

Fiches BAC

T^{le}
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



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

T^{le}
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 »



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

© 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.

SOMMAIRE

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

Langages, programmation et algorithmique

Modularité et mise au point des programmes

- 1 Modules et documentations ☐ 7
- 2 Déboguer un programme ☐ 9
- 3 Tests ☐ 11
- 4 Guide de style ☐ 13
- 5 **QUIZ EXPRESS** ☐ 15

Réversivité

- 6 Fonctionnement d'un programme récursif ☐ 17
- 7 Méthode « diviser pour régner » ☐ 19
- 8 Exemples d'utilisation de la récursivité ☐ 21
- 9 **QUIZ EXPRESS** ☐ 23

Paradigmes de programmation

- 10 Les principaux paradigmes de programmation ☐ 25
- 11 Programmation fonctionnelle ☐ 27
- 12 Programmation orientée objet ☐ 29
- 13 **QUIZ EXPRESS** ☐ 31

Structures de données

Structures de données : interface et implémentation

14	Types abstraits	☐	33
15	Implémentation des files	☐	35
16	Structures de données Python	☐	37
17	QUIZ EXPRESS	☐	39

Structures arborescentes

18	Définition des arbres	☐	41
19	Implémentation des arbres binaires	☐	43
20	Utilisation des arbres	☐	45
21	QUIZ EXPRESS	☐	47

Graphes

22	Vocabulaire des graphes	☐	49
23	Représentations d'un graphe	☐	51
24	Parcours de graphes – Algorithme de Dijkstra	☐	53
25	Implémentations et applications des parcours de graphes	☐	55
26	QUIZ EXPRESS	☐	57

Bases de données

Conception de bases de données

27	Vocabulaire des bases de données	☐	59
28	Le modèle relationnel	☐	61
29	Conception de bases de données	☐	63
30	QUIZ EXPRESS	☐	65

Systèmes de gestion de bases de données – SQL

31	Les systèmes de gestion de bases de données	67
32	Le langage SQL	69
33	Le langage SQL avancé	71
34	QUIZ EXPRESS	73

Architectures matérielles, systèmes d'exploitation et réseaux

Composants et processus

35	Les visages multiples du numérique	75
36	Processus et ressources	77
37	Notions d'interblocage	79
38	Composants intégrés d'un système sur puce.	81
39	QUIZ EXPRESS	83

Protocoles de routage et sécurisation des communications

40	Le routage	85
41	Algorithmes de routage dynamique	87
42	La sécurisation des communications : HTTPS	89
43	QUIZ EXPRESS	91

Programmation avancée et algorithmique

Programmes et données – Calculabilité

44	Programmes et données	93
45	Interprétation et compilation – Bytecode	95
46	Trois femmes entrées dans l'histoire de l'informatique.	97
47	Histoire de la théorie de la calculabilité	99
48	Le problème de l'arrêt	101
49	QUIZ EXPRESS	103

Programmation avancée

50	Vers la programmation dynamique	105
51	Programmation dynamique et alignement de séquences	107
52	Recherche de motifs dans des chaînes de caractères	109
53	QUIZ EXPRESS	111

DÉPLIANT

Réversibilité

- Programme réversible
- 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



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.

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.

À noter

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

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
```