Dernière mise à jour: November 1, 2026

🗺️ Parcours sur quatre ans · Année 1, Printemps

À propos de ce cours

Ce cours gratuit compte 9 unités et 4 travaux pratiques. Ce sont les vrais totaux, pas un programme plus long raccourci.

Un contrôle est un nombre ou un résultat de code sur une page de cours qu'un script recalcule avant publication. Le nombre affiché et le nombre recalculé doivent correspondre.

> Ceci est le plan d'un cours gratuit en autonomie rédigé par Saving Planets. Ce n'est pas un cours universitaire : il ne donne aucun crédit académique et aucun établissement n'y participe.
>
> Son contenu suit les cours de programmation de première année comme MIT 6.100L (anciennement 6.0001), CS50 de Harvard, CS106A de Stanford et CS 1371 de Georgia Tech pour ingénieurs. Nous renvoyons partout à des ressources gratuites. Aucune de ces organisations ne cautionne cette page.
>
> Ce que nous ajoutons : chaque exemple est du vrai Python que notre script de construction exécute avant la mise en ligne de la page, et sa sortie affichée doit correspondre exactement à ce que vous voyez ici. Les données proviennent de vraies sources publiques : la série de CO_2 de Mauna Loa de la NOAA, l'analyse de température GISTEMP de la NASA et les stations hydrométriques de l'USGS.
>
> Vous n'avez rien à installer : utilisez Google Colab ou JupyterLite, qui exécute Python dans votre navigateur grâce à Pyodide. Votre progression n'est enregistrée que dans ce navigateur.

Comment ce cours aide à sauver la planète

Les sciences du climat et de l'environnement reposent sur le code. La courbe du CO_2, les relevés de température mondiale et les prévisions de débit des rivières sont tous produits par des programmes qui lisent des données publiques, et vous pouvez lire les mêmes fichiers.

Chaque unité se termine par une courte section sourcée montrant comment ses idées aident les gens et la planète.

Pour les enseignants

Vous pouvez enseigner ce cours, ou laisser un élève le faire seul. Il est gratuit. Saving Planets ne donne ni note ni crédit scolaire.

- Cette page est le cours entier : 9 unités et 4 travaux pratiques (ENGR 130).
- Prévoyez 15 semaines, environ 6 à 9 heures par semaine, y compris les semaines de travaux pratiques.
- Suivez le calendrier de 15 semaines. Lisez l'unité, faites les exercices sur papier, puis ouvrez la réponse. La semaine d'un TP, faites le TP ou la version à la maison.
- Un contrôle est un nombre ou un résultat de code sur la page qu'un script recalcule avant publication. Le nombre affiché et le nombre recalculé doivent correspondre.
- Utilisez Imprimer cette page, ou téléchargez le guide d'étude PDF. La feuille de formules est vers la fin.
- Demandez : en quoi cette unité aide-t-elle les gens et la planète ? Chaque unité a un encadré vert avec un fait sourcé.
- Copiez le modèle de guide enseignant et écrivez le nom de votre classe, le temps dont vous disposez et les questions que vous poserez. Ouvrir le modèle de guide enseignant

Avant de commencer

Prérequis

- Algèbre du lycée.
- Aucune expérience de programmation requise.
- Physique I et Analyse I aident pour l'unité de simulation, mais ne sont pas obligatoires.

Comment étudier

- Tapez vous-même chaque exemple au lieu de le coller ; vos doigts apprennent la syntaxe.
- Prédisez la sortie avant d'exécuter le code.
- Quand vous êtes bloqué, lisez le message d'erreur de bas en haut.
- Changez une seule chose à la fois et relancez.
- Faites chaque exercice avant d'ouvrir la réponse.
- Refaites les quiz quelques jours plus tard. La pratique espacée vaut mieux qu'une longue séance.

Ce qu'il vous faut

- Un ordinateur ou une tablette avec un navigateur web récent.
- Un compte Google gratuit pour Colab, ou JupyterLite sans aucun compte.
- Facultatif : Python 3 installé localement avec NumPy et Matplotlib.

Programme suggéré sur 15 semaines

Environ 6 à 9 heures par semaine : lecture, exercices et, les semaines de TP, le TP.

Semaine	Unité	Thèmes	Lecture
1	Premiers pas : variables, types et expressions	Notebooks, variables, types, expressions	Tutoriel Python §3
2	Structures de contrôle : conditions et boucles	Conditions et boucles	Tutoriel Python §4
3	Fonctions	Fonctions	MIT 6.100L
4	Listes, dictionnaires et chaînes	Listes, dictionnaires, chaînes	Tutoriel Python §5
5	Fichiers et données CSV	Fichiers et CSV	Documentation du module csv
6	Fichiers et données CSV	Données réelles avec erreurs ; TP A	NOAA GML
7	NumPy et Matplotlib	Tableaux NumPy ; exercices de mi-parcours	Bases de NumPy
8	NumPy et Matplotlib	Matplotlib ; TP B	Matplotlib pyplot
9	Débogage et tests	Débogage et tests ; TP C	Documentation de pytest
10	Algorithmes et complexité	Recherche et complexité	MIT 6.100L
11	Algorithmes et complexité	Dichotomie et méthodes numériques	MIT 6.100L
12	Simulation numérique	Simulation de projectile	Physique I
13	Simulation numérique	Orbites et populations	Analyse II
14	Simulation numérique	Traînée et convergence ; TP D	Introduction à l’aérospatiale
15	Révision	Révision et série d’exercices finale	Toutes les unités

Votre progression

Cochez les unités et les TP à mesure que vous les terminez, et faites les quiz. La progression et les scores sont enregistrés uniquement dans ce navigateur et ne nous sont jamais envoyés.

Unités

Premiers pas : variables, types et expressions

Exécutez votre premier code Python dans un notebook de navigateur, stockez des valeurs dans des variables et calculez avec des nombres et du texte.

Objectifs d'apprentissage

À la fin de cette unité, vous saurez :

- Ouvrir un notebook Colab ou JupyterLite et exécuter des cellules.
- Utiliser des variables et les types int, float, str et bool.
- Utiliser les opérateurs arithmétiques, dont //, % et **.
- Mettre en forme la sortie avec des f-strings.

Notions clés

- Un programme est une suite d'instructions. Un notebook permet de les exécuter une cellule à la fois et de voir les résultats immédiatement.
- Une variable est un nom lié à une valeur : mass_kg = 1100. Choisissez des noms qui disent quoi et dans quelles unités.
- Chaque valeur a un type. int (nombres entiers), float (décimaux), str (texte) et bool (True/False) sont les types de base ; type() vous dit lequel.
- / donne toujours un float ; // est la division entière ; % est le reste ; ** est une puissance.
- Les floats sont des approximations binaires : 0.1 + 0.2 n'est pas exactement 0.3, donc comparez les floats avec une tolérance.

Équations clés

$$W = mg$$

`W=mg`

L'exemple 1.1 calcule un poids

$$\text{change} = \frac{x_{new} - x_{old}}{x_{old}}$$

`change=_new-x_oldx_old`

Variation relative (exemple 1.2)

Exemples résolus pas à pas

Exemple 1.1 Calculez le poids d'un avion de 1100 kg et vérifiez les types.

```
mass_kg = 1100
g = 9.81
weight = mass_kg * g
print(weight)
print(type(mass_kg), type(weight))
```

1. mass_kg est un int ; le multiplier par le float g donne un float.

Sortie

```
10791.0
<class 'int'> <class 'float'>
```

Réponse : 10791 N, un float.

Exemple 1.2 De combien le CO_2 de Mauna Loa a-t-il augmenté entre la moyenne annuelle de 1959 et celle de 2025 (NOAA) ?

```
ppm_1959 = 315.98
ppm_2025 = 427.35
rise = ppm_2025 - ppm_1959
print(f"CO2 rose {rise:.2f} ppm, or {rise / ppm_1959:.1%}")
```

1. Soustrayez, puis divisez par la valeur de départ.
2. La f-string formate :.2f (2 décimales) et :.1% (pourcentage).

Sortie

```
CO2 rose 111.37 ppm, or 35.2%
```

Réponse : 111.37 ppm, une hausse de 35.2%.

Exemple 1.3 Explorez les opérateurs et les surprises des floats.

```
print(7 / 2, 7 // 2, 7 % 2, 2 ** 10)
print(0.1 + 0.2 == 0.3, abs((0.1 + 0.2) - 0.3) < 1e-9)
print(int("42") + 1, str(42) + "1")
```

1. 7/2 = 3.5, 7//2 = 3, 7%2 = 1.
2. 0.1 + 0.2 == 0.3 vaut False ; un test avec tolérance vaut True.
3. int("42") convertit du texte en nombre ; str(42) + "1" concatène du texte.

Sortie

```
3.5 3 1 1024
False True
43 421
```

Réponse : Utilisez des tolérances pour les floats et convertissez les types volontairement.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Que donne 17 % 5 ?

   2.

2. De quel type est 3 * 2.0 ?

   float (6.0).

3. Écrivez une f-string qui affiche pi avec 3 décimales.

   f"{math.pi:.3f}" → 3.142 (après import math).

4. Pourquoi mass_kg est-il un meilleur nom que m ?

   Il indique la grandeur et son unité, ce qui évite les erreurs d'unités.

Idées reçues fréquentes

Idée reçue : = signifie « égal ».
En réalité : En Python, = affecte ; == compare.

Idée reçue : 0.1 + 0.2 == 0.3 vaut True.
En réalité : Cela vaut False à cause de l'arrondi binaire.

Idée reçue : "5" + 5 donne 10.
En réalité : C'est une TypeError ; convertissez avec int("5").

Cartes mémoire

Variable	int / float
Nom lié à une valeur.	Nombre entier / nombre décimal.

str

Texte.

bool

True ou False.

f-string

Mise en forme f"{x:.2f}".

//, %

Division entière, reste.

Quiz d’autocontrôle

1. 7 // 2 =

3.5

3

4

1

3. Division entière.

2. type(2.0) est…

int

float

str

bool

float. Il a un point décimal.

3. 2 ** 10 =

20

100

1024

12

1024. Puissance.

4. L’affectation en Python utilise…

==

=

:=only

->

=. = lie un nom.

Refaire ce quiz

Comment cela aide à sauver la planète

Les nombres de l’exemple 1.2 viennent directement du Global Monitoring Laboratory de la NOAA, qui publie la série de CO_2 de Mauna Loa sous forme de fichier CSV gratuit. Quelques lignes de Python en tirent les faits phares de la science du climat.

Sur Saving Planets : Analyse I

Sources : NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · Google Colab (notebooks gratuits) (https://colab.research.google.com/)

Ressources d’apprentissage gratuites

- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Sections 3 et 4.
- Google Colab (notebooks gratuits) (https://colab.research.google.com/): Démarrez un notebook.
- JupyterLite : Python dans votre navigateur (Pyodide) (https://jupyter.org/try-jupyter/lab/): Option sans compte.

☐ **J’ai terminé cette unité**

Structures de contrôle : conditions et boucles

Faire décider les programmes avec if/elif/else et répéter avec les boucles for et while.

Objectifs d'apprentissage

À la fin de cette unité, vous saurez :

- Écrire des if/elif/else avec des opérateurs de comparaison et booléens.
- Parcourir des intervalles et des séquences avec for.
- Utiliser des boucles while avec une condition d'arrêt claire.
- Accumuler des totaux et des comptes.

Notions clés

- L'indentation définit les blocs en Python. Quatre espaces par niveau est la norme.
- Les comparaisons (<, >=, ==, !=) produisent des booléens ; combinez-les avec and, or, not.
- for x in sequence se répète une fois par élément ; range(a, b) donne a, a+1, ..., b−1.
- while se répète jusqu'à ce que sa condition vaille False. Assurez-vous que quelque chose dans la boucle la modifie, sinon elle ne s'arrête jamais.
- Le motif de l'accumulateur : initialisez un total à 0 et ajoutez dans la boucle.

Équations clés

$$\sum_{n=1}^{100} n = \frac{100 \cdot 101}{2} = 5050$$

_n=1^100n=(100·101)/(2)=5050

La somme de Gauss vérifie l'exemple 2.2

$$x_{k+1} = x_k + \Delta$$

x_k+1=x_k+Δ

Une boucle qui fait avancer une valeur

Exemples résolus pas à pas

Exemple 2.1 Classez l'anomalie de température mondiale 2024 de la NASA.

```
anomaly = 1.28   # NASA GISTEMP, 2024, deg C vs 1951-1980
if anomaly >= 1.0:
    print("at least 1.0 C above the 1951-1980 average")
```

```
elif anomaly > 0:
    print("warmer than average")
else:
    print("at or below average")
```

1. 1.28 ≥ 1.0, donc la première branche s'exécute et les autres sont ignorées.

Sortie

```
at least 1.0 C above the 1951-1980 average
```

Réponse : Les branches sont testées dans l'ordre ; seule la première vraie s'exécute.

Exemple 2.2 Additionnez les nombres de 1 à 100 avec une boucle.

```
total = 0
for n in range(1, 101):
    total += n
print(total)
```

1. range(1, 101) s'arrête avant 101.
2. total += n ajoute chaque nombre.

Sortie

```
5050
```

Réponse : 5050, conforme à la formule de Gauss.

Exemple 2.3 Si le CO_2 continuait d'augmenter de 2.5 ppm par an à partir de la moyenne 2025, quand dépasserait-il 450 ppm ?

```
ppm = 427.35   # 2025 annual mean, Mauna Loa
year = 2025
while ppm < 450:
    ppm += 2.5
    year += 1
print(year, round(ppm, 2))
```

1. La boucle ajoute 2.5 ppm et un an jusqu'à ppm ≥ 450.

Sortie

```
2035 452.35
```

Réponse : En 2035, à 452.35 ppm dans ce scénario simple à taux constant (pas une prévision).

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Combien de fois s'exécute for i in range(3, 10, 2) ?

   4 fois (3, 5, 7, 9).

2. Écrivez une boucle qui compte les nombres impairs inférieurs à 20.

   count = sum(1 for n in range(20) if n % 2) : 10.

3. Qu'est-ce qui ne va pas dans : while x < 10: print(x)

   x ne change jamais, donc la boucle est infinie.

Idées reçues fréquentes

Idée reçue : range(1, 10) inclut 10.
En réalité : Il s'arrête à 9.

Idée reçue : else après if est obligatoire.
En réalité : Il est facultatif.

Idée reçue : L'indentation n'est qu'une question de style.
En réalité : En Python, elle définit la structure du programme.

Cartes mémoire

if / elif / else

Choisir une branche.

for

Répéter pour chaque élément.

while

Répéter tant que c'est vrai.

range(a, b, s)

De a jusqu'à b−1, avec un pas s.

Accumulateur

break / continue

total += x dans une boucle.

Sortir de la boucle / passer au tour suivant.

Quiz d’autocontrôle

1. list(range(2, 6)) vaut…

[2, 3, 4, 5, 6]

[2, 3, 4, 5]

[3, 4, 5, 6]

[2, 6]

[2, 3, 4, 5]. S’arrête avant 6.

2. Quel mot-clé ajoute un autre test après if ?

else if

elif

then

case

elif. elif.

3. Une boucle while s’arrête quand…

sa condition vaut False

après 10 tours

jamais

à un print

sa condition vaut False. La condition est vérifiée à chaque tour.

4. not (True and False) vaut…

True

False

None

une erreur

True. True and False vaut False.

Refaire ce quiz

Comment cela aide à sauver la planète

Des seuils comme 1.0 °C ou 450 ppm sont exactement ce que les programmes de surveillance testent avec des instructions if, signalant automatiquement records et alertes à l'arrivée de nouvelles données.

Sur Saving Planets : Physique I

Sources : NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/) · NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

Ressources d'apprentissage gratuites

- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Section 4 : structures de contrôle.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Cours magistraux sur les branchements et l'itération.

☐ **J'ai terminé cette unité**

Fonctions

Regrouper la logique dans des fonctions réutilisables et testables, avec paramètres, valeurs par défaut et valeurs de retour.

Objectifs d’apprentissage

À la fin de cette unité, vous saurez :

- Définir des fonctions avec def, des paramètres et return.
- Utiliser des arguments par défaut et nommés.
- Écrire des docstrings.
- Comprendre la portée locale.

Notions clés

- Une fonction donne un nom à un morceau de logique. Écrivez-la une fois, testez-la une fois, utilisez-la partout.
- Les paramètres reçoivent des valeurs ; return renvoie un résultat. Une fonction sans return renvoie None.
- Les arguments par défaut (cl_max=1.6) raccourcissent les cas courants ; les arguments nommés rendent les appels lisibles.
- Les variables créées dans une fonction sont locales et disparaissent à son retour.
- Une docstring (la chaîne de la première ligne) documente ce que fait la fonction.

Équations clés

$$F = \frac{9}{5}C + 32$$

F=95C+32

Exemple 3.1

$$V_s = \sqrt{\frac{2W}{\rho S C_{L,\max}}}$$

V_s=2Wρ SC_L,

Exemple 3.2, tiré d’Introduction à l’aérospatiale

Exemples résolus pas à pas

Exemple 3.1 Écrivez et testez une fonction de conversion Celsius-Fahrenheit.

```
def c_to_f(c):
    """Convert Celsius to Fahrenheit."""
    return c * 9 / 5 + 32

for c in (-40, 0, 1.28, 100):
    print(c, "->", round(c_to_f(c), 2))
```

1. −40 est identique sur les deux échelles, une vérification pratique.
2. round(..., 2) arrondit proprement les floats.

Sortie

```
-40 -> -40.0
0 -> 32.0
1.28 -> 34.3
100 -> 212.0
```

Réponse : Une anomalie de 1.28 °C correspond à un écart de 2.3 °F, mais 1.28 °C en tant que température vaut 34.3 °F ; les unités comptent.

Exemple 3.2 Transformez la formule de vitesse de décrochage en fonction et utilisez-la à deux masses volumiques.

```
def stall_speed(weight_n, rho, area, cl_max=1.6):
    return (2 * weight_n / (rho * area * cl_max)) ** 0.5

print(round(stall_speed(10791, 1.227, 16.2), 1))
print(round(stall_speed(10791, 0.738, 16.2), 1))
```

1. cl_max a une valeur par défaut, donc l'appelant peut l'omettre.
2. ** 0.5 est une racine carrée.

Sortie

```
26.0
33.6
```

Réponse : 26.0 m/s au niveau de la mer et 33.6 m/s à 5000 m, comme dans le cours d'aérospatiale.

Exemple 3.3 Écrivez une fonction qui renvoie les variations d'une année sur l'autre d'une liste (moyennes annuelles de Mauna Loa 2022–2025).

```
def growth(values):
    """Year-to-year changes of a list."""
    return [b - a for a, b in zip(values, values[1:])]

co2 = [418.53, 421.08, 424.61, 427.35]   # 2022-2025
print([round(d, 2) for d in growth(co2)])
```

1. zip associe chaque valeur à la suivante.
2. Une compréhension de liste construit le résultat.

Sortie

```
[2.55, 3.53, 2.74]
```

Réponse : Hausses de 2.55, 3.53 et 2.74 ppm.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Écrivez area_circle(r) et testez-la à r = 2.

   return math.pi * r**2 → 12.566....

2. Que renvoie une fonction sans return ?

   None.

3. Pourquoi tester c_to_f à −40 ?

   Les deux échelles y concordent, donc la réponse attendue est connue.

Idées reçues fréquentes

Idée reçue : print et return, c'est pareil.
En réalité : print affiche une valeur ; return la renvoie pour un usage ultérieur.

Idée reçue : Modifier un paramètre modifie la variable de l'appelant.
En réalité : Réaffecter un paramètre reste local (les objets mutables sont une exception).

Idée reçue : Les fonctions doivent être longues et tout faire.
En réalité : Les petites fonctions à but unique sont plus faciles à tester.

Cartes mémoire

def

Définit une fonction.

return

Renvoie une valeur.

Paramètre

Nom pour une entrée.

Argument par défaut

Valeur utilisée s’il est omis.

Docstring

Chaîne de documentation.

Portée

Où un nom est visible.

Quiz d’autocontrôle

1. On définit une fonction avec…

func

def

function

lambda seulement

def. def name(params):

2. Une fonction sans instruction return renvoie…

0

None

False

une erreur

None. None implicite.

3. Dans def f(x, k=2), k est…

obligatoire

un argument par défaut

global

une constante

un argument par défaut. A une valeur par défaut.

4. Les variables créées dans une fonction sont...

global

locales

partagées

permanentes

locales. Portée locale.

Refaire ce quiz

Comment cela aide à sauver la planète

Les codes de climatologie et d'hydrologie sont construits à partir de petites fonctions testées comme celles-ci, pour les conversions d'unités, les taux de variation et les formules physiques ; c'est ainsi que les grands modèles restent fiables.

Sur Saving Planets : Introduction à l'ingénierie aérospatiale

Sources : NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html) · Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/)

Ressources d'apprentissage gratuites

- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Section 4.8 : définir des fonctions.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): Fonctions et variables.

☐ **J'ai terminé cette unité**

Listes, dictionnaires et chaînes

Stocker des collections de données : des listes pour les séquences, des dictionnaires pour les recherches et des chaînes pour le texte.

Objectifs d’apprentissage

À la fin de cette unité, vous saurez :

- Indexer, découper et modifier des listes.
- Utiliser len, sum, min, max et sorted.
- Construire et interroger des dictionnaires.
- Découper des chaînes et compter des mots.

Notions clés

- Une liste contient une séquence ordonnée : flows[0] est le premier élément, flows[-1] le dernier, flows[2:5] une tranche.
- Les listes sont mutables : append, insert, remove et sort les modifient sur place.
- Un dictionnaire associe des clés à des valeurs : gistemp[2024] → 1.28. Les recherches sont rapides quelle que soit sa taille.
- Les compréhensions de liste construisent des listes de façon concise : [x * 2 for x in xs if x > 0].
- Les chaînes sont des séquences de caractères avec des méthodes comme split, strip, lower et startswith.

Équations clés

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

x=1n_i=1^n x_i

Moyenne d’une liste (exemple 4.1)

Exemples résolus pas à pas

Exemple 4.1 Résumez dix jours de débit moyen journalier USGS du Potomac à Little Falls (1–10 juin 2025, en pieds cubes par seconde).

```
flows = [44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600]
print(len(flows), max(flows), min(flows))
print(sum(flows) / len(flows))
print(sorted(flows)[:3])
```

1. len, max et min fonctionnent sur n'importe quelle liste.
2. sum/len est la moyenne ; sorted renvoie une nouvelle liste.

Sortie

```
10 44700 12200
22980.0
[12200, 12600, 14100]
```

Réponse : Moyenne 22980 cfs ; étendue 12200–44700 cfs.

Exemple 4.2 Trouvez l'année la plus chaude dans un dictionnaire d'anomalies NASA GISTEMP.

```
gistemp = {2016: 1.01, 2020: 1.01, 2023: 1.17, 2024: 1.28, 2025: 1.18}
warmest = max(gistemp, key=gistemp.get)
print(warmest, gistemp[warmest])
print([y for y, a in gistemp.items() if a > 1.1])
```

1. max avec key=gistemp.get compare les valeurs.
2. Une compréhension filtre les éléments.

Sortie

```
2024 1.28
[2023, 2024, 2025]
```

Réponse : 2024, à 1.28 °C ; 2023–2025 sont toutes au-dessus de 1.1 °C.

Exemple 4.3 Comptez des mots avec un dictionnaire.

```
words = "reduce reuse recycle reduce repair reduce".split()
counts = {}
for w in words:
    counts[w] = counts.get(w, 0) + 1
print(counts)
```

1. get(w, 0) renvoie 0 pour un nouveau mot.

Sortie

```
{'reduce': 3, 'reuse': 1, 'recycle': 1, 'repair': 1}
```

Réponse : « reduce » apparaît 3 fois.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Que vaut [10, 20, 30, 40][1:3] ?

   [20, 30].

2. Comment ajouter 2026: 1.2 à gistemp ?

   gistemp[2026] = 1.2.

3. Élevez au carré chaque nombre de xs avec une compréhension.

   [x**2 for x in xs].

Idées reçues fréquentes

Idée reçue : Les listes commencent à l'indice 1.
En réalité : Elles commencent à 0.

Idée reçue : sorted(xs) modifie xs.
En réalité : sorted renvoie une nouvelle liste ; xs.sort() la modifie.

Idée reçue : Les clés d'un dictionnaire peuvent se répéter.
En réalité : Chaque clé est unique ; une nouvelle affectation remplace la valeur.

Cartes mémoire

Liste

Séquence ordonnée et mutable.

Indice / tranche

xs[i], xs[a:b].

Dictionnaire

Association clé → valeur.

Compréhension

[f(x) for x in xs if cond].

Tuple

str.split

Séquence immuable.

Texte → liste de morceaux.

Quiz d’autocontrôle

1. xs[-1] est…

le premier élément

le dernier élément

une erreur

la longueur

le dernier élément. Les indices négatifs comptent à partir de la fin.

2. len({"a": 1, "b": 2}) =

1

2

4

3

2. Deux clés.

3. "a,b,c".split(",") donne…

"abc"

["a", "b", "c"]

("a","b","c")

3

["a", "b", "c"]. Une liste de chaînes.

4. Les listes sont…

immuables

mutables

des dictionnaires

toujours triées

mutables. Elles peuvent changer sur place.

Refaire ce quiz

Comment cela aide à sauver la planète

L'USGS publie le débit journalier de milliers de stations hydrométriques. Les dix nombres de l'exemple 4.1 sont réels : une crue du Potomac qui décroît sur une semaine, le genre de série utilisée pour l'approvisionnement en eau et la planification des inondations.

Sur Saving Planets : Page Rivières

Sources : USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

Ressources d'apprentissage gratuites

- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Section 5 : structures de données.
- USGS Water Services (API de données) (https://waterservices.usgs.gov/): Comment demander des données à l'USGS.

☐ **J'ai terminé cette unité**

Fichiers et données CSV

Lire et écrire des fichiers, analyser des données CSV et gérer les lignes défectueuses sans planter.

Objectifs d’apprentissage

À la fin de cette unité, vous saurez :

- Ouvrir, lire et écrire des fichiers texte avec with.
- Analyser du CSV avec le module csv.
- Convertir des chaînes en nombres et gérer les erreurs avec try/except.
- Télécharger un vrai jeu de données et l’analyser.

Notions clés

- with open(name) as f: ouvre un fichier et le ferme automatiquement.
- Le CSV (valeurs séparées par des virgules) est le format le plus courant des données publiques. csv.DictReader transforme chaque ligne en dictionnaire dont les clés sont les en-têtes.
- Tout ce qu’on lit dans un fichier est du texte : convertissez avec int() et float().
- Les données réelles contiennent des commentaires, des vides et de mauvaises valeurs. try/except ValueError permet de les ignorer ou de les journaliser.
- Le fichier des moyennes annuelles de Mauna Loa de la NOAA commence par des lignes de commentaire #, puis year,mean,unc.

Équations clés

$$\text{rate} = \frac{x_{last} - x_{first}}{t_{last} - t_{first}}$$

```
rate=_last-x_firstt_last-t_first
```

Taux de variation moyen (exemple 5.1)

$$1\ \text{cfs} = 0.0283168\ \text{m}^3/\text{s}$$

```
1\ cfs=0.0283168\ m^3/s
```

Conversion d’unités (exemple 5.2)

Exemples résolus pas à pas

Exemple 5.1 Analysez les cinq dernières lignes du CSV des moyennes annuelles de Mauna Loa de la NOAA et trouvez le taux moyen de hausse.

```
import csv, io
text = """year,mean,unc
2021,416.41,0.12
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
years = [int(r["year"]) for r in rows]
ppm = [float(r["mean"]) for r in rows]
print(years[0], years[-1], ppm[-1])
print(round((ppm[-1] - ppm[0]) / (years[-1] - years[0]), 2), "ppm per year")
```

1. io.StringIO permet à csv de lire une chaîne comme si c'était un fichier.
2. Convertissez les champs texte en int et float.

Sortie

```
2021 2025 427.35
2.73 ppm per year
```

Réponse : 2.73 ppm par an de 2021 à 2025.

Exemple 5.2 Écrivez un petit CSV de débits USGS, relisez-le et convertissez en m³/s.

```
with open("flows.csv", "w") as f:
    f.write("date,cfs\n2025-06-01,44700\n2025-06-02,40200\n2025-06-03,29100\n")
with open("flows.csv") as f:
    next(f)                      # skip the header
    for line in f:
        date, cfs = line.strip().split(",")
        print(date, round(int(cfs) * 0.0283168, 1), "m3/s")
```

1. next(f) saute la ligne d'en-tête.
2. strip enlève le saut de ligne ; split sépare les champs.

Sortie

```
2025-06-01 1265.8 m3/s
2025-06-02 1138.3 m3/s
2025-06-03 824.0 m3/s
```

Réponse : 44700 cfs ≈ 1265.8 m^3/s.

Exemple 5.3 Lisez des lignes contenant des commentaires et de mauvaises valeurs sans planter.

```
lines = ["# comment", "year,mean", "2024,424.61", "2025,bad", "2026,"]
good = []
for line in lines:
    if line.startswith("#") or line.startswith("year"):
        continue
    year, value = line.split(",")
    try:
        good.append((int(year), float(value)))
    except ValueError:
        print("skipping", repr(line))
print(good)
```

1. continue saute les commentaires et l'en-tête.
2. float("bad") et float("") lèvent ValueError, qui est interceptée.

Sortie

```
skipping '2025,bad'
skipping '2026,'
[(2024, 424.61)]
```

Réponse : Seule la ligne valide est conservée ; les mauvaises sont signalées.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Comment lire le fichier NOAA complet dans Colab ?

   Utilisez urllib.request.urlopen(url) ou téléchargez-le, puis sautez les lignes commençant par #.

2. Que donne csv.DictReader pour chaque ligne ?

   Un dict dont les clés sont les noms de colonnes, avec des valeurs de type chaîne.

3. Pourquoi utiliser with open(...) ?

   Le fichier est fermé automatiquement, même en cas d'erreur.

Idées reçues fréquentes

Idée reçue : Les nombres d'un CSV sont des nombres.

En réalité : Ils sont lus comme des chaînes tant qu'ils ne sont pas convertis.

Idée reçue : Des lignes défectueuses rendent le jeu de données inutilisable.

En réalité : Ignorez-les et journalisez-les, puis indiquez combien il y en a.

Idée reçue : Ouvrir un fichier en mode "w" ajoute à la fin.

En réalité : "w" écrase ; "a" ajoute à la fin.

Cartes mémoire

with open()

Ouvre un fichier et le ferme automatiquement.

CSV

Valeurs séparées par des virgules.

csv.DictReader

Des lignes sous forme de dicts.

try / except

Gérer les erreurs.

ValueError

Conversion impossible.

Mode "w" / "a"

Écraser / ajouter.

Quiz d'autocontrôle

1. float("3.5") donne…

"3.5"

3.5

3

une erreur

3.5. Convertit du texte en float.

2. float("") lève…

TypeError

ValueError

KeyError

rien

ValueError. Une chaîne vide n’est pas un nombre.

3. Le mode de fichier "a"...

écrase

ajoute

lit

supprime

ajoute. Ajout.

4. Les clés de csv.DictReader proviennent...

des numéros de ligne

de la ligne d’en-tête

du nom du fichier

du hasard

de la ligne d’en-tête. Première ligne.

Refaire ce quiz

Comment cela aide à sauver la planète

La NOAA, la NASA et l’USGS partagent leurs mesures sous forme de fichiers CSV gratuits. Savoir les lire vous-même permet de confronter les gros titres aux données sources.

Sur Saving Planets : Analyse II

Sources : NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · Documentation Python : le module csv (https://docs.python.org/3/library/csv.html)

Ressources d’apprentissage gratuites

- Documentation Python : le module csv (https://docs.python.org/3/library/csv.html): Lire et écrire du CSV.
- NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv): Le vrai fichier.
- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Section 7.2 : fichiers.

☐ **J’ai terminé cette unité**

NumPy et Matplotlib

Calculer sur des tableaux entiers d’un coup avec NumPy, ajuster des tendances et faire des graphiques clairs avec Matplotlib.

Objectifs d’apprentissage

À la fin de cette unité, vous saurez :

- Créer des tableaux NumPy et utiliser l’arithmétique vectorisée.
- Utiliser mean, diff, les masques booléens et polyfit.
- Tracer des données avec des axes légendés et des titres.
- Enregistrer des figures dans des fichiers.

Notions clés

- Un tableau NumPy contient des nombres d’un même type et calcule sur tous à la fois : cfs * 0.0283168 convertit chaque valeur.
- np.diff donne les différences successives ; un masque booléen comme cfs > 20000 sélectionne ou compte des valeurs.
- np.polyfit(x, y, 1) ajuste une droite et renvoie la pente et l’ordonnée à l’origine.
- Le module pyplot de Matplotlib dessine des figures : fig, ax = plt.subplots(); ax.plot(x, y); légendez les axes avec les unités ; fig.savefig enregistre l’image.
- Un bon graphique a des axes légendés avec unités, un titre et une source de données citée.

Équations clés

$$y \approx mx + b$$

Droite des moindres carrés avec np.polyfit

$$V = \sum Q_i \, \Delta t$$

```
V= Q_i Δ t
```

Volume à partir des débits journaliers (exemple 6.2)

Exemples résolus pas à pas

Exemple 6.1 Utilisez NumPy sur les moyennes annuelles de Mauna Loa 2021–2025 : différences, moyenne et tendance.

```
import numpy as np
years = np.array([2021, 2022, 2023, 2024, 2025])
ppm = np.array([416.41, 418.53, 421.08, 424.61, 427.35])
print(np.diff(ppm))
print(round(ppm.mean(), 3), round(np.diff(ppm).mean(), 3))
slope, intercept = np.polyfit(years, ppm, 1)
print(round(slope, 3))
```

1. np.diff donne quatre hausses annuelles.
2. La pente de polyfit est le taux le mieux ajusté.

Sortie

```
[2.12 2.55 3.53 2.74]
421.596 2.735
2.796
```

Réponse : Hausse ajustée ≈ 2.80 ppm par an sur 2021–2025.

Exemple 6.2 Convertissez dix jours de débit du Potomac en m³/s, comptez les jours de fort débit et trouvez le volume total.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(np.round(m3s[:3], 1))
print(int((cfs > 20000).sum()), "days above 20,000 cfs")
print(round(m3s.sum() * 86400 / 1e9, 3), "km3 in 10 days")
```

1. Multipliez tout le tableau d'un coup.
2. (cfs > 20000).sum() compte les valeurs True.
3. Somme des m³/s × 86400 s par jour.

Sortie

```
[1265.8 1138.3  824. ]
5 days above 20,000 cfs
0.562 km3 in 10 days
```

Réponse : Cinq jours au-dessus de 20000 cfs ; environ 0.56 km³ d'eau sont passés en dix jours.

Exemple 6.3 Tracez une sélection d'anomalies annuelles NASA GISTEMP et enregistrez la figure.

```
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
years = [1880, 1951, 1980, 2000, 2016, 2020, 2023, 2024, 2025]
anom = [-0.17, -0.07, 0.26, 0.39, 1.01, 1.01, 1.17, 1.28, 1.18]
fig, ax = plt.subplots()
ax.plot(years, anom, "o-")
ax.set_xlabel("Year"); ax.set_ylabel("Anomaly (C)")
ax.set_title("NASA GISTEMP v4, selected years")
fig.savefig("gistemp.png")
print(len(ax.lines[0].get_xdata()), "points saved")
```

1. matplotlib.use("Agg") permet au code d'enregistrer des images sans écran (Colab affiche de toute façon les graphiques en ligne).
2. Légendez les axes et le titre ; enregistrez avec savefig.

Sortie

```
9 points saved
```

Réponse : Une courbe enregistrée de 9 points ; ouvrez gistemp.png pour la voir.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. np.array([1, 2, 3]) * 2 donne ?

   array([2, 4, 6]).

2. Comment sélectionner les valeurs supérieures à 1.0 dans un tableau a ?

   a[a > 1.0].

3. Quel appel ajoute une légende d'axe ?

   ax.set_xlabel("Year").

Idées reçues fréquentes

Idée reçue : Les tableaux NumPy et les listes s'additionnent de la même façon.

En réalité : [1, 2] + [3, 4] concatène ; les tableaux s'additionnent élément par élément.

Idée reçue : Une droite ajustée prouve une cause.

En réalité : Elle résume une tendance ; les causes demandent une compréhension physique.

Idée reçue : Des graphiques sans unités, ça va.

En réalité : Indiquez toujours les unités et citez la source.

Cartes mémoire

np.array

Tableau numérique rapide.

Vectorisé

Calcul sur des tableaux entiers.

Masque booléen

a[a > x].

np.polyfit

Ajustement polynomial par moindres carrés.

plt.subplots

Crée une figure et des axes.

savefig

Enregistre un graphique.

Quiz d’autocontrôle

1. np.diff([1, 4, 9]) =

[3, 5]

[1, 4, 9]

[0, 3, 5]

14

[3, 5]. Différences successives.

2. np.polyfit(x, y, 1) renvoie…

la moyenne

la pente et l’ordonnée à l’origine

un graphique

le maximum

la pente et l’ordonnée à l’origine. Ajustement de degré 1.

3. (a > 0).sum() compte…

les valeurs positives

toutes les valeurs

les zéros

rien

les valeurs positives. True compte pour 1.

4. fig.savefig("x.png")…

affiche

enregistre la figure

la supprime

montre le code

enregistre la figure. Écrit un fichier image.

Refaire ce quiz

Comment cela aide à sauver la planète

L’analyse GISTEMP de la NASA est publiée avec ses données pour que chacun puisse reproduire la série de température mondiale. La tracer vous-même, avec des unités indiquées et la source, est la manière honnête de partager des faits climatiques.

Sur Saving Planets : Données climatiques sur ce site : CO2M

Sources : NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/) · NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html)

Ressources d’apprentissage gratuites

- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): Les tableaux pour débutants.
- Matplotlib: pyplot tutorial (https://matplotlib.org/stable/tutorials/pyplot.html): Bases du tracé.
- NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv): Les vraies données.

☐ **J’ai terminé cette unité**

Débogage et tests

Trouver les bogues méthodiquement, lire les tracebacks et écrire des tests qui détectent les erreurs avant qu'elles ne comptent.

Objectifs d'apprentissage

À la fin de cette unité, vous saurez :

- Lire un traceback et localiser la ligne en échec.
- Utiliser print et des assertions pour cerner les bogues.
- Lever des erreurs claires pour les entrées invalides.
- Écrire des tests unitaires avec assert et pytest.

Notions clés

- Lisez les messages d'erreur par le bas : la dernière ligne nomme le type d'erreur, les lignes au-dessus montrent où elle s'est produite.
- Déboguez de façon scientifique : formulez une hypothèse, testez-la avec une petite entrée, changez une seule chose.
- Les cas limites (listes vides, zéro, nombres négatifs, données manquantes) causent la plupart des bogues.
- Un assert énonce ce qui doit être vrai ; un assert qui échoue arrête le programme à l'endroit du bogue.
- Les tests sont de petites fonctions qui vérifient des réponses connues ; pytest exécute automatiquement toute fonction nommée test_....

Équations clés

$$|x_{computed} - x_{expected}| < \varepsilon$$

Test avec tolérance pour les floats

Exemples résolus pas à pas

Exemple 7.1 Trouvez le bogue d'une fonction de moyenne appelée sur une liste vide.

```
def mean(xs):
    return sum(xs) / len(xs)

try:
    print(mean([]))
```

```
except ZeroDivisionError as e:
    print("bug found:", e)
```

1. Diviser par len([]) = 0 lève ZeroDivisionError.
2. L'intercepter affiche le message.

Sortie

```
bug found: division by zero
```

Réponse : La fonction doit gérer une entrée vide.

Exemple 7.2 Corrigez-la avec une erreur claire et ajoutez des tests par assertion.

```
def mean(xs):
    if not xs:
        raise ValueError("mean() of empty list")
    return sum(xs) / len(xs)

assert mean([1, 2, 3]) == 2
assert abs(mean([0.1, 0.2]) - 0.15) < 1e-12
try:
    mean([])
except ValueError:
    print("all tests passed")
```

1. Levez ValueError avec un message en cas d'entrée vide.
2. Les résultats en float sont comparés avec une tolérance.

Sortie

```
all tests passed
```

Réponse : Tous les tests passent.

Exemple 7.3 Écrivez un test de style pytest pour f_to_c à partir de points connus.

```
def f_to_c(f):
    return (f - 32) * 5 / 9

def test_f_to_c():
    assert f_to_c(32) == 0
```

```
    assert f_to_c(212) == 100
    assert abs(f_to_c(-40) - (-40)) < 1e-12

test_f_to_c()
print("ok")
```

1. Paires connues : 32 °F = 0 °C, 212 °F = 100 °C, −40 = −40.
2. Dans un fichier, pytest découvrirait test_f_to_c automatiquement ; ici, nous l'appelons.

Sortie

```
ok
```

Réponse : La fonction passe les trois vérifications.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Que signifie généralement « IndexError: list index out of range » ?

   Vous avez utilisé un indice ≥ len(list), souvent une erreur de décalage d'un cran.

2. Écrivez un assert vérifiant que x est positif.

   assert x > 0, "x must be positive".

3. Citez trois cas limites pour une fonction qui fait la moyenne de données.

   Liste vide, une seule valeur, et valeurs manquantes ou non numériques.

Idées reçues fréquentes

Idée reçue : S'il s'exécute, il est correct.

En réalité : Un code peut s'exécuter et donner de mauvaises réponses ; testez-le contre des résultats connus.

Idée reçue : Comparez les floats avec ==.

En réalité : Utilisez une tolérance.

Idée reçue : Les tests, c'est seulement pour les gros projets.

En réalité : De petits tests font gagner du temps dès le premier jour.

Cartes mémoire

Traceback

Rapport d’erreur ; à lire par le bas.

Cas limite

Entrée inhabituelle à la frontière.

assert

Arrête si une condition est fausse.

Test unitaire

Vérifie une fonction contre des réponses connues.

pytest

Lanceur de tests.

raise

Signale une erreur délibérément.

Quiz d’autocontrôle

1. La ligne la plus utile d’un traceback est généralement…

la première

la dernière

celle du milieu

aucune

la dernière. Elle nomme l’erreur.

2. pytest exécute les fonctions nommées…

run_*

test_*

check_*

main

test_*. Convention.

3. Un assert qui échoue lève…

ValueError

AssertionError

rien

TypeError

AssertionError. AssertionError.

4. Une bonne première étape de débogage est de…

tout réécrire

reproduire avec une petite entrée

ajouter des print au hasard partout

l'ignorer

reproduire avec une petite entrée. Réduisez le problème.

Refaire ce quiz

Comment cela aide à sauver la planète

Le code scientifique qui éclaire les politiques doit être vérifié. Ce cours applique ce qu'il prêche : chaque exemple de code de cette page est exécuté par notre script de construction, et sa sortie doit correspondre exactement avant la mise en ligne.

Sur Saving Planets : Cours

Sources : Documentation de pytest (https://docs.pytest.org/) · Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/)

Ressources d'apprentissage gratuites

- Documentation de pytest (https://docs.pytest.org/): Écrire des tests.
- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Section 8 : erreurs et exceptions.

Regarder le cours magistral

Lecture 7 testing debugging exceptions and assertions (https://ocw.mit.edu/courses/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/resources/lecture-7-testing-debugging-exceptions-and-assertions/) (en anglais)

MIT 6.0001 Introduction to Computer Science and Programming in Python (Fall 2016), MIT OpenCourseWare. Licence : CC BY-NC-SA 4.0 (https://creativecommons.org/licenses/by-nc-sa/4.0/) · Conditions d'utilisation (https://ocw.mit.edu/pages/privacy-and-terms-of-use/). Lien fourni uniquement pour l'étude gratuite ; Saving Planets n'est ni affilié à cette université ni soutenu par elle.

☐ **J'ai terminé cette unité**

Algorithmes et complexité

Recherche, recherche de racines et croissance du temps d'exécution avec la taille de l'entrée : raisonner en grand O.

Objectifs d'apprentissage

À la fin de cette unité, vous saurez :

- Implémenter la recherche linéaire et la recherche dichotomique.
- Utiliser la dichotomie pour résoudre des équations.
- Estimer la complexité : O(1), O(log n), O(n), O(n²).
- Choisir des structures de données pour la vitesse.

Notions clés

- Un algorithme est une méthode précise, étape par étape. Son coût est la façon dont le nombre d'étapes croît avec la taille n de l'entrée.
- La recherche linéaire vérifie les éléments un par un : O(n). La recherche dichotomique coupe une liste triée en deux à chaque étape : O(log n).
- La dichotomie trouve une racine de f sur [a, b] où f change de signe en coupant l'intervalle en deux ; après n étapes, l'intervalle vaut (b − a)/2ⁿ.
- Des boucles imbriquées sur n éléments sont en O(n²) : correct pour 1000 éléments, sans espoir pour un million.
- Les recherches dans un dictionnaire ou un ensemble sont en O(1) en moyenne, dans une liste en O(n).

Équations clés

$$\text{steps}_{binary} \approx \lceil \log_2 n \rceil$$

Étapes de la recherche dichotomique

$$n = \left\lceil \log_2 \frac{b-a}{\text{tol}} \right\rceil$$

`n=_2-atol`

Itérations de la dichotomie

Exemples résolus pas à pas

Exemple 8.1 Cherchez le dernier élément d'une liste triée d'un million de nombres pairs, des deux façons.

```
def linear_search(xs, target):
    for i, x in enumerate(xs):
        if x == target:
            return i
    return -1

def binary_search(xs, target):
    lo, hi, steps = 0, len(xs) - 1, 0
    while lo <= hi:
        steps += 1
        mid = (lo + hi) // 2
        if xs[mid] == target:
            return mid, steps
        if xs[mid] < target:
            lo = mid + 1
        else:
            hi = mid - 1
    return -1, steps

data = list(range(0, 2_000_000, 2))
print(linear_search(data, 1_999_998))
print(binary_search(data, 1_999_998))
```

1. La recherche linéaire examine 1000000 éléments pour atteindre l'indice 999999.
2. La recherche dichotomique n'a besoin que de 20 étapes.

Sortie

```
999999
(999999, 20)
```

Réponse : 20 étapes contre un million : $\log_2(10^6) \approx 20$.

Exemple 8.2 Trouvez $\sqrt{2}$ par dichotomie sur $x^2 - 2$ à partir de $[0, 2]$ à 10^{-6} près.

```
def bisect_root(f, a, b, tol=1e-6):
    n = 0
    while b - a > tol:
        m = (a + b) / 2
        if f(a) * f(m) <= 0:
            b = m
```

```
        else:
            a = m
        n += 1
    return (a + b) / 2, n

root, n = bisect_root(lambda x: x**2 - 2, 0, 2)
print(round(root, 6), n)
```

1. Chaque étape coupe l'intervalle en deux.
2. $2/2^{21} < 10^{-6}$ après 21 étapes.

Sortie

```
1.414214 21
```

Réponse : 1.414214 après 21 itérations.

Exemple 8.3 Comparez la croissance de $\log_2 n$ et de n^2.

```
import math
for n in (10, 1_000, 1_000_000):
    print(n, math.ceil(math.log2(n)), n * n)
```

1. math.ceil(math.log2(n)) est le nombre d'étapes de la recherche dichotomique.

Sortie

```
10 4 100
1000 10 1000000
1000000 20 1000000000000
```

Réponse : Pour $n = 10^6$, $\log_2 n$ vaut 20 alors que n^2 vaut 10^{12}.

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Combien d'étapes de recherche dichotomique pour 1 milliard d'éléments triés ?

   $\lceil \log_2 10^9 \rceil = 30$.

2. Quelle est la complexité de la somme d'une liste ?

   $O(n)$.

3. Dichotomie sur $[0, 1]$ à 10^{-3} près : combien d'étapes ?

$\lceil \log_2 1000 \rceil = 10$.

Idées reçues fréquentes

Idée reçue : Des ordinateurs plus rapides rendent la complexité sans importance.

En réalité : Une méthode en $O(n^2)$ sur un million d'éléments, c'est 10^{12} étapes ; aucun ordinateur n'y remédie.

Idée reçue : La recherche dichotomique fonctionne sur n'importe quelle liste.

En réalité : Seulement sur des données triées.

Idée reçue : Le grand O donne le temps exact.

En réalité : Il décrit la croissance, en ignorant les constantes.

Cartes mémoire

Algorithme

Méthode précise, étape par étape.

Big-O

Taux de croissance du coût.

Recherche linéaire

$O(n)$.

Recherche dichotomique

$O(\log n)$, données triées.

Dichotomie

Recherche de racine par division en deux.

$O(n^2)$

Boucles imbriquées.

Quiz d'autocontrôle

1. La recherche dichotomique est en…

$O(n)$

$O(\log n)$

$O(n^2)$

$O(1)$

O(log n). Divise par deux à chaque étape.

2. La dichotomie exige…

une dérivée

un changement de signe sur [a, b]

des données triées

des entiers

un changement de signe sur [a, b]. Théorème des valeurs intermédiaires.

3. Une boucle dans une boucle sur n éléments est en…

O(n)

$O(n^2)$

O(log n)

O(1)

$O(n^2)$. n × n.

4. Une recherche moyenne dans un dict est en…

O(n)

O(1)

$O(n^2)$

O(log n)

O(1). Hachage.

Refaire ce quiz

Comment cela aide à sauver la planète

Des algorithmes efficaces permettent de traiter le déluge de données d'observation de la Terre, des satellites aux milliers de stations hydrométriques. Choisir O(n log n) plutôt que $O(n^2)$ peut transformer une semaine de calcul en secondes, et moins de calcul signifie aussi moins d'énergie.

Sur Saving Planets : Analyse I

Sources : USGS Water Services (API de données) (https://waterservices.usgs.gov/) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/)

Ressources d'apprentissage gratuites

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Cours magistraux sur la recherche, la dichotomie et la complexité.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): Entraînement à la résolution de problèmes.

- [ ] **J'ai terminé cette unité**

Simulation numérique

Faire avancer des modèles physiques dans le temps : un projectile, une orbite et une population, le cœur de tout modèle climatique et d'ingénierie.

Objectifs d'apprentissage

À la fin de cette unité, vous saurez :

- Utiliser le pas de temps (Euler et Euler-Cromer) pour simuler un mouvement.
- Vérifier les simulations par rapport à des réponses exactes.
- Simuler une orbite circulaire et une population logistique.
- Explorer l'effet de la taille du pas sur la précision.

Notions clés

- Une simulation met à jour l'état (position, vitesse, population) par petits pas de temps dt à l'aide des taux de variation : x_new = x + v·dt.
- Euler-Cromer met d'abord à jour la vitesse, puis la position avec la nouvelle vitesse ; il garde les orbites stables bien mieux que la méthode d'Euler simple.
- Validez toujours : comparez avec une solution exacte dans un cas où elle existe (sans traînée, orbite circulaire, formule logistique).
- Un dt plus petit est plus précis mais plus lent ; divisez dt par deux et regardez si la réponse change.
- Les modèles météorologiques et climatiques sont d'immenses versions de ces boucles, qui font avancer les équations de la physique sur une grille mondiale.

Équations clés

$$v_{k+1} = v_k + a_k\,\Delta t, \quad x_{k+1} = x_k + v_{k+1}\,\Delta t$$

```
v_k+1=v_k+a_k Δ t, x_k+1=x_k+v_k+1 Δ t
```

Pas d'Euler-Cromer

$$R = \frac{v_0^2 \sin 2\theta}{g}$$

```
R=(v_0^22θ)/(g)
```

Portée d'un projectile sans traînée

$$\vec{a} = -\frac{GM}{r^3}\vec{r}$$

```
a=-(GM)/(r^3) r
```

Gravité pour une orbite

$$P_{k+1} = P_k + h\,r P_k \left(1 - \frac{P_k}{K}\right)$$

P_k+1=P_k+h rP_k(1-_kK)

Modèle logistique, pas d'Euler

Exemples résolus pas à pas

Exemple 9.1 Simulez un projectile à 20 m/s lancé à 45° et comparez avec la portée exacte.

```
import math
v0, angle, g, dt = 20.0, math.radians(45), 9.81, 0.001
x, y = 0.0, 0.0
vx, vy = v0 * math.cos(angle), v0 * math.sin(angle)
while y >= 0:
    vy -= g * dt
    x += vx * dt
    y += vy * dt
print(round(x, 2), "m simulated;", round(v0**2 / g, 2), "m exact")
```

1. Faites avancer la vitesse, puis la position, jusqu'à l'atterrissage.
2. Portée exacte = v_0^2 sin 90°/g.

Sortie

```
40.77 m simulated; 40.77 m exact
```

Réponse : Les deux donnent 40.77 m à deux décimales avec dt = 0.001 s.

Exemple 9.2 Simulez une orbite à 400 km au-dessus de la Terre avec des pas de 1 seconde.

```
import math
GM = 3.986e14
r0 = 6.771e6
x, y = r0, 0.0
vx, vy = 0.0, math.sqrt(GM / r0)
dt, t = 1.0, 0
period = 2 * math.pi * math.sqrt(r0**3 / GM)
while t < period:
    r3 = (x * x + y * y) ** 1.5
    vx -= GM * x / r3 * dt          # Euler-Cromer: update velocity first
```

```
    vy -= GM * y / r3 * dt
    x += vx * dt
    y += vy * dt
    t += dt
print(round(period / 60, 1), "min;", round(math.hypot(x, y) / 1000), "km from Earth's center
```

1. Partez de r_0 avec la vitesse circulaire $\sqrt{(GM/r_0)}$.
2. Mettez à jour la vitesse à partir de la gravité, puis la position.

Sortie

```
92.4 min; 6771 km from Earth's center
```

Réponse : Après une période (92.4 minutes), l'engin est toujours à 6771 km du centre de la Terre : l'orbite reste circulaire.

Exemple 9.3 Faites évoluer une population logistique de poissons ($r = 0.5$, $K = 1000$, $P_0 = 100$) sur quatre ans.

```
r, K, P, h = 0.5, 1000, 100.0, 1.0
for year in range(1, 5):
    P += h * r * P * (1 - P / K)
    print(year, round(P, 1))
```

1. Chaque année ajoute $rP(1 - P/K)$.

Sortie

```
1 145.0
2 207.0
3 289.1
4 391.8
```

Réponse : 145.0, 207.0, 289.1, 391.8, comme dans le TP A d'Analyse II (391.9 là-bas, une différence d'arrondi).

Exercices

Faites chacun d'abord sur papier, puis vérifiez.

1. Que devient l'erreur de l'exemple 9.1 si dt vaut 0.1 s ?

   L'atterrissage est détecté tard, donc l'erreur de portée augmente ; essayez et comparez.

2. Pourquoi la méthode d’Euler simple fait-elle partir une orbite en spirale vers l’extérieur ?

   Elle ajoute un peu d’énergie à chaque pas ; Euler-Cromer la conserve presque.

3. Ajoutez une traînée $-k\,v\,\vec{v}$ à l’exemple 9.1. Que devient le meilleur angle ?

   Il passe sous 45° (TP D).

Idées reçues fréquentes

Idée reçue : Une simulation qui tourne est correcte.

En réalité : Validez sur des cas exacts et vérifiez que diviser dt par deux change à peine la réponse.

Idée reçue : Un dt plus petit est toujours meilleur.

En réalité : Cela coûte du temps et peut ajouter des erreurs d’arrondi ; choisissez dt par des tests de convergence.

Idée reçue : Les modèles prédisent exactement l’avenir.

En réalité : Ce sont des approximations dont les hypothèses doivent être énoncées.

Cartes mémoire

Pas de temps dt

Taille de chaque mise à jour.

Méthode d’Euler

x += v·dt.

Euler-Cromer

La vitesse d’abord, puis la position.

Validation

Vérification contre des réponses connues.

Convergence

La réponse cesse de changer quand dt diminue.

État

Variables qui décrivent le système à l’instant présent.

Quiz d’autocontrôle

1. Euler-Cromer met à jour…

la position d’abord

la vitesse d’abord, puis la position avec la nouvelle vitesse

seulement la vitesse

rien

la vitesse d’abord, puis la position avec la nouvelle vitesse. Stable pour les orbites.

2. Pour valider une simulation…

lui faire confiance

la comparer à une solution exacte

prendre un dt énorme

retirer la physique

la comparer à une solution exacte. Cas connus.

3. Portée exacte à 20 m/s, 45°, g = 9.81 :

20.4 m

40.8 m

81.5 m

9.8 m

40.8 m. 400/9.81.

4. Diviser dt par deux sans voir de changement signifie…

un bogue

le résultat a convergé

le modèle est faux

rien

le résultat a convergé. Test de convergence.

Refaire ce quiz

Comment cela aide à sauver la planète

Les modèles de population comme l'exemple 9.3 font partie des outils avec lesquels les halieutes évaluent les stocks et fixent des limites de capture durables.

La même idée de pas de temps, à une échelle énormément plus grande, est à la base des modèles météorologiques et climatiques.

Sur Saving Planets : Analyse II, TP A · Guide des orbites

Sources : NOAA Fisheries : évaluations des populations (https://www.fisheries.noaa.gov/topic/population-assessments) · MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/)

Ressources d'apprentissage gratuites

- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Cours sur la simulation et la modélisation.
- NumPy: the absolute basics for beginners (https://numpy.org/doc/stable/user/absolute_beginners.html): Vectoriser les simulations.

☐ **J'ai terminé cette unité**

Exercices

Ces exercices sont dans chaque unité. Faites-les sur papier, puis ouvrez la réponse.

1. Premiers pas : variables, types et expressions
   4 exercices
2. Structures de contrôle : conditions et boucles
   3 exercices
3. Fonctions
   3 exercices
4. Listes, dictionnaires et chaînes
   3 exercices
5. Fichiers et données CSV
   3 exercices
6. NumPy et Matplotlib
   3 exercices
7. Débogage et tests
   3 exercices
8. Algorithmes et complexité

3 exercices

9. Simulation numérique
   3 exercices

TP de programmation

Quatre TP de programmation utilisent des données publiques réelles et la simulation : la série de CO_2 de Mauna Loa, la série mondiale des températures de la NASA, une station hydrométrique de l'USGS et un projectile avec traînée.

Le code de départ de chaque TP s'exécute tel quel ; les résultats affichés ont été obtenus en l'exécutant. Téléchargez ensuite le jeu de données complet et approfondissez l'analyse.

Référence de sécurité : Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/)

TP A : la série de CO_2 de Mauna Loa (NOAA)

SEMAINE 6

Objectif : Chargez le CSV des moyennes annuelles de CO_2 de la NOAA, calculez la hausse totale et le taux de croissance récent, puis tracez la courbe.

$$\text{rate} = \text{slope of } \text{np.polyfit}(\text{year}, \text{ppm}, 1)$$

```
rate=slope of np.polyfit(year, ppm, 1)
```

Taux de croissance par ajustement linéaire

Mode opératoire

1. Exécutez le code de départ (quelques lignes réelles du fichier de la NOAA).
2. Téléchargez le CSV complet de la NOAA (lien ci-dessous), ignorez les lignes #, et chargez toutes les années.
3. Tracez les ppm en fonction de l'année, avec des axes étiquetés et la source.
4. Ajustez des droites sur 1960–1969 et 2016–2025. Comment le taux a-t-il évolué ?

```
import csv, io
import numpy as np
text = """year,mean,unc
1959,315.98,0.12
1960,316.91,0.12
2020,414.21,0.12
2021,416.41,0.12
2022,418.53,0.12
2023,421.08,0.12
2024,424.61,0.12
2025,427.35,0.12
"""
rows = list(csv.DictReader(io.StringIO(text)))
yr = np.array([int(r["year"]) for r in rows])
ppm = np.array([float(r["mean"]) for r in rows])
print(round(ppm[-1] - ppm[0], 2), "ppm since 1959")
recent = yr >= 2020
slope = np.polyfit(yr[recent], ppm[recent], 1)[0]
print(round(slope, 2), "ppm per year, 2020-2025")
```

Sortie

```
111.37 ppm since 1959
2.65 ppm per year, 2020-2025
```

Tableau de données

Année	CO_2 moyen annuel (ppm)
1959	315.98
1960	316.91
2020	414.21
2023	421.08
2024	424.61
2025	427.35

Valeurs réelles du fichier des moyennes annuelles de Mauna Loa de la NOAA GML, consulté le 24 oct. 2026.

Exemple de calcul

1. 427.35 − 315.98 = 111.37 ppm depuis 1959.
2. Taux ajusté 2020–2025 ≈ 2.65 ppm par an.

> **Sécurité**
>
> - Aucune, au-delà d'un usage normal de l'ordinateur.

Alternative à la maison ou virtuelle

- Colab ou JupyterLite dans un navigateur.

NOAA GML: Mauna Loa CO_2 annual means (CSV) (https://gml.noaa.gov/webdata/ccgg/trends/co2/co2_annmean_mlo.csv) · Google Colab (notebooks gratuits) (https://colab.research.google.com/)

Lien avec la planète : C'est la série derrière la courbe de Keeling, la plus longue mesure continue de la hausse du CO_2. NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

☐ **J'ai terminé ce TP (réel ou virtuel)**

TP B : la température mondiale avec NASA GISTEMP

SEMAINE 8

Objectif : Analysez les anomalies annuelles de température mondiale de la NASA : moyennes, année la plus chaude et tendance.

$$\Delta T \text{ per decade} = 10 \times \text{slope}$$

```
Δ T\ per decade=10×slope
```

Tendance par décennie

Mode opératoire

1. Exécutez le code de départ avec les anomalies annuelles (J–D) de 2016–2025.
2. Téléchargez le CSV mondial complet de GISTEMP ; ignorez sa première ligne de titre ; gérez les valeurs manquantes ***.
3. Tracez 1880–2025 et calculez la tendance depuis 1975.
4. Comparez un ajustement sur 10 ans à un ajustement sur 50 ans et expliquez pourquoi les tendances courtes sont plus bruitées.

```
import numpy as np
years = np.arange(2016, 2026)
anom = np.array([1.01, 0.91, 0.85, 0.98, 1.01, 0.85, 0.89, 1.17, 1.28, 1.18])
print(round(anom.mean(), 3))
print(int(years[anom.argmax()]), anom.max())
print(round(np.polyfit(years, anom, 1)[0] * 10, 2), "C per decade (10-year fit)")
```

Sortie

```
1.013
2024 1.28
0.32 C per decade (10-year fit)
```

Tableau de données

Année	Anomalie (°C par rapport à 1951–1980)
2016	1.01
2020	1.01
2023	1.17
2024	1.28
2025	1.18

Valeurs réelles de NASA GISTEMP v4 (GLB.Ts+dSST, colonne J–D), consultées le 24 oct. 2026. La NASA met à jour l'analyse, donc des téléchargements ultérieurs peuvent légèrement différer.

Exemple de calcul

1. Moyenne 2016–2025 = 1.013 °C.
2. Un ajustement sur dix ans donne environ 0.32 °C par décennie, mais dix ans, c'est court et bruité ; utilisez des périodes plus longues pour la tendance de long terme.

> **Sécurité**
>
> - Aucune, au-delà d'un usage normal de l'ordinateur.

Alternative à la maison ou virtuelle

- Colab ou JupyterLite.

NASA GISTEMP v4 global means (CSV) (https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv) · NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

Lien avec la planète : Reproduire soi-même la série de la NASA est la meilleure réponse à « comment le sait-on ? » NASA GISS: GISTEMP v4 surface temperature analysis (https://data.giss.nasa.gov/gistemp/)

☐ **J'ai terminé ce TP (réel ou virtuel)**

TP C : une rivière vue par une station de l’USGS

SEMAINE 9

Objectif : Analysez le débit journalier de la station de l’USGS sur le Potomac à Little Falls : unités, statistiques, décrue et volume.

$$V = \sum_i Q_i \times 86{,}400 \text{ s}$$

V=_iQ_i×86,400\ s

Volume à partir des débits moyens journaliers

Mode opératoire

1. Exécutez le code de départ avec dix jours réels (du 1er au 10 juin 2025).
2. Demandez une année complète à USGS Water Services (format=rdb) pour le site 01646500 et le paramètre 00060, puis analysez le fichier séparé par tabulations.
3. Tracez le débit journalier, trouvez le pic et calculez le volume annuel en km³.
4. Écrivez des tests pour votre fonction de conversion d’unités.

```
import numpy as np
cfs = np.array([44700, 40200, 29100, 22700, 17300, 14100, 12600, 12200, 14300, 22600])
m3s = cfs * 0.0283168
print(round(m3s.mean(), 1), round(np.median(m3s), 1))
drops = np.diff(cfs)
print(int(drops.min()), "cfs biggest one-day drop")
print(round(m3s.sum() * 86400 / 1e6, 1), "million m3 in 10 days")
```

Sortie

```
650.7 564.9
-11100 cfs biggest one-day drop
562.2 million m3 in 10 days
```

Tableau de données

Date (2025)	Débit moyen journalier (cfs)
1er juin	44,700
3 juin	29,100
5 juin	17,300
8 juin	12,200

Date (2025)	Débit moyen journalier (cfs)
10 juin	22,600

Valeurs journalières réelles et approuvées du site USGS 01646500, consultées le 24 oct. 2026.

Exemple de calcul

1. Moyenne 650.7 m³/s, médiane 564.9 m³/s (la moyenne est tirée vers le haut par le pic du 1er juin).
2. La plus forte baisse en un jour a été de 11100 cfs ; environ 562 millions de m³ sont passés en dix jours.

> **Sécurité**
>
> - Aucune, au-delà d'un usage normal de l'ordinateur.

Alternative à la maison ou virtuelle

- Colab ou JupyterLite.

USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/) · USGS Water Services (API de données) (https://waterservices.usgs.gov/)

Lien avec la planète : Les stations hydrométriques comme celle-ci servent à l'approvisionnement en eau potable, aux alertes de crue et aux débits écologiques. USGS: Potomac River near Washington, DC (Little Falls), site 01646500 (https://waterdata.usgs.gov/monitoring-location/USGS-01646500/)

☐ **J'ai terminé ce TP (réel ou virtuel)**

TP D : projectile avec traînée

SEMAINE 14

Objectif : Étendez la simulation du projectile avec une traînée quadratique et trouvez le meilleur angle de tir quand la traînée augmente.

$$\vec{a} = -g\hat{y} - k\,|\vec{v}|\,\vec{v}$$

`a=-g y-k | v| v`

Traînée quadratique (k par mètre)

Mode opératoire

1. Exécutez le code de départ, qui teste les angles entiers de 20° à 60°.
2. Vérifiez que k = 0 donne 45° et la portée exacte.
3. Divisez dt par deux et vérifiez que les résultats changent à peine.
4. Tracez la portée en fonction de l'angle pour chaque k.

```
import math
def fly(v0, deg, k=0.0, dt=0.001, g=9.81):
    vx, vy = v0 * math.cos(math.radians(deg)), v0 * math.sin(math.radians(deg))
    x = y = 0.0
    while y >= 0:
        v = math.hypot(vx, vy)
        vx -= k * v * vx * dt
        vy -= (g + k * v * vy) * dt
        x += vx * dt
        y += vy * dt
    return x
for k in (0.0, 0.01, 0.05):
    best = max(range(20, 61), key=lambda a: fly(20, a, k))
    print(k, best, round(fly(20, best, k), 1))
```

Sortie

```
0.0 45 40.8
0.01 43 31.4
0.05 39 17.8
```

Tableau de données

Traînée k (1/m)	Meilleur angle	Portée (m)
0	45°	40.8

Traînée k (1/m)	Meilleur angle	Portée (m)
0.01	43°	31.4
0.05	39°	17.8

Calculé par le code de départ (v_0 = 20 m/s, dt = 0.001 s).

Exemple de calcul

1. Sans traînée, le meilleur angle est 45° et la portée 40.8 m (exactement 40.77 m).
2. La traînée réduit la portée et abaisse le meilleur angle.

> **Sécurité**
>
> - Aucune, au-delà d'un usage normal de l'ordinateur.

Alternative à la maison ou virtuelle

- Colab ou JupyterLite.

Google Colab (notebooks gratuits) (https://colab.research.google.com/) · JupyterLite : Python dans votre navigateur (Pyodide) (https://jupyter.org/try-jupyter/lab/)

Lien avec la planète : La traînée relève de la même physique que celle qui fixe la consommation de carburant d'un avion (Introduction à l'ingénierie aérospatiale).

☐ **J'ai terminé ce TP (réel ou virtuel)**

Séries d'entraînement aux examens

Chronométrez-vous : environ 10 minutes par problème. Comparez ensuite avec les solutions complètes.

Série d'exercices de mi-parcours (unités 1–5)

1. Qu'affiche print(7 // 2, 7 % 2) ?

    1. Division entière et reste.

    Réponse : 3 1

2. Écrivez une boucle qui additionne les entiers de 1 à 100.

    1. total = 0; for n in range(1, 101): total += n.

    Réponse : 5050

3. Écrivez stall_speed(W, rho, S, cl_max=1.6) et évaluez-la à 10791 N, 1.227 kg/m³, 16.2 m².

    1. (2W/(ρSC))**0.5.

    Réponse : 26.0 m/s

4. Étant donné co2 = [418.53, 421.08, 424.61, 427.35], calculez les variations d'une année à l'autre.

    1. [b - a for a, b in zip(co2, co2[1:])].

    Réponse : [2.55, 3.53, 2.74]

5. Pourquoi float("") a-t-il besoin de try/except lors de la lecture de données ?

    1. Il lève ValueError.

    Réponse : Pour ignorer ou journaliser les lignes invalides au lieu de planter.

Série d'entraînement finale (toutes les unités)

1. Convertissez 44700 cfs en m³/s.

    1. × 0.0283168.

    Réponse : ≈ 1,265.8 m³/s

2. Combien d'étapes de recherche dichotomique pour un million d'éléments triés ?

    1. $\lceil \log_2 10^6 \rceil$.

    Réponse : 20

3. Combien d'étapes de bissection ramènent [0, 2] sous 10^{-6} ?

    1. $2/2^n < 10^{-6}$.

    Réponse : 21

4. Portée exacte d'un projectile à 20 m/s lancé à 45° (g = 9.81) ?

    1. v_0^2/g.

    Réponse : 40.77 m

5. Quatre pas d'Euler de P += 0.5P(1 − P/1000) à partir de 100 ?

    1. 145.0, 207.0, 289.1, …

    Réponse : 391.8

6. Écrivez un test pour f_to_c.

1. assert f_to_c(212) == 100, et d’autres.

Réponse : Paires connues : 32 → 0, 212 → 100, −40 → −40.

Fiche de formules

$$7//2 = 3,\ 7\,\%\,2 = 1,\ 2^{10} = 1024$$

$$\bar{x} = \tfrac{1}{n}\sum x_i$$

$$\text{rate} = \Delta y / \Delta x$$

$$y \approx mx + b\ (\text{np.polyfit})$$

$$1\ \text{cfs} = 0.0283168\ \text{m}^3/\text{s}$$

$$\text{binary search} \approx \lceil \log_2 n \rceil$$

$$n_{bisect} = \lceil \log_2((b-a)/\text{tol}) \rceil$$

$$v\ +\!= a\,\Delta t,\ x\ +\!= v\,\Delta t$$

$$R = v_0^2 \sin 2\theta / g$$

$$P\ +\!= hrP(1 - P/K)$$

Constantes

Constante	Valeur
1 pied cube par seconde (cfs)	0.0283168 m³/s
Secondes par jour	86,400
CO_2 de Mauna Loa, moyenne annuelle 1959	315.98 ppm
CO_2 de Mauna Loa, moyenne annuelle 2025	427.35 ppm
Anomalie GISTEMP 2024 (par rapport à 1951–1980)	1.28 °C
GM de la Terre	3.986 × 10¹⁴ m³/s²
g	9.81 m/s²

Source : NOAA GML: Trends in atmospheric CO_2, data (https://gml.noaa.gov/ccgg/trends/data.html)

Outils et références gratuits

Exécutez Python gratuitement dans votre navigateur et consultez la documentation ici.

- Google Colab (notebooks gratuits) (https://colab.research.google.com/): Notebooks gratuits avec NumPy et Matplotlib préinstallés.

- JupyterLite : Python dans votre navigateur (Pyodide) (https://jupyter.org/try-jupyter/lab/): Python dans le navigateur, sans compte, propulsé par Pyodide.
- Le tutoriel officiel de Python (https://docs.python.org/3/tutorial/): Le tutoriel de référence.
- MIT OpenCourseWare 6.100L: Introduction to CS and Programming Using Python (https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/): Le cours d'introduction complet du MIT.
- Harvard CS50's Introduction to Programming with Python (https://cs50.harvard.edu/python/): Le cours Python gratuit de Harvard.

Glossaire

Algorithme
Méthode précise, étape par étape.

Tableau
Conteneur NumPy pour des calculs numériques rapides.

Assertion
Vérification qui arrête le programme si elle est fausse.

Big-O
Comment le coût croît avec la taille de l'entrée.

Dichotomie
Recherche de racine en divisant un intervalle par deux.

CSV
Fichier de valeurs séparées par des virgules.

Dictionnaire
Association clé → valeur.

Cas limite
Entrée inhabituelle à une limite.

Euler-Cromer
Pas de temps stable : la vitesse d'abord.

Exception
Erreur levée pendant l'exécution.

Fonction
Bloc de code nommé et réutilisable.

Liste
Séquence ordonnée et mutable.

Boucle
Exécution répétée.

Notebook
Document composé de cellules de code exécutables.

Pyodide
Python compilé pour s'exécuter dans un navigateur web.

Simulation
Faire avancer un modèle dans le temps.

Traceback
Rapport indiquant où une erreur s'est produite.

Test unitaire
Petite vérification d'une seule fonction.

Variable
Nom lié à une valeur.

f-string
Littéral de chaîne formatée.

Grille d'autoévaluation

Aucune note ni aucun crédit ne sont attribués. Utilisez cette grille pour juger honnêtement où vous en êtes dans chaque unité.

Niveau	À quoi cela ressemble
4 · Maîtrise	Écrit des fonctions claires et testées ; valide les résultats par rapport à des réponses connues ; justifie ses choix.
3 · Compétent	Les programmes fonctionnent, avec des bogues occasionnels trouvés par les tests.
2 · En progression	Sait modifier des exemples mais a du mal à écrire à partir de zéro.
1 · Débutant	Copie du code sans le comprendre ; revoir l'unité.

Pondération suggérée si vous voulez une note

Partie	Poids
Exercices	20%
TP de programmation	35%
Série d'entraînement de mi-parcours	20%
Série d'entraînement finale	25%

Visez le niveau 3 ou mieux dans chaque unité avant de passer au cours suivant.

Cours associés

- Dessin technique / CAO ENGR 120
- Analyse III MATH 210
- Chimie générale I + TP CHEM 110 + CHEM 110L
- Tous les cours