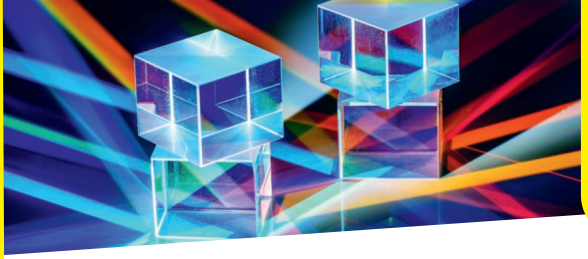
PrépaBAC



1re GÉNÉRALE

Maths

SPÉCIALITÉ

Réussir l'examen

BAC 2026

- Les **cours** en fiches
- Les **méthodes** clés
- **300 exos & sujets** guidés
- Tous les **corrigés** expliqués

Pour **réussir** la nouvelle épreuve de **maths !**

INCLUS

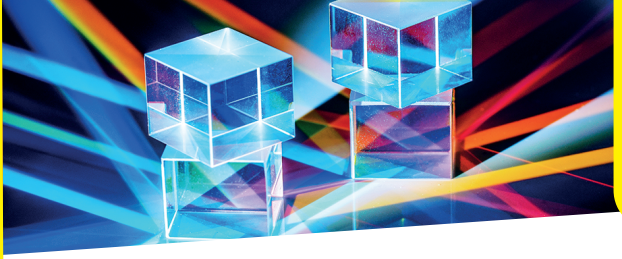
QUIZ et SUJETS de BAC

+ OFFERT

- Un abonnement annabac.com
- Des quiz interactifs

Hatier

PrépaBAC



1re GÉNÉRALE

Maths

SPÉCIALITÉ

Réussir l'examen

Michel Abadie
Professeur agrégé de mathématiques
Lycée Galillée (Gennevilliers)

Annick Meyer
Professeure agrégée de mathématiques
Lycée Aristide Briand (Saint-Nazaire)

Jean-Dominique Picchiottino
Professeur agrégé de mathématiques
Lycée Pasquet (Arles)

Martine Salmon
Professeure agrégée de mathématiques

Hatier

Mode d'emploi

Voici les ressources que vous trouverez dans ce Prépabac :

Dans chaque chapitre

Un cours visuel

1. Des fiches synthétiques et illustrées
2. Des méthodes détaillées
3. Une carte mentale récapitulative

Un entraînement progressif

1. Des quiz pour se tester
2. Des exercices guidés
3. Des sujets de type bac

Des corrigés détaillés

Préparer l'épreuve

QCM

Sujet de bac

- Des **vidéos** de révision au fil des chapitres.
- Des **quiz interactifs** sur les automatismes.

Nous vous recommandons d'utiliser régulièrement l'ouvrage. Ainsi vous pourrez aborder vos contrôles en toute sérénité et réussir la nouvelle épreuve anticipée de maths en Première.

Les auteurs


Michel Abadie


Annick Meyer


Jean-Dominique Picchiottino


Martine Salmon

SOMMAIRE

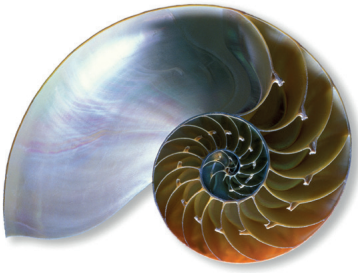
Algorithmique et programmation

1 Notion de liste

Algèbre

2 Équations et fonctions polynômes du second degré

3 Suites numériques, modèles discrets



COURS & MÉTHODES

Analyse

4 Dérivation

COURS & MÉTHODES

5 Fonction exponentielle

COURS & MÉTHODES

6 Étude de fonctions

7 Fonctions trigonométriques

Géométrie

8 Calcul vectoriel – Produit scalaire

9 Géométrie repérée

Probabilités et statistiques

10 Probabilités conditionnelles et indépendance

11 Variables aléatoires réelles

Spécial bac

12 Préparer le bac

Les vidéos de l'ouvrage

Retrouvez l'ensemble de ces vidéos sur la chaîne Youtube annabac

Les notions-clés

hatier-clic.fr/26mat1_08

Les cartes mentales

hatier-clic.fr/26mat1_11

L'épreuve de maths en 1re

L'épreuve anticipée du bac en spécialité maths

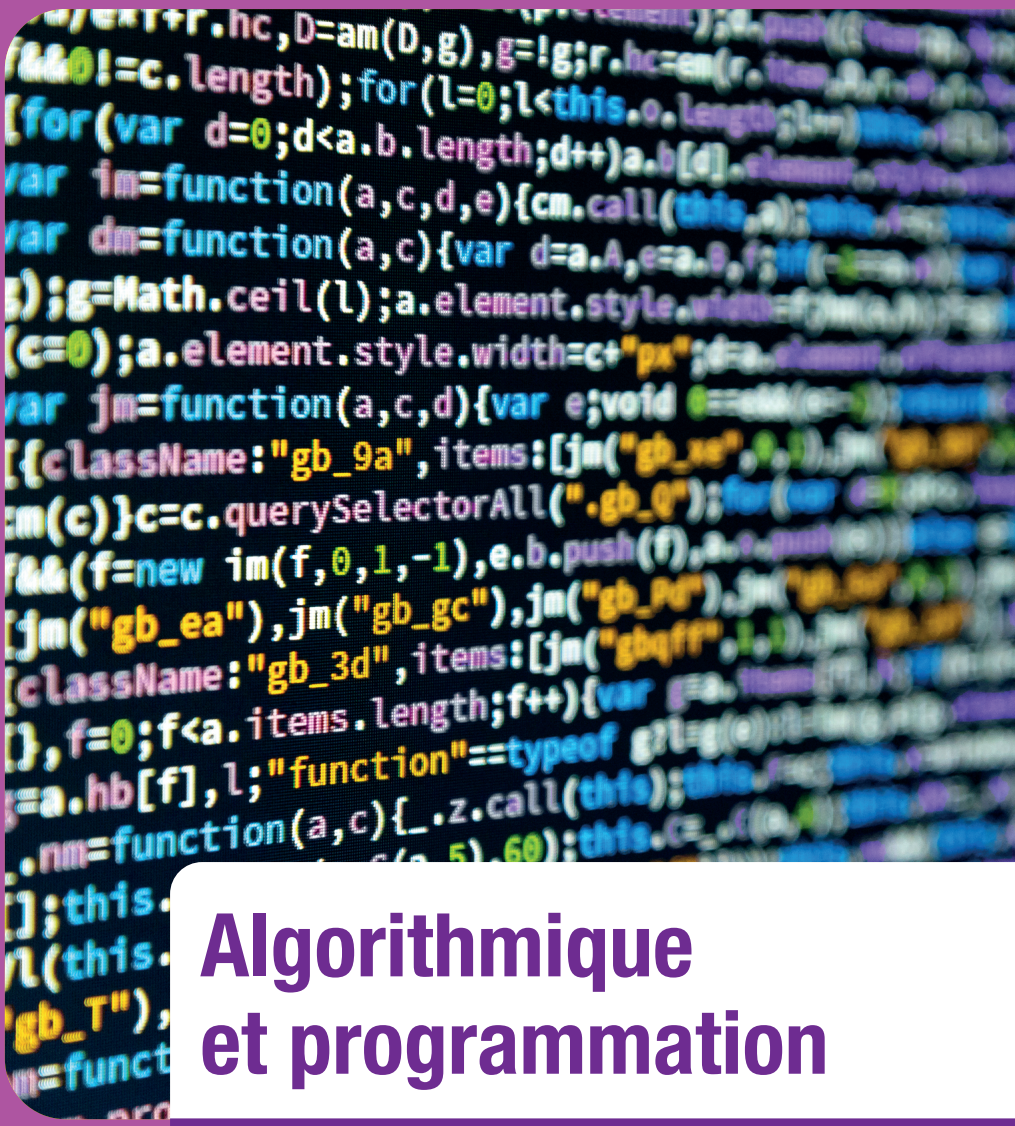
Retrouver toutes les infos sur le déroulement de la nouvelle épreuve anticipée de maths en classe de Première

hatier-clic.fr/26mat1_09

QCM – Automatismes

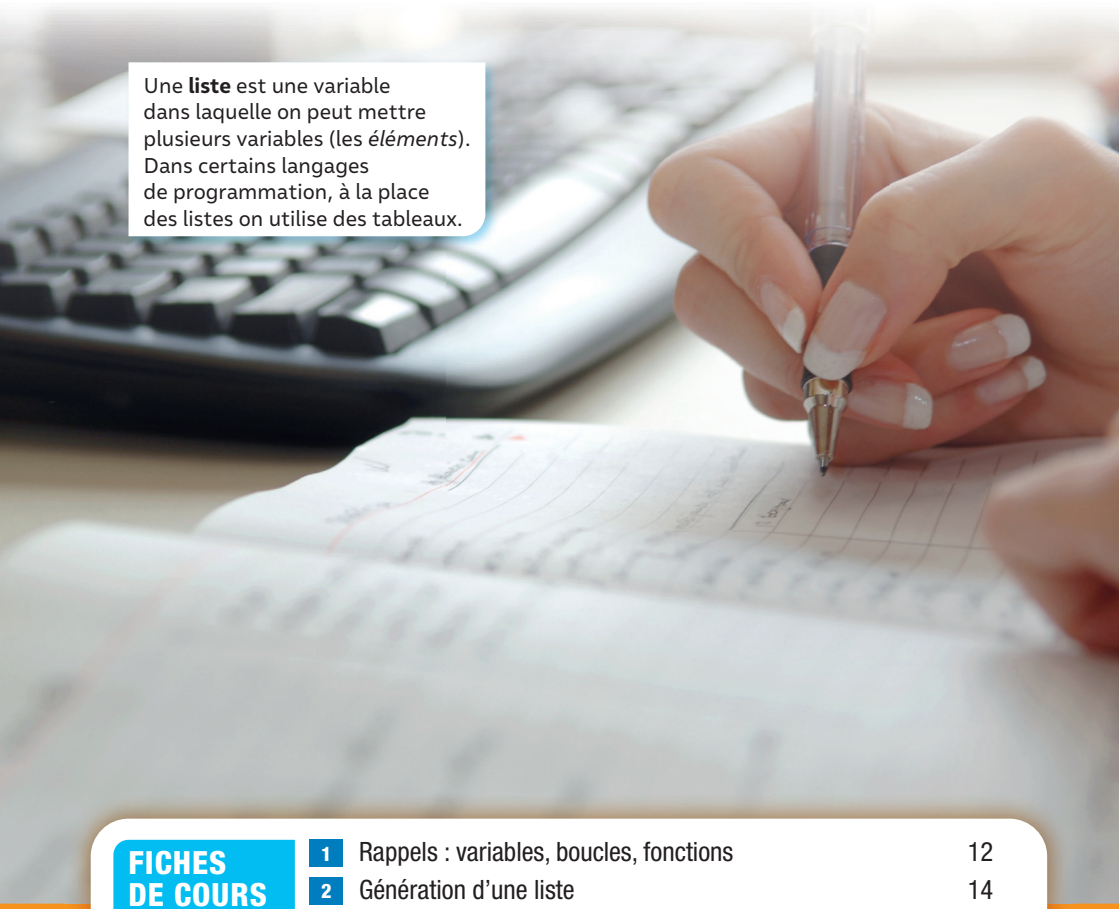
Pour préparer l'épreuve, retrouver des quiz interactifs supplémentaires sur le site annabac

hatier-clic.fr/26mat1_10



Algorithmique et programmation

1 Notion de liste



Une **liste** est une variable dans laquelle on peut mettre plusieurs variables (les *éléments*). Dans certains langages de programmation, à la place des listes on utilise des tableaux.

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 $7k$, 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.