
TP 3 : Boucles for en Python

1 Introduction

1.1 Syntaxe de la commande

En algorithmique, on peut parfois vouloir répéter un nombre donné de fois un bloc d'instruction. Par exemple pour calculer une somme, les termes d'une suite définie par récurrence, un produit,... Pour cela, on utilise une boucle `for`. La structure est la suivante :

```
1 for var in range(debut, fin, pas):  
2     instructions
```

Détaillons les éléments de cette syntaxe.

- `var` : est une variable qui prend successivement chacune des valeurs comprises entre `début` et `fin` (exclu) avec une distance de `pas`.
- `instruction` : désigne un bloc d'instruction qui va être exécuté de manière successive, pour toutes les valeurs de `var`.

Par exemple, `var in range(2,5,1)` signifie que `var` prend les valeurs 2, 3, 4.

`var in range(10,20,2)` signifie que `var` prend les valeurs 10, 12, 14, 16, 18.

A noter ! La valeur `fin` est toujours exclue.

Si la quantité `pas` n'est pas précisée, elle prendra par défaut la valeur 1. Ainsi `var in range(1,10)` signifie que `var` prend les valeurs 1, 2, 3, 4, 5, 6, 7, 8, 9.

Si la quantité `debut` n'est pas non plus précisée, elle prendra par défaut la valeur 0. Ainsi `var in range(5)` signifie que `var` prend les valeurs 0, 1, 2, 3, 4.

1.2 Illustration sur un premier exemple

Considérons l'exemple consistant à calculer les 10 premiers termes d'une suite (u_n) . Soit la suite de terme général $u_n = n^2$.

```
1 #Calcul des 10 premiers termes de la suite u_n=n^2  
2 u=np.zeros(10)  
3 for k in range(10):  
4     u[k]=k**2
```

Revenons tout d'abord sur les différents éléments de ce script :

- `u=np.zeros(10)` permet de créer un vecteur/tableau `u` de taille 10 rempli de 0, les cases étant numérotées de 0 à 9.
- `u[k]` désigne la k -ème coordonnée du vecteur `u`,
- la boucle `for` permet de stocker chacune des valeurs de la suite (u_n) dans le vecteur `u`.
Plus précisément, l'instruction `u[k]=k**2` permet de stocker la valeur k^2 dans la k -ème coordonnée du vecteur `u`.

La boucle `for` ci-dessus s'exécute comme suit :

1. initialement, la variable `k` est affectée à la valeur 0.
→ l'instruction `u[0]=0**2` est alors exécutée.

2. la variable `k` est ensuite affectée à la valeur 1 (0 + le pas de 1).
 $\hookrightarrow$ l'instruction `u[1]=2**2` est alors exécutée.
3. la variable `k` est ensuite affectée à la valeur 2 (1+1)
 $\hookrightarrow$ l'instruction `u[2]=2**2` est alors exécutée.
4. ...

A la fin de la boucle, le vecteur `u` contient les 10 (de 0 jusqu'à 9) premières valeurs de la suite de terme général $u_n = n^2$. La variable `k`, quant à elle, contient alors la valeur 9.

Ne pas oublier ! La valeur de fin (ici 10) est toujours exclue !

Attention ! La première coordonnée du vecteur `u` a le **numéro 0** !

C'est une spécificité de Python qui peut être parfois un peu (voire beaucoup !) prise de tête..

2 Calcul des m premiers termes d'une suite

2.1 Suites explicites : $u_n = f(n)$

On s'intéresse ici à la suite $(v_n)_{n \in \mathbb{N}}$ définie par :

$$\forall n \in \mathbb{N}, v_n = \frac{\sqrt{n}e^n}{(\pi + n)^2}$$

Le but est de calculer les m premiers éléments de cette suite.

- Quelle fonction `f` permet d'écrire v_n sous la forme $v_n = f(n)$?

- Coder cette fonction dans un nouveau script. On la nommera `f1`.

- Rappeler la commande permettant de construire un vecteur de taille `m` (donnée) dont tous les coefficients sont nuls.

- Ecrire une fonction :

- dont le nom est `calculSuite1`,
- qui prend une fonction `f` et un entier `m` en entrée,
- qui calcule en sortie un vecteur `u` contenant les `m` premiers termes de la suite de terme général : $u_n = f(n)$.

En début de fonction, `u` sera assigné au vecteur de taille `m` dont tous les coefficients sont nuls.

- Quel appel permet de calculer les 12 premiers éléments de la suite (v_n) définie ci-dessus ? Exécuter cet appel dans la console.

2.2 Suites récurrentes d'ordre 1 $u_{n+1} = f(u_n)$

On s'intéresse ici à la suite $(w_n)_{n \in \mathbb{N}^*}$ définie par :

$$\begin{cases} w_1 = 0 \\ \forall n \in \mathbb{N}, \quad w_{n+1} = \frac{w_n^2 + 1}{2} \end{cases}$$

Le but est de calculer les m premiers éléments de cette suite.

- Quelle fonction f permet d'écrire $w_{n+1} = f(w_n)$?

- Coder cette fonction dans un nouveau script. On la nommera **f2**.

- Ecrire une fonction :

- dont le nom est **calculSuite2**,
- qui prend une fonction **f**, un réel **init** et un entier **m** en entrée,
- qui calcule en sortie un vecteur **u** contenant les **m** premiers termes de la suite (u_n) définie par :
$$\begin{cases} u_0 = \text{init} \\ \forall n \in \mathbb{N}, \quad u_{n+1} = f(u_n) \end{cases}$$

En début de fonction, le vecteur **u** sera assigné au vecteur de taille **m** dont tous les coefficients sont nuls.

- Quel appel permet de calculer les 16 premiers éléments de la suite (w_n) définie ci-dessus ? Exécuter cet appel dans la console.

3 Représentation graphique des termes d'une suite

Pour tracer des graphes en Python, il nous faudra charger la bibliothèque `matplotlib.pyplot`. Pour cela, écrire en haut de votre script la commande

```
import matplotlib.pyplot as plt
```

— Recopier et exécuter le programme suivant :

```
1 u2=calculSuite2(f2,0,10)
2 plt.plot(u2,'+')
```

— Que constatez-vous ? Retrouvez-vous bien la valeur $w_1 = 0$?

— Recopier et exécuter le programme suivant :

```
1 x=np.arange(1,11)
2 u2=calculSuite2(f2,0,10)
3 plt.plot(x,u2,'+')
```

— A quoi correspond `x` ? Que fait la commande `np.arange(1,11)` ? Quelles sont les différences entre les commandes `plt.plot(u2,'+')` et `plt.plot(x,u2,'+')` ?

4 A vous de jouer

Exercice 1. Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie pour tout $n \in \mathbb{N}$, par $u_n = \frac{(-1)^n}{n+1}$.

1. En vous inspirant des parties précédentes, calculer les 30 premiers termes de la suite.
2. Représenter graphiquement les 30 premiers termes de cette suite
3. Vers quelle valeur semble converger la suite ?

Exercice 2. Soit $(u_n)_{n \in \mathbb{N}}$ la suite définie par $u_0 = 3$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \frac{2u_n + 1}{u_n + 2}$.

1. En vous inspirant des parties précédentes, calculer les 15 premiers termes de la suite.
2. Représenter graphiquement les 30 premiers termes de cette suite.

Exercice 3. Soit la suite $(u_n)_{n \in \mathbb{N}}$ définie par $u_0 = 0$, $u_1 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+2} = u_n + u_{n+1}$.

1. Ecrire une fonction nommée `calculFibo` permettant de calculer les m premiers termes de cette suite pour un entier m donné en argument d'entrée.
2. Représenter graphiquement les 30 premiers termes de cette suite.
3. **Sur feuille** Trouver l'expression de u_n en fonction de n .

5 Calculs de sommes et produits

Garder en mémoire tous les termes d'une suite n'est pas toujours utile et occupe parfois inutilement la mémoire de l'ordinateur.

Ecrire un programme Python qui étant donné $init$ et f renvoie le n -eme terme de la suite (u_n) définie par $\begin{cases} u_0 = init \\ u_{n+1} = f(u_n) \end{cases}$ sans garder en mémoire tous les autres termes.

Les boucles `for` sont également utile pour calculer les sommes ou produits. En effet, si l'on veut étudier $\sum_{k=0}^n u_k$, on peut noter $S_n = \sum_{k=0}^n u_k$ et regarder la suite (S_n) . On a alors la relation

$$S_{n+1} = S_n + u_{n+1}$$

Par exemple, si l'on veut un programme qui calcule $\sum_{k=0}^n k^4$, on peut utiliser le programme suivant :

Exemple 5.0.1.

```
1 #Calcul de somme de k=0 à n (k^4)
2 S=0
3 for k in range(n+1):
4     S=S+k**4
```

Exercice 4.

En s'inspirant de ce programme écrire un programme Python permettant de calculer $\sum_{k=0}^n (4k^5 + e^k - 2)$ pour $n \in \mathbb{N}$ donné.

Exercice 5.

Adapter le programme pour calculer $\sum_{k=5}^{2n} \ln(k) + \cos(k)$ pour $n \geq 5$ donné.

Exercice 6.

Adapter le programme pour écrire un programme qui prend $n \in \mathbb{N}^*$ et qui renvoie $n!$.