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

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

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 **systématiquement 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

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

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

On peut ensuite utiliser ces variables dans des calculs :

```
x = 5
y = 3
somme = x + y
print(somme)       # affiche 8

z=3                # z contient la valeur 3
z=z+4              # z contient 3+4=7
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 ?

```
capital = 1000
print(capital)              # 1000
capital = capital * 1.05    # capitalisation à 5%
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)`.

```
n = 7
x = 3.14
prenom = "Alice"
notes = [15, 12, 18]

print(type(n))        # <class 'int'>
print(type(x))        # <class 'float'>
print(type(prenom))   # <class 'str'>
print(type(notes))    # <class 'list'>
```

Exemple 1.4 – Pièges courants avec les types

```
a = 7        # int
b = 7.0      # float  -- pas la meme chose !
c = "7"      # str    -- ce n'est pas le nombre 7

print(a + b)    # 14.0  (Python convertit a en float)
# print(a + c) # ERREUR : on ne peut pas additionner
                # un int et une str
```

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

```
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

```
x = 3.7
n = int(x)            # tronque (pas arrondi) : n vaut 3
print(n)              # 3

p = 5
r = float(p)          # r vaut 5.0
print(r)              # 5.0

age = 17
msg = "J'ai " + str(age) + " ans"
print(msg)            # J'ai 17 ans
```

Exemple 1.6 – Division entière et division réelle

Python distingue deux opérateurs de division :

```
print(7 / 2)   # 3.5 -- division reelle (résultat de type float)
print(7 // 2)  # 3   -- quotient entier  (résultat de type int)
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 :

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

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

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

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

```
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

```
x, y, z = 1, 2, 3
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 $b = 3$:

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

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

```
a, b = 5, 7
if a > b:
    a, b = b, a     # inversion si nécessaire
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 :

```
import math

# Vitesse moyenne
d = 150          # distance en km
t = 2            # temps en heures
v = d / t        # on calcule la vitesse en km/h
print(v)         # 75.0

# Aire d'un disque
r = 5
S = math.pi * r**2     # r**2 signifie r^2
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.