

# EXERCICES

## 1. Répéter l'affichage du contenu d'une liste

Soit une liste  $L$  d'entiers (entre 0 et 9, pour des raisons d'esthétique de l'affichage). On demande d'afficher  $n$  fois cette liste où  $n > 0$  est un entier donné. Par exemple, si  $L$  est la liste de contenu :

1 4 8 0 9 6 3 2 8

et si  $n = 6$ , l'affichage obtenu doit être :

1 4 8 0 9 6 3 2 8  
2 4 8 0 9 6 3 2 8  
3 4 8 0 9 6 3 2 8  
4 4 8 0 9 6 3 2 8  
5 4 8 0 9 6 3 2 8  
6 4 8 0 9 6 3 2 8

Deux « cases » successives seront séparées d'une espace.

## 2. Damier de nombres

On dit que deux entiers  $a$  et  $b$  ont même parité si  $a$  et  $b$  sont tous les deux pairs ou bien si  $a$  et  $b$  sont tous les deux impairs. Par exemple, 81 et 31 ont même parité, 12 et 82 ont même parité et 81 et 12 ont des parités différentes.

- Ecrire un code qui à partir de deux entiers calcule la valeur d'une variable **memParite** qui vaut **True** si les deux entiers ont même parité, et **False** sinon.
- Réaliser un programme qui affiche un damier de forme carrée, de côté de longueur  $n > 0$  et rempli alternativement du nombre 4 et du nombre 2. La case en haut à gauche sera toujours 4. Par exemple, pour  $n = 7$ , le damier aura l'allure suivante :

1 4 2 4 2 4 2 4  
2 2 4 2 4 2 4 2  
3 4 2 4 2 4 2 4  
4 2 4 2 4 2 4 2  
5 4 2 4 2 4 2 4  
6 2 4 2 4 2 4 2  
7 4 2 4 2 4 2 4

## 3. Somme de sommes

Soit la suite  $(U)$  des entiers positifs qui ne s'écrivent, en base 10, qu'avec le chiffre 1 :

1, 11, 111, 1111, 11111, ...

On remarquera qu'un tel nombre est une somme de puissances de 10, par exemple pour *mille cent-onze* :

$$1111 = 10^0 + 10^1 + 10^2 + 10^3 = 1 + 10 + 100 + 1000.$$

- Soit  $n \geq 1$  un entier donné. Calculer, avec une boucle **for**, le  $n$ -ième nombre de la suite  $(U)$  (on ne se contentera pas d'afficher le nombre mais on le placera dans une variable). Par exemple, si  $n = 4$ , le programme doit calculer le nombre 1111.
- Calculer la somme des  $N$  premiers nombres de la suite  $(U)$ . Par exemple, si  $N = 20$  :

$$1 + 11 + 111 + 1111 + 11111 + \dots + 11111111111111111111$$

Pour cela, on placera une partie du code de la question précédente dans le corps d'une boucle **for**.

#### 4. Somme qui vaut 42

On donne deux listes d'entiers L et M. Créer un booléen **somme42** qui vaut True si la somme de deux nombres, l'un dans L et l'autre dans M, vaut 42. Sinon, le booléen vaudra False. Voici quelques exemples de comportements attendus :

<sup>1</sup> [17, 22, 5, 5, 33, 8] [34, 8, 20] -> True  
<sup>2</sup> [6, 22, 5, 5, 33, 8] [35, 8, 25] -> False

#### 5. Afficher les effectifs d'une liste d'entiers

On donne une liste L d'entiers, par exemple

L = [81, 31, 81, 12, 81, 9, 12, 65]

On demande d'afficher le nombre de fois que les différents nombres de la liste L apparaissent dans L. L'ordre d'affichage devra respecter l'ordre d'apparition dans la liste L. Avec l'exemple ci-dessus, le programme devra afficher

<sup>1</sup> 81 : 3  
<sup>2</sup> 31 : 1  
<sup>3</sup> 12 : 2  
<sup>4</sup> 9 : 1  
<sup>5</sup> 65 : 1  
 La ligne

<sup>1</sup> 12 : 2

de l'affichage signifie juste que 12 apparaît 2 fois dans la liste L.

#### 6. Multiple de 42 qui soit somme de deux carrés

Un carré <sup>1</sup> est un nombre qui est de la forme  $n \times n = n^2$  où  $n$  est un entier positif, par exemple 36 ou 49 sont des carrés mais pas 42.

Certains entiers peuvent s'écrire comme la somme de deux carrés. Par exemple, le nombre  $13 = 9 + 4$  est une somme des carrés 9 et 4. En revanche, en essayant les différents cas possibles :

$$12 = 0 + 12 = 1 + 11 = 4 + 8 = 9 + 2$$

on voit que 12 ne peut pas s'écrire comme une somme de deux carrés.

Il se trouve qu'il existe un unique entier entre 1 et 1000 qui est multiple de 42 et peut s'écrire comme somme de deux carrés. On vous demande de trouver ce nombre. On appliquera la méthode suivante. On observe d'abord que  $32^2 = 1024 > 1000$  donc une solution du problème est la somme de carrés de nombres inférieurs à 32. Ensuite, on remarque qu'il suffit d'examiner tous les cas possibles en calculant toutes les sommes de la forme  $a^2 + b^2$  où  $a$  et  $b$  sont des entiers strictement inférieurs à 32.

#### 7. Supprimer les doublons

Partant d'une liste d'entiers, définir une variable sansDoublons qui référence une liste composée des valeurs de L, mais dans laquelle les différentes valeurs égales de L (s'il y

en a) n'ont été

1. Sous entendu, « carré parfait ».

2

copiées qu'une seule fois. On dira ainsi de la liste construite qu'elle ne contient pas de doublon. Exemples :

<sub>1</sub> [42, 81, 42, 65, 12, 81, 31, 42] -> [42, 81, 65, 12, 31]

<sub>2</sub> [42] -> [42]

<sub>3</sub> [42, 42, 42, 42, 42] -> [42]

### 8. Motif carré formé des chiffres 4 ou 2

Écrire un programme qui à partir d'un entier  $n > 0$  affiche un carré de côté  $n$ , dont la bordure est faite avec le nombre 4 et l'intérieur rempli par le nombre 2. Deux chiffres successifs seront séparés par une espace.

Voici un exemple avec un carré de côté  $n = 8$  :

```
1 4 4 4 4 4 4 4 4
2 4 2 2 2 2 2 2 4
3 4 2 2 2 2 2 2 4
4 4 2 2 2 2 2 2 4
5 4 2 2 2 2 2 2 4
6 4 2 2 2 2 2 2 4
7 4 2 2 2 2 2 2 4
8 4 4 4 4 4 4 4 4
```

### 9. Triangle de Floyd

Le triangle suivant est un triangle de Floyd à 5 lignes :

```
1 1
2 2 3
3 4 5 6
4 7 8 9 10
5 11 12 13 14 15
```

Le triangle de Floyd à  $n$  lignes est obtenu en plaçant sur  $n$  lignes successives, dans l'ordre croissant, des entiers consécutifs à partir de 1, la ligne numéro  $k$  comportant exactement  $k$  entiers.

A partir d'un nombre de lignes  $n > 0$ , écrire un code qui affiche le triangle de Floyd à  $n$  lignes. Par exemple, si  $n = 5$ , le code affichera le triangle ci-dessus.

### 10. Somme nulle

On donne une liste  $L$  d'entiers et on demande de construire un booléen `somme0` valant `True` s'il existe des entiers successifs dans  $L$  et dont la somme vaut 0. Par exemple, si  $L = [-4, 2, -4, 1, 9, -6, -4]$  alors `somme0 = True` puisque  $(-4) + 1 + 9 + (-6) = 0$ . De même, si  $L = [-4, 0, 5, 1]$  alors `somme0 = True` puisque la liste contient un terme nul. En revanche, si  $L = [-3, 2, 4]$  alors `somme0 = False`.

### 11. Grille : remplissage par colonnes

Vous devez écrire un code qui affiche un motif dont un exemple est visible

ci-dessous : 3

```
1 42 52 62 72
2 43 53 63 73
3 44 54 64 74
4 45 55 65 75
5 46 56 66 76
6 47 57 67 77
7 48 58 68 78
8 49 59 69 79
9 50 60 70 80
10 51 61 71 81
```

Ce motif consiste en une grille rectangulaire de dimensions  $n \times p$  données ( $n$  : nombre de lignes,  $p$  : nombre de colonnes) de tous les entiers consécutifs à partir d'un entier donné  $d$ , le remplissage devant se faire *colonne par colonne*.

Ci-dessus, l'affichage obtenu pour  $n=10$ ,  $p=4$  et  $d=42$ . Noter que deux colonnes sont séparées par un espace.

Vous pourrez remarquer que dans chaque ligne, les valeurs des nombres se suivent avec un écart valant le nombre  $p$  de colonnes, par exemple 42, 52, 62 et 72 dans la première ligne.

## 12. Que des 81 puis que des 12

Soit une liste  $L$  formée d'abord d'un certain nombre de fois de l'entier 81 puis d'un certain nombre de fois de l'entier 12. Par exemple,  $L=[81, 81, 81, 12, 12, 12, 12]$  ou encore  $L=[81, 12]$  ou même  $L=[12, 12]$  ainsi que  $L=[81, 81, 81]$ .

Écrire, à l'aide d'une boucle **while**, une variable `indicePremier_12` qui définit le premier indice du terme de  $L$  qui vaut 12. Par exemple, si  $L = [81, 81, 81, 12, 12, 12, 12]$  alors `indicePremier_12 = 3`. Si 12 n'apparaît pas dans  $L$ , la variable `indicePremier_12` vaudra  $n$  où  $n$  est la longueur de la liste  $L$ .

Exemples de comportement :

```
1 [81, 81, 81, 12, 12, 12, 12] -> 3
2 [81, 12] -> 1
3 [81, 81] -> 2
4 [12, 12] -> 0
```

### 13. Listes « opposées » (boucle **while**)

Ecrire un code qui partant deux listes d'entiers L et M crée un booléen **sontOpposees** valant **True** si les deux listes sont « opposées » et **False** sinon. Deux listes sont considérées comme « opposées » si elles ont le même nombre d'éléments et si, à des indices identiques, elles possèdent des éléments opposés (comme -81 et 81). Voici quelques exemples de comportements attendus :

```
1 [81, -12, 0, -81, -31] [-81, 12, 0, 81, 31] -> True
2 [-81] [81] -> True
3 [0, 0] [0, 0] -> True
4 [] [] -> True
5 [81, -12] [-81, -12] -> False
6 [-81, 12, 0] [81, -12] -> False
```

Vous ne devez pas utiliser de boucle **for** mais une boucle **while**

### 14. Plus petit entier non nul (boucle **while**)

On donne une liste d'entiers qui ne sont pas tous nuls, par exemple

4

— >L = [5, 8, 0, 9, 12, 0, 6, 4] ou encore L = [0, 0, 0, 4, 0, 3, 0].

Construire une variable **plusPetitNonNul** ayant pour valeur le plus petit entier de la liste qui soit non nul. Voici quelques exemples de comportement :

```
1 [5, 8, 0, 9, 1, 0, 6, 4] -> 1
2 [0, 0, 0, 4, 0, 3, 0] -> 3
3 [5, 8, 1, 6, 4] -> 1
4 [-5, -8] -> -8
5 [0, -5, -8] -> -8
6 [-3, 0, -5, -8] -> -8
7 [0, 5, -8] -> -8
8 [42] -> 42
9 [-42] -> -42
```

On pourra appliquer la méthode suivante :

- chercher à l'aide d'une boucle **while** le plus petit indice i de L tel que L[i] soit non nul
- chercher le plus petit élément de la liste L à partir l'indice i

### 15. Pairs d'abord, impairs ensuite

On donne une liste L d'entiers et on demande d'écrire un booléen **PairsImpairs** qui renvoie **True** si dans la liste L apparaissent D'ABORD les entiers pairs de la liste et ENSUITE les entiers impairs de la liste.

Voici quelques exemples de comportements :

L	Pairs puis impairs ?	Commentaire
[12, 82, 81, 9, 31]	<b>True</b>	D'abord 12, 82 (pairs) puis les impairs
[81, 9, 31]	<b>True</b>	Que des impairs

[12, 82]	True	Que des pairs
[12, 82, 81, 9, 46, 31]	False	9 (impair) est suivi d'un pair (46)

### 16. Liste d'entiers en miroir

On donne une liste d'entiers, par exemple  $L = [4, 2, 2, 4]$  et on demande de dire si cette liste est en miroir, autrement dit si

- le premier et le dernier élément de  $L$  sont égaux,
- le deuxième et l'avant-dernier élément de  $L$  sont égaux,
- et ainsi de suite jusqu'à épuisement de la liste.

Dans le cas de la liste  $L = [4, 2, 2, 4]$ , cette liste est en miroir. Dans le cas de la liste  $L = [4, 2, 1]$ , la liste n'est pas en miroir.

### 17. Pas d'impair (avec **break**)

Soit une liste  $L$  formée d'entiers. Écrire, à l'aide d'une boucle **for** et d'une instruction **break**, un booléen `queDesPairs` qui dit si, oui ou non,  $L$  ne contient que des entiers pairs. Comportement attendu :

5

- <sub>1</sub> [82, 31, 82] -> False
- <sub>2</sub> [82, 12, 46] -> True
- <sub>3</sub> [82] -> True
- <sub>4</sub> [81] -> False

### 18. Suite croissante d'entiers consécutifs (avec **while**)

L'exercice doit être codé en utilisant une boucle **while**.

Écrire un code qui à partir d'une liste  $L$  d'entiers définit une variable booléenne nommée **consecutifs** qui vaut **True** si la liste est constituée d'entiers CONSÉCUTIFS croissants et **False** sinon. Ci-dessous, voici quelques exemples de comportements attendus

- <sub>1</sub> [81, 82, 83] -> True
- <sub>2</sub> [82, 81, 83] -> False
- <sub>3</sub> [2013, 2038, 3000] -> False
- <sub>4</sub> [81] -> True

### 19. Suite croissante d'entiers consécutifs (avec **break**)

L'exercice doit être codé en utilisant une boucle **for** et une instruction **break**. Écrire un code qui à partir d'une liste  $L$  d'entiers définit une variable booléenne nommée **consecutifs** qui vaut **True** si la liste est constituée d'entiers CONSÉCUTIFS croissants et **False** sinon. Ci-dessous, voici quelques exemples de comportements attendus

- <sub>1</sub> [81, 82, 83] -> True
- <sub>2</sub> [82, 81, 83] -> False
- <sub>3</sub> [2013, 2038, 3000] -> False
- <sub>4</sub> [81] -> True

## 20. Calculer le nombre de chiffres d'un entier

On donne un entier  $n \geq 0$  et on cherche le nombre de chiffres  $n_{\text{chiffres}}$  de  $n$ . Par exemple, si  $n = 2020$  alors  $n_{\text{chiffres}} = 4$  ou encore si  $n = 42$  alors  $n_{\text{chiffres}} = 2$ . L'idée pour calculer  $n_{\text{chiffres}}$  est de compter le nombre de divisions successives de  $n$  par 10 jusqu'à ce que le quotient (entier) soit nul. Par exemple

- le quotient entier de 2020 par 10 est 202,
- le quotient entier de 202 par 10 est 20,
- le quotient entier de 20 par 10 est 2,
- le quotient entier de 2 par 10 est 0.

C'est parce 2020 a justement 4 chiffres qu'on a effectué 4 divisions successives par 10 avant d'obtenir un quotient nul.

Ecrire un code Python utilisant une boucle **while** et qui détermine  $n_{\text{chiffres}}$  connaissant  $n$ . Voici quelques exemples de comportements

```
1 42 -> 2
2 2020 -> 4
3 10 -> 2
4 7 -> 1
5 0 -> 1
6 741520036365253625145211741523636854198541 -> 42
```