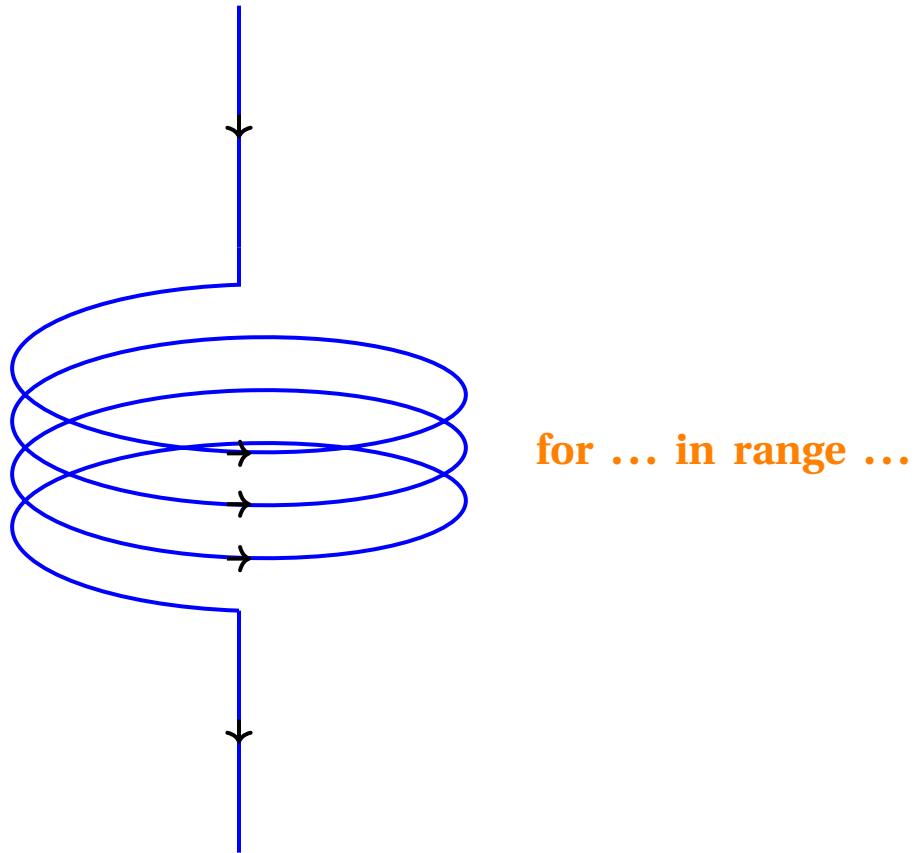


T.P. chapitre 6 : la boucle for ... in ...



```
1  >>> for i in range(1, 5):  
2      ...     print(i)  
3      1  
4      2  
5      3  
6      4  
7  >>>
```

1 Pour découvrir

<http://www.algoprogram.fr/06-qcm/qcm.php?contenu=8&titre=La%20boucle%20for%20...%20in%20...%201>

Penser à réaliser des tests dans une console Python !

2 La boucle for ... in ... en Python

En anglais, **range** signifie ampleur, gamme, envergure, étendue.

Même si cette traduction n'est pas correcte, nous pourrons penser à un intervalle.

Exercice 1 :

1. Tester dans la console les deux lignes suivantes :

```
1  >>> for compteur in range(1, 10) :  
2      ...     print(compteur)
```

2. Par quelles valeurs passe la variable `compteur` ?

Exercice 2 :

En utilisant une boucle for, écrire dans la console des instructions permettant d'afficher :

- les entiers de 1 à 10,
- les entiers de 1 à 100,
- les entiers de 5 à 15,
- les nombres pairs de 0 à 20.

Exercice 3 :

Écrire dans l'éditeur une fonction **jusque(n)** :

- prenant un nombre entier *n* en argument,
- affichant successivement tous les entiers de 0 à *n*.

Exercice 4 :

Écrire dans l'éditeur une fonction **affiche_entre(a, b)** :

- prenant deux nombres entiers *a* et *b* en argument,
- affichant successivement tous les entiers de *a* jusqu'à *b*.

★ Synthèse : syntaxe d'une boucle for

| Langage naturel | Python |
|--|---|
| pour variable allant de 1 à n faire bloc d'instructions | for variable in range(1, n+1): bloc d'instructions |

Remarques :

- La variable prend successivement les valeurs de 1 à n.
- Nous remarquons que pour aller jusqu'à *n*, nous écrivons *n + 1* : `range(1, n+1)`.
Nous pourrons penser à un intervalle fermé à gauche, et **ouvert à droite** : `[1 ; n + 1[`.
- Les **deux points** en fin de première ligne introduisent le bloc d'instructions à répéter.
Celui-ci doit être **indenté**.
- La fin de l'indentation signale la fin du bloc d'instruction à répéter.

3 Application à la résolution de problèmes

Exercice 5 : permis de conduire

Afin de passer son permis de conduire dans quelques années, une élève place la somme de 400€. Celle-ci est rémunérée à l'issue de chaque année au taux de 3% (pourcentage calculé sur le montant présent sur le compte pour l'année en cours).

Écrire une fonction **permis(n)** :

- prenant en argument un nombre d'années n ,
- renvoyant la somme présente sur le compte au bout de n années de placement.

Exercice 6 : le marchand de farine

Au XVIII^{ème} siècle, un négociant en farine vivant à Sète décide de se rendre à Toulouse, pour y vendre sa farine. Il emprunte pour son périple le canal du Midi, qui relie la mer Méditerranée et la Garonne.

Le canal traverse 63 écluses. A chaque écluse, le négociant doit donner 1 % de son chargement en péage royal, puis échanger 3 sacs de farine contre de la nourriture.

a. Écrire une fonction **farine(sac)** :

- prenant en argument le nombre de sacs dont il dispose au départ de Sète,
- renvoyant le nombre de sacs qu'il lui restera à son arrivée à Toulouse.

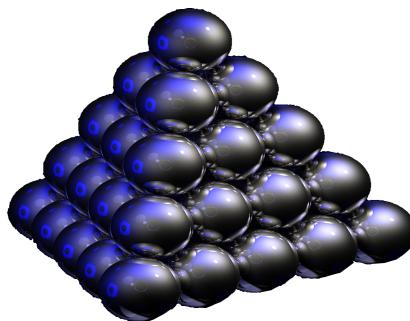
b. Écrire une seconde fonction, **minimum()** :

- sans argument en entrée,
- renvoyant le nombre minimum de sacs dont il faut disposer au départ pour arriver jusqu'à Sète avec au moins 500 sacs à vendre.

c. N'y-a-t-il pas un anachronisme⁽¹⁾ dans cet exercice ?

Exercice 7 : un problème d'empilement de sphères

On empile des sphères comme sur la figure ci-dessous pour former une pyramide à base carrée.



- Combien de sphères faut-il pour réaliser une pyramide à 2 étages? 3 étages?
- Sachant qu'il faut 55 sphères pour former une pyramide à 5 étages, combien en faudra-t-il pour former une pyramide à 6 étages?
- Écrire une fonction **pyramide(n)** :
 - prenant en argument un entier naturel n ,
 - renvoyant le nombre de sphères nécessaires pour réaliser une pyramide à n étages.
- En déduire le nombre d'étages de la plus haute pyramide que l'on peut réaliser avec 1000 sphères.

(1). <https://fr.wikipedia.org/wiki/Anachronisme>

4 Applications en arithmétique

4.1 L'opérateur modulo %

Il existe une infinité de nombres entiers naturels : 0, 1, 2, 3 ...

En mathématiques et en informatique, il sera souvent utile de travailler dans des structures ne contenant, par exemple, que 10 entiers : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Après, nous considérerons que :

- 10, c'est 0,
- 11, c'est 1,
- 12, c'est 2,
- 13, c'est 3,
- ...

Cela peut sembler curieux ou inutile. Cependant, il est intéressant de remarquer que nous comptons déjà modulo 24 depuis la petite enfance. En effet, nous ne numérotions les heures que de 0 à 23, et revenons à 0 toutes les 24 heures.

Nous comptons ainsi modulo 24 :

- au bout de 24 heures, il est 0 heure : $24 \% 24 = 0$
- au bout de 25 heures, il est 1 heure : $25 \% 24 = 1$
- au bout de 26 heures, il est 2 heures : $26 \% 24 = 2$
- au bout de 48 heures, il est 0 heure : $48 \% 24 = 0$
- ...

Définition :

- Soient a et b deux nombres entiers naturels tels que $b \neq 0$.

Effectuer la **division euclidienne** de a par b , c'est trouver les deux entiers naturels q et r tels que :

$$a = b \times q + r \quad \text{et} \quad 0 \leq r < b.$$

- Le nombre q est appelé le **quotient** de la division euclidienne de a par b .
- Le nombre r est appelé le **reste** de la division euclidienne de a par b .

Syntaxe :

En Python, le nombre $a \% b$ est le **reste de la division euclidienne** de a par b .

Exercice 8 :

Prévoir les résultats renvoyés par les instructions suivantes. Vérifier si besoin dans la console.

| | | | | | | | | | |
|-----|------------|-----|------------|-----|------------|-----|------------|------|-------------|
| do. | $3 \% 2$ | ré. | $4 \% 2$ | mi. | $5 \% 2$ | fa. | $6 \% 2$ | sol. | $0 \% 2$ |
| la. | $1 \% 2$ | si. | $2 \% 2$ | do. | $7 \% 2$ | | | | |
| a. | $13 \% 10$ | b. | $8 \% 10$ | c. | $27 \% 10$ | d. | $3 \% 10$ | e. | $134 \% 10$ |
| f. | $4 \% 5$ | g. | $25 \% 5$ | h. | $53 \% 5$ | | | | |
| 1. | $11 \% 13$ | 2. | $13 \% 13$ | 3. | $14 \% 13$ | 4. | $15 \% 13$ | 5. | $25 \% 13$ |
| 6. | $26 \% 13$ | 7. | $27 \% 13$ | 8. | $28 \% 13$ | | | | |

4.2 Divisibilité et parité

Exercice 9 :

Soit n un entier naturel.

- a. Compléter l'équivalence suivante en utilisant l'opérateur `%` :

n est divisible par 3 \Leftrightarrow

- b. Écrire une fonction `divisible_par_3(n)` :

- prenant en argument un entier naturel n ,
- et renvoyant : • `True`, si n est divisible par 3,
- `False`, sinon.

- c. Tester votre fonction avec les nombres suivants : 9, 7, 21, 341, 351.

- d. Rappeler le critère de divisibilité par 3 étudié au collège.

- e. Vérifier que nous obtenons bien les mêmes résultats que dans la question c. à l'aide de ce critère.

Exercice 10 :

- a. Écrire une fonction `pair(n)` :

- prenant en argument un entier naturel n ,
- et renvoyant : • `True`, si n est pair,
- `False`, sinon.

- b. Rappeler le critère de divisibilité par 2 étudié au collège.

Exercice 11 :

Soit n un entier naturel.

- a. Compléter l'équivalence suivante en utilisant l'opérateur `%` :

a est divisible par b \Leftrightarrow

- b. Écrire une fonction `divisible(a,b)` :

- prenant en argument un entier naturel a et un entier naturel b non nul,
- et renvoyant : • `True`, si a est divisible par b ,
- `False`, sinon.

- c. Utiliser la fonction de la question b. pour tester si :

- | | | |
|------------------------------|------------------------------|-----------------------------|
| • 12 est divisible par 4? | • 9 est divisible par 4? | • 81 est divisible par 9? |
| • 144 est divisible par 12? | • 169 est divisible par 13? | • 400 est divisible par 20? |
| • 8043 est divisible par 17? | • 8041 est divisible par 17? | |

4.3 Déterminer si un nombre est premier

Définition :

Un nombre entier naturel est **premier** s'il admet exactement deux diviseurs positifs : 1 et lui-même.

Exemples :

- 6 n'est pas premier : il admet quatre diviseurs positifs : 1, 2, 3, 6.
- 7 est premier : il admet deux diviseurs positifs : 1 et lui-même.
- 1 n'est pas premier : il n'admet qu'un diviseur positif : lui-même.

| Nombre | Diviseurs |
|--------|-----------|
| 6 | 6·3·2·1 |
| 7 | 7·1 |
| 8 | 8·4·2·1 |
| 9 | 9·3·1 |

Exercice 12 : une fonction pour déterminer si un nombre est premier

- a. Compléter les algorithmes ci-dessous :

Algorithme 1 : divisible(a, b)

```
1: si .....  
2:     retourner .....  
3: sinon  
4:     retourner .....
```

Algorithme 2 : premier(n)

```
1: reponse ← Vrai  
2: pour compteur allant de ... à ... faire  
3:     si .....  
4:         .....  
5: retourner reponse
```

- b. Copier-coller la fonction **divisible(a, b)**, écrite dans l'exercice précédent.

- c. Écrire une fonction **premier(n)** :

- prenant en argument un entier naturel $n \geq 2$,
- renvoyant :
 - **True** si n est premier,
 - **False** sinon.