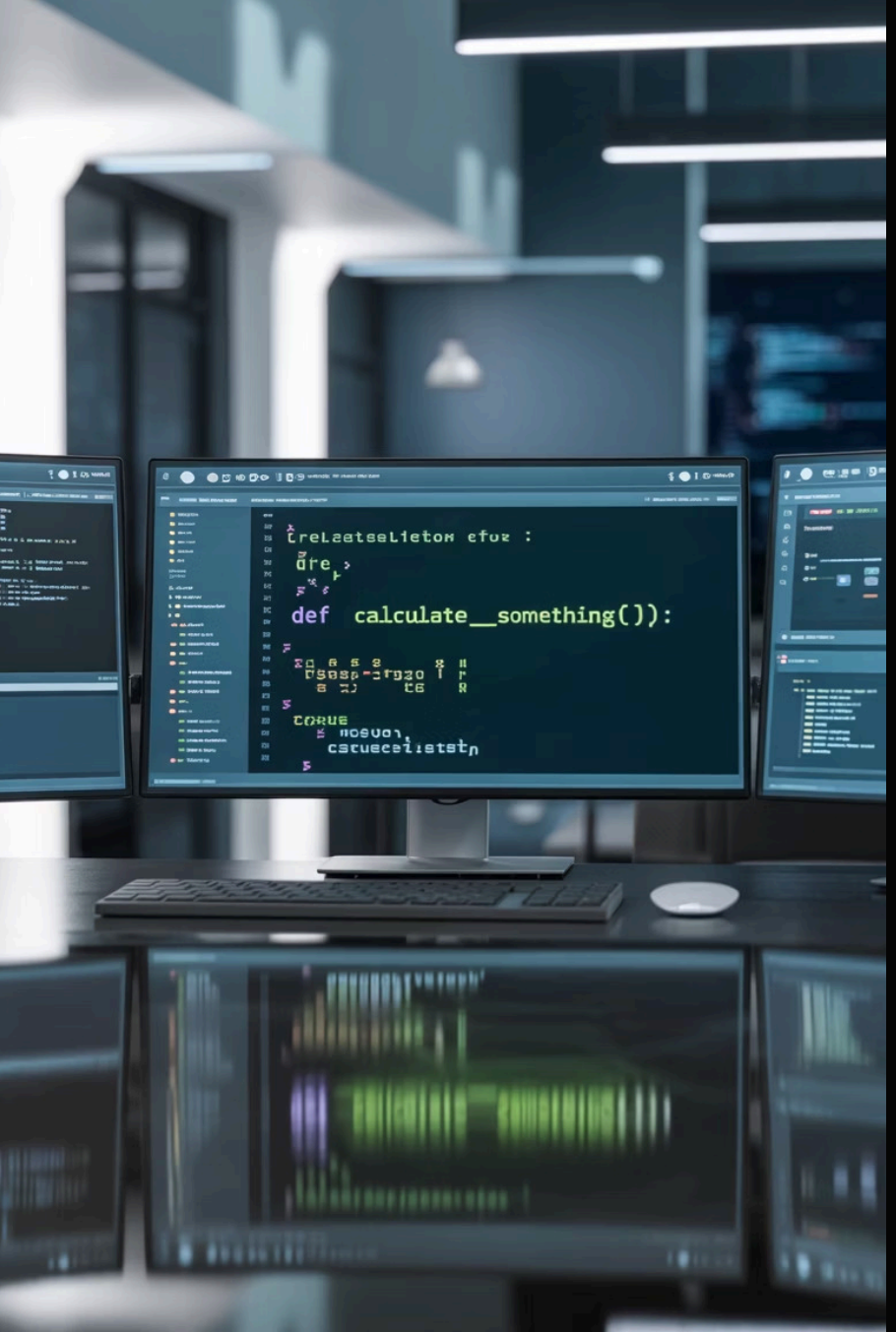


Les Listes en Python

Bienvenue dans ce cours sur les listes Python, un concept fondamental pour la gestion de données en informatique. Les listes sont des structures de données essentielles qui permettent de stocker et organiser plusieurs éléments, particulièrement utiles dans le domaine de la comptabilité et gestion.

Par David DONISA , Enseignant en BTS SIO

Qu'est-ce qu'une liste Python ?

Définition

Une liste est une structure de données ordonnée qui permet de stocker plusieurs éléments de différents types dans une seule variable. En comptabilité, imaginez-la comme un classeur pouvant contenir différents documents : factures, montants, dates, etc.

Syntaxe de base

```
ma_variable = [1, 2, 3, 'Suite de mots', True]
```

Avantages

- Stockage de données hétérogènes
- Ordre préservé des éléments
- Modification possible après création
- Accès direct aux éléments par index

Création et affichage des listes

Création d'une liste comptable

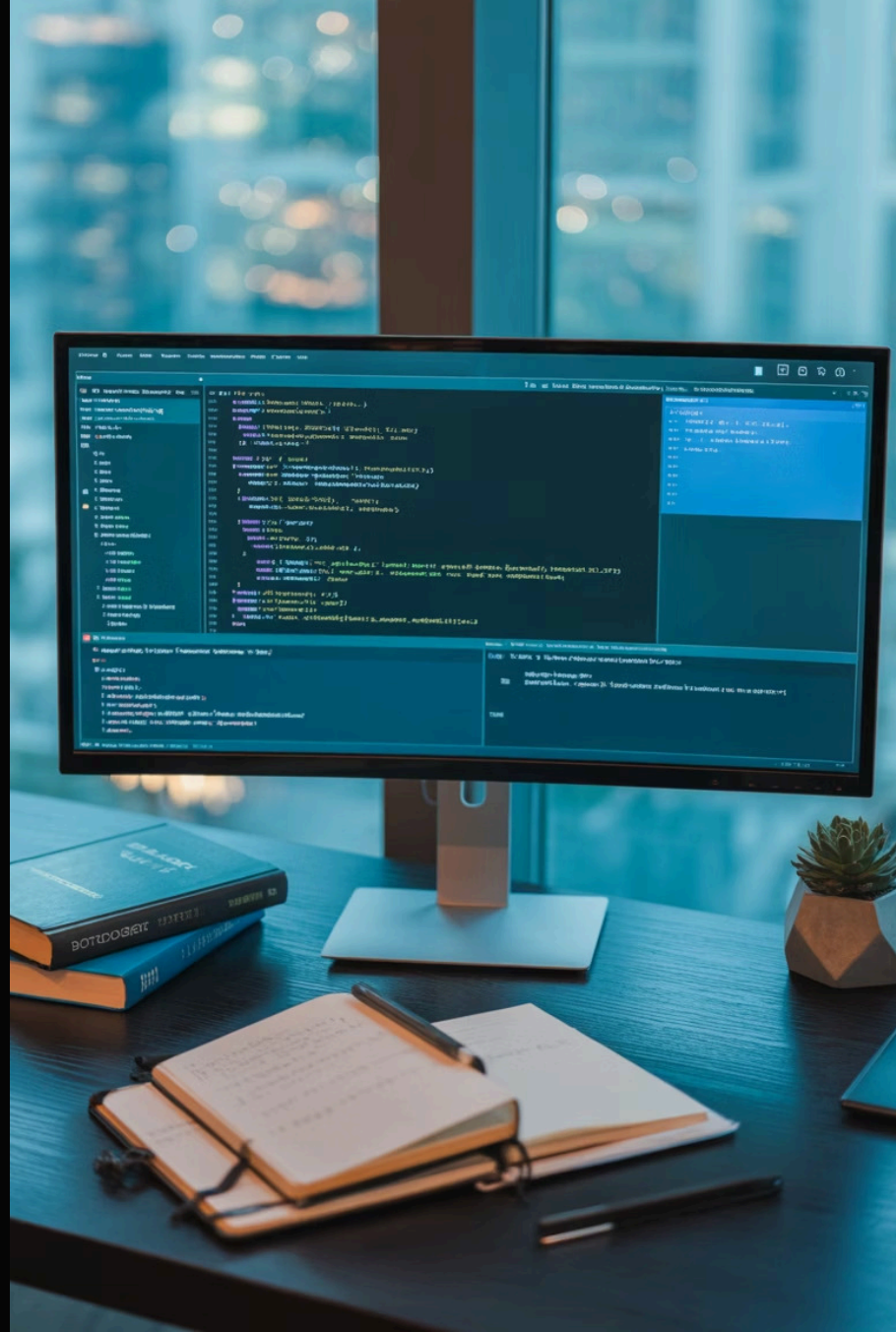
```
factures = [1250.50, 890.00, 1500.75, 'Fournisseur A', True]
print(factures)
```

Cette liste contient des montants de factures, un nom de fournisseur et un statut de paiement.

Liste avec sous-listes

```
comptes = [410, 'Fournisseurs', [1250.50, 890.00]]
print(comptes[2][0])  # Affiche 1250.50
```

Pratique pour organiser les comptes comptables avec leurs détails.



Accès aux éléments par index

01

Index commence à 0

```
prix_articles = [15.99, 25.50, 8.75, 12.00]
print(prix_articles[0])  # Affiche 15.99
```

Le premier élément est toujours à l'index 0

02

Index négatif

```
print(prix_articles[-1])  # Affiche 12.00
```

L'index -1 correspond au dernier élément

03

Accès aux sous-listes

```
commandes = [['Art001', 15.99], ['Art002', 25.50]]
print(commandes[0][1])  # Affiche 15.99
```

Double indexation pour accéder aux éléments imbriqués

Tri et organisation des données

Tri des montants de vente

```
ventes_mensuelles = [1250, 980, 1450, 750, 2100]
print("Original :", ventes_mensuelles)

# Tri croissant
ventes_mensuelles.sort()
print("Croissant :", ventes_mensuelles)

# Tri décroissant
ventes_mensuelles.sort(reverse=True)
print("Décroissant :", ventes_mensuelles)
```

Tri temporaire

```
print("Trié pour affichage :", sorted(ventes_mensuelles))
print("Liste originale :", ventes_mensuelles)
```

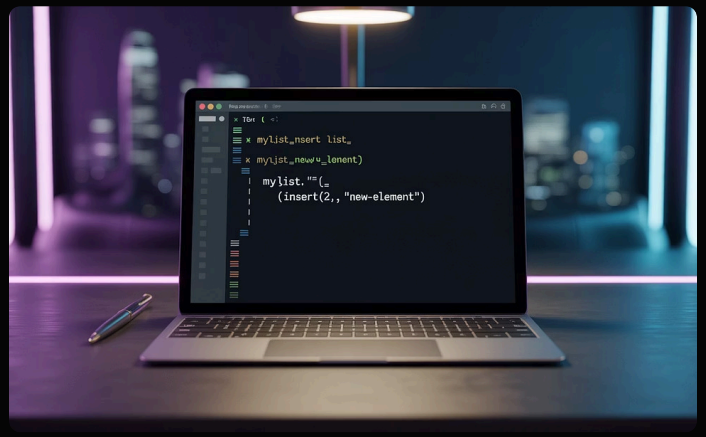
Astuce comptable : Le tri est essentiel pour analyser les performances commerciales et identifier rapidement les meilleures ventes.

Ajout d'éléments aux listes

Ajoute un élément à la fin de la liste

Méthode .append()

```
factures_en_cours = [1200, 850, 950]
factures_en_cours.append(1100)
print(factures_en_cours)  # [1200, 850, 950, 1100]
```



Insère un élément à une position spécifique

Méthode .insert()

```
factures_en_cours.insert(1, 750)
print(factures_en_cours)  # [1200, 750, 850, 950, 1100]
```

Ajoute tous les éléments d'une autre liste

Méthode .extend()

```
factures_en_cours.extend([1300, 1450])
print(factures_en_cours)
```

Suppression d'éléments

Suppression par index - .pop()

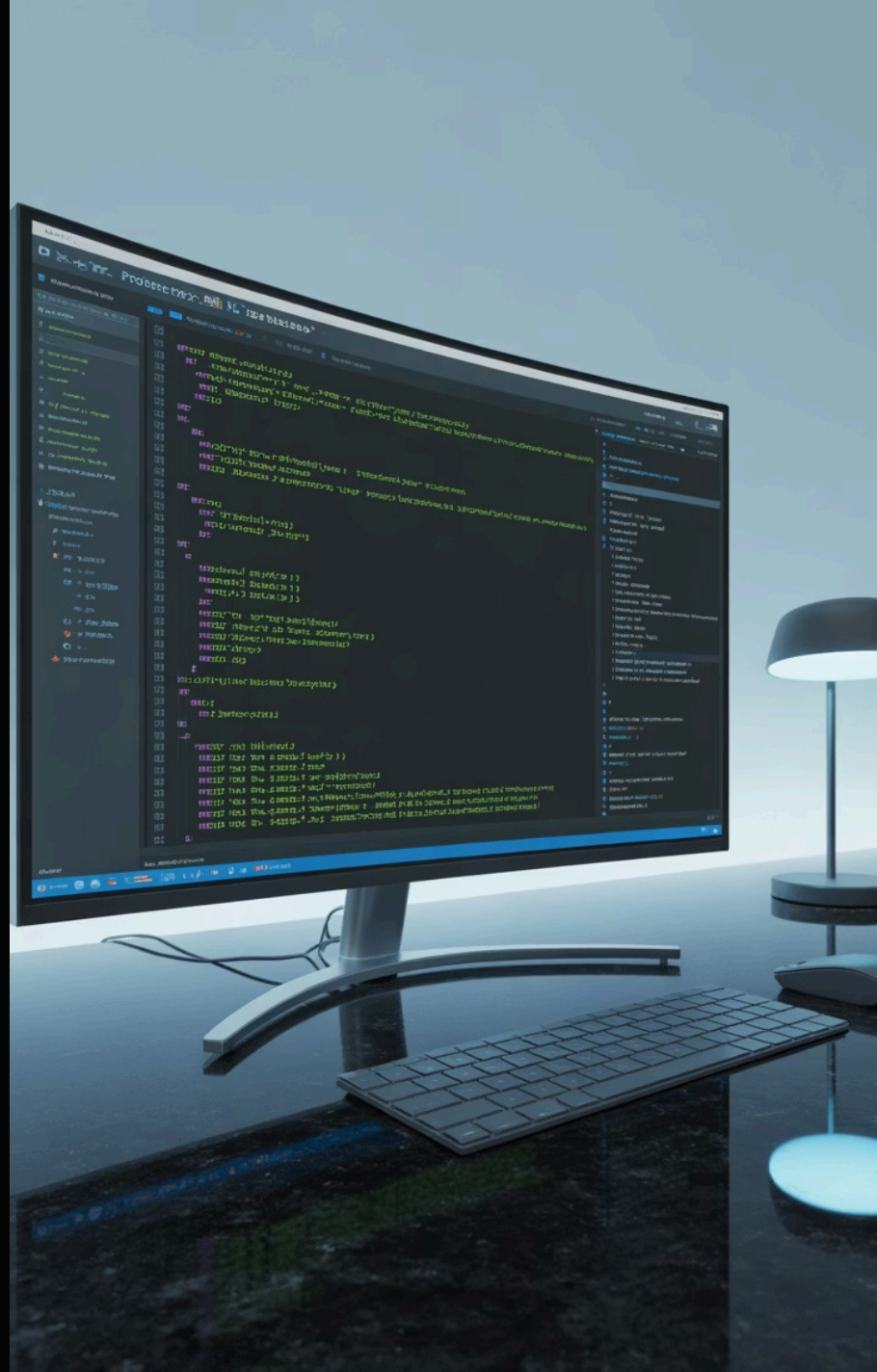
```
depenses = [500, 750, 1200, 300, 900]
depenses.pop(2)  # Supprime l'élément à l'index 2
print(depenses)  # [500, 750, 300, 900]
```

Utile quand vous connaissez la position de l'élément à supprimer

Suppression par valeur - .remove()

```
depenses.remove(750)  # Supprime la première occurrence de 750
print(depenses)  # [500, 300, 900]
```

Pratique pour supprimer une facture annulée par son montant



Parcours des listes avec les boucles

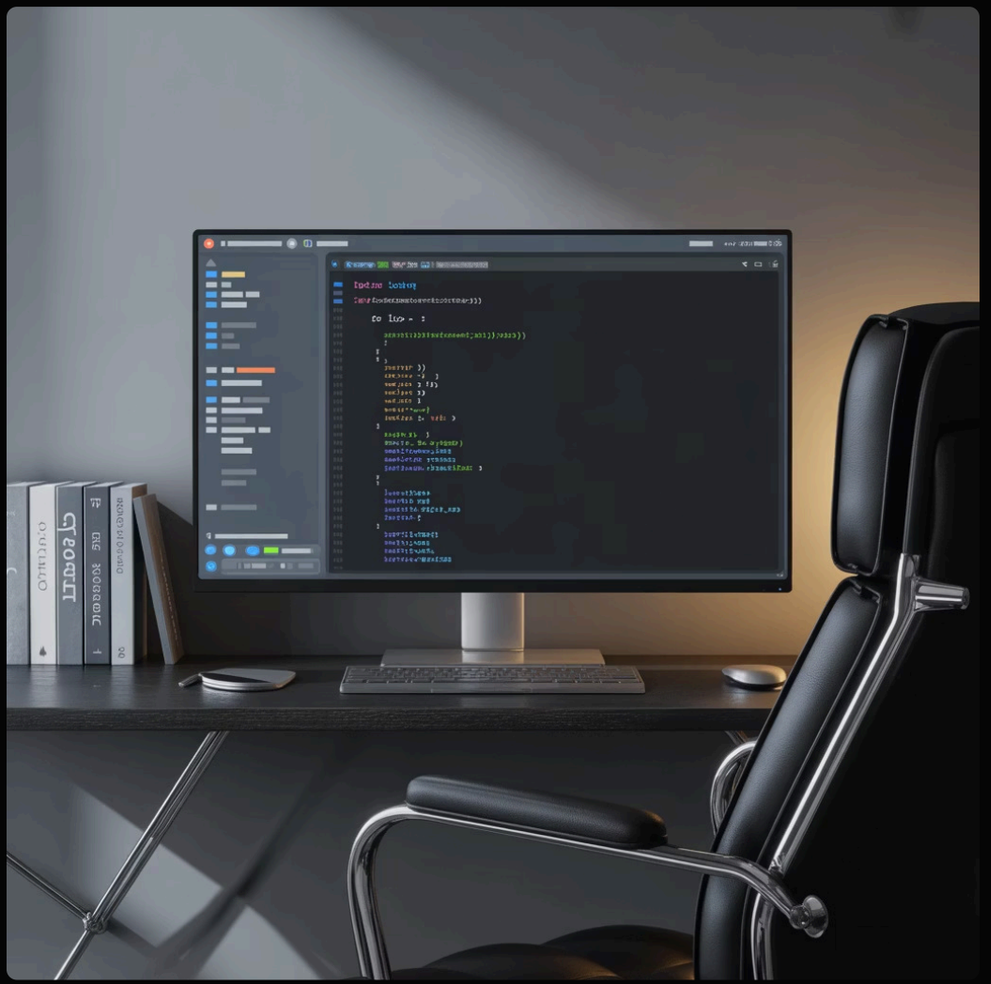
Boucle for simple

```
chiffres_affaires = [15000, 18500, 22000, 17500]

print("Chiffres d'affaires mensuels :")
for ca in chiffres_affaires:
    print(f"- {ca}€")
```

Avec les index

```
for i, montant in enumerate(chiffres_affaires):
    print(f"Mois {i+1} : {montant}€")
```



Le parcours de listes est fondamental en comptabilité pour :

- Calculer des totaux
- Générer des rapports
- Analyser les tendances
- Vérifier les données

List Comprehensions : Création avancée

1 Liste simple

```
numeros_factures = [x for x in range(1001, 1011)]
# [1001, 1002, 1003, ..., 1010]
```

2 Avec transformation

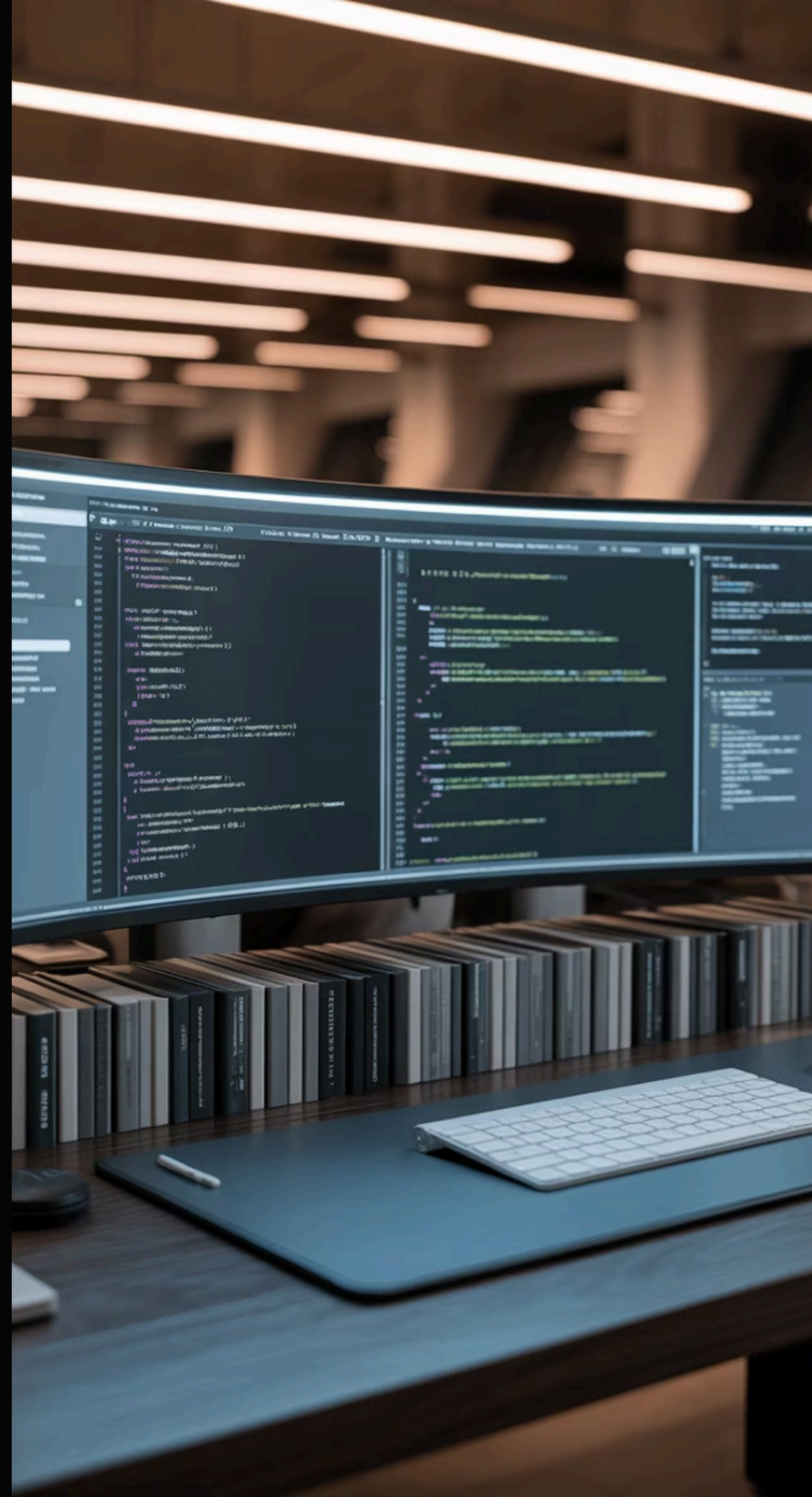
```
prix_ht = [100, 200, 150, 300]
prix_ttc = [prix * 1.20 for prix in prix_ht]
# [120.0, 240.0, 180.0, 360.0]
```

3 Avec condition

```
ventes = [850, 1200, 750, 1500, 900]
grosses_ventes = [v for v in ventes if v > 1000]
# [1200, 1500]
```

4 Avec condition complexe

```
statuts = ['PAYÉ' if v > 1000 else 'EN ATTENTE'
 for v in ventes]
```



Exemple pratique : Gestion comptable complète

```
factures_clients = [
    ['FAC001', 1250.00, 'Client A'],
    ['FAC002', 850.50, 'Client B'],
    ['FAC003', 2100.75, 'Client C']
]

print("=== RAPPORT DE FACTURATION ===")
total = 0

for index, facture in enumerate(factures_clients):
    numero, montant, client = facture
    total += montant
    print(f"{index+1}. {numero} - {client} : {montant}€")

print(f"\nTotal des factures : {total}€")
print(f"Nombre de factures : {len(factures_clients)}")
print(f"Montant moyen : {total/len(factures_clients):.2f}€")
```

3	100%	1
Types de données	Flexibilité	Concept
Combinés dans une structure	Des listes pour tous vos besoins	Pour de nombreuses applications