Fiches BAC

1re générale

NOUVEAU BAC

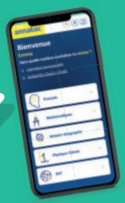
Maths

SPÉCIALITÉ

avec des cartes mentales récapitulatives

Tout le programme en 46 FICHES

- L'essentiel du cours
- Les méthodes & exercices types
- Des quiz et leurs corrigés pour s'évaluer



OFFERT ! Un abonnement au site de révision annabac

Hatier

MODE D'EMPLOI

Comment utiliser vos **FICHES BAC** ?

Lisez les fiches de manière active

- ☑ **Anticipez** le contenu en lisant d'abord le plan.
- ☑ **Faites une pause** après chaque paragraphe pour reformuler mentalement l'idée clé.
- ☑ **Cachez la fiche** et essayez de restituer un maximum d'informations au brouillon.

Testez-vous !

- ➜ Vérifiez vos connaissances avec les Quiz EXPRESS.
- ➜ Refaites les exercices types, commentés dans la rubrique Méthodes.

Comment **organiser** vos révisions ?

Vous révisez toute l'année

VOS OBJECTIFS

- ☑ **Comprendre et retenir**
  - ➜ Fiches de cours
  - ➜ Cartes mentales
- ☑ **Vérifier que vous avez retenu les points clés**
  - ➜ Quiz express
- ☑ **Réactiver régulièrement vos connaissances**
  - ➜ Méthodes

Vous révisez au dernier moment

VOS OBJECTIFS

- ☑ **Cibler vos lacunes**
  - ➜ Quiz express
- ☑ **Revoir les points clés**
  - ➜ Fiches correspondant à vos lacunes
- ☑ **Mémoriser l'essentiel**
  - ➜ Cartes mentales

Fiches BAC

1re générale

NOUVEAU BAC

Maths

SPÉCIALITÉ

Michel Abadie
Professeur agrégé de mathématiques
Lycée Galilée (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
Lycée Évariste Galois (Sartrouville)

Hatier

SOMMAIRE

Quand vous avez révisé une fiche, cochez la case ❒ correspondante !

Algorithmique et programmation

Notion de liste

Algèbre

Équations et fonctions polynômes du second degré

Suites numériques, modèles discrets

Analyse

Dérivation

Fonction exponentielle

Étude de fonctions

Fonctions trigonométriques

Géométrie

Calcul vectoriel – Produit scalaire

Géométrie repérée

Probabilités et statistiques

Probabilités conditionnelles et indépendance

Variables aléatoires réelles

DÉPLIANT

4 cartes mentales sur les notions essentielles

Maquette de principe : Frédéric Jély
Mise en pages : STDI
Schémas : STDI
Édition : Jean-Marc Cheminée

Rappels : variables, boucles, fonctions

1

OK

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 alors 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.

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 $u_{n+1} = 0{,}8u_n + 18$. On trouve $N = 8$. On vérifie que $u_7 < 85$ et $u_8 > 85$.
Programme en Python ci-contre.

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

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

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.

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

Génération d'une liste

2

OK

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 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 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**.

Exemple : Les éléments de la liste [1,2,3] sont répétés 3 fois : liste6 = [1,2,3]*3 crée la liste [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 $B = A$, les variables A et B contiendraient la même valeur (celle de B). On stocke temporairement la valeur de A dans une variable auxiliaire.

SOLUTION

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

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