Fiches BAC

Tle générale

NOUVEAU BAC

NSI

SPÉCIALITÉ

Un dépliant mémento

Tout le programme en 53 FICHES détachables

- Les notions clés du cours
- Les méthodes avec des exemples
- 60 quiz 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 !

- ➜ Faites (et refaites !) les Quiz EXPRESS.
- ➜ Testez les extraits de code dans un éditeur Python.

Comment **organiser** vos révisions ?

Vous révisez **toute l'année**

VOS OBJECTIFS

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

Vous révisez **au dernier moment**

VOS OBJECTIFS

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

Fiches BAC

Tle générale

NOUVEAU BAC

NSI

SPÉCIALITÉ

Guillaume Connan

Professeur agrégé de mathématiques
Enseignant de mathématiques et d'informatique en classes préparatoires au lycée Externat-Chavagnes de Nantes
Formateur au DIU « Enseigner l'informatique au Lycée »

Vojislav Petrov

Professeur agrégé de mathématiques
Ancien élève de l'École normale supérieure de Lyon
Enseignant de mathématiques et d'informatique en classes préparatoires au lycée Baimbridge en Guadeloupe

Gérard Rozsavolgyi

Professeur agrégé de mathématiques
Responsable de la Licence professionnelle métiers de l'informatique à l'IUT d'Orléans
Formateur au DIU « Enseigner l'informatique au lycée »

Laurent Signac

Docteur en informatique
Enseignant en informatique à l'École nationale supérieure d'ingénieurs de Poitiers
Formateur au DIU « Enseigner l'informatique au lycée »

Hatier

Crédits photographiques
81 ph © akg-images
97 h Coll. British Government art collection
97 b Coll. Shorpy Photo Archives
98 Coll. Nasa
99 h ph © DR
99 m Coll. wikipedia
100 h Coll. wikipedia
100 m ph © DR

Maquette de principe : Frédéric Jély • **Mise en pages :** STDI
Schémas : STDI • **Iconographie :** Hatier illustration
Édition : Jean-Marc Cheminée

SOMMAIRE

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

Langages, programmation et algorithmique

Modularité et mise au point des programmes

Récursivité

Paradigmes de programmation

Structures de données

Structures de données : interface et implémentation

Structures arborescentes

Graphes

Bases de données

Conception de bases de données

Systèmes de gestion de bases de données – SQL

Architectures matérielles, systèmes d'exploitation et réseaux

Composants et processus

Protocoles de routage et sécurisation des communications

Programmation avancée et algorithmique

Programmes et données – Calculabilité

Programmation avancée

DÉPLIANT

Récursivité
- Programme récursif
- Diviser pour régner

Programmation orientée objet
- Une classe type en Python

Types de données en Python
- Opérations sur les listes/piles
- Opérations sur les files
- Opérations sur les dictionnaires
- Opérations sur les ensembles

Routage et sécurisation des communications
- Le chiffrement asymétrique
- Comparatif du matériel réseau
- Le Transport Layer Security (TLS)
- Comparatif des algorithmes de routage

Modules et documentations

1

OK

I Programmation modulaire

1 Principes généraux

Le **principe de modularité** est particulièrement important dans tout développement logiciel. Il simplifie les tests, permet de réutiliser du code, et facilite la maintenance. Ce principe peut opérer à plusieurs niveaux :
– découper le code en fonctions ;
– grouper les fonctions dans une classe (on les appelle alors méthodes) ;
– grouper des fonctions par thèmes, dans un fichier séparé ;
– grouper des fichiers en bibliothèques.

À noter
La manipulation des modules et de la documentation fait partie intégrante du développement logiciel.

La **programmation modulaire** intervient :
– lors de l'écriture de portions de code pour les réutiliser plus tard, ou les distribuer ;
– lors de l'utilisation, pour répondre à un besoin, de code écrit par une autre personne.

2 Modularité en Python

On peut (on doit…) écrire des **fonctions** et des **procédures**, et les réutiliser.

Python permet de faire de la **programmation orientée objets** ▶ FICHE 12 et donc de grouper des fonctions (alors appelées méthodes) au sein de classes, selon le type d'objet sur lequel elles agissent.

Fonctions, procédures et classes peuvent être groupées par thèmes dans des fichiers séparés alors appelé **modules** qui peuvent être groupés en **packages**.

II Importation des modules

Certains modules Python sont installés par défaut (bibliothèque standard) et d'autres peuvent être ajoutés en utilisant des outils comme `pip`. Le dépôt Pypi (Python Package Index, https://pypi.org/) référence la plupart des modules tiers.

Pour importer un module installé, on utilise le mot clé `import` dans une de ses variantes. Ci-après des exemples avec le module `random`.

• On importe le module `random`, les fonctions du module doivent être préfixées du nom du module :

```
import random
a = random.randint(1, 10)
```

• On importe le module `random` et on le renomme en `rnd` (le nouveau nom donné est généralement plus court) :

```
import random as rnd
a = rnd.randint(1, 10)
```

• On importe uniquement la fonction `randint` du module `random` (plus de préfixe, mais seule la fonction `randint` est disponible) :

```
from random import randint
a = randint(1, 10)
```

Toutes les variantes contenant des * dans l'instruction d'importation sont à bannir. Séduisantes pour le débutant, elles sont sources de bugs et rendent le code moins lisible.

III Documentation

Que l'on écrive ou qu'on utilise une fonction ou un module, la documentation est centrale pour que le travail soit réutilisable.

Parmi les différents mécanismes, l'un des plus simples est la **docstring** qui peut être attachée à une fonction, à une procédure, à une méthode, à une classe, à un module, ou à un package. Dans tous les cas, c'est une chaîne de caractères qui doit figurer au début de l'entité qu'elle documente. Cette documentation est ensuite consultable à l'aide de la commande `help` :

```
>>> import random
>>> help(random) # affiche la docstring du module
>>> help(print)  # affiche la docstring de la fonction
```

Exemple d'écriture d'une docstring :

```
def factorielle(n) :
    """ Calcul de la factorielle :
        n! = 1x2x…xn
        >>> factorielle(5)
        120
    """
    res = 1
    for i in range(2, n + 1):
        res = res * i
    return res
```