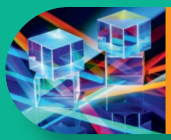
PrépaBAC

1re GÉNÉRALE

Mes SPÉCIALITÉS

Cours & entraînement

NOUVEAU BAC



Maths

Physique-chimie

SVT

Vidéos de révision

Pour maîtriser tes 3 spécialités

OFFERT Un abonnement au site de révision annabac

Hatier

Avec cet ouvrage :
des compléments numériques

Des vidéos de révision sur la chaîne Youtube annabac

En maths
Les notions-clés
hatier-clic.fr/8297850_01

En physique-chimie
Les notions-clés
hatier-clic.fr/8297850_02

Les schémas bilans
hatier-clic.fr/8297850_03

Un abonnement* au site de révision annabac

Cet ouvrage vous permet d'accéder gratuitement*
à toutes les ressources du site – fiches, quiz, vidéos, sujets corrigés… –
dans les différentes matières (tronc commun et spécialités) de 1re.

Rendez-vous sur **www.annabac.com**
dans la rubrique « J'ai acheté
un ouvrage Hatier »

* Selon les conditions précisées sur le site

prépaBAC

Cours & entraînement

1re GÉNÉRALE

Mes SPÉCIALITÉS

Maths

Michel Abadie • Annick Meyer
• Jean-Dominique Picchiottino • Martine Salmon

Physique-Chimie

Joël Carrasco • Alexandra Chauvin
• Gaëlle Cormerais • Éric Langlois

SVT

Nicolas Ducasse • Benjamin Forichon • Johanna Garcia
• Hervé Mulard • Bruno Vah

Hatier

Maquette de principe : Frédéric Jély
Mise en pages : STDI
Schémas : Nord Compo, Coredoc
Iconographie : Hatier illustration
Édition : Béatrix Lot

© Hatier, Paris, 2024
Sous réserve des exceptions légales, toute représentation ou reproduction intégrale ou partielle, faite, par quelque procédé que ce soit, sans le consentement de l'auteur ou de ses ayants droit, est illicite et constitue une contrefaçon sanctionnée par le Code de la Propriété Intellectuelle. Le CFC est le seul habilité à délivrer des autorisations de reproduction par reprographie, sous réserve en cas d'utilisation aux fins de vente, de location, de publicité ou de promotion de l'accord de l'auteur ou des ayants droit.

AVANT-PROPOS

VOUS ÊTES EN PREMIÈRE générale et vous avez choisi les trois spécialités :

Maths • Physique-chimie • SVT

Vous savez que la réussite dans ces matières demande un travail régulier tout au long de l'année ?
Alors ce gros Prépabac est pour vous !

Dans chaque discipline, l'ouvrage vous permet en effet :
– de **mémoriser les connaissances essentielles** sur chacun des thèmes du nouveau programme,
– d'**acquérir progressivement des méthodes clés :** mettre en œuvre une recherche, conduire un raisonnement, appliquer la technique de calcul appropriée…

Cet objectif est rendu possible grâce à un **ensemble de ressources très complet :** des fiches de cours et de méthode synthétiques et visuelles, mais aussi un entraînement progressif, incluant des sujets guidés, pour vous préparer à l'épreuve finale du bac.

Nous vous recommandons de les utiliser régulièrement, en fonction de vos besoins. Ainsi vous pourrez aborder vos contrôles en toute sérénité et acquérir les compétences nécessaires en Terminale.

Bonnes révisions !

Les auteurs

SOMMAIRE GÉNÉRAL

Pour trouver un chapitre précis dans une matière, reportez-vous au sommaire de chaque partie.

MATHS

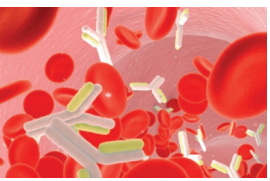
SOMMAIRE 9

- Algorithmique et programmation 12
- Algèbre 26
- Analyse 52
- Géométrie 98
- Probabilités et statistique 124

PHYSIQUE-CHIMIE

SOMMAIRE 153

- Constitution et transformations de la matière 156
- Mouvement et interactions 228
- L'énergie : conversions et transferts 252
- Ondes et signaux. 280

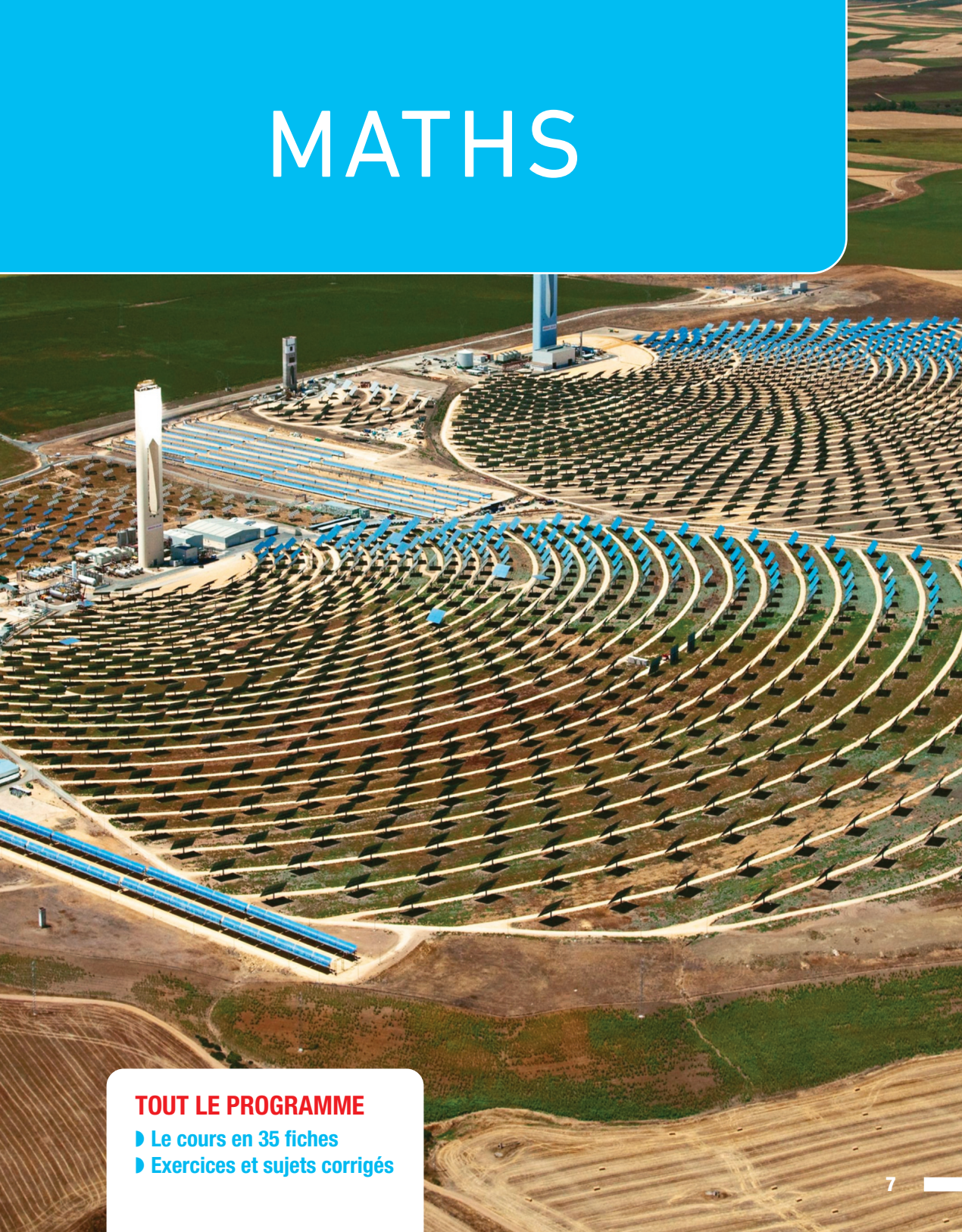


SVT

SOMMAIRE 319

- La Terre, la vie et l'organisation du vivant 322
- Les enjeux contemporains de la planète 400
- Corps humain et santé 424

MATHS



TOUT LE PROGRAMME

- Le cours en 35 fiches
- Exercices et sujets corrigés

SOMMAIRE

Quand vous avez révisé un chapitre, cochez la case ☐ correspondante !

Algorithmique et programmation

1. Notion de liste

1 Rappels : variables, boucles, fonctions . . . . . 12 ☐
2 Génération d'une liste . . . . . 14 ☐
3 Manipulations sur les éléments d'une liste . . . . . 16 ☐
4 Opérations globales sur les listes . . . . . 18 ☐
EXERCICES ET CORRIGÉS . . . . . 20 ☐

Algèbre

2. Équations et fonctions polynômes du second degré

5 Équations et fonctions polynômes du second degré . . . . . 26 ☐
6 Forme canonique, factorisation, racines et signe . . . . . 28 ☐
7 Somme et produit des racines . . . . . 30 ☐
EXERCICES ET CORRIGÉS . . . . . 32 ☐

3. Suites numériques, modèles discrets

8 Généralités sur les suites . . . . . 38 ☐
9 Suites arithmétiques . . . . . 40 ☐
10 Suites géométriques . . . . . 42 ☐
11 Notion de limite . . . . . 44 ☐
EXERCICES ET CORRIGÉS . . . . . 46 ☐

Analyse

4. Dérivation

12 Nombre dérivé et tangente . . . . . 52 ☐
13 Fonction dérivée – Dérivées usuelles . . . . . 54 ☐
14 Opérations sur les dérivées . . . . . 56 ☐
EXERCICES ET CORRIGÉS . . . . . 58 ☐

SOMMAIRE

5. Fonction exponentielle

15 Définitions et propriétés algébriques 64
16 Propriétés analytiques et applications 66
EXERCICES ET CORRIGÉS 68

6. Étude de fonctions

17 Sens de variation d'une fonction 74
18 Extremum d'une fonction 76
19 Positions relatives de deux courbes 78
EXERCICES ET CORRIGÉS 80

7. Fonctions trigonométriques

20 Cercle trigonométrique. Mesure d'un angle 86
21 Cosinus et sinus d'un nombre réel 88
22 Fonctions cosinus et sinus 90
EXERCICES ET CORRIGÉS 92

Géométrie

8. Calcul vectoriel – Produit scalaire

23 Rappels sur les vecteurs 98
24 Produit scalaire de deux vecteurs 100
25 Produit scalaire et orthogonalité 102
26 Transformation d'expressions et formule d'Al-Kashi 104
EXERCICES ET CORRIGÉS 106

9. Géométrie repérée

27 Vecteur normal à une droite 112
28 Équation de cercle 114
29 Parabole représentative d'une fonction polynôme de degré 2 116
EXERCICES ET CORRIGÉS 118

Probabilités et statistique

10. Probabilités conditionnelles et indépendance

30 Succession de deux épreuves indépendantes . . . . . 124 ☐
31 Probabilités conditionnelles et indépendance . . . . . 126 ☐
32 Formule des probabilités totales . . . . . 128 ☐
EXERCICES ET CORRIGÉS . . . . . 130 ☐

11. Variables aléatoires réelles

33 Définition et loi de probabilité d'une variable aléatoire . . . . . 136 ☐
34 Espérance, variance et écart type . . . . . 138 ☐
35 Variables aléatoires et algorithmes . . . . . 140 ☐
EXERCICES ET CORRIGÉS . . . . . 142 ☐

1 Rappels : variables, boucles, fonctions

En bref *Les notions de variables, de boucles, de fonctions ont été étudiées en classe de seconde. Les algorithmes dans lesquels elles interviennent seront écrits en langage naturel (ou en « pseudocode ») ou traduits en langage Python.*

I Affectation de variables

■ Une **variable** est un emplacement (une « boîte ») dans lequel on peut stocker une « valeur » (nombre, texte…), repéré par une « étiquette » (son nom) qui permet d'avoir accès à son contenu. Ce contenu peut varier au cours de l'exécution d'un programme.

■ L'affectation est notée « ← » dans un algorithme (pseudocode) et « = » en Python. *Exemples* :

Algorithme	Code Python	Sens
$A \leftarrow 12$	$A = 12$	Affecte à A la valeur 12, la valeur précédemment contenue dans A est « écrasée ».
$B \leftarrow A$	$B = A$	Affecte à B la valeur contenue dans A, ne modifie pas A.
$A \leftarrow A + 3$	$A = A + 3$	Ajoute 3 à la valeur numérique contenue dans A. Si A valait 12, A vaut maintenant 15.

II Boucles et fonctions

■ **Boucle bornée (avec nombre d'itérations connu)**

L'instruction *Pour…* (*for…* en langage Python) permet de répéter l'exécution d'un bloc d'instructions **un nombre connu de fois**.

■ **Boucle non bornée (avec arrêt conditionnel)**

L'instruction *Tant Que…* (*while…* en langage Python) permet de répéter l'exécution d'un bloc d'instructions **tant qu'une certaine condition est réalisée** : l'exécution s'arrête dès que la condition n'est plus remplie.

■ **Fonctions**

• Si, dans un programme, on doit exécuter plusieurs fois la même série d'instructions, on peut enregistrer ces instructions dans une **fonction**, qu'on appelle quand on en a besoin.
• La fonction peut prendre des **arguments** (ou paramètres) et renvoyer une **valeur** (en Python, l'instruction correspondante est *return* « valeur »).
• Certaines fonctions prédéfinies en Python peuvent être utilisées en chargeant des « bibliothèques » en début de programme (fonctions aléatoires…).

Méthodes

1 Utiliser une boucle

Une société propose, sur abonnement, la livraison hebdomadaire d'un panier de fruits et légumes. Initialement, le nombre d'abonnés est 65. Les responsables font l'hypothèse que d'un mois à l'autre 20 % des abonnements sont résiliés et 18 abonnements supplémentaires sont souscrits.

Écrire un programme déterminant et affichant le nombre de mois au bout duquel le nombre d'abonnés sera pour la première fois supérieur ou égal à 85.

CONSEILS

Expliciter la relation entre les nombres d'abonnés de deux mois successifs.
Le programme comportera une boucle qui calcule le nombre d'abonnés tant qu'il est inférieur à 85, et un « compteur » du nombre d'exécutions.

SOLUTION

Si on note u_n le nombre d'abonnés au bout de n mois et u_{n+1} le nombre d'abonnés au bout de $n + 1$ mois, alors $u_{n+1} = 0{,}8u_n + 18$. En Python :

```
U=65
N=0
while U<85 :
    N=N+1
    U=0,8×U+18
print(N)
```

On trouve $N = 8$. On vérifie que $u_7 < 85$ et $u_8 > 85$.

2 Définir et utiliser une fonction

Écrire un programme Python qui définit une fonction « *somcar* » renvoyant, pour tout entier naturel n non nul, la somme $1^2 + 2^2 + \ldots + n^2$.

CONSEILS

La somme cherchée est « stockée » dans une variable s qui doit être initialisée.
À chaque étape, on ajoute un terme à la valeur précédente de s.

SOLUTION

```
def somcar(n) :
    s=0
    for k in range(1, n+1) :
        s=s+k**2
    return s
```

CONSEIL

En Python, la fin d'une boucle est marquée par la fin de l'indentation (du décalage).

L'instruction « for *k* in range(1, n+1) » signifie « pour *k* variant de 1 (compris) à $n + 1$ (exclu) », c'est-à-dire de 1 à n.

2 Génération d'une liste

En bref *En Python, une liste est délimitée par des crochets, l'index du premier élément est 0, celui du deuxième est 1… Les éléments ne sont pas nécessairement tous de même type, une liste peut contenir plusieurs fois le même élément.*

I Créer une liste

■ Pour créer une liste **en extension**, il suffit de saisir ses éléments entre crochets, séparés par des virgules.

Exemple : liste1 = [36,11,"bonjour"].
L'instruction print(liste1) renvoie : [36, 11, 'bonjour'].

■ On crée une liste **en compréhension** à partir des éléments d'une liste initiale, éventuellement uniquement ceux vérifiant une certaine condition (filtrage).

Exemples : Les instructions suivantes définissent une liste dont les éléments sont les carrés :

- des éléments de la liste *aliste* : [n**2 for n in aliste] ;
- des entiers de 2 à 8 inclus : [n**2 for n in range(2,9)] ;
- des entiers de 2 à 8 qui sont multiples de 3 :
[n**2 for n in range(2,9) if n% 3==0].

À NOTER
« n%3 » est le reste de *n* dans la division euclidienne par 3 ou « reste de *n* modulo 3 ».

II Modifier une liste

On suppose que l'on a déjà créé une liste nommée *liste2*, éventuellement vide (on crée une liste vide en tapant liste2 = [ ]).

■ **La méthode « append »** ajoute un élément en dernière position d'une liste. La syntaxe est liste2.append(56) : ajoute 56 en dernier élément (la nouvelle *liste2* a un élément de plus).

On peut ainsi créer une liste à partir d'une liste vide en ajoutant les éléments un par un. Dans une boucle *for* ou *while*, c'est une manière de créer une liste de données qu'on calcule ou qui arrivent au fur et à mesure.

■ **La méthode « extend »** ajoute tous les éléments de la liste prise en argument (ici *liste3*) à la fin de la *liste2* : liste2.extend(liste3).

■ On obtient les mêmes résultats par **concaténation**, mais en créant une nouvelle liste qui enregistre le résultat.

Exemples : liste4 = liste2 + [56] et liste5 = liste2 + liste3.

■ On peut créer une liste par **répétition des éléments d'une liste** existante.

Exemple : Les éléments de la liste [1,2,3] sont répétés 3 fois : liste6 = [1,2,3]*3.
L'instruction print(liste6) renvoie [1, 2, 3, 1, 2, 3, 1, 2, 3].

Méthodes

1 Créer une liste d'entiers

a. Écrire un programme Python permettant de créer puis d'afficher la liste des 10 premiers multiples strictement positifs de 7.

b. Écrire un programme Python permettant de créer puis d'afficher la liste des diviseurs (positifs) de 360.

CONSEILS

a. Les multiples de 7 sont de la forme 7*k*, *k* entier de 1 à 10 (in range(1,11)).

b. On crée une liste vide. Pour chaque entier *k* de 1 à 360 (in range(1,361)), on teste si *k* est un diviseur de 360. Si oui, *k* est ajouté à la liste.

SOLUTION

a. La première instruction correspond à la création de la liste *mult7*, la deuxième instruction permet de l'afficher.

```
mult7=[7*k for k in range(1,11)]
print(mult7)
```

b. « if 360 % *k*==0 » signifie « si le reste dans la division euclidienne de 360 par *k* est égal à 0 ». On obtient une liste de 24 diviseurs.

```
div=[ ]
for k in range(1,361) :
    if 360%k==0 :
        div.append(k)
print(div)
```

2 Permuter deux éléments d'une liste

Écrire un programme permettant de transformer la liste [cylindre, hexagone, octogone, pyramide, quadrilatère] en la liste [quadrilatère, hexagone, octogone, pyramide, cylindre].

CONSEILS

Pour permuter les valeurs des variables *A* et *B*, il ne suffit pas d'écrire *A* = *B* puis *B* = *A*. Après exécution de ce programme, les variables *A* et *B* contiennent la même valeur (celle de *B*). Pour ne pas « perdre » la valeur initialement contenue dans *A*, on la stocke temporairement dans une variable auxiliaire.

SOLUTION

```
figures = ["cylindre","hexagone","octogone",
"pyramide","quadrilatère"]
aux = figures[0]
figures[0] = figures[4]
figures[4] = aux
```

À NOTER

Avec Python, on peut aussi plus simplement écrire :
figures[0], figures[4]
= figures[4], figures[0].

3 Manipulations sur les éléments d'une liste

En bref *Lorsqu'une liste a été créée, on peut être amené à effectuer sur ses éléments différentes manipulations, soit à partir de l'index de ces éléments, soit avec leur valeur.*

I Ajouter, modifier ou supprimer un élément

- Les méthodes « append » et « extend » permettent d'**ajouter** ou de **modifier** un élément dans une liste → FICHE 2.
- La méthode « insert » permet d'**insérer** un élément dans une liste.

Exemple : liste5.insert(2, 27) insère 27 à l'index 2 (c'est-à-dire en troisième position) de *liste5* et décale les éléments suivants.

- Pour **supprimer** un élément d'une liste d'**index déterminé** en Python, on utilise « del ».

Exemple : del liste5[1] supprime l'élément d'index 1 (c'est-à-dire le deuxième élément) de *liste5*.

- Pour **supprimer** un élément d'une **valeur donnée**, on utilise la méthode de liste « remove ».

Exemple : liste5.remove(1) supprime la première occurrence de la valeur « 1 » dans *liste5*, pas les éventuelles autres occurrences de cette valeur.

II Parcourir, itérer ou rechercher dans une liste

- **Itérer sur les éléments d'une liste :**
  - sans index, par exemple :

```
for x in liste6 :
    print(x)
```

  - avec index, par exemple :

```
for n, x in enumerate(liste6) :
    print(n, x)
```

- **Accéder** à l'élément d'index donné, **modifier** sa valeur :

• liste6[2] : pour accéder à l'élément d'index 2 de *liste6* ;
• liste6[2]=18 donne à l'élément d'index 2 de *liste6* la valeur 18.

- **Rechercher une valeur dans une liste :**

2 in liste6 : renvoie « **True** » si l'un des éléments de *liste6* est 2, « **False** » sinon.

- **Étudier les occurrences d'une valeur :**

• liste6.count(2) donne le nombre d'**occurrences** de la valeur 2 dans *liste6* (renvoie 0 si 2 ne figure pas dans *liste6*) ;
• liste6.index(2) renvoie l'**index de la première occurrence** de 2 dans *liste6* si 2 figure dans *liste6* (sans tenir compte des autres occurrences éventuelles), renvoie un message d'erreur sinon.

MOT CLÉ
Cet opérateur est un **booléen**, c'est-à-dire un type de données qui ne peut prendre que deux valeurs : True (vrai) et False (faux).

Méthode

Supprimer une à une les différentes occurrences d'une valeur dans une liste

On considère la liste (nommée simplement *liste*) :

[1, 1, 2, 1, 2, 3, 4, 1, 2, 1, 8, 7, 1].

Écrire un programme supprimant un par un dans cette liste les éléments de valeur 1, en affichant à chaque étape la liste obtenue :

a. avec une boucle *for* ;

b. avec une boucle *while*.

CONSEILS

On saisit la liste en extension.

a. On peut utiliser une boucle *for* exécutée *n* fois, où *n* est le nombre d'occurrences de la valeur 1 dans la liste initiale, *n* étant défini au préalable. À chaque étape, on demande au programme de supprimer la première occurrence de la valeur 1 ; cette manipulation est effectuée *n* fois, les occurrences de la valeur 1 sont supprimées une à une (dans l'ordre croissant de leurs index).

b. Si on utilise une boucle *while*, elle est exécutée tant que la valeur 1 figure dans la liste. Dans ce cas, il n'est pas nécessaire de compter au préalable le nombre d'occurrences du 1.

a. *n* est le nombre d'occurrences de la valeur 1, et *k* est un « compteur » qui prend successivement comme valeurs tous les entiers de 0 à *n* – 1.

```
n=liste.count(1)
for k in range(n) :
    liste.remove(1)
    print(liste)
```

L'instruction « liste.remove(1) » peut être remplacée par la séquence d'instructions :

```
i = liste.index(1)
del liste[i]
```

b. « liste.count(1)!=0 » signifie « le nombre d'occurrences du 1 dans la liste est différent de 0 ». On peut remplacer « différent de 0 » par « strictement positif ».

```
while liste.count(1) != 0 :
    liste.remove(1)
    print(liste)
```

À NOTER

L'indentation indique que l'instruction d'affichage est dans la boucle.
On demande à chaque étape l'affichage de la liste : on obtient 6 listes successives de plus en plus courtes.

4 Opérations globales sur les listes

En bref *Il est possible d'effectuer sur une liste des opérations globales, c'est-à-dire des opérations qui concernent l'ensemble de ses éléments sans différenciation, sans utilisation des index ou des valeurs des éléments.*

I Réaliser des opérations sur des listes

- On obtient le nombre d'éléments (la **longueur**) de la liste nommée *liste* avec l'instruction len(liste).
- La **somme** des éléments de la liste (si les valeurs sont numériques) s'obtient par sum(liste).

À NOTER
En combinant ces deux opérations, on peut aisément calculer, lorsque les éléments de la liste ont tous des valeurs numériques, la moyenne de toutes les valeurs.

- Pour **inverser** l'ordre des éléments d'une liste, on utilise la méthode « reverse() ».

Exemple : avec liste7.reverse(), la liste *liste7* est modifiée, remplacée par la liste « inversée ».

- Pour simplement **afficher dans l'ordre inverse** les éléments de *liste7* (sans modification de *liste7*), on peut saisir :

```
for i in reversed(liste7) :
    print(i)
```

Les valeurs sont affichées les unes en dessous des autres (dans l'ordre inverse de celui où elles figurent dans *liste7*).

II Trier les éléments d'une liste

- Pour **trier** les éléments d'une liste suivant leur valeur (**ordre croissant** pour des nombres, **ordre alphabétique** pour des chaînes de caractères), on utilise :

• La méthode de liste « sort() ». *Exemple* : liste7.sort() : la liste *liste7* est modifiée, sa « valeur » initiale est « perdue », remplacée par la nouvelle liste ordonnée.
• La fonction « sorted() ». *Exemple* : sorted(list7) : la liste *liste7* n'est pas modifiée, il faut nommer et enregistrer la nouvelle liste.

- Pour obtenir un tri dans l'**ordre inverse** (décroissant) : liste7.sort(reverse = True) ou sorted(list7, reverse = True).

À NOTER
Le tri permet de déterminer les valeurs minimale et maximale lorsque toutes les valeurs sont numériques. Il existe aussi dans Python des méthodes min(*liste*) et max(*liste*) prédéfinies.

Méthode

Déterminer la valeur médiane d'une liste de valeurs numériques

Écrire une fonction en Python permettant de déterminer la note médiane d'une série de notes.

On rappelle que pour une série statistique ordonnée $(x_1 ; x_2 ; x_3 ; \dots ; x_n)$, la médiane M est définie par :

- si n est impair, alors $M = x_{\frac{n+1}{2}}$;
- si n est pair avec $n = 2k$, alors $M = \dfrac{x_k + x_{k+1}}{2}$.

Vérifier avec les deux listes suivantes :

a. L1 = [11, 7, 2, 14, 10, 11, 5, 12, 8, 16, 6, 15, 12, 7, 7, 18, 10].

b. L2 = [10, 1, 5, 6, 11, 3, 19, 11, 10, 9, 3, 14, 7, 13, 4, 15].

CONSEILS

Il est plus commode de classer par ordre croissant les éléments de la liste.
On effectue ensuite un test sur la parité du nombre d'éléments.
Il faut faire attention aux index des éléments de la liste : si la série ordonnée est $(x_1 ; x_2 ; x_3 ; \dots ; x_n)$ et si la liste correspondante est notée L, alors, pour tout entier j compris entre 1 et n, $x_j = L[j - 1]$.

SOLUTION

```
def mediane(L):
    L.sort()
    n = len(L)
    if n%2==0:
        return (L[int(n/2)-1]+L[int(n/2)])/2
    else:
        return L[int((n-1)/2)]
```

Avec l'instruction « L.sort() », la liste L est ordonnée, la liste initiale (mêmes valeurs mais dans un ordre éventuellement différent) est « écrasée ».

« if n%2==0 » signifie « si le reste de n dans la division euclidienne par 2 est 0 », autrement dit « si n est pair ».

a. En appliquant cette fonction à L1, on trouve que sa médiane est **10**.

b. En l'appliquant à L2, on trouve une médiane égale à **9,5**.

À NOTER

Les index des éléments d'une liste doivent être des nombres entiers. Même si n est pair, $\frac{n}{2}$ n'est pas considéré par Python comme un entier, d'où la syntaxe int(n/2) qui transforme cette valeur en entier.

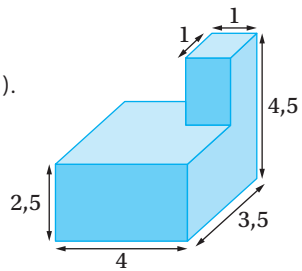
SE TESTER QUIZ

Vérifiez que vous avez bien compris les points clés des **fiches 1 à 4**.

1 Variables, boucles, fonctions

→ FICHE 1

Une cuve de récupération des eaux pluviales est formée de deux parallélépipèdes rectangles superposés (schéma ci-contre, dimensions en m).
On considère le programme ci-dessous :




```
def vol(h)
    if h < 2,5 :
        v = 14 * h
    else :
        v = ... + h
    return v
```

Pour que ce programme définisse une fonction qui retourne le volume d'eau (en m^3) dans la cuve en fonction de la hauteur d'eau h (en m), par quelle valeur doit-on compléter l'avant-dernière ligne ?

☐ **a.** 35 ☐ **b.** 32,5 ☐ **c.** 63

2 Génération d'une liste

→ FICHE 2

L'instruction [n**2 for n in range(3,6)] définit la liste :

☐ **a.** [9, 16, 25, 36] ☐ **b.** [9, 16, 25] ☐ **c.** [16, 25, 36]

3 Manipulations sur les éléments d'une liste

→ FICHE 3

On considère le programme ci-contre.
À la fin de son exécution, le résultat affiché est :

☐ **a.** 1
☐ **b.** 2
☐ **c.** 3

```
L=[2,12,41,18,24,16]
m3=0
for k in L :
    if k%3==0 :
        m3=m3+1
print(m3)
```

4 Opérations globales sur les listes

→ FICHE 4

On considère la liste L = [8, 18, 3, 8, 15, 1]. On applique successivement les instructions « print(sorted(L)) » (affichage 1) puis « print(L) » (affichage 2). On obtient :

☐ **a.** affichage 1 : [1, 3, 8, 8, 15, 18], puis affichage 2 : [8, 18, 3, 8, 15, 1] ;
☐ **b.** affichage 1 : [1, 3, 8, 8, 15, 18], puis affichage 2 : [1, 3, 8, 8, 15, 18] ;
☐ **c.** affichage 1 : [18, 15, 8, 8, 3, 1], puis affichage 2 : [8, 18, 3, 8, 15, 1].

S'ENTRAÎNER

5 Supprimer le premier ou le dernier élément

→ FICHES 1 et 3

Écrire une fonction en Python prenant en argument une liste et supprimant :
a. le premier élément de cette liste ;
b. le dernier élément de cette liste.

6 Calculer des sommes cumulées

→ FICHES 1, 3 et 4

Écrire une fonction en Python prenant en argument une liste de nombres et renvoyant les « sommes cumulées », c'est-à-dire une nouvelle liste où le *k*-ième élément est la somme des *k* premiers éléments de la liste originale.

7 Déterminer si une liste est triée

→ FICHES 1 et 4

Écrire une fonction en Python appelée *est_triee* qui prend en argument une liste (de nombres ou de « mots ») et renvoie « True » si la liste est triée par ordre croissant, « False » sinon.

OBJECTIF BAC

8 Lister les élèves qui n'ont pas la moyenne

45 min

Cet exercice, comme le précédent, fait appel à une fonction du module « random » qui permet la génération aléatoire d'un entier entre deux bornes données. Il donne aussi l'occasion de réutiliser certaines manipulations sur les listes.

LE SUJET

On considère un groupe de 15 élèves : Alice, Bob, Claude, Dany, Éloi, Fred, Greg, Harry, Inès, Jean, Kelly, Luis, Mona, Nick, Olga, et les notes (sur 20) obtenues aux deux derniers devoirs par les 15 élèves de ce groupe.
Les élèves pour lesquels la moyenne des deux notes est strictement inférieure à 10 doivent participer aux séances d'AP (accompagnement personnalisé).
On cherche à écrire un programme qui donne la composition du groupe d'élèves devant venir en AP.
Écrire un programme en Python qui :
a. enregistre la liste des 15 élèves ;
b. génère aléatoirement deux listes de 15 nombres compris entre 0 et 20 (0 et 20 compris), qui représentent les notes des 15 élèves aux deux derniers devoirs ;
c. affiche la liste à 30 éléments où chaque nom d'élève est suivi de sa moyenne sur les deux derniers devoirs ;
d. affiche la liste des élèves qui doivent venir en AP.

LIRE LE SUJET

a. On crée la liste en extension.

b. Comme à l'exercice précédent, on crée deux listes en utilisant une fonction du module « random ». On met donc en tête du programme « from random import * ».

c. On calcule d'abord la moyenne de chaque élève sur les deux derniers devoirs et on intercale les noms et les moyennes.

d. On récupère dans une liste les noms des élèves dont la moyenne est inférieure à 10.

LA FEUILLE DE ROUTE

a. Créer une liste en extension

→ FICHE 2

Les éléments de la liste sont des mots, c'est-à-dire des chaînes de caractères. Afin que Python ne les confonde pas avec des variables, on les saisit entre guillemets.

b. Générer aléatoirement deux listes de nombres

→ FICHES 1 et 2

On crée d'abord deux listes vides et on ajoute à chacune un à un 15 entiers entre 0 et 20 générés aléatoirement. On peut aussi, puisque la longueur des listes est connue, créer au départ deux listes formées de 15 zéros.

c. Créer une liste en intercalant les éléments de deux listes

→ FICHES 1 et 2

Dans la nouvelle liste, les éléments d'index pair sont les noms des élèves, les éléments d'index impair sont les moyennes.

d. Constituer une liste à partir d'une autre liste

→ FICHES 1 et 2

On teste successivement les notes (moyennes). On constitue une liste des noms des élèves dont la note est strictement inférieure à 10. Cette liste est construite à partir d'une liste vide, car initialement on ne connaît pas sa longueur.