TP 2 : Les boucles **for**

PCSI, lycée Bertran de Born

1 La boucle for

Les boucles **for** permettent de réaliser la répétition d'un bloc d'instruction lorsque l'on connaît à l'avance le nombre d'itérations à effectuer. En Python, la syntaxe est la suivante :

```
for variable in structure_iterable :
    instructions
```

Pour vous faire la main, implémentez dans le fichier *temp.py* le programme suivant, puis exécutez-le :

```
for i in range(0,20,2):
    print(i)
```

Dans ce programme exclusivement formé d'une boucle for :

- La variable i prend pour commencer la valeur initiale 0.
- L'instruction `print(i)` est réalisée et le programme affiche la valeur de i à savoir 0.
- La variable i est incrémentée de 2 et vaut alors 2.
- L'instruction `print(i)` est réalisée et le programme affiche la valeur de i à savoir 2.
- La variable i est incrémentée de 2 et vaut alors 4.
- L'instruction `print(i)` est réalisée et le programme affiche la valeur de i à savoir 4.
- ...
- L'instruction `print(i)` est réalisée et le programme affiche la valeur de i à savoir 18.
- La variable i est incrémentée de 2 et vaut alors 20.
- L'instruction `print(i)` n'est pas réalisée puisque i a atteint sa valeur finale
- Le programme s'arrête.

Remarque : On aurait pu s'attendre à ce que la dernière valeur affichée soit 20 et non 18. Cela vient du fait que la condition de départ est inclusive (on commence à 0 inclus) et la condition d'arrêt est exclusive (on s'arrête à 20 exclus). Ceci est clairement indiqué au moment où vous tapez `range(` : En effet, à ce moment là, un cadre d'aide apparaît dans lequel sont précisés les différents arguments de la fonction range et leur utilisation.

Voici un autre programme très simple à implémenter utilisant une boucle **for** :

```
k=int(input("Jusqu'à combien voulez-vous que j'affiche les carrés ?"))
for i in range(0,k+1):
    c=i**2
    print(str(i)+" * "+str(i)+" = "+str(c))
```

2 Utilisation des boucles for pour calculer les termes d'une suite

2.1 Suite définie de manière explicite

On considère la suite (u_n) définie par $\forall n \in \mathbb{N}$, $u_n = n^2 + n + 3$.

Le programme suivant permet de calculer et d'afficher les 5 premiers termes de la suite :

```
for i in range(0,5):
    u=i**2+i+3
    print(u)
```

Remarque : Le programme a effectivement affiché 5 termes : les termes de u_0 jusqu'à u_4

2.2 Suite définie par récurrence

On considère la suite (u_n) définie par $u_0 = 2$ et $\forall n \in \mathbb{N}$, $u_{n+1} = 0.5 \times u_n + 5$.

1. A votre avis, sans implémenter ces programmes dans Python, lequel de ces programmes permet de calculer et d'afficher les 20 premiers termes de la suite ?

```
u=2
print(u)
for i in range(1,20):
    u=0.5*u+5
print(u)
```

```
u=2
print(u)
for i in range(1,20):
    u=0.5*u+5
    print(u)
```

```
u=2
print(u)
for i in range(1,19):
    u=0.5*u+5
    print(u)
```

2. Implémenter les 3 programmes dans Python pour vérifier votre résultat
3. Proposer un programme qui demande à l'utilisateur un entier naturel n et qui affiche uniquement la valeur de u_n

Remarque : On retiendra qu'en Python, l'**indentation** (c'est-à-dire le fait de décaler le code grâce à la touche tabulation) est essentielle, puisqu'une simple différence d'indentation donne un programme différent. Le bloc d'instructions associé à une boucle `for` doit donc toujours être **indenté**

3 Différentes manières de faire une boucle for

3.1 Avec la fonction range

Comme nous venons de le voir, pour parcourir une liste d'entiers, on utilise l'itérable `range`. La syntaxe générale est la suivante :

```
for variable in range(debut,fin,pas):
    instructions
```

- le premier argument `début` donne l'entier de départ (cet argument est optionnel et vaut 0 par défaut)
- le plus grand élément est `fin-1`
- `pas` indique l'écart entre un élément et son successeur immédiat (cet argument est optionnel et vaut 1 par défaut).

3.2 Avec une chaîne de caractères

Implémentez et exécutez le programme suivant afin d'en observer le résultat :

```
for m in 'bonjour':
    print(m)
```

Considérons par exemple la chaîne de caractères `msg = 'chaine'`. Pour la parcourir, on peut :

- utiliser la syntaxe générale `for l in msg` : dans ce cas `l` prend successivement les valeurs `'c'`,`'h'`, ..., `'e'`.
- utiliser un indice `k` avec la syntaxe `for k in range(len(msg))` : on accède alors aux caractères de `msg` avec l'expression `msg[k]`.

3.3 Avec une liste

Implémentez et exécutez le programme suivant afin d'en observer le résultat :

```
for truc in ('a',12,'bcd',(0,1)):
    print(truc)
```

3.4 Sans rien !

Si l'on veut exécuter plusieurs fois une instruction ne dépendant d'aucune variable, autant éviter d'introduire une variable et les affectations qui vont avec. Cela est possible avec la syntaxe suivante, qui permet dans cet exemple précis d'afficher 10 fois la chaîne de caractères `'ab'`

```
for _ in range(10):
    print('ab')
```

Exercices de synthèse

Exercice 1. Soit (u_n) la suite définie par $u_1 = 12$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \sqrt{u_n} + 3$

Ecrire un programme qui demande à l'utilisateur un entier n et qui affiche la valeur de u_n

Exercice 2.

Ecrire un programme qui affiche les carrés de tous les nombres inférieurs à 200 qui terminent par un 5

Exercice 3.

Soit (u_n) la suite définie par $u_0 = 3$ et $\forall n \in \mathbb{N}$, $u_{n+1} = u_n + n - 12$.

Ecrire un programme qui affiche tous les termes jusqu'à u_{30}. Le programme devra afficher des phrases du type : « $u(1) = -9$ »

Exercice 4.

Soit (u_n) la suite définie par $u_1 = 3$ et $\forall n \in \mathbb{N}$, $u_{n+1} = u_n^2 - 2n - 10$

Ecrire un programme qui demande à l'utilisateur un entier n et qui calcule et affiche le terme général u_n ainsi que la somme S_n des n premiers termes.

Exercice 5. Ecrire un programme qui affiche un rectangle de `n` lignes sur `p` colonnes rempli de caractères * sur le modèle suivant (pour `n=3` et `p=5`) :

```
*****
*****
*****
```

Exercice 6. Ecrire un programme qui affiche un triangle de taille `n` constitué de caractères * sur le modèle suivant (pour `n=3`) :

```
  *
 ***
*****
```

Exercice 7.

Voici un exercice sur le thème des suites :

On considère les suites (u_n) et (v_n) définies par

$$u_0 = 4,\ \forall n \in \mathbb{N},\ u_{n+1} = \frac{u_n^2 + 9}{2u_n} \text{ et } v_n = \frac{u_n - 3}{u_n + 3}$$

1. Montrer que les suites (u_n) et (v_n) sont bien définies.
2. Démontrer que $\forall n \in \mathbb{N}$, $v_{n+1} = v_n^2$
3. Calculer v_0 et v_1.
4. Montrer que $\forall n \in \mathbb{N}$, $v_n = (v_0)^{2^n}$
5. En déduire l'expression de u_n en fonction de n.

Après avoir fait entièrement cet exercice, écrire un programme qui calcule et affiche les 10 premiers termes de suites (u_n) et (v_n), puis vérifier informatiquement que l'expression trouvée de u_n en fonction de n est correcte en comparant les 10 premiers termes de la suite (u_n) avec les 10 premiers termes trouvés grâce à cette expression.