

Fiche ECE

méthode · if · for · tranches · hasard · affichage formaté

LE JOUR DE L'ÉPREUVE

Le programme est **fourni**, presque entièrement écrit et commenté : on ne vous demande jamais d'écrire un programme entier. Les fonctions utiles sont rappelées dans le sujet.

Vous saisissez vos mesures, complétez parfois une formule, une conversion, un titre ou un axe ; puis vous **exécutez et exploitez** : lire une valeur, comparer à un modèle, écrire un résultat avec son incertitude, conclure.

Méthode pour la partie Python d'un sujet

1. **Lire tout le programme** d'abord : repérer les lignes à compléter (....., « À COMPLÉTER ») et les unités en commentaire.
2. **Saisir les mesures** : point décimal, virgule entre les valeurs, autant de valeurs dans chaque liste.
3. **Convertir** si le sujet le demande : $\mu\text{m} \rightarrow \text{m}$ * 1e-6, $\text{cm} \rightarrow \text{m}$ * 1e-2, $\text{mmol/L} \rightarrow \text{mol/L}$ * 1e-3.
4. **Exécuter tout de suite** ; en cas d'erreur, lire la ligne indiquée, puis le type d'erreur.
5. **Critiquer le résultat** : ordre de grandeur, unité. Python n'avertit jamais d'un résultat absurde.
6. **Exploiter par écrit** : incertitude à 1 chiffre significatif, arrondie à l'excès ; valeur au même rang ; conclusion justifiée.

DÉCIDER

comparer	< > <= >= == !=
réponse	True (vrai) ou False (faux)
valeur absolue	abs(x)
si, sinon si, sinon	if · elif · else, puis :
le bloc	décalé de 4 espaces

RÉPÉTER

chaque élément	for x in liste:
0, 1, 2, 3	range(4)
1, 2, 3	range(1, 4)
chaque rang	for i in range(len(c)):
liste vide, ajout	L = [] · L.append(x)

TRANCHES

5 premiers	x[:5] (rangs 0 à 4)
à partir du rang 5	x[5:]
rangs 2 et 3	x[2:4] (fin exclue)
le dernier	x[-1]

HASARD ET MONTE-CARLO

incertitude- type u	np.random.normal(x, u, N)
tolérance ± t	np.random.uniform(x - t, x + t, N)
u du résultat	np.std(resultats, ddof=1)

LIRE UN AFFICHAGE FORMATÉ

f"λ = {x} m"	{...} remplacé par la valeur
{x:.2f} ·	2 chiffres après la virgule · à l'unité
{x:.0f}	
{x:.1e}	écriture scientifique : 2.9e-09

CONCLURE AVEC UN Z-SCORE

formule	$z = x_{\text{exp}} - x_{\text{réf}} / u(x)$ · deux mesures : $z = x_1 - x_2 / \sqrt{u_1^2 + u_2^2}$
$z < 1$	valeurs très compatibles : l'écart est plus petit que l'incertitude
$1 \leq z < 2$	valeurs compatibles : l'écart est du même ordre que l'incertitude
$z \geq 2$	valeurs non compatibles : chercher une cause (mesure, incertitude, modèle)

EXEMPLE COMMENTÉ 1 · TYPE A, Z-SCORE ET CONCLUSION (D'APRÈS LES SUJETS 78 ET 89)

```
import numpy as np

# 1. les mesures : listes de même longueur, point décimal, unités en commentaire
D = 1.50 # en m
a = np.array([40, 50, 70, 100, 120, 280, 400]) * 1e-6 # μm -> m
L = np.array([4.9, 3.9, 2.8, 2.0, 1.6, 0.70, 0.49]) * 1e-2 # cm -> m
lam_ref = 650e-9 # valeur de référence

# 2. un λ par fente, d'un coup : L = 2λD/a donne λ = L·a/(2D) ; parenthèses obligatoires
lam = L * a / (2 * D)

# 3. incertitude de type A : écart-type expérimental (ddof=1), divisé par racine de n
n = len(lam)
lam_moy = np.mean(lam)
u_lam = np.std(lam, ddof=1) / np.sqrt(n)
print('λ =', round(lam_moy * 1e9, 1), 'nm   u =', round(u_lam * 1e9, 1), 'nm')

# 4. z-score, puis conclusion : un seul bloc est exécuté, le premier dont la condition est vraie
z = abs(lam_moy - lam_ref) / u_lam
print('z =', round(z, 2))
if z < 1: # deux-points, puis bloc décalé de 4 espaces
    print('valeurs très compatibles')
elif z < 2: # testé seulement si z >= 1
    print('valeurs compatibles')
else:
    print('valeurs non compatibles')
```

```
λ = 652.9 nm   u = 2.9 nm
z = 0.97
valeurs très compatibles
```

Résultat écrit : $\lambda = (653 \pm 3) \text{ nm}$. L'incertitude 2,9 nm est arrondie à l'excès, à un chiffre : 3 nm ; la valeur, au même rang.

EXEMPLE COMMENTÉ 2 · SIMULATION DE MONTE-CARLO (D'APRÈS LES SUJETS 1, 22 ET 78)

```
import numpy as np

# fentes de Young : λ = i × b / D, chaque grandeur avec son incertitude-type
N = 100000 # nombre de tirages
i_sim = np.random.normal(2.20e-3, 0.05e-3, N) # valeur, incertitude-type, N
b_sim = np.random.normal(0.50e-3, 0.01e-3, N)
D_sim = np.random.normal(1.00, 0.01, N)
# une tolérance « ± t » de fabricant se simule par np.random.uniform(x - t, x + t, N)

lam_sim = i_sim * b_sim / D_sim # N valeurs possibles de λ, calculées d'un coup
print(f"λ = {np.mean(lam_sim):.2e} m") # .2e : 2 chiffres après la virgule
print(f"u(λ) = {np.std(lam_sim, ddof=1):.1e} m") # 1'écart-type des résultats = u(λ)
```

```
λ = 1.10e-06 m
u(λ) = 3.5e-08 m
```

Les tirages changent à chaque exécution, mais le résultat reste stable à ce nombre de chiffres. La formule de composition des incertitudes relatives donne la même valeur : $\lambda = (1,10 \pm 0,04) \mu\text{m}$.

LES FORMULES DES SIX MINI-PROJETS

signal sinusoïdal	$A * \text{np.sin}(2 * \text{np.pi} * f * t + \text{phi})$; en phase (0) : amplitude 2A ; en opposition (π) : somme nulle
diffraction	$L = 2\lambda D / a$: L proportionnelle à $1/a$, de coefficient $k = 2\lambda D$
distribution AH / A ⁻	%AH = $100 / (1 + 10^{\text{pH} - \text{pK}_A})$; %A ⁻ = $100 - \%AH$; croisement à pH = pK _A
taux d'avancement	$x^2 + K_A x - K_A C = 0$; $\Delta = K_A^2 + 4K_A C$; $x = (-K_A + \sqrt{\Delta}) / 2$; $\tau = x / C$
effet Doppler	$v = c \times \Delta f / f_E$; $u(v) = c \times u(\Delta f) / f_E$
étalonnage	$c = \sigma / k$; $c_{\text{mère}} = F \times c_{\text{filles}}$; $t = c \times M$ (c en mol/L)

```
import numpy as np

# 20 mesures du pH de l'eau du robinet ; la liste continue sur deux lignes
mesures = np.array([7.62, 7.58, 7.65, 7.60, 7.71, 7.55, 7.63, 7.59, 7.66, 7.61,
                    7.57, 7.64, 7.68, 7.60, 7.62, 7.56, 7.67, 7.63, 7.59, 7.61])

incertitudes = [] # liste vide, remplie au fil de la boucle
for n in [5, 10, 20]: # n vaut 5, puis 10, puis 20
    serie = mesures[:n] # tranche : les n premières mesures
    u = np.std(serie, ddof=1) / np.sqrt(n)
    incertitudes.append(u) # ajout à la fin de la liste
    print(n, 'mesures : u =', round(u, 3))
print(incertitudes[-1] < incertitudes[0]) # hors de la boucle : une comparaison
```

```
5 mesures : u = 0.023
10 mesures : u = 0.014
20 mesures : u = 0.009
True
```

Quatre fois plus de mesures divisent l'incertitude par 2 environ : c'est le $\sqrt{n}$ de la formule. Résultat : pH = 7,62 ± 0,01.

LES PIÈGES DE L'ECE

0,124	0.124	Le séparateur décimal est le point ; la virgule sépare les valeurs d'une liste.
µm → m	* 1e-6	10e-6 vaut 10 ⁻⁵ , dix fois trop ; ^ n'est pas une puissance.
5 mesures	5 valeurs	Deux listes de longueurs différentes : <code>ValueError</code> au tracé.
λ = L·a/2D	/ (2 * D)	Sans parenthèses, <code>L * a / 2 * D</code> multiplie par <i>D</i> .
if, for	: + 4 espaces	Deux-points oubliés : <code>SyntaxError</code> ; bloc non décalé : <code>IndentationError</code> .
comparer	==	= range une valeur, == compare.
écart-type	ddof=1	Sans lui, numpy divise par <i>n</i> au lieu de <i>n</i> – 1.
mmol/L	* 1e-3	<i>t</i> = <i>c</i> × <i>M</i> demande <i>c</i> en mol/L : sinon, mille fois trop.
un graphique	l'exploiter	Tracé mais ni lu ni commenté, il ne rapporte rien.

CE QUE JE SAIS FAIRE

- ☐ Conclure avec `if`, `elif`, `else` et un z-score.
- ☐ Répéter un calcul avec `for`, `range` et `append`.
- ☐ Extraire une partie d'une série avec une tranche.
- ☐ Évaluer une incertitude par Monte-Carlo.
- ☐ Superposer deux signaux synchrones déphasés.
- ☐ Tracer un diagramme de distribution ; calculer τ .
- ☐ Tester un modèle et en tirer une grandeur.
- ☐ Mener une exploitation complète, jusqu'à la conclusion.

Point à revoir :