Lycée Marcelin BERTHELOT
BCPST 1C- Madame MOREL

Boucle FOR

Les boucles permettent de répéter une suite d'instructions. Il en existe deux types : les boucles itératives `for` et les boucles conditionnelles `while`.
Dans ce cours, on s'intéresse à la boucle `for`

1 Boucle FOR

1.1 range()

`range(b)` est l'ensemble des valeurs de `0` inclus à `b` exclu (entiers entre `0` et `b-1`)
`range(a, b)` est l'ensemble des valeurs de `a` inclus à `b` exclu (entiers entre `a` et `b-1`))
`range(a, b, c)` est l'ensemble des valeurs de `a` inclus à `b` exclu en avançant (ou reculant si $c < 0$) de `c` en `c` (c'est `a`, `a+c`, `a+c+c`, ...).

Exemple : `range(11)` `range(1,20)` `range(12,0,-1)`

Remarque : combien y a-t-il d'entiers "dans" `range(n)`?

1.2 Syntaxe

```
for i in range(a,b,c):
   [bloc instructions
stop: fin indentation
```

- Comme pour l'instruction `if`, les **deux points :** au bout de la ligne du `for` et **l'indentation** du bloc d'instructions sont obligatoires!!
- La variable boucle (ou compteur) n'a pas d'obligation à s'appeler i. C'est une variable muette, et peut donc prendre n'importe quel nom (sauf celui d'une variable qui existe déjà...)

Exemple: quelles sont les valeurs de `a` et `b` à la fin de chaque algorithme?

1.
```
a=1;b=2
for k in range(5):
   print(a)
print(b)
```

2.
```
a=1;b=2
for k in range(5):
   print(a)
   print(b)
```

Exemple: quelle est la valeur de `a` à la fin de chaque algorithme?

1.
```
a=1
for k in range(1,10):
   a=a*k
print(a)
```

2.
```
for k in range(1,10):
   a=1
   a=a*k
print(a)
```

Remarque : Lorsque le nombre d'itérations est connu à l'avance, on utilise une boucle `for`.

Exercice 1 Écrire un code Python qui :

1. affiche les 20 premières puissances de 2, c'est-à-dire de 2^0 à 2^{19} inclus.
2. affiche les 50 premiers entiers naturels dans l'ordre **croissant**
3. affiche les 50 premiers entiers naturels dans l'ordre **décroissant**

2 Calcul de sommes et de produits

2.1 Calcul d'une somme

Une suite u étant donnée, le but de cette partie est d'écrire un algorithme qui permet de calculer la somme $S_n = \sum_{k=i}^{n} u_k$ ($i = 0$ ou 1 en général).
L'idée est de réécrire cette somme en $n - i + 1$ (nombre de termes de la somme) itérations qui vont pouvoir être programmées par une boucle `for` **si** n **est connu**.
On a la relation de récurrence suivante ($n - i + 1$ itérations):

$$\begin{cases} S_i = u_i \\ \text{pour tout } k \text{ tq } i+1 \leqslant k \leqslant n\,,\ S_k = S_{k-1} + u_k \end{cases}$$

Algorithme: on prend une variable s, initialisée à 0, qui prend les valeurs de S_i, S_{i+1}, ..., S_n.
Exemple: Écrire une fonction `somme(n)` qui calcule et affiche la valeur de la somme arithmétique $S_n = \sum_{k=0}^{n} k$ pour tout entier naturel n.

Exercice 2 Écrire une fonction de paramètre `n` qui renvoie la valeur de la somme:

1. $\sum_{k=1}^{n} k^{\sqrt{k}}$
2. $\frac{1}{n} \sum_{k=1}^{n} \frac{1}{k^2}$

Algorithme: calcul de $\sum_{k=i}^{n} u_k$.

```
def somme(n):
    s=0
    for k in range(i,n+1):
        s=s+<valeur uk>
    return s
```

Puisque c'est la valeur **finale** de s qui contient la somme S_n, il suffit juste d'afficher cette valeur et non toutes les valeurs "intermédiaires" de s.

2.2 Calcul d'un produit

Une suite u étant donnée, le but de cette partie est d'écrire un algorithme qui permet de calculer le produit $P_n = \prod_{k=i}^{n} u_k$ ($i = 0$ ou 1 en général).
L'idée est de réécrire ce produit en $n - i + 1$ (nombre de termes du produit) itérations qui vont pouvoir être programmées par une boucle `for` **si** n **est connu**.
On a la relation de récurrence suivante ($n - i + 1$ itérations):

$$\begin{cases} P_i = u_i \\ \text{Pour tout } k \text{ tq } i+1 \leqslant k \leqslant n\,,\ P_k = P_{k-1} \times u_k \end{cases}$$

Algorithme: on prend une variable p, initialisée à 1, qui prend les valeurs de P_i, P_{i+1}, …, P_n.
Exemple: écrire un code Python qui calcule $n!$ pour tout entier naturel n.

Exercice 3 $\forall n \in \mathbb{N}^*$, on définit le produit $P_n = \prod_{k=1}^{n} \frac{k^2}{k^2+1}$. Écrire une fonction qui renvoie la valeur de P_n pour tout entier naturel n non nul.

Algorithme: calcul du produit $\prod_{k=i}^{n} u_k$.

```
def produit(n):
   p=1
   for k in range(i,n+1):
      p=p*<valeur uk>
   return p
```

Puisque c'est la valeur **finale** de p qui contient le produit P_n, il suffit juste d'afficher cette valeur et non toutes les valeurs "intermédiaires" de p.

3 Suites récurrentes: calcul du terme d'ordre n

Il s'agit d'écrire une fonction qui calcule et affiche la valeur de u_n pour tout n.

3.1 Suites récurrentes d'ordre 1

Certaines suites sont définies par leur premier terme u_{n_0} et une relation de récurrence de la forme:

$$\forall n \geqslant n_0 + 1\,,\ \boxed{u_n = f(n, u_{n-1})}$$

Très souvent, n_0 vaut 0 ou 1 dans les exercices.

Exemple: $\begin{cases} u_1 = 1 \\ \forall n \geqslant 2\,,\ u_n = \sqrt{n + u_{n-1}} \end{cases}$

L'algorithme permettant de calculer les n premiers termes de telles suites comporte deux parties:

1. **Initialisation:**

   ∗ Donner une valeur à n (variable n),

   ∗ Donner une valeur à u_{n_0} (variable u).

2. **Boucle for:**

   ∗ Dans la boucle, la valeur de u_k est contenue dans la nouvelle variable u, la valeur de u_{k-1} étant celle de l'ancienne variable u.

   ∗ **Lorsque le compteur prend la valeur k, la variable u contient la valeur de** u_k.

Algorithme:

```
def terme(n):
   u=<valeur un0>
   for k in range(n0+1,n+1):
      u=<valeur uk>
   return u
```

Exercice 4 On considère la suite $(u_n)_{n\geqslant 0}$ définie par $u_0 = e - 1$ et la relation de récurrence:

$$\forall n \geqslant 1\,,\ u_n = n\,u_{n-1} - 1\,.$$

Écrire une fonction de paramètre `n` qui calcule et affiche le $n$ième terme de la suite.

3.2 Suites récurrentes d'ordre 2

Ce paragraphe concerne les suites dont les deux premiers termes u_{n_0} et u_{n_0+1} sont donnés, et qui sont définies par une relation de récurrence (donnée ou à trouver) de la forme:

$$\forall n \geqslant n_0 + 2\,, \; \boxed{u_n = f(n, u_{n-1}, u_{n-2})}$$

Exemple: $\begin{cases} u_0 = 1\,,\; u_1 = 2 \\ \forall n \geqslant 2\,,\; u_n = 3\,u_{n-1} - u_{n-2} \end{cases}$

Algorithme:

```
def terme (n):
   w=un0 ; v=un0+1
   for k in range(n0+2,n+1):
      u=<valeur uk>
      w=v
      v=u
   return u
```

- Pour programmer, on utilise trois variables:
  $*$ Avant l'entrée dans la boucle, v contient u_{k-1} et w contient u_{k-2}.
  $*$ Dans la boucle, u contient u_k.
- ATTENTION!! Dans la ligne `w=v; v=u` il faut bien respecter **l'ordre** des affectations, sinon le résultat est faux ...

Exercice 5 On considère la suite (u_n) définie par: $u_0 = 4$, $u_1 = 10$, et $\forall n \geqslant 2$, $u_n = \dfrac{1}{n}\,u_{n-1} - \dfrac{n}{2}\,u_{n-2}$.
Écrire une fonction qui renvoie la valeur de u_n pour tout entier $n \geqslant 2$.