

# Python et les Mathématiques

de la Terminale à la prépa ECG

Les notions Python que tout étudiant doit maîtriser  
avant d'entrer en classe préparatoire



## Lycée Saint-Jean de Passy

J-P Spriet  
2 avril 2026

## **Mentions légales**

Cet ouvrage est la propriété intellectuelle de son auteur, JP Spriet, professeur en classes préparatoires ECG au Lycée Saint-Jean de Passy (Paris). Il est mis à disposition gratuitement à des fins exclusivement pédagogiques, sous licence **Creative Commons BY-NC-ND 4.0** (Attribution — Pas d'utilisation commerciale — Pas de modification).

Vous êtes libre de partager cet ouvrage, dans sa forme intégrale et non modifiée, à condition de mentionner l'auteur et l'établissement d'origine. Toute utilisation commerciale et toute modification sont interdites.

<https://creativecommons.org/licenses/by-nc-nd/4.0>

## **Exercices de baccalauréat**

Plusieurs exercices sont reproduits ou adaptés de sujets officiels du baccalauréat général (épreuve de mathématiques 2024-2025), mis à disposition par le Ministère de l'Éducation nationale. Leur origine est précisée dans chaque exercice concerné.

## **Contact**

Pour toute question ou remarque : [jp.spriet@saintjeandepassy.fr](mailto:jp.spriet@saintjeandepassy.fr)

---

# Introduction

---

Cet ouvrage s'adresse aux élèves de Terminale qui envisagent d'intégrer une classe préparatoire économique et commerciale (ECG) et qui souhaitent aborder cette nouvelle étape avec sérénité.

La classe préparatoire ECG mobilise Python de manière croissante, tant dans les enseignements quotidiens qu'aux concours. L'algorithmique et la programmation représentent aujourd'hui **10 à 12 % des points aux écrits** et sont présentes **systematiquement à l'oral** des grandes écoles (HEC, ESCP...). Maîtriser Python n'est donc pas un atout : c'est une exigence.

L'ambition de ce livre est de construire un **pont entre le lycée et la prépa**. Son principe directeur est simple : les mathématiques et Python se nourrissent mutuellement. Chaque notion de programmation est introduite au service d'un problème mathématique concret, et chaque concept mathématique trouve en Python un outil pour être exploré, vérifié, visualisé.

Aucune connaissance préalable en programmation n'est requise. En revanche, les notions mathématiques du programme de Terminale sont supposées connues : suites, fonctions, probabilités. Ce sont elles qui fournissent le fil conducteur de l'ouvrage.

Les exercices sont classés par niveau de difficulté ( accessible, intermédiaire, exigeant) et les corrigés détaillés figurent en fin de chaque chapitre. Le dernier chapitre propose des exercices de synthèse, dont plusieurs extraits de sujets de baccalauréat, pour s'entraîner dans les conditions les plus proches de l'examen.

*J-P Spriet*  
*Lycée Saint-Jean de Passy*





---

# Table des matières

---

<b>Introduction</b>	<b>iii</b>
<b>1 Variables et types</b>	<b>1</b>
1.1 Cours . . . . .	1
1.1.1 Variable et affectation . . . . .	1
1.1.2 Les quatre types essentiels . . . . .	2
1.1.3 Convertir un type . . . . .	3
1.1.4 Affectations multiples . . . . .	3
1.2 Exercices . . . . .	4
Corrigés des exercices . . . . .	6
<b>2 Les listes</b>	<b>9</b>
2.1 Cours . . . . .	9
2.1.1 Définition et création d'une liste . . . . .	9
2.1.2 Accéder à un élément – indices . . . . .	9
2.1.3 Longueur d'une liste . . . . .	10
2.1.4 Modifier un élément . . . . .	10
2.1.5 Ajouter un élément – <code>append()</code> . . . . .	11
2.1.6 Fonctions utiles sur les listes . . . . .	11
2.2 Exercices . . . . .	12
Corrigés des exercices . . . . .	14
<b>3 Tests et conditions</b>	<b>19</b>
3.1 Cours . . . . .	19
3.1.1 Booléens et comparaisons . . . . .	19
3.1.2 Instruction <code>if</code> – test simple . . . . .	20
3.1.3 Alternative <code>if / else</code> . . . . .	20
3.1.4 Conditions multiples <code>if / elif / else</code> . . . . .	20
3.1.5 Opérateurs logiques : <code>and</code> , <code>or</code> , <code>not</code> . . . . .	21
3.1.6 L'opérateur <code>in</code> . . . . .	22
3.2 Exercices . . . . .	23
Corrigés des exercices . . . . .	25
<b>4 Les boucles</b>	<b>33</b>
4.1 Cours . . . . .	33
4.1.1 La boucle <code>for</code> – parcourir une liste . . . . .	33
4.1.2 <code>range()</code> – générer des entiers . . . . .	34
4.1.3 La boucle <code>while</code> . . . . .	35
4.1.4 <code>for + append()</code> – construire une liste . . . . .	35
4.1.5 <code>break</code> et <code>continue</code> . . . . .	36
4.2 Exercices . . . . .	37
Corrigés des exercices . . . . .	39



<b>5 Les fonctions</b>	<b>45</b>
5.1 Cours . . . . .	45
5.1.1 Définir une fonction – <code>def</code> et <code>return</code> . . . . .	45
5.1.2 Paramètres multiples . . . . .	46
5.1.3 Paramètres par défaut . . . . .	46
5.1.4 Fonctions avec boucles . . . . .	47
5.1.5 Portée des variables . . . . .	48
5.1.6 Une fonction qui appelle une fonction . . . . .	49
5.2 Exercices . . . . .	49
Corrigés des exercices . . . . .	51
<b>6 Simulation du hasard</b>	<b>57</b>
6.1 Cours . . . . .	57
6.1.1 Bibliothèques et <code>import</code> . . . . .	57
6.1.2 <code>rd.random()</code> et <code>rd.randint()</code> . . . . .	58
6.1.3 <code>rd.choice()</code> et <code>rd.shuffle()</code> . . . . .	58
6.1.4 Simuler une loi de Bernoulli . . . . .	59
6.1.5 Loi binomiale – <code>np.random.binomial()</code> . . . . .	60
6.1.6 Loi des grands nombres . . . . .	61
6.2 Exercices . . . . .	62
Corrigés des exercices . . . . .	63
<b>7 Représentations graphiques</b>	<b>69</b>
7.1 Cours . . . . .	69
7.1.1 Le module <code>matplotlib</code> . . . . .	69
7.1.2 Tracer une courbe – <code>plt.plot()</code> . . . . .	69
7.1.3 Personnaliser le graphique . . . . .	71
7.1.4 Droites de référence – <code>axhline</code> et <code>axvline</code> . . . . .	72
7.1.5 Nuage de points – <code>plt.scatter()</code> . . . . .	74
7.1.6 Plusieurs courbes sur le même graphe . . . . .	75
7.2 Exercices . . . . .	78
Corrigés des exercices . . . . .	81
<b>8 Maths et Python – exercices de synthèse</b>	<b>91</b>
8.1 Suites . . . . .	91
8.2 Fonctions . . . . .	93
8.3 Probabilités et simulation . . . . .	94
Corrigés des exercices . . . . .	97

# Chapitre 1

## Variables et types

### 1.1 Cours

#### 1.1.1 Variable et affectation

En mathématiques, on écrit  $x = 5$  pour désigner la valeur d'une grandeur (appelée variable)  $x$ .

En Python, c'est la même idée : une **variable** est un nom qui désigne une valeur stockée en mémoire.

##### Définition 1.1 – Variable

Une **variable** est une association entre un nom et une valeur. L'instruction qui crée cette association s'appelle une **affectation**. En Python, elle s'écrit avec le signe  $=$ .

##### Le signe $=$ n'est pas une égalité !

En Python,  $x = 5$  signifie « mettre la valeur 5 dans la variable  $x$  » – pas «  $x$  est égal à 5 » au sens mathématique.

On peut par exemple écrire en Python :  $x = x + 1$  ; ce qui n'a aucun sens en mathématiques mais est parfaitement valide en Python : cela signifie que  $x$  prend la valeur qu'avait  $x$  plus un.

##### Exemple 1.1 – Premières affectations

```
1 x = 5           # x contient l'entier 5
2 prix = 19.90   # prix contient le reel 19.90
3 nom = "Alice"  # nom contient la chaine "Alice"
```

On peut ensuite utiliser ces variables dans des calculs :

```
1 x = 5
2 y = 3
3 somme = x + y
4 print(somme)   # affiche 8
5
6 z=3            # z contient la valeur 3
7 z=z+4         # z contient 3+4=7
8 print(z)      # affiche 7
```

##### Exemple 1.2 – Affecter plusieurs fois la même variable

Une variable peut changer de valeur au cours du programme. La nouvelle valeur **remplace** l'ancienne.

Par exemple : un capital de 1 000 euros est placé à un taux de rémunération de 5% d'intérêts,



quel est le nouveau capital au bout d'un an ?

```

1 capital = 1000
2 print(capital)           # 1000
3 capital = capital * 1.05 # capitalisation à 5%
4 print(capital)           # 1050.0

```

### 1.1.2 Les quatre types essentiels

En Python, chaque valeur a un **type**. Le type détermine ce qu'on peut faire avec la valeur.

#### Définition 1.2 – Les quatre types de base

- `int` : entier (ex. 3, -7, 100)
- `float` : réel, nombre à virgule (ex. 3.14, -0.5)
- `str` : chaîne de caractères, texte (ex. "Bonjour Madeleine")
- `list` : liste de valeurs (ex. [1, 2, 3])

La fonction `type()` permet de connaître le type d'une variable.

#### Exemple 1.3 – Identifier le type d'une variable

La fonction `type()` permet de connaître le type d'une variable : après affectation, on teste le type d'une variable `x` en écrivant `type(x)`.

```

1 n = 7
2 x = 3.14
3 prenom = "Alice"
4 notes = [15, 12, 18]
5
6 print(type(n))           # <class 'int'>
7 print(type(x))           # <class 'float'>
8 print(type(prenom))     # <class 'str'>
9 print(type(notes))      # <class 'list'>

```

#### Exemple 1.4 – Pièges courants avec les types

```

1 a = 7           # int
2 b = 7.0         # float -- pas la meme chose !
3 c = "7"        # str   -- ce n'est pas le nombre 7
4
5 print(a + b)    # 14.0 (Python convertit a en float)
6 # print(a + c) # ERREUR : on ne peut pas additionner
7                # un int et une str

```

Le type `int` donne un résultat **exact**; le type `float` peut introduire de petites erreurs d'arrondi :

```

1 print(0.1 + 0.2) # 0.30000000000000004 (pas 0.3 exactement)

```

### 1.1.3 Convertir un type

On peut **convertir** une valeur d'un type vers un autre avec les fonctions `int()`, `float()` et `str()`.

#### Exemple 1.5 – Conversions courantes

```

1 x = 3.7
2 n = int(x)           # tronque (pas arrondi) : n vaut 3
3 print(n)            # 3
4
5 p = 5
6 r = float(p)        # r vaut 5.0
7 print(r)            # 5.0
8
9 age = 17
10 msg = "J'ai " + str(age) + " ans"
11 print(msg)         # J'ai 17 ans

```

#### Exemple 1.6 – Division entière et division réelle

Python distingue deux opérateurs de division :

```

1 print(7 / 2) # 3.5 -- division réelle (résultat de type float)
2 print(7 // 2) # 3 -- quotient entier (résultat de type int)
3 print(7 % 2) # 1 -- reste (modulo) (résultat de type int)

```

En mathématiques :  $7 = 2 \times 3 + 1$ , donc  $7//2 = 3$  et  $7\%2 = 1$ .

Ainsi, le quotient dans la division euclidienne de 7 par 2 est  $7//2 = 3$  et le reste est  $7\%2 = 1$ .

### 1.1.4 Affectations multiples

#### Méthode – Affectations multiples et inversion

Python permet d'affecter plusieurs variables en une seule ligne :

```
1 a, b = 3, 5 # a vaut 3 et b vaut 5
```

Et surtout d'échanger deux variables **sans variable temporaire** :

```

1 a, b = 5, 3
2 a, b = b, a # échange : a vaut 3, b vaut 5
3 print(a, b) # 3 5

```

Dans la plupart des langages (C, Java...), il faudrait écrire trois lignes et créer une variable temporaire `temp` :

```
1 temp = a ; a = b ; b = temp
```

Python rend cela inutile – et vous croiserez cette facilité souvent en prépa ECG.

#### Exemple 1.7 – Initialiser plusieurs variables d'un coup

```

1 x, y, z = 1, 2, 3
2 print(x + y + z) # 6

```

**Exemple 1.8 – Inverser les bornes d'un intervalle**

En anticipant un peu sur les test, voici un exemple d'utilisation de cette facilité offerte par Python : on se donne deux variables  $a$  et  $b$  et on veut les classer dans l'ordre croissant : Prenons  $a = 10$  et  $b = 3$  :

```

1 a, b = 10, 3
2 if a > b:
3     a, b = b, a      # inversion si nécessaire
4 print(a, b)         # 3 10

```

Prenons  $a = 5$  et  $b = 7$  :

```

1 a, b = 5, 7
2 if a > b:
3     a, b = b, a      # inversion si nécessaire
4 print(a, b)         # 5 7

```

**Application mathématique – Traduire une formule mathématique en Python**

En mathématiques, on manipule des formules comme  $v = d/t$  ou  $S = \pi r^2$ . En Python, on les traduit directement :

```

1 import math
2
3 # Vitesse moyenne
4 d = 150      # distance en km
5 t = 2        # temps en heures
6 v = d / t    # on calcule la vitesse en km/h
7 print(v)     # 75.0
8
9 # Aire d'un disque
10 r = 5
11 S = math.pi * r**2    # r**2 signifie r^2
12 print(round(S, 2))     # 78.54

```

La puissance s'écrit `**` en Python :  $r**2 = r^2$ ,  $r**3 = r^3$ , etc.

## 1.2 Exercices

**Exercice 1.1 ★ – Rectangle**

→ corrige p. 6

On considère un rectangle de longueur  $\ell = 12$  cm et de largeur  $L = 7$  cm.

1. Affecter ces valeurs à deux variables `longueur` et `largeur`.
2. Calculer et afficher le périmètre  $P = 2(\ell + L)$  et l'aire  $A = \ell \times L$ .

**Exercice 1.2 ★ – Conversion de température**

→ corrige p. 6

La formule de conversion Celsius → Fahrenheit est :  $F = C \times 1,8 + 32$ .

Écrire un programme qui calcule et affiche la température en Fahrenheit pour  $C = 0$  °C,  $C = 25$  °C et  $C = 232 + 7/9$  °C.