

La boîte à outils de l'ECE

calculs · variables · numpy · listes · pyplot · erreurs

CALCULER

opérations	<code>+ - * /</code> , puissance <code>**</code>
décimal	<code>0.68</code> (point, jamais virgule)
$3,00 \times 10^8$	<code>3.00e8</code>
$1,0 \times 10^{-12}$	<code>1.0e-12</code>
ordre des calculs	<code>(2 + 3) * 4</code> → 20

RANGER, AFFICHER

ranger une valeur	<code>c = 340</code>
modifier	<code>f = f * 2</code>
commentaire	<code># en m/s</code>
afficher	<code>print('c =', c, 'm/s')</code>
arrondir	<code>round(x, 2)</code>
type	<code>int</code> · <code>float</code> · <code>str</code>

NUMPY

charger	<code>import numpy as np</code>
log décimal	<code>np.log10(x)</code>
racine, π	<code>np.sqrt(x)</code> · <code>np.pi</code>
10 puissance x	<code>10 ** (-pH)</code>

LISTES ET TABLEAUX

liste	<code>pH = [2.4, 2.5, 2.9]</code>
nombre d'éléments	<code>len(pH)</code>
1 ^{er} , 3 ^e élément	<code>pH[0]</code> · <code>pH[2]</code>
calculer sur tout	<code>pH = np.array(pH)</code>
100 valeurs de 0 à 11	<code>np.linspace(0, 11, 100)</code>

STATISTIQUES

moyenne	<code>np.mean(x)</code>
écart-type	<code>np.std(x, ddof=1)</code>
incertitude-type	<code>s / np.sqrt(len(x))</code>

TRACER AVEC PYPLOT

charger	<code>import matplotlib.pyplot as plt</code>
points	<code>plt.plot(x, y, 'b+')</code>
courbe	<code>plt.plot(x, y, 'r-')</code>
nommer la série	<code>label='mesures' + plt.legend()</code>
légender	<code>plt.xlabel</code> · <code>plt.ylabel</code> · <code>plt.title</code>
histogramme	<code>plt.hist(x, bins=5)</code>
afficher	<code>plt.grid()</code> · <code>plt.show()</code>

EXEMPLE COMMENTÉ 1 · UNE COURBE D'ÉTALONNAGE, COMME À L'ECE

```
import numpy as np                                # 1. charger les bibliothèques, toujours en premier
import matplotlib.pyplot as plt

# 2. saisir les mesures : listes entre crochets, POINT décimal, même longueur
c = np.array([1.0, 2.5, 5.0, 8.0, 10.0])          # en mmol/L
sigma = np.array([0.162, 0.330, 0.605, 0.925, 1.158]) # en mS/cm

# 3. le modèle : on calcule sur 100 valeurs d'un coup (tableau numpy)
k = 0.1172
c_mod = np.linspace(0, 11, 100)                  # 100 concentrations de 0 à 11
sigma_mod = k * c_mod                             # loi de Kohlrausch, sur les 100 valeurs

# 4. exploiter : une concentration inconnue, affichée avec son unité
c_serum = 0.869 / k
print('c du sérum =', round(c_serum, 2), 'mmol/L')

# 5. tracer : mesures en points, modèle en ligne, axes TOUJOURS légendés
plt.plot(c, sigma, 'b+', label='mesures')
plt.plot(c_mod, sigma_mod, 'r-', label='modèle')    # c_mod avec sigma_mod : même longueur
plt.xlabel('c (mmol/L)')
plt.ylabel('σ (mS/cm)')
plt.legend()

plt.show()                                         # 6. afficher, en dernier
```

c du sérum = 7.41 mmol/L

EXEMPLE COMMENTÉ 2 · UNE SÉRIE DE MESURES ET SON INCERTITUDE

```
import numpy as np
import matplotlib.pyplot as plt

# 15 pesées du même cylindre d'aluminium, en g ; la liste continue sur deux lignes
m = np.array([55.47, 55.43, 55.45, 55.44, 55.48, 55.42, 55.46, 55.45,
              55.43, 55.47, 55.44, 55.46, 55.42, 55.45, 55.46])

n = len(m)                                # nombre de mesures : 15
m_moy = np.mean(m)                        # moyenne : meilleure estimation de la masse
s = np.std(m, ddof=1)                     # écart-type EXPÉRIMENTAL (divisé par n - 1) : ddof=1 obligatoire
u = s / np.sqrt(n)                        # incertitude-type de la moyenne (type A)

print('n =', n)
print('moyenne =', round(m_moy, 3), 'g')
print('u =', round(u, 4), 'g')            # on écrira u à 1 chiffre significatif, arrondi à l'excès

plt.hist(m, bins=5, edgecolor='black')    # répartition des mesures en 5 classes
plt.xlabel('masse (g)')
plt.ylabel('effectif')
plt.show()
```

```
n = 15
moyenne = 55.449 g
u = 0.0048 g
```

Résultat : $m = (55,449 \pm 0,005)$ g. L'histogramme montre les mesures regroupées autour de la moyenne.

EXEMPLE COMMENTÉ 3 · DES VARIABLES, NUMPY ET UN AFFICHAGE SOIGNÉ

```
import numpy as np

# effet Doppler : émetteur de 40 kHz qui s'approche à 2,0 m/s
f_E = 40000                                # Hz ; pas d'espace dans un nombre
v = 2.0                                    # m/s
c = 340                                    # m/s
lam = c / f_E                              # « lambda » est un mot réservé : on écrit lam

f_R = f_E * c / (c - v)                    # expression exacte ; les parenthèses fixent l'ordre
delta_f = f_R - f_E
delta_f_approx = f_E * v / c               # expression approchée (v faible devant c)

print('lambda =', lam, 'm')                 # dans un texte, « lambda » est permis
print('Δf exact =', round(delta_f, 1), 'Hz')
print('Δf approché =', round(delta_f_approx, 1), 'Hz')
print('L =', 10 * np.log10(1.0e-5 / 1.0e-12), 'dB') # log10, pas log
```

```
lambda = 0.0085 m
Δf exact = 236.7 Hz
Δf approché = 235.3 Hz
L = 70.0 dB
```

LES PIÈGES

0,124	<code>0.124</code>	Avec une virgule, <code>0,124</code> devient une paire de deux nombres : 0 et 124.
10^{-12}	<code>1e-12</code>	<code>10e-12</code> vaut 10×10^{-12} , dix fois trop. <code>^</code> n'est pas une puissance : <code>10^-2</code> vaut -12, sans message d'erreur.
$\log(x)$	<code>np.log10(x)</code>	<code>np.log(x)</code> calcule $\ln(x)$, le logarithme népérien.
2D, 2θ	<code>2 * D</code>	Le signe de la multiplication est obligatoire.
λ	<code>lam</code>	<code>lambda</code> , <code>if</code> , <code>in</code> , <code>def</code> ... sont des mots réservés : interdits comme noms.
un texte	<code>'Hz'</code>	Entre guillemets, sinon Python cherche une variable de ce nom.
c et C	deux variables	Python distingue majuscules et minuscules.
1 ^{re} valeur	<code>x[0]</code>	Python compte à partir de 0 ; <code>x[5]</code> n'existe pas dans une liste de 5 valeurs.
écart-type	<code>ddof=1</code>	Sans lui, numpy divise par n au lieu de $n - 1$.
nm, μ m, m	tout en m	Des unités mélangées donnent un résultat faux, sans message d'erreur.
' ' (iPad)	<code>'</code>	Le clavier remplace parfois ' par ' : Python ne comprend que le guillemet droit.

LIRE UN MESSAGE D'ERREUR

Regarder la **dernière ligne** (le type d'erreur) et le **numéro de ligne**. Si la ligne paraît juste, remonter d'une ligne.

<code>SyntaxError</code>	la ligne n'est pas du Python : parenthèse ou guillemet non fermé, virgule décimale, mot réservé
<code>NameError</code>	nom inconnu : majuscule, faute de frappe, variable pas encore définie, texte sans guillemets
<code>TypeError</code>	opération impossible : texte + nombre, calcul sur une liste au lieu d'un <code>np.array</code>
<code>IndexError</code>	élément qui n'existe pas : <code>x[5]</code> dans une liste de 5 valeurs
<code>ValueError</code>	dans <code>plt.plot</code> : <code>x and y must have same first dimension</code> , listes de longueurs différentes
aucun message	résultat faux : vérifier l'ordre de grandeur, les unités, <code>log</code> ou <code>log10</code> , <code>^</code> ou <code>**</code>

CE QUE JE SAIS FAIRE

- ☐ Calculer dans la console, avec le point décimal et la notation `e`.
- ☐ Ranger des valeurs dans des variables, sans mot réservé.
- ☐ Distinguer nombre et texte ; afficher valeur arrondie et unité.
- ☐ Utiliser `np.log10` et `np.sqrt`, sans confondre avec `np.log`.
- ☐ Saisir des mesures dans une liste et calculer sur tout le tableau.
- ☐ Calculer moyenne, écart-type et incertitude-type.
- ☐ Tracer points, modèle et histogramme, axes légendés.
- ☐ Lire un message d'erreur ; douter d'un résultat aberrant.

Point à revoir :