while‘.

Méthode Pour chercher le premier rang n tel que $u_n \geqslant S$:

1. initialiser n et u ;
2. répéter tant que $u < S$;
3. mettre à jour u ;
4. augmenter n ;
5. afficher ou renvoyer n.



Exemple Pour chercher au bout de combien d’années la population dépasse 1000 :

```
n = 0
u = 500
while u < 1000:
    u = 1.12*u
    n = n + 1
print(n)
```

Propriété (Condition d’arrêt) Dans une boucle ‘while‘, la condition est testée avant chaque répétition. Quand la condition devient fausse, la boucle s’arrête.

Vigilance. Une boucle ‘while‘ doit finir par s’arrêter. Il faut donc vérifier que les variables évoluent bien vers la sortie de la boucle.

⇒ **TD N14 – Partie B : seuils, boucles ‘while‘ et conditions d’arrêt.**

4. Fonctions Python

Pour réutiliser un calcul, on l’encapsule dans une fonction. Une fonction reçoit éventuellement des paramètres et renvoie un résultat.

Définition (Fonction Python) Une fonction Python est un bloc d’instructions nommé, défini avec ‘def‘. Elle peut recevoir des paramètres et renvoyer une valeur avec ‘return‘.

Méthode Pour écrire une fonction :

1. choisir un nom explicite ;
2. indenter le bloc d’instructions ;
3. utiliser des paramètres lorsque les données peuvent changer ;
4. terminer par ‘return‘ si l’on veut renvoyer un résultat.

Exemple La fonction suivante calcule le terme u_n de la suite définie par $u_0 = 500$ et $u_{n+1} = 1{,}12u_n$.

```
def terme(n):
    u = 500
    for k in range(n):
        u = 1.12*u
    return u
```

L'appel 'terme(8)' renvoie la valeur de u_8.

Exemple La fonction suivante renvoie le premier rang à partir duquel $u_n \geqslant S$.

```
def seuil(S):
    n = 0
    u = 500
    while u < S:
        u = 1.12*u
        n = n + 1
    return n
```

Vigilance. 'print' affiche une valeur à l'écran ; 'return' renvoie une valeur utilisable dans un autre calcul. Dans une fonction, on privilégie souvent 'return'.

⇒ **TD N14 – Partie B : écrire et modifier des fonctions Python.**

5. Simuler pour conjecturer

Une simulation permet d'explorer une situation lorsque le calcul exact est long, ou lorsque l'on veut observer un phénomène avant de le démontrer.

Définition (Simulation) Une simulation est une expérience réalisée par un programme pour imiter une situation mathématique ou aléatoire. Elle ne remplace pas une preuve, mais elle peut aider à conjecturer.

Méthode Pour simuler une expérience aléatoire, on répète un grand nombre de fois l'expérience, on compte les succès, puis on calcule une fréquence.

Exemple Le programme suivant simule 1000 lancers d'une pièce équilibrée et calcule la fréquence d'apparition de pile.

```
from random import random

compteur = 0
for k in range(1000):
    if random() < 0.5:
        compteur = compteur + 1

frequence = compteur/1000
print(frequence)
```

La valeur obtenue change d'une exécution à l'autre, mais elle est généralement proche de 0,5.

Propriété (Rôle d'une simulation) Une simulation donne une indication numérique ou graphique. Elle peut suggérer une conjecture, mais une conclusion mathématique nécessite une preuve ou un raisonnement.

Vigilance. Augmenter le nombre de répétitions stabilise souvent les fréquences observées, mais ne transforme pas une simulation en démonstration.

⇒ **TD N14 – Partie C : simulations simples et interprétation de fréquences.**

6. Lire, corriger et valider un programme

En algorithmique, il ne suffit pas d'obtenir un résultat : il faut vérifier que le programme répond bien à la question posée.

Méthode Pour contrôler un programme :

1. identifier les variables ;
2. préciser ce que représente chaque variable ;
3. faire une trace sur quelques étapes ;
4. vérifier la condition d'arrêt ;
5. tester le programme sur un cas simple dont on connaît la réponse.

Exemple On veut calculer le premier rang n tel que $u_n \geqslant 1000$, pour $u_0 = 500$ et $u_{n+1} = 1{,}12u_n$. La trace commence ainsi :

étape	n	u
départ	0	500
1	1	560
2	2	626
3	3	698,6

On vérifie que n augmente d'une unité à chaque calcul d'un nouveau terme. Ainsi, lorsque la boucle s'arrête, la variable 'n' correspond bien au rang recherché.

Propriété **(Erreur fréquente)** Dans une boucle de seuil, augmenter n au mauvais endroit peut décaler la réponse d'une unité. Une trace d'exécution permet souvent de repérer cette erreur.

Vigilance. Un programme correct doit répondre à la question exacte. Il faut distinguer « dépasser strictement un seuil » et « atteindre ou dépasser un seuil ». Les conditions 'u < S' et 'u <= S' ne donnent pas toujours le même rang.

⇒ **TD N14 – Partie D : corriger, compléter et expliquer des programmes.**

7. Synthèse

A RETENIR

— Une variable stocke une valeur qui peut changer pendant l'exécution.
— Une affectation comme 'u = 1.12*u' calcule une nouvelle valeur à partir de l'ancienne.
— Une boucle 'for' sert à répéter un nombre connu de fois.
— Une boucle 'while' sert à répéter tant qu'une condition est vraie.
— Une fonction Python est définie avec 'def' et peut renvoyer une valeur avec 'return'.
— Une simulation permet d'explorer ou de conjecturer, mais elle ne constitue pas une preuve.
— Une trace d'exécution permet de vérifier le rôle des variables et de repérer des erreurs.

APPLICATIONS IMMÉDIATES

1. Compléter la trace de 'u = 2*u + 1' répétée trois fois à partir de $u = 4$.
2. Écrire une boucle 'for' qui calcule u_{10} si $u_0 = 3$ et $u_{n+1} = 2u_n + 5$.
3. Écrire une boucle 'while' qui cherche le premier rang tel que $u_n > 10000$.
4. Transformer un programme avec 'print' en fonction avec 'return'.
5. Expliquer la différence entre 'u < S' et 'u <= S' dans une boucle de seuil.
6. Dire pourquoi une simulation ne suffit pas à prouver une probabilité exacte.

BESOIN D'UNE REPRISE ?

À retravailler en priorité :
— lire une affectation ;
— suivre une trace d'exécution ;
— distinguer boucle 'for' et boucle 'while' ;
— choisir la bonne condition d'arrêt ;
— écrire une fonction avec paramètres ;
— interpréter le résultat d'un programme.

⇒ **TD N14 – Toutes les parties : traces, boucles, seuils, fonctions et simulations.**