100% EXOS

2de

NOUVEAU BAC

Maths

l'entraînement intensif !

- **300 EXERCICES** progressifs & minutés
- **50 SUJETS** de contrôle
- **CORRIGÉS** détaillés & commentés
- **COURS & MÉTHODES**

GRATUIT*: des ressources interactives et des parcours de révision sur **annabac.com**

Hatier

100% EXOS

2de

Maths

Laurent Darré
Professeur agrégé de mathématiques
au lycée François Magendie (Bordeaux).

Hatier

Maquette de principe : Frédéric Jély
Mise en pages : Grafatom
Schémas : Grafatom
Édition : Julien Lionnet

Avant-propos

Votre ouvrage 100 % exos

- Conforme au nouveau programme de Maths 2de entré en vigueur à la rentrée 2019, ce « 100 % exos » vous propose une **méthode de travail** complète et un **entraînement intensif** sur mesure, faisant une large place à la préparation du **nouveau bac**.
- Pour chaque thème du programme, vous trouverez un **COURS** structuré, les **MÉTHODES** qu'il faut maîtriser, des **EXERCICES** progressifs et leurs **CORRIGÉS** détaillés.
- Assortis d'**indications de solutions**, de **commentaires** et de **conseils** des auteurs, tous les exercices corrigés vous permettent :
  - de **comprendre** les notions essentielles et de **maîtriser** le cours ;
  - de **progresser** et de vous **entraîner** à votre rythme ;
  - de vous **évaluer** et de **réussir** vos devoirs et contrôles ;
  - de vous **préparer** à l'entrée en Première.

Le site de vos révisions

- L'achat de cet ouvrage vous permet de bénéficier d'un **ACCÈS GRATUIT*** à toutes les **ressources** d'annabac.com : fiches, quiz, sujets corrigés… et à ses **parcours de révision** personnalisés.
- Pour profiter de cette offre, rendez-vous sur **www.annabac.com**, dans la rubrique « Je profite de mon avantage client ».

* Selon les conditions précisées sur le site.

Sommaire

ALGORITHMIQUE ET PROGRAMMATION

NOMBRES ET CALCULS

FONCTIONS

GÉOMÉTRIE

STATISTIQUES ET PROBABILITÉS

1 Algorithmique et programmation

I ALGORITHMIQUE

1. Présentation d'un algorithme

◗ Un **algorithme** est une suite d'instructions, compréhensibles par qui doit l'exécuter, ayant pour but de conduire à un résultat donné.

◗ Les **variables** sont des objets (nombre entier, nombre réel, texte,…) utilisés au cours de l'algorithme qui prennent une valeur précise à un moment donné.

◗ Les **instructions** nécessaires au bon déroulement de l'algorithme peuvent être :

- les **entrées** et les **sorties** (peu utilisées avec les algorithmes)
- **l'affectation de variables**, notée avec une flèche ←
- les **tests**
- les **boucles**

2. Tests

Si (test) **alors** (bloc d'instructions) **Fin Si**

Si (test) **alors** (bloc d'instructions) **Sinon** (bloc d'instructions) **Fin Si**

Après le mot « Si », le test est formé d'une ou plusieurs conditions de comparaison utilisant les symboles : $=, \neq, <, >, \leqslant, \geqslant$, portant sur les variables de l'algorithme.

3. Boucles

Boucle conditionnelle	Boucle itérative
Tant que (condition) (bloc d'instructions) **Fin Tant que**	**Pour** (compteur) allant de N_1 à N_2 (bloc d'instructions) **Fin Pour**

II PROGRAMMATION EN PYTHON

1. Variables

Les principaux types de variables sont :
int(nombre entier), float(nombre à virgule), str(texte) et bool(booléen ou logique).

EXEMPLE :

```
A = 1
B = 2.5
C = «1»
D = (1+1==2)
```

A prend la valeur 1, elle est du type int
B prend la valeur 2.5, elle est du type float
C prend la valeur «1», elle est du type str
D prend la valeur True, elle est du type bool

Pour déclarer A comme un réel, tout en initialisant sa valeur à 1, on écrit A = 1.0
Une variable booléenne ne peut prendre que deux valeurs : True et False.

2. Entrée/sortie

L'entrée (ou lecture ou saisie) se fait au moyen de la fonction input().

EXEMPLE :

```
A = input()
A = int(input())
A = int(input(«Saisir un entier»))
```

Permet de déclarer A comme entier
Une question ou un commentaire guide la saisie

Pour déclarer un nombre à virgule ou un texte, on écrit float ou str à la place de int.

La sortie (ou écriture ou affichage) se fait au moyen de la fonction print().

EXEMPLE :

```
print(A)
print(«Bonjour»)
print(«Bonjour»,A)
```

Affiche la valeur courante de A
Affiche le texte entre guillemets
Affiche un texte suivi de la valeur de A

3. Tests

Un test ou instruction conditionnelle se fait au moyen du mot « if » et éventuellement du mot « else ».

if (test) :
  (instructions)
(suite du programme)
	if (test) :
	  (instructions)
	else :
	  (instructions)
	(suite du programme)

EXEMPLE :

```
A = int(input(« quel est votre âge ?»))
if A<18 :
    print(« Vous êtes mineur »)
else :
    print(« Vous êtes majeur »)
```

Le mot « Alors » n'est pas utilisé dans Python.
S'il y a plusieurs cas, on peut utiliser elif qui est la contraction de else et if, pour le « sinon si ».
Avec un test qui utilise une égalité, on emploie le signe égal doublé : ==

4. Les boucles

Boucle conditionnelle (TANT QUE)

```
while (condition):
    (bloc d'instructions)
```

EXEMPLE :

```
K ← 0
while K < 4 :
        K ← K + 1
```

Veiller à ce que la condition ne soit plus vérifiée à un moment donné, sinon la boucle sera exécutée de façon illimitée.

Boucle itérative (POUR)

```
for i in range(N1,N2):
    (bloc d'instructions)
```

EXEMPLES :

range(2,9) donne les entiers de 2 à 8 (et pas de 2 à 9).
range(9) donne les entiers de 0 à 8 (et pas de 0 à 9).
range(2,9,3) donne les entiers de 2 à 8 par sauts de 3, c'est-à-dire 2,5,8.

La variable i est le compteur. Elle prend les valeurs de N_1 à $N_2 - 1$.

5. Les fonctions

Les **fonctions** permettent de regrouper plusieurs instructions en un seul bloc. Certaines fonctions sont disponibles comme input() ou print(), d'autres sont utilisables si l'on a importé un module (voir paragraphe 6).
Enfin, on peut créer sa propre fonction avec la syntaxe :

```
def nomdefonction() :
```

EXEMPLE :

```
def cube(n) :
        return n**3
(programme)
```

Dans le programme, cube(4) prend la valeur 64.
La double étoile ** permet de calculer une puissance. Par Exemple, 3^4 se note 3**4.

Remarque : Voici deux points importants qui concernent **les tests**, **les boucles** et **les fonctions**.

- Les deux points à la fin de la première ligne sont obligatoires.
- Le décalage du bloc d'instructions est très important, ceci s'appelle l'indentation. Il n'y a pas de fin, c'est le retour au niveau antérieur du texte qui signifie cette fin.

6. Les modules

◗ Un **module** est un fichier contenant des définitions et des fonctions. On utilise essentiellement les modules math, random et turtle.
Exemples de modules d'utilisation fréquente : math, random, turtle
Le module math contient, entre autres, la constante pi, la fonction racine carrée sqrt(), la fonction puissance pow(), les fonctions d'arrondis floor() et ceil() …
◗ Pour accéder au contenu d'un module, celui-ci doit être importé.
◗ Une étoile * permet d'importer toutes les fonctions du module.
EXEMPLE : from random import *
◗ On peut aussi importer une fonction en particulier :
EXEMPLE : from random import randint

Afin de simuler le lancer d'un dé, on emploie randint(1,6) qui renvoie un nombre entier entre 1 et 6.

MÉTHODE 1

Utiliser une boucle POUR dans un algorithme

➜ Voir les exos 10 et 24.

Étape 1. Choisir un nom de variable pour le compteur de la boucle.
Prévoir les valeurs que ce compteur doit prendre.
Étape 2. Entre deux lignes qui matérialisent le début et la fin de la boucle, écrire les instructions qui seront répétées autant de fois que cela a été prévu à l'étape 1.

Exo résolu

Calculer les 10 premiers multiples de 7 (de 7×1 à 7×10).

CORRIGÉ

Étape 1. La variable I est le compteur de la boucle.
L'instruction « Pour I allant de 1 à 10 » permet à la variable I de prendre toutes les valeurs entières entre 1 et 10.
Étape 2. Entre le début et la fin de la boucle, on insère une ligne qui permet de calculer un multiple pour chaque valeur du compteur I.

```
Pour I allant de 1 à 10
Début de boucle
        M ← 7*I
Fin de boucle
```

Lors de la première boucle, I prend la valeur 1, M est calculé et vaut 7.
Puis le compteur I prend la valeur 2, M est calculé et vaut 14.
Ainsi de suite jusqu'au dixième multiple.

MÉTHODE 2

Définir une fonction dans un algorithme

➜ Voir les exos 11 et 15.

Étape 1. Choisir un nom pour la fonction, se demander s'il y a un ou des arguments, que l'on écrit entre parenthèses (séparés par des virgules s'il y en a plusieurs).
S'il n'y a pas d'argument, on laisse des parenthèses vides.

Étape 2. Écrire toutes les instructions nécessaires.
Parfois, ces instructions donnent lieu à un résultat que le programme va utiliser.
On utilise le mot "Renvoyer" ou "Retourner".

Exo résolu

Écrire une fonction qui renvoie 1 si la somme de trois nombres écrits en arguments est supérieure ou égale à 12, 0 sinon.

CORRIGÉ

Étape 1 Le nom choisi pour la fonction est : Somme().
En arguments, il y a trois nombres, qu'on nomme par exemple A, B et C.

Étape 2 Un test calcule la somme A + B + C et vérifie si la somme obtenue est supérieure à 12.
La fonction renvoie 1 ou 0 selon les cas.

```
Fonction Somme(A,B,C)
    Si A + B + C ⩾ 12 alors
        Renvoyer 1
    Sinon
        Renvoyer 0
    Fin Si
```

Les instructions concernées sont décalées pour une meilleure lisibilité.

MÉTHODE 3

Écrire un programme en Python avec des entrées et des sorties

➜ Voir les exos 8, 9, 14, 16 et 21.

Étape 1 Choisir des noms de variables simples et explicites.
Les initialiser si besoin.

Étape 2 Les entrées (saisies) se font avec la fonction input(), les sorties (affichages) avec la fonction print().

Étape 3 Vérifier le programme en l'exécutant. S'il y a des saisies, le faire avec des valeurs différentes pour envisager tous les cas.

Exo résolu

Les médecins testent parfois l'adaptation à l'effort avec le test de Ruffier-Dickson.
Tout d'abord, ils prennent le pouls (nombre de battements cardiaques par minute) au repos (P1). Après 30 flexions, le pouls est à nouveau mesuré (P2). Il est une dernière fois mesuré après 1 minute de récupération (P3).
L'indice de Ruffier-Dickson se calcule avec la formule :

$$\frac{P2 - 70 + 2(P3 - P1)}{10}.$$

Écrire un programme qui calcule l'indice de Ruffier-Dickson à partir des trois mesures.

CORRIGÉ

Étape 1 P1, P2 et P3 sont des noms de variables possibles pour les trois mesures. Ici, il n'y a pas d'initialisation.
L'indice de Ruffier-Dickson est noté IRD.

Étape 2 La saisie de P1, P2 et P3, qui sont des entiers, s'écrit : int(input()).
L'affichage de l'indice IRD s'écrit : print(IRD).
Le programme complet s'écrit :

```
P1=int(input("Mesure du pouls au repos ?"))
P2=int(input("Mesure du pouls après 30 flexions ?"))
P3=int(input("Mesure du pouls 1 minute après l'effort ?"))
IRD=(P2-70+2*(P3-P1))/10
print(IRD)
```

Étape 3 Lors de l'exécution, on teste avec $P1 = 70$, $P2 = 110$ et $P3 = 80$.
Le programme affiche $IRD = 6.0$.

La flèche ← d'affectation d'une valeur à une variable, que l'on trouve dans les algorithmes, se traduit par =.
Entre les parenthèses de la fonction input(), il est recommandé d'écrire, entre guillemets, une question ou un texte qui guide la saisie.
Au niveau médical, plus l'indice est bas, meilleure est l'adaptation à l'effort.
Un indice supérieur à 10 est mauvais, un indice inférieur à 4 est bon.