Exercices du chapitre 6 : la boucle `for`

1 La boucle for

Exercice 1 :

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

Qu'affiche le programme ci-dessus lors de son exécution ?

.....

Exercice 2 :
Dans chacun des cas suivants, proposer une séquence d'instructions permettant d'afficher les nombres demandés :

1. Les nombres entiers de 0 à 10.

.....

.....

2. Les nombres entiers de 5 à 19.

.....

.....

3. Les nombres pairs de 0 à 20.

.....

.....

4. Les nombres impairs de 1 à 15.

.....

.....

5. La table de multiplication de 7 : de 7 à 70.

.....

.....

Exercice 3 :

Écrire une fonction **affiche_jusque(n)** :

- prenant en argument un entier n,
- affichant les entiers de 1 à n.

.....

.....

.....

.....

Exercice 4 :

Écrire une fonction **affiche_entre(a, b)** :

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

.....

.....

.....

.....

Exercice 5 :

1. On donne la fonction ci-dessous :

```
1  def ajoute_deux(x, n):
2      nombre = x
3      for compteur in range(1, n+1):
4          nombre = nombre + 2
5      return nombre
```

Que renvoie : `>>> ajoute_deux(5, 3)` ?

.....

Que fait cette fonction ?

.....

2. Écrire une fonction **multiplie_par_deux(x, n)**

- multipliant n fois le nombre x par 2,
- renvoyant le résultat finalement obtenu.

.....

.....

.....

.....

.....

.....

2 Application à la résolution de problèmes

Exercice 6 :
Un adolescent place la somme de 350€ sur un compte bancaire au 1^{er} Janvier de l'année 0.
Chaque année, à partir du 1^{er} Janvier de l'année 1, ses grand-parents versent la somme de 85€ sur ce compte d'épargne.

Écrire une fonction **epargne(n)** :

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Exercice 7 :
Un adolescent place la somme de 450€ sur un compte bancaire au 1^{er} Janvier de l'année 0.
Chaque année, à partir du 1^{er} Janvier de l'année 1, il touche 3% d'intérêts :
c'est-à-dire 3% de la somme présente sur le compte.

Écrire une fonction **epargne(n)** :

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

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Écrire une fonction **seuil(k)** :

- prenant en argument une somme k,
- renvoyant la première année où la somme présente sur le compte dépasse k €.

.....

.....

.....

.....

.....

.....

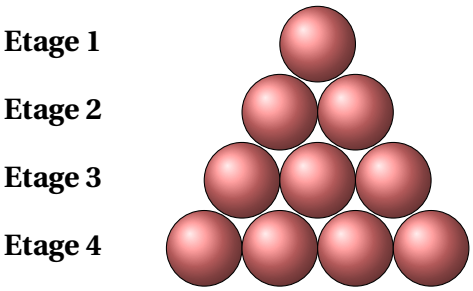
.....

.....

.....

Exercice 8 :

On empile des balles de la façon suivante :



1. Combien faut-il de balles pour réaliser une pyramide à 5 étages ?

.....

2. Écrire une fonction **triangle(n)** :

- prenant en argument un entier naturel `n`,
- renvoyant le nombre de balles nécessaires pour réaliser une construction à n étages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

3 Applications en arithmétique

3.1 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 numérotons 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 :
  $$\boldsymbol{a = b \times q + r} \quad \text{et} \quad \boldsymbol{0 \leq r < b}.$$
- Le nombre $\boldsymbol{q}$ est appelé le **quotient** de la division euclidienne de $\boldsymbol{a}$ par $\boldsymbol{b}$. .
- Le nombre $\boldsymbol{r}$ est appelé le **reste** de la division euclidienne de $\boldsymbol{a}$ par $\boldsymbol{b}$.

Syntaxe :
En Python, le nombre **`a%b`** est le **reste de la division euclidienne** de a par b.

Exercice 9 :
révoir 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`				

3.2 Divisibilité et parité

Exercice 10 :
Soit n un entier naturel.

1. Compléter l'équivalence suivante en utilisant l'opérateur % :

n est divisible par 5 $\Leftrightarrow$ .....

2. Écrire une fonction **divisible_par_5(n)** :
- prenant en argument un entier naturel `n`,
- et renvoyant : • `True`, si n est divisible par 5,
  • `False`, sinon.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Exercice 11 :
Soit n un entier naturel.

1. Compléter l'équivalence suivante en utilisant l'opérateur % :

n est divisible par 7 $\Leftrightarrow$ .....

2. Écrire une fonction **divisible_par_7(n)** :
- prenant en argument un entier naturel `n`,
- et renvoyant : • `True`, si n est divisible par 7,
  • `False`, sinon.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Exercice 12 :

Soit n un entier naturel.

1. Compléter l'équivalence suivante en utilisant l'opérateur % :

a est divisible par b $\Leftrightarrow$ .....

2. 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.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

3.3 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

Exercice 13 : une fonction pour déterminer si un nombre est premier

1. 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
```

2. Écrire en Python la fonction **divisible(a, b)**.

3. Écrire une fonction **premier(n)** :

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