PRÉPA RATIONNELLE
FORMATION PYTHON ECG

Extrait gratuit
25 pages sur 408

COURS · EXERCICES CORRIGÉS · ANNALES 2023-2026 · RÉFÉRENCE DES COMMANDES

Python pour la *prépa ECG*

Le livre : toutes les commandes du programme, puis chapitre par chapitre le cours, les exercices d'application, les exercices d'entraînement, les exercices calqués sur les annales et un QCM.

Maths approfondies & appliquées | ECG 1 · ECG 2 | 273 exercices corrigés | 117 commandes | 131 questions de concours analysées

↘ à travailler un stylo à la main

LE RÉFLEXE

Avant d'écrire un programme, trois questions : *qu'est-ce que j'initialise ? quelles boucles ? qu'est-ce que je renvoie ?*

`S = 0`	`for k in`	`return S`	`rd.random() < p`
INITIALISATION	TRAITEMENT	SORTIE	UNE SIMULATION

Sommaire

D'abord la méthode et toutes les commandes du programme. Puis cinq chapitres, chacun dans le même ordre : le cours, les exercices (application, entraînement, annales), le QCM. Enfin les annales décortiquées et le mémo. Chaque chapitre s'ouvre sur la liste de ses exercices avec leurs pages.

AVANT DE COMMENCER

Mode d'emploi 4
Le réflexe : Initialisation → Traitement → Sortie 5
Les trois gestes des sujets 5
Ce que disent les jurys 6
La règle d'or du style 7

LES COMMANDES DU PROGRAMME

Importer les bibliothèques 8
Bases du langage 9
Listes 9
Fonctions mathématiques 10
Créer des vecteurs et des matrices (numpy) 10
Calculer avec des matrices 11
Algèbre linéaire (numpy.linalg) 11
Simulations aléatoires (numpy.random) 11
Graphiques (matplotlib.pyplot) 12
scipy 12
Traitement de données (pandas) 12
SQL — Requêtes 13
SQL — Opérateurs et options 13
SQL — Fonctions d'agrégation 14
Les paramètres qui piègent 14

CHAPITRE 1 · LES BASES

Le cours 16
01 · Pourquoi miser sur Python ? 16
02 · Les principales librairies 17
03 · Types fondamentaux et variables 18
04 · Affichage et saisie 19
05 · Fonctions natives, numpy & personnalisées 19
06 · Les listes Python 19

07 · Structures de contrôle 20
08 · Script vs fonction 21
09 · Méthode de résolution 22
10 · Les 3 types d'exercices 23
11 · Bien se préparer 23
12 · Récapitulatif des commandes 24
13 · Algorithmes classiques sur les listes 25
Les exercices (61) 27
QCM (15 questions) 96

CHAPITRE 2 · ANALYSE

Le cours 101
01 · Calcul de sommes 101
02 · Calcul de produits 103
03 · Suites par récurrence 104
04 · Fonctions et constantes NumPy 107
05 · Définir une fonction (def) 107
06 · Méthode de dichotomie 108
07 · Tracer une fonction et une suite 110
08 · La méthode des rectangles 110
Les exercices (61) 112
QCM (15 questions) 181

CHAPITRE 3 · ALGÈBRE

Le cours 186
01 · Introduction à NumPy 186
02 · Création de matrices 186
03 · Indexation et accès 187
04 · Opérations matricielles 188
05 · Fonctions d'analyse et algèbre linéaire 189
06 · Construire une matrice par une formule 189
07 · Algèbre linéaire avec numpy.linalg 190
08 · Graphes et matrice d'adjacence 190
Les exercices (59) 192
QCM (15 questions) 258

CHAPITRE 4 · PROBABILITÉS

Le cours 262
01 · Imports et principes 262
02 · Génération aléatoire de base 263
03 · Lois discrètes et continues 263
04 · Fonctions « maison » 264
05 · Statistiques descriptives 264
06 · Graphiques de base 265
07 · Méthode de Monte-Carlo 265
08 · Deux variables : nuage et régression 265
09 · Chaînes de Markov 266

10 · Loi des grands nombres et théorème central limite 267
11 · Estimation et intervalles de confiance 267
12 · Simuler une loi à partir de rd.random 267
13 · Trois scripts classiques 268
14 · Mémo-commandes 268
15 · Quiz express 269
Les exercices (76) 270
QCM (15 questions) 361

CHAPITRE 5 · SQL

Le cours 366
01 · Une table, une requête 366
02 · Compter, sommer, moyenner : les agrégats 367
03 · Croiser deux tables : la jointure 367
Les exercices (6) 369

DÉCORTIQUER LES ANNALES 2023-2026

Le tableau de bord 376
La méthode de lecture d'un script à trous 378
Quatre sujets décortiqués 379
Les 131 questions, une par une 382
À l'oral d'HEC 388
Les classiques d'avant 2023, traduits en Python (10 exercices) 389

MÉMO

Dix patrons 404
Les vingt erreurs qui coûtent des points 406

AVANT DE COMMENCER

Comment *travailler* avec ce livre

Cinq pages à lire d'abord : le mode d'emploi, le réflexe à prendre avant chaque programme, les trois gestes des sujets, ce que disent les jurys, la règle d'or du style.

01 Mode d'emploi

Le livre suit l'ordre dans lequel on apprend : d'abord **toutes les commandes du programme**, puis cinq chapitres (les bases, l'analyse, l'algèbre, les probabilités et statistiques, SQL). Chaque chapitre a la même structure.

ÉTAPE	CE QU'ON Y TROUVE	COMMENT S'EN SERVIR
Le cours	Les notions, un tableau de commandes par notion, des paires « ça marche / ça ne marche pas », les conseils du prof	Lire une fois, puis revenir quand un exercice bloque
Application	Des exercices courts, un geste à la fois	À faire tous, sans regarder le corrigé
Entraînement	Les exercices de la formation Python ECG de Prépa Rationnelle	Le cœur du travail : un par jour tient dans 20 minutes
Annales	Des exercices calqués sur les questions des concours 2023-2026, avec la source	Dans les conditions du concours : au stylo, sans ordinateur
QCM	Quinze questions, réponses et explications à la page suivante	Pour vérifier qu'on a retenu

Un exercice par page. L'énoncé, le corrigé et une explication en quatre blocs : *En bref* (l'idée), *Étapes* (le déroulé), *Pièges* (ce qui coûte des points), *Notions* (le vocabulaire). Cache le corrigé avec une feuille, écris ta réponse, puis compare.

PASTILLE	SENS
LIRE	Un script est donné : dire ce qu'il fait, ce qu'il affiche, ou l'expliquer
COMPLÉTER	Un script à trous : chaque case `____` attend une expression
ÉCRIRE	Partir de zéro : une fonction ou un script
CLASSIQUE	Une mécanique qui revient d'un sujet à l'autre : à savoir refaire de tête
APPLIQUÉES / APPROFONDIES	Au programme d'une seule voie ; sans pastille, les deux voies sont concernées

LES QUATRE IMPORTS, TOUJOURS LES MÊMES

```
import numpy as np
import numpy.linalg as al
import numpy.random as rd
import matplotlib.pyplot as plt
```

Tous les corrigés utilisent ces quatre alias. Ils sont rappelés en tête des sujets de concours : les connaître évite de perdre une minute à les chercher.

02 Le réflexe : *Initialisation → Traitement → Sortie*

LE CONSEIL DU PROF

Quand tu commences un programme Python, il y a trois choses à te demander. **Qu'est-ce que j'ai besoin d'initialiser ? Quelles boucles vais-je employer ? Qu'est-ce qu'on me demande en sortie ?** Ces trois questions suffisent pour écrire n'importe quel script du concours.

ÉTAPE	SCRIPT	FONCTION
Initialisation	Variables et compteurs avant la boucle	Paramètres d'entrée + compteurs locaux
Traitement	Calculs, boucles, conditions	Même logique, variables locales
Sortie	`print(...)` (à l'utilisateur)	`return ...` (à l'appelant)

1 Initialisation

Préparer variables, compteurs, listes. Une somme part de `S = 0`, un produit de `P = 1`, une liste de `L = []`, un compteur de `n = 0`.

2 Traitement

Répéter avec `for` quand le nombre de tours est connu, avec `while` quand il dépend d'une condition ; choisir avec `if`.

3 Sortie

Afficher dans un script (`print`), renvoyer dans une fonction (`return`). C'est l'appel qu'on affiche : `print(f(10))`.

LE MÊME EXERCICE, DÉROULÉ DES DEUX FAÇONS

SCRIPT : N! POUR N = 6

```
n = 6
P = 1                       #
initialisation
for k in range(1, n+1):     # traitement
    P *= k
print(P)                    # sortie :
720
```

FONCTION : N! POUR N QUELCONQUE

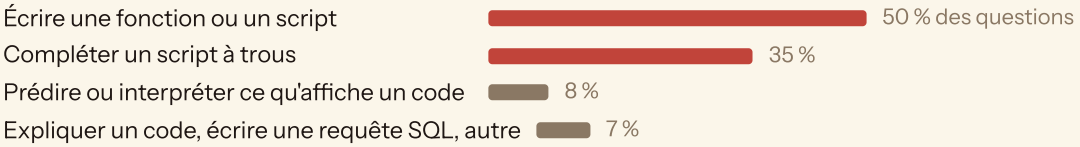
```
def factorielle(n):
    P = 1                   #
initialisation
    for k in range(1, n+1):
        P *= k              # traitement
    return P                # sortie

print(factorielle(6))       # 720
```

LE PIÈGE QUE VOIT CHAQUE CORRECTEUR

Une fonction qui affiche au lieu de renvoyer. `print` montre une valeur à l'écran et la perd ; `return` la rend disponible pour la suite du calcul. Un candidat sur deux l'ignore encore le jour du concours.

03 Les trois *gestes* des sujets

Sur les 131 questions Python des sujets BCE et Ecricome de 2023 à 2026 (les deux voies), la tâche demandée se répartit ainsi :



Chaque geste a sa méthode. Les trois sont reprises au fil des chapitres ; les voici une fois pour toutes.

MÉTHODE · COMPLÉTER UN SCRIPT À TROUS

1. Avant la boucle : que doit contenir chaque variable ? (souvent le terme de rang 0 ou 1).
2. À la fin d'un tour : que doit-elle contenir ? Écris la mise à jour qui y mène.
3. Dernier tour : quelle est la dernière valeur de l'indice ? Ajuste range.
4. Vérifie à la main sur n = 2.

MÉTHODE · TROUVER LE PREMIER RANG : ÉCRIRE LA NÉGATION

1. Écris la condition d'arrêt : « u > a ».
2. Mets sa négation dans le while : « u <= a ».
3. Dans la boucle : terme suivant, puis compteur + 1.

```
n = 0
u = 1
while u ⩽ a:
    u = u + 1/u
    n += 1
```

MÉTHODE · LIRE UN CODE : LE FAIRE TOURNER À LA MAIN

1. Un tableau : une colonne par variable, une ligne par tour.
2. Déroule trois tours, reconnais la formule, puis conclus.

ÉCRIRE UNE FONCTION : LES TROIS QUESTIONS

Initialiser quoi ? Quelles boucles ? Renvoyer quoi ? Puis vérifier à la main sur une petite valeur (n = 2 ou 3) avant de passer à la question suivante.

04 Ce que disent les *jurys*

Les rapports de jury se répètent d'une année à l'autre : les questions Python sont abordables, une partie des candidats les saute, les scripts à trous sont mal complétés, et une réponse qui ne suit pas la méthode demandée ne rapporte rien.

> *Le programme à trous a presque toujours été mal complété.*
>
> Rapport HEC-ESSEC, maths appliquées 2024

> *Une majorité de candidats traitent les questions de python, mais une (trop grosse) minorité fait visiblement l'impasse dessus.*
>
> Rapport Ecricome, maths approfondies 2024

> *Les questions Python [...] étaient abordables et n'ont pas été suffisamment traitées. [...] Davantage de questions Python pourraient être présentes à l'avenir.*
>
> Rapport emlyon, maths approfondies 2024

> *La commande GROUP BY a rarement été comprise par les candidats.*
>
> Rapport Ecricome, maths appliquées 2026

> *Tout candidat qui sait programmer n'a pas à se restreindre à une réponse dans le cadre du programme [...] une réponse correcte, même hors programme, est complètement validée.*
>
> Rapport emlyon, maths approfondies 2026

> *Toutes les planches comportaient une question d'informatique faisant intervenir soit le langage Python, soit du SQL.*
>
> Rapport de l'oral HEC, maths appliquées 2024

> *En ce qui concerne le SQL, le jury a pu constater que sa maitrise était trop souvent insuffisante.*
>
> Rapport de l'oral HEC, maths appliquées 2025

> *Les candidats doivent savoir faire tourner du code Python à la main sur des exemples simples.*
>
> Rapport de l'oral HEC, maths appliquées 2025

LE CONSEIL DU PROF

Python, ce sont 3 à 4 points « vraiment trop simples à obtenir ». Les questions Python sont indépendantes du reste : tu peux les traiter même si tu as bloqué avant. En fin d'épreuve, si le temps file, reviens dessus.

05 La règle d'or du *style*

> *La version la plus simple à penser et à inventer. Pas la plus courte, pas la plus optimisée : celle qu'un élève de prépa peut reproduire de tête, en partant de zéro.*
>
> La règle de tous les corrigés de ce livre

LES RÈGLES DE LA MAISON

- Indice de boucle `k` quand il sert, `_` sinon (et `i`, `j` pour une matrice).
- Accumulateurs courts : `S`, `P`, `U`, `L` ; compteurs `n`, `c`.
- Initialiser **avant** la boucle : `S = 0`, `P = 1`, `L = []`.
- Accumuler avec `+=`, `*=`, `.append`.
- Une fonction **renvoie** (`return`) ; le `print` se fait à l'appel.
- Pile/Face, urnes, dés : des **nombres**, jamais des chaînes de caractères : `rd.random() < p`, compteurs `r`, `v`.
- Pas de commentaire ligne par ligne : le code se lit seul.

HORS PROGRAMME : À NE JAMAIS ÉCRIRE

- `import random`, `random.choice`, `random.shuffle` : seul `numpy.random` est au programme.
- `math.exp`, `math.factorial` : seul `numpy` est au programme ; une factorielle se calcule avec une boucle.
- `enumerate`, `lambda`, `break`, `np.fromfunction`.
- `A @ B` pour le produit matriciel : écrire `np.dot(A, B)`.
- Les f-strings décoratives : `print(S)` suffit.

Le jury d'emlyon 2026 précise qu'une réponse correcte hors programme est validée. Mais une réponse dans le programme est toujours plus sûre : elle est lue par un correcteur qui attend exactement ces commandes.

RÉFÉRENCE

Les *117 commandes* du programme

Tout ce qu'un sujet peut demander, famille par famille. Pour chaque commande : ce qu'elle fait, son statut dans le programme officiel, la voie concernée, et le nombre de sujets 2023-2026 où elle apparaît.

81 commandes exigibles (le programme officiel, BO spécial n° 1 du 11 février 2021), auxquelles s'ajoutent les commandes **rappelées dans le sujet** quand il en a besoin, et quelques commandes **hors liste** que les sujets ont tout de même utilisées. La colonne « Sujets » compte les épreuves écrites BCE et Ecricome, sur 40, où la commande figure dans une question ou dans l'encadré de rappel du sujet.

LES COMMANDES LES PLUS DEMANDÉES	SUJETS SUR 40
`def … return`	30
`for k in range(…)`	25
`if / elif / else`	19
`np.zeros` · `plt.plot`	15
`while`	12
`np.exp` , `np.log` , `np.sqrt`	12
`L.append` · `rd.random`	11
`len(L)`	10

IMPORTER LES BIBLIOTHÈQUES

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`import numpy as np`	Importation de numpy avec l'alias np	EXIGIBLE		
`import numpy.linalg as al`	Importation de l'algèbre linéaire avec l'alias al	EXIGIBLE		
`import numpy.random as rd`	Importation des simulations aléatoires avec l'alias rd	EXIGIBLE		
`import matplotlib.pyplot as plt`	Importation des graphiques avec l'alias plt	EXIGIBLE		
`import scipy.special as sp`	Importation de scipy.special (pour sp.ndtr)	EXIGIBLE	APPROFONDIES	
`import pandas as pd`	Importation de pandas avec l'alias pd	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`from numpy import *`	Importer toutes les fonctions sans préfixe (déconseillé : on ne sait plus d'où vient chaque fonction)	EXIGIBLE		

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`import math as m`	Le module math : ses fonctions ne s'appliquent pas aux vecteurs, on lui préfère numpy	HORS LISTE		

BASES DU LANGAGE

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`x = expression`	Affectation : la variable x reçoit la valeur de l'expression (nombre, booléen, matrice, texte)	EXIGIBLE		
`# commentaire`	Tout ce qui suit # sur la ligne est ignoré par Python	EXIGIBLE		
`+ - * / **`	Opérations de base ; **est la puissance (2**10 vaut 1024)	EXIGIBLE		
`== ≠ < > ⩽ ⩾`	Comparaisons : elles renvoient True ou False	EXIGIBLE		
`True False and or not`	Booléens et opérateurs logiques	EXIGIBLE		
`if ... : / elif ... : / else :`	Instruction conditionnelle : le bloc indenté ne s'exécute que si la condition est vraie	EXIGIBLE		19
`for k in range(a, b):`	Boucle pour k = a, a+1, …, b−1 (la borne b est exclue)	EXIGIBLE		25
`for x in T:`	Boucle sur les éléments d'une liste, d'un vecteur ou d'une chaîne	EXIGIBLE		
`while condition:`	Boucle tant que la condition est vraie : on y écrit la négation de la condition d'arrêt	EXIGIBLE		12
`def f(x, y): … return …`	Définition d'une fonction : elle renvoie un résultat avec return, elle ne l'affiche pas	EXIGIBLE		30
`return a, b`	Renvoyer plusieurs valeurs ; à l'appel : a, b = f(…)	EXIGIBLE		
`a, b = b, a + b`	Affectation simultanée : les deux nouvelles valeurs sont calculées avec les anciennes	HORS LISTE		
`print(x)`	Affiche la valeur de x	HORS LISTE		
`n = int(input('n = '))`	Demande un entier à l'utilisateur (input renvoie du texte, int le convertit)	HORS LISTE		3
`break continue`	Sortir d'une boucle, passer au tour suivant : non exigibles, on préfère une condition while	RAPPELÉE DANS LE SUJET		

LISTES

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`L = []`	Créer une liste vide	EXIGIBLE	APPLIQUÉES	
`[a]*n`	Créer une liste avec n fois l'élément a	EXIGIBLE	APPLIQUÉES	
`L.append(a)`	Ajoute l'élément a à la fin de la liste L	EXIGIBLE	APPLIQUÉES	11
`len(L)`	Renvoie le nombre d'éléments de la liste L	EXIGIBLE	APPLIQUÉES	10
`L[k]`	L'élément d'indice k ; les indices vont de 0 à len(L) − 1	EXIGIBLE	APPLIQUÉES	
`[f(k) for k in range(n)]`	Liste en compréhension : la liste des f(k) pour k de 0 à n − 1	EXIGIBLE	APPLIQUÉES	9

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`a in L`	Vaut True si a se trouve au moins une fois dans L et False sinon	EXIGIBLE	APPLIQUÉES	
`del L[k]`	Supprime l'élément d'indice k de la liste L	EXIGIBLE	APPLIQUÉES	
`L.count(a)`	Renvoie le nombre d'occurrences de a dans la liste L	EXIGIBLE	APPLIQUÉES	
`L1 + L2`	Concatène les deux listes L1 et L2	EXIGIBLE	APPLIQUÉES	
`range(a, b, p)`	Les entiers de a à b − 1 avec un pas p (range(0, 10, 2) : 0, 2, 4, 6, 8)	EXIGIBLE		
`max(L) min(L) sum(L)`	Plus grand, plus petit élément et somme d'une liste	HORS LISTE		
`L.pop(k) L.remove(a)`	Retirer un élément (par sa position ou par sa valeur)	HORS LISTE		

FONCTIONS MATHÉMATIQUES

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`np.exp(x) np.log(x) np.sqrt(x)`	Exponentielle, logarithme népérien, racine carrée (aussi sur un vecteur, coefficient par coefficient)	EXIGIBLE		12
`np.abs(x)`	Valeur absolue	EXIGIBLE		5
`np.floor(x)`	Partie entière	EXIGIBLE		1
`np.sin(x) np.cos(x)`	Sinus et cosinus	EXIGIBLE	APPROFONDIES	
`np.e np.pi`	Les constantes e et π	EXIGIBLE		
`f = np.vectorize(f)`	Rend une fonction applicable à un vecteur, coefficient par coefficient	RAPPELÉE DANS LE SUJET		
`m.factorial(k)`	k! avec le module math (une boucle suffit et vaut mieux au concours)	HORS LISTE		

CRÉER DES VECTEURS ET DES MATRICES (NUMPY)

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`np.array([[1, 2], [3, 4]])`	Crée une matrice à partir de la liste de ses lignes	EXIGIBLE		8
`np.zeros([n, m])`	Crée la matrice nulle de taille n × m (np.zeros(n) : vecteur nul)	EXIGIBLE		15
`np.ones([n, m])`	Crée la matrice de taille n × m dont tous les coefficients valent 1	EXIGIBLE		5
`np.eye(n)`	Crée la matrice identité de taille n	EXIGIBLE		4
`np.linspace(a, b, n)`	Crée un vecteur de n valeurs régulièrement réparties entre a et b inclus	EXIGIBLE		9
`np.arange(a, b, p)`	Crée le vecteur a, a + p, a + 2p, … strictement inférieur à b	EXIGIBLE		7
`M[i, j] M[i, :] M[:, j]`	Un coefficient, la ligne i, la colonne j (indices à partir de 0)	EXIGIBLE		
`a, b = np.shape(M)`	Le nombre de lignes et de colonnes de M	EXIGIBLE		8
`np.diag(L)`	Crée la matrice diagonale dont la diagonale est L	HORS LISTE		

CALCULER AVEC DES MATRICES

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`A + B A * B A ** 2`	Opérations coefficient par coefficient (A * B n'est PAS le produit matriciel)	EXIGIBLE		
`M == N M >= 1`	Comparaisons coefficient par coefficient : on obtient une matrice de booléens	EXIGIBLE		
`np.dot(A, B)`	Produit matriciel AB (syntaxe exigible ; A.dot(B) ne l'est pas)	EXIGIBLE		5
`np.transpose(M)`	Transposée de M	EXIGIBLE		3
`np.sum(M) np.sum(M, 0) np.sum(M, 1)`	Somme de tous les coefficients, de chaque colonne, de chaque ligne	EXIGIBLE		3
`np.cumsum(x)`	Vecteur des sommes cumulées x_0, $x_0 + x_1$, …	EXIGIBLE		—
`np.min(x) np.max(x)`	Plus petit et plus grand coefficient	EXIGIBLE		1
`np.mean(x) np.median(x)`	Moyenne et médiane (avec 0 ou 1 en second argument : par colonne ou par ligne)	EXIGIBLE		2
`np.var(x) np.std(x)`	Variance et écart-type empiriques	EXIGIBLE		1
`np.prod(x)`	Produit des coefficients	HORS LISTE		

ALGÈBRE LINÉAIRE (NUMPY.LINALG)

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`al.inv(M)`	Inverse de la matrice carrée M, si elle est inversible	EXIGIBLE		3
`al.matrix_rank(M)`	Rang de la matrice M (le programme l'appelle al.rank)	EXIGIBLE		3
`al.matrix_power(M, n)`	Puissance n-ième de la matrice carrée M	EXIGIBLE		5
`al.solve(A, B)`	Solution X du système AX = B	EXIGIBLE		—
`D, P = al.eig(M)`	D : vecteur des valeurs propres ; P : matrice dont les colonnes sont des vecteurs propres	EXIGIBLE		3

SIMULATIONS ALÉATOIRES (NUMPY.RANDOM)

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`rd.random() rd.random(n)`	Un réel, ou n réels, de loi uniforme sur [0, 1[	EXIGIBLE		11
`rd.randint(a, b)`	Un entier de loi uniforme sur [[a, b − 1]] (la borne b est exclue)	EXIGIBLE		9
`rd.binomial(n, p)`	Loi binomiale B(n, p) ; rd.binomial(n, p, N) : un vecteur de N tirages	EXIGIBLE		2
`rd.geometric(p)`	Loi géométrique G(p), à valeurs dans $\mathbb{N}^*$	EXIGIBLE		3
`rd.poisson(a)`	Loi de Poisson de paramètre a	EXIGIBLE		1
`rd.exponential(a)`	Loi exponentielle d'ESPÉRANCE a, donc de paramètre 1/a (piège classique)	EXIGIBLE		5
`rd.normal(m, s)`	Loi normale de moyenne m et d'ÉCART-TYPE s (pas la variance)	EXIGIBLE		6
`rd.gamma(a)`	Loi gamma de paramètre a	EXIGIBLE		3
`rd.binomial(10, 0.2, [100, 10])`	Le dernier argument donne la taille : un nombre, ou [lignes, colonnes]	EXIGIBLE		
`rd.uniform(a, b)`	Loi uniforme sur [a, b] ; au programme, on écrit a + (b − a)*rd.random()	HORS LISTE		1

GRAPHIQUES (MATPLOTLIB.PYPLOT)

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`plt.plot(X, Y)`	Trace la courbe passant par les points (X[k], Y[k]) ; plt.plot(X, Y, '+') : des points	EXIGIBLE		15
`plt.show()`	Affiche le graphique (tous les plt.plot précédents sur la même figure)	EXIGIBLE		
`plt.hist(x, bornes)`	Histogramme ; density=True pour le comparer à une densité	EXIGIBLE		—
`plt.bar(X, Y)`	Diagramme en bâtons	EXIGIBLE	APPLIQUÉES	1
`plt.boxplot(x)`	Diagramme en boîte	EXIGIBLE	APPLIQUÉES	—
`plt.xlim plt.ylim plt.axis plt.grid plt.legend`	Réglages du graphique : non exigibles	RAPPELÉE DANS LE SUJET		
`X, Y = np.meshgrid(x, y)`	Grille de points pour une fonction de deux variables	RAPPELÉE DANS LE SUJET	APPROFONDIES	—
`plt.contour(X, Y, Z)`	Lignes de niveau d'une fonction de deux variables	RAPPELÉE DANS LE SUJET	APPROFONDIES	—
`plt.quiver(X, Y, U, V)`	Champ de vecteurs, par exemple les gradients	RAPPELÉE DANS LE SUJET	APPROFONDIES	
`ax = Axes3D(plt.figure()) ax.plot_surface(X, Y, Z)`	Surface en 3D (from mpl_toolkits.mplot3d import Axes3D)	RAPPELÉE DANS LE SUJET	APPROFONDIES	
`plt.scatter(x, y)`	Nuage de points (on peut aussi écrire plt.plot(x, y, '.'))	HORS LISTE		

SCIPY

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`sp.ndtr(x)`	$\Phi(x)$, fonction de répartition de la loi normale centrée réduite	EXIGIBLE	APPROFONDIES	2
`sp.ndtri(y)`	$\Phi^{-1}(y)$, le quantile de la loi normale (ndtri(0.975) ≈ 1,96)	HORS LISTE		1
`scipy.integrate.odeint`	Résolution numérique d'une équation différentielle	RAPPELÉE DANS LE SUJET	APPLIQUÉES	

TRAITEMENT DE DONNÉES (PANDAS)

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`df = pd.read_csv('data.csv')`	Crée un tableau pandas à partir d'un fichier CSV	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`df.head(n)`	Les n premières lignes du tableau df	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`df.shape`	Le couple (nombre de lignes, nombre de colonnes)	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`df.describe()`	Un résumé statistique de chaque colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`df.mean() df.std() df.median()`	Moyenne, écart-type et médiane de chaque colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`df.count()`	Les effectifs de chaque colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`df.sort_values(by='col')`	Trie le tableau suivant la colonne 'col' (ascending=False pour décroissant)	RAPPELÉE DANS LE SUJET	APPLIQUÉES	

SQL — REQUÊTES

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`SELECT col1, col2 FROM table`	Les colonnes demandées de la table (* pour toutes)	EXIGIBLE	APPLIQUÉES	3
`SELECT … FROM … WHERE condition`	Ne garde que les lignes qui vérifient la condition	EXIGIBLE	APPLIQUÉES	3
`SELECT * FROM t1 INNER JOIN t2 ON t1.a = t2.b`	Jointure : la seule au programme	EXIGIBLE	APPLIQUÉES	1
`INSERT INTO table VALUES (v1, v2, …)`	Ajoute une ligne à la table	EXIGIBLE	APPLIQUÉES	
`UPDATE table SET col = val WHERE condition`	Modifie les lignes qui vérifient la condition (sans WHERE : toutes)	EXIGIBLE	APPLIQUÉES	
`DELETE FROM table WHERE condition`	Supprime les lignes qui vérifient la condition	EXIGIBLE	APPLIQUÉES	
`CREATE TABLE table (col TYPE, …)`	Crée une table ; types au programme : INTEGER et TEXT	EXIGIBLE	APPLIQUÉES	
`PRIMARY KEY FOREIGN KEY`	Clé primaire (identifie une ligne) et clé étrangère (renvoie à une autre table)	EXIGIBLE	APPLIQUÉES	
`UNION INTERSECT EXCEPT`	Réunion, intersection et différence de deux requêtes	RAPPELÉE DANS LE SUJET	APPLIQUÉES	

SQL — OPÉRATEURS ET OPTIONS

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`= <> < ⩽ > ⩾`	Comparaisons ; <> signifie « différent de »	EXIGIBLE	APPLIQUÉES	
`AND OR NOT`	Opérateurs logiques pour combiner des conditions	EXIGIBLE	APPLIQUÉES	
`+ - *`	Opérations arithmétiques dans une requête	EXIGIBLE	APPLIQUÉES	
`ORDER BY col (DESC)`	Trie le résultat (DESC : décroissant)	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`DISTINCT`	Supprime les lignes en double du résultat	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`GROUP BY col`	Un résultat par groupe de lignes de même valeur (demandé à Ecricome 2026, « rarement compris » selon le jury)	HORS LISTE	APPLIQUÉES	1

SQL — FONCTIONS D'AGRÉGATION

COMMANDE	CE QU'ELLE FAIT	STATUT	VOIE	SUJETS
`COUNT(*)`	Le nombre de lignes	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`AVG(col)`	La moyenne d'une colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`SUM(col)`	La somme d'une colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	
`MIN(col)` `MAX(col)`	La plus petite et la plus grande valeur d'une colonne	RAPPELÉE DANS LE SUJET	APPLIQUÉES	

LES PARAMÈTRES QUI PIÈGENT

COMMANDE	CE QU'ON CROIT	CE QUI EST VRAI
`range(a, b)`	de a à b	de a à **b − 1** : une somme de 1 à n s'écrit `range(1, n+1)`
`rd.randint(a, b)`	un entier de a à b	de a à **b − 1** : un dé, c'est `rd.randint(1, 7)`
`rd.exponential(a)`	le paramètre λ	l'**espérance** 1/λ : E(λ) s'écrit `rd.exponential(1/lam)`
`rd.normal(m, s)`	la variance	l'**écart-type** s
`np.linspace(a, b, n)`	un pas de n	**n points** de a à b inclus
`A * B`	le produit matriciel	le produit **coefficient par coefficient** : écrire `np.dot(A, B)`
`np.ones(3)`	une matrice 3 × 3	un **vecteur** de trois 1 : la matrice est `np.ones((3, 3))`
`L[n]` , `A[n, n]`	le n-ième élément	hors de la liste : le dernier est `L[n-1]`
`plt.hist(x)`	des fréquences	des **effectifs** : ajouter `density=True` pour comparer à une loi
`M = L`	une copie	un **deuxième nom** pour la même liste : copier avec `list(L)`

CHAPITRE 1 · LES BASES

Les *exercices*

Un exercice par page : l'énoncé, le corrigé et l'explication. Cache le corrigé, écris ta réponse, puis compare.

APPLICATION

1.1 Afficher les entiers de 1 à n ÉCRIRE
1.2 Que valent a et b ? LIRE
1.3 Somme des n premiers entiers ÉCRIRE
1.4 Division entière et modulo LIRE
1.5 Année bissextile ÉCRIRE
1.6 Somme des carrés COMPLÉTER
1.7 Fonction signe et valeur absolue ÉCRIRE
1.8 Boucle while mystère LIRE
1.9 Construire une liste de termes ÉCRIRE
1.10 Indices d'une liste LIRE
1.11 Compter les éléments pairs COMPLÉTER
1.12 Moyenne et minimum sans fonction toute faite ÉCRIRE
1.13 Nombre de chiffres d'un entier ÉCRIRE
1.14 Fonction composée LIRE

ENTRAÎNEMENT

1.15 Somme et produit ÉCRIRE
1.16 Que fait ce script ? LIRE
1.17 Suite récurrente : calculer un terme COMPLÉTER
1.18 Premier rang sous un seuil COMPLÉTER
1.19 Double somme ÉCRIRE
1.20 Factorielle et coefficient binomial ÉCRIRE
1.21 Suite récurrente d'ordre 2 COMPLÉTER
1.22 Suites adjacentes : la mise à jour simultanée COMPLÉTER
1.23 Trouver les deux erreurs LIRE
1.24 Fonction définie par morceaux ÉCRIRE
1.25 Placement : combien d'années pour atteindre 5000 € ? ÉCRIRE
1.26 Listes : append ou compréhension COMPLÉTER
1.27 Compter et calculer des fréquences ÉCRIRE
1.28 Maximum et second maximum ÉCRIRE
1.29 Recherche dichotomique dans une liste triée COMPLÉTER
1.30 Rendu de monnaie : l'algorithme glouton ÉCRIRE
1.31 Les deux valeurs les plus proches ÉCRIRE
1.32 Triangle de Pascal avec des listes ÉCRIRE
1.33 Double somme avec division ÉCRIRE
1.34 Produit double ÉCRIRE
1.35 Coefficient binomial ÉCRIRE
1.36 Puissance d'un nombre ÉCRIRE

1.37 Factorielle et seuil ÉCRIRE
1.38 Partie entière LIRE
1.39 Résolution d'équation du 2e degré ÉCRIRE
1.40 Suite de Fibonacci ÉCRIRE
1.41 Pair ou impair ÉCRIRE
1.42 Jeu du nombre secret ÉCRIRE
1.43 Table de multiplication ÉCRIRE
1.44 Maximum de deux nombres ÉCRIRE
1.45 PGCD par algorithme d'Euclide ÉCRIRE
1.46 Test de primalité ÉCRIRE
1.47 Conjecture de Syracuse ÉCRIRE
1.48 Crible d'Ératosthène ÉCRIRE
1.49 Décomposition en facteurs premiers ÉCRIRE
1.50 Tri par insertion ÉCRIRE
1.51 Maximum d'une liste et son indice ÉCRIRE
1.52 Moyenne des éléments d'une liste ÉCRIRE

ANNALES

1.53 Somme harmonique : le programme qui ne s'arrête pas LIRE
1.54 Seuil : écrire la négation de la condition COMPLÉTER
1.55 Une espérance avec np.arange et np.sum COMPLÉTER
1.56 La liste [1, 2, 2, 3, 3, 3, ...] et le tirage au hasard ÉCRIRE
1.57 Ajouter sans doublon LIRE
1.58 Trier un tableau : le tri par sélection ÉCRIRE
1.59 Trier en gardant la trace des positions COMPLÉTER
1.60 Le programme à trous qui piège COMPLÉTER
1.61 Que s'affiche-t-il ? Une boucle while à la main LIRE

COMPLÉTER FACILE CLASSIQUE

EXERCICE 1.6

Somme des carrés

Le script à trous le plus simple des sujets.

ÉNONCÉ

On veut $S = \sum_{k=1}^{n} k^2$ pour $n = 20$.

Question 1. Complète le script.

```
n = 20
S = _____
for k in range(_____, _____):
    S += _____
print(S)
```

Question 2. Que se passe-t-il si on écrit `range(n)` ?

CORRIGÉ

Question 1

```
n = 20
S = 0
for k in range(1, n+1):
    S += k**2
print(S)
```

Le script affiche 2870, c'est-à-dire n(n+1)(2n+1)/6.

Question 2

La boucle parcourt 0 à 19 : le terme $20^2 = 400$ manque et le terme 0^2 ne change rien. Le script affiche 2470.

EN BREF

Dans un script à trous, chaque trou a une réponse unique dictée par la formule : l'initialisation, les bornes, le terme.

ÉTAPES

1. S = 0 : la somme est vide au départ.
2. range(1, n+1) reproduit k = 1 à n.
3. S += k**2 ajoute le terme général.

PIÈGES

- k^2 n'est pas la puissance en Python : c'est k**2.
- Oublier le +1 de la borne droite.

NOTIONS MOBILISÉES

- ****** — L'opérateur puissance.
- **Terme général** — L'expression de la somme en fonction de k.

EXERCICE 1.16

Que fait ce script ?

Lire une boucle while et dire, en une phrase mathématique, ce que le script calcule.

ÉNONCÉ

Au concours, on te donne souvent un script et on te demande ce qu'il fait. La méthode : suivre les variables à la main sur les premiers tours, puis formuler le résultat en langage mathématique.

Question 1. Qu'affiche ce script ?

```
n = 0
p = 1
while p ⩽ 1000:
    n += 1
    p *= 2
print(n)
```

Question 2. Que représente le nombre affiché ? Réponds par une phrase mathématique.

Question 3. Modifie le script pour obtenir le plus petit entier n tel que $3^n > 10^6$.

CORRIGÉ

Question 1

Il affiche `10`. La boucle double `p` tant qu'il ne dépasse pas 1000 : $2^9 = 512 \leq 1000$ mais $2^{10} = 1024 > 1000$.

Question 2

C'est le plus petit entier n tel que $2^n > 1000$. À chaque tour, `p` vaut 2^n : la boucle s'arrête au premier rang où la condition `p ⩽ 1000` devient fausse.

Question 3

```
n = 0
p = 1
while p ⩽ 10**6:
    n += 1
    p *= 3
print(n)
```

Seuls la base et le seuil changent. Le script affiche 13 : $3^{12} = 531\,441$ et $3^{13} = 1\,594\,323$.

EN BREF

Le script fabrique les puissances de 2 une par une et compte combien il en faut pour dépasser 1000. C'est le modèle de toutes les recherches de seuil.

ÉTAPES

1. Au départ, n = 0 et p = 1 = 2^0.
2. Chaque tour ajoute 1 à n et multiplie p par 2 : on garde toujours p = 2^n.
3. La boucle s'arrête dès que p > 1000.
4. La valeur affichée est donc le premier n tel que 2^n > 1000.

PIÈGES

- Répondre 9 : c'est le dernier n tel que $2^n \leq 1000$, pas le premier tel que $2^n > 1000$.
- Paraphraser le code (« on multiplie p par 2 ») au lieu de donner le sens mathématique.

NOTIONS MOBILISÉES

- **Boucle while** — Elle répète tant que la condition est vraie. On l'utilise quand on ne connaît pas à l'avance le nombre de tours.
- **Invariant** — Une relation vraie à chaque tour, ici p = 2^n. La repérer, c'est comprendre le script.

COMPLÉTER FACILE CLASSIQUE

EXERCICE 2.6

Premier n tel que 2^n dépasse 1000

Le seuil : un while, un compteur, la négation de l'objectif.

ÉNONCÉ

On cherche le plus petit entier n tel que $2^n > 1000$.

Question 1. Complète le script.

```
n = 0
u = 1
while _____:
    u = _____
    n += 1
print(n)
```

Question 2. Retrouve ce résultat par le calcul.

CORRIGÉ

Question 1

```
n = 0
u = 1
while u ⩽ 1000:
    u = 2*u
    n += 1
print(n)
```

Affiche 10 : 2^10 = 1024.

Question 2

$2^n > 1000 \iff n > \frac{\ln 1000}{\ln 2} \approx 9{,}97$: le plus petit entier est $n = 10$.

EN BREF

Tant que l'objectif n'est pas atteint, on passe au terme suivant et on compte.

ÉTAPES

1. n = 0, u = 2^0 = 1.
2. while u <= 1000 : la négation de u > 1000.
3. u = 2*u puis n += 1 : u vaut toujours 2^n.

PIÈGES

- Écrire while u > 1000 : la boucle ne démarre pas, le script affiche 0.
- Incrémenter n avant de mettre u à jour change le lien entre n et u.

NOTIONS MOBILISÉES

- **Négation** — La condition du while est le contraire de ce qu'on cherche.
- **Invariant** — À chaque instant, u = 2^n.

EXERCICE 3.5

A * B ou np.dot(A, B) ?

Le piège numéro un des matrices en Python.

ÉNONCÉ

On pose `A` et `B` ci-dessous.

Question 1. Prédis les deux affichages.

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
B = np.array([[0, 1], [1, 0]])
print(A * B)
print(np.dot(A, B))
```

Question 2. Que vaut `np.dot(B, A)` ? Qu'en conclut-on ?

CORRIGÉ

Question 1

`A * B` multiplie case par case : `[[0 2] [3 0]]`. `np.dot(A, B)` est le produit matriciel : `[[2 1] [4 3]]` (B échange les colonnes de A).

Question 2

```
import numpy as np

A = np.array([[1, 2], [3, 4]])
B = np.array([[0, 1], [1, 0]])
print(np.dot(B, A))
```

`[[3 4] [1 2]]` : B à gauche échange les lignes de A. Le produit matriciel n'est pas commutatif.

EN BREF

** agit coefficient par coefficient ; le produit matriciel s'écrit np.dot.*

ÉTAPES

1. A * B : (A*B)[i, j] = A[i, j] × B[i, j].
2. np.dot(A, B) : lignes de A contre colonnes de B.
3. np.dot(A, B) et np.dot(B, A) diffèrent en général.

PIÈGES

- Écrire A * B pour un produit matriciel : le résultat a la bonne taille, l'erreur passe inaperçue.
- L'opérateur @ existe mais n'est pas au programme : écris np.dot.

NOTIONS MOBILISÉES

- **np.dot** — Le produit matriciel.
- **Non-commutativité** — AB ≠ BA en général.

EXERCICE 4.1

Simuler un événement de probabilité p

rd.random() < p : la brique de toutes les simulations.

ÉNONCÉ

On fixe $p = 0{,}3$.

Question 1. Écris un script qui simule 10 000 fois l'événement et affiche la fréquence des succès.

Question 2. Que simule `rd.random() < 0.5` ? Et `rd.random() > 0.9` ?

CORRIGÉ

Question 1

```
import numpy.random as rd

p = 0.3
N = 10000
S = 0
for _ in range(N):
    if rd.random() < p:
        S += 1
print(S/N)
```

Affiche une valeur proche de 0,3 : la fréquence approche la probabilité.

Question 2

Un Pile ou Face équilibré ; un événement de probabilité 0,1 (la longueur de $]0{,}9\,;1[$).

EN BREF

rd.random() tombe dans [0, 1[de façon uniforme : la probabilité d'être sous p vaut exactement p.

ÉTAPES

1. S = 0, compteur de succès.
2. N fois : si rd.random() < p, S += 1.
3. La fréquence S/N estime p.

PIÈGES

- Tester rd.random() == p.
- Tirer rd.random() une fois pour toutes avant la boucle : il faut un tirage par essai.

NOTIONS MOBILISÉES

- **Fréquence** — Nombre de succès divisé par le nombre d'essais.
- **Loi des grands nombres** — La fréquence tend vers la probabilité.

EXERCICE 4.68

Que estime ce script ?

Reconnaître une fréquence, puis calculer la probabilité qu'elle estime.

D'après EDHEC 2023 · maths appliquées · question b.

ÉNONCÉ

Le script suivant demande un entier m à l'utilisateur.

Qu'affiche ce script quand m est grand ? Calcule la valeur théorique.

```
m = int(input('m = '))
C = 0
for _ in range(m):
    X = rd.geometric(1/2)
    Y = rd.geometric(1/2)
    if X == Y:
        C += 1
print(C/m)
```

CORRIGÉ

`C/m` est la fréquence de l'événement $(X = Y)$ sur m essais : il estime $P(X = Y)$.

$P(X = Y) = \sum_{k\geq 1} P(X = k)^2 = \sum_{k\geq 1} \left(\frac{1}{2^k}\right)^2 = \sum_{k\geq 1} \frac{1}{4^k} = \frac{1}{3}$.

> Ici `input` renvoie 5 dans le vérificateur : la fréquence affichée est grossière. Avec $m = 10\,000$, on trouve environ 0,333.

EN BREF

Un compteur divisé par le nombre d'essais est une fréquence : elle estime la probabilité de l'événement testé dans le if.

ÉTAPES

1. Repérer l'événement testé : X == Y.
2. C compte ses réalisations, C/m est sa fréquence.
3. Calculer P(X = Y) avec la formule des probabilités totales.

PIÈGES

- Répondre « une valeur entre 0 et 1 » sans dire de quelle probabilité.
- Oublier le carré : P(X = k et Y = k) = P(X = k)².

NOTIONS MOBILISÉES

- **Fréquence** — Par la loi des grands nombres, elle tend vers la probabilité.
- **Série géométrique** — La somme des 1/4^k pour k ≥ 1 vaut 1/3.

EXERCICE 5.1

Filtrer, trier, compter

Les trois requêtes de base sur une seule table.

ÉNONCÉ

La table `evenement(id, nom, jour, lieu, places)` liste des événements avec leur date, leur ville et leur nombre de places.

Question 1. Écris la requête qui donne le nom et la date des événements de Lyon.

Question 2. Écris la requête qui donne les trois événements qui offrent le plus de places, du plus grand au plus petit.

Question 3. Écris la requête qui donne le nombre d'événements par ville.

CORRIGÉ

Question 1

```sql
SELECT nom, jour FROM evenement WHERE lieu = 'Lyon'
```

Question 2

```sql
SELECT nom, places FROM evenement ORDER BY places DESC LIMIT 3
```

`DESC` pour l'ordre décroissant, `LIMIT 3` pour ne garder que les trois premières lignes.

Question 3

```sql
SELECT lieu, COUNT(*) FROM evenement GROUP BY lieu
```

Une seule requête : `GROUP BY lieu` fait un paquet par ville, `COUNT(*)` compte dans chaque paquet.

EN BREF

SELECT choisit les colonnes, WHERE filtre les lignes, ORDER BY trie, GROUP BY regroupe avant un agrégat.

ÉTAPES

1. Les textes s'écrivent entre apostrophes : 'Lyon'.
2. ORDER BY … DESC puis LIMIT n.
3. Toute colonne non agrégée du SELECT figure dans le GROUP BY.

PIÈGES

- Écrire WHERE lieu == 'Lyon' : un seul = en SQL.
- Mettre COUNT(*) dans un WHERE.

NOTIONS MOBILISÉES

- **Agrégat** — COUNT, SUM, AVG, MIN, MAX : un résultat par table ou par groupe.

EXERCICE A.1

Traduire floor(6*rand()) + 1

Le dé des sujets Scilab, et sa version numpy.

D'après un classique des annales ECE/ECS (époque Scilab), traduit en Python.

ÉNONCÉ

Les anciens sujets simulaient un dé avec `floor(6*rand()) + 1`. On lit la traduction Python ci-dessous.

Question 1. Quelles valeurs `d` peut-il prendre, avec quelles probabilités ?

```
import numpy as np
import numpy.random as rd

d = int(np.floor(6*rd.random())) + 1
```

Question 2. Écris une fonction `uniforme(a, b)` qui renvoie un entier uniforme de a à b inclus, à partir de `rd.random()` seulement.

CORRIGÉ

Question 1

`6*rd.random()` est uniforme sur $[0, 6[$; sa partie entière vaut 0, 1, ..., 5 avec probabilité $1/6$ chacune ; on ajoute 1 : un dé équilibré. Aujourd'hui on écrit `rd.randint(1, 7)`.

Question 2

```
import numpy as np
import numpy.random as rd

def uniforme(a, b):
    return a + int(np.floor((b - a + 1)*rd.random()))

print(uniforme(1, 6), uniforme(10, 20))
```

Il y a b − a + 1 valeurs possibles : on découpe [0, 1[en autant de cases de même longueur.

EN BREF

Multiplier rd.random() étire l'intervalle ; la partie entière le découpe en cases équiprobables.

ÉTAPES

1. 6*rd.random() ∈ [0, 6[.
2. np.floor donne 0 à 5, équiprobables.
3. +1 ramène sur 1 à 6.

PIÈGES

- Oublier le +1 : on obtient 0 à 5.
- Écrire np.floor(6*rd.random() + 1) : même résultat ici, mais la lecture est moins claire.

NOTIONS MOBILISÉES

- **Partie entière** — np.floor, ou int pour un réel positif.

FIN DE L'EXTRAIT GRATUIT

Tu viens de lire 25 pages. Le livre en fait 408.

Les commandes, tu les as. Ce qui fait les points au concours, c'est de les avoir utilisées des dizaines de fois, un stylo à la main.

DANS LE LIVRE COMPLET	EXERCICES CORRIGÉS
Chapitre 1 · Les bases	**61**
Chapitre 2 · Analyse	**61**
Chapitre 3 · Algèbre	**59**
Chapitre 4 · Probabilités	**76**
Chapitre 5 · SQL	**6**
Les classiques d'avant 2023, traduits en Python	**10**
Le cours de chaque chapitre, 60 QCM, les annales 2023-2026 décortiquées, le mémo	**273**

Le livre complet est ici :
preparationnelle.com/tableau