

Les Structures Conditionnelles en Python

Bienvenue dans ce cours dédié aux **structures conditionnelles** en Python ! Vous allez découvrir comment créer des programmes qui prennent des décisions intelligentes en fonction des données qu'ils reçoivent.

Ce chapitre est essentiel pour tout futur développeur car maîtriser les conditions vous permettra de créer des applications robustes et interactives .

Par David DONISA , Enseignant en BTS SIO





Qu'est-ce qu'une Structure Conditionnelle ?

Une **structure conditionnelle** permet à votre programme de prendre des décisions. Imaginez un garde à l'entrée d'une discothèque : il vérifie l'âge avant de laisser entrer ou non.

En Python, nous utilisons les mots-clés `if`, `elif` et `else` pour créer ces "gardes numériques" qui orientent l'exécution de notre code selon les conditions que nous définissons.

📄 **Analogie :** C'est comme les panneaux de signalisation sur une route - ils indiquent quel chemin prendre selon la situation !

La Structure if Basique

Syntaxe Fondamentale

```
mon_age = int(input("Quel est votre âge ? "))
```

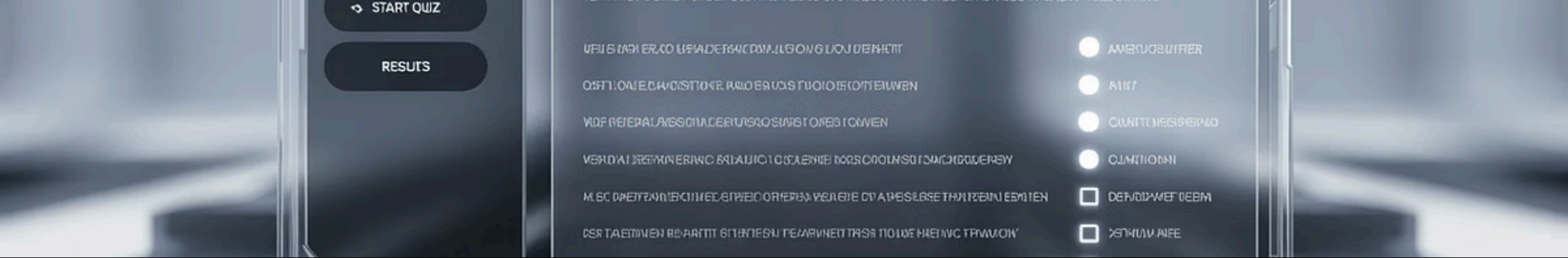
```
if mon_age >= 18:  
    print("Vous êtes majeur !")  
    print("Bienvenue dans l'application")  
else:  
    print("Accès refusé - âge insuffisant")
```

La condition `mon_age >= 18` est évaluée. Si elle est vraie, le premier bloc s'exécute. Sinon, c'est le bloc `else` qui prend le relais.



Points Importants

- Le `:` après la condition est obligatoire
- L'**indentation** définit les blocs de code
- Python utilise 4 espaces par niveau



Conditions Multiples avec elif

Lorsque vous avez plusieurs conditions à tester, `elif` (contraction de "else if") devient votre meilleur ami. Voici un exemple concret d'application de remises commerciales :

```
chiffre_affaires = float(input("Entrez le CA (€) : "))

if chiffre_affaires >= 1000000:
    print("Remise exceptionnelle de 15% !")
elif chiffre_affaires >= 500000:
    print("Remise de 10% accordée")
elif chiffre_affaires >= 100000:
    print("Remise de 5% accordée")
else:
    print("Pas de remise cette fois")
```

Astuce : Python teste les conditions dans l'ordre et s'arrête dès qu'une condition est vraie. L'ordre des tests est donc crucial !

Opérateurs Logiques : AND, OR, NOT

Les opérateurs logiques permettent de combiner plusieurs conditions. Voici un exemple de système d'authentification :

```
identifiant = input("Identifiant : ")
mot_de_passe = input("Mot de passe : ")

if identifiant == "admin" and mot_de_passe == "1234":
    print("Connexion administrateur réussie")
elif identifiant == "user" and mot_de_passe == "6789":
    print("Connexion utilisateur réussie")
else:
    print("Échec de connexion")
```



AND (et)

Toutes les conditions doivent être vraies



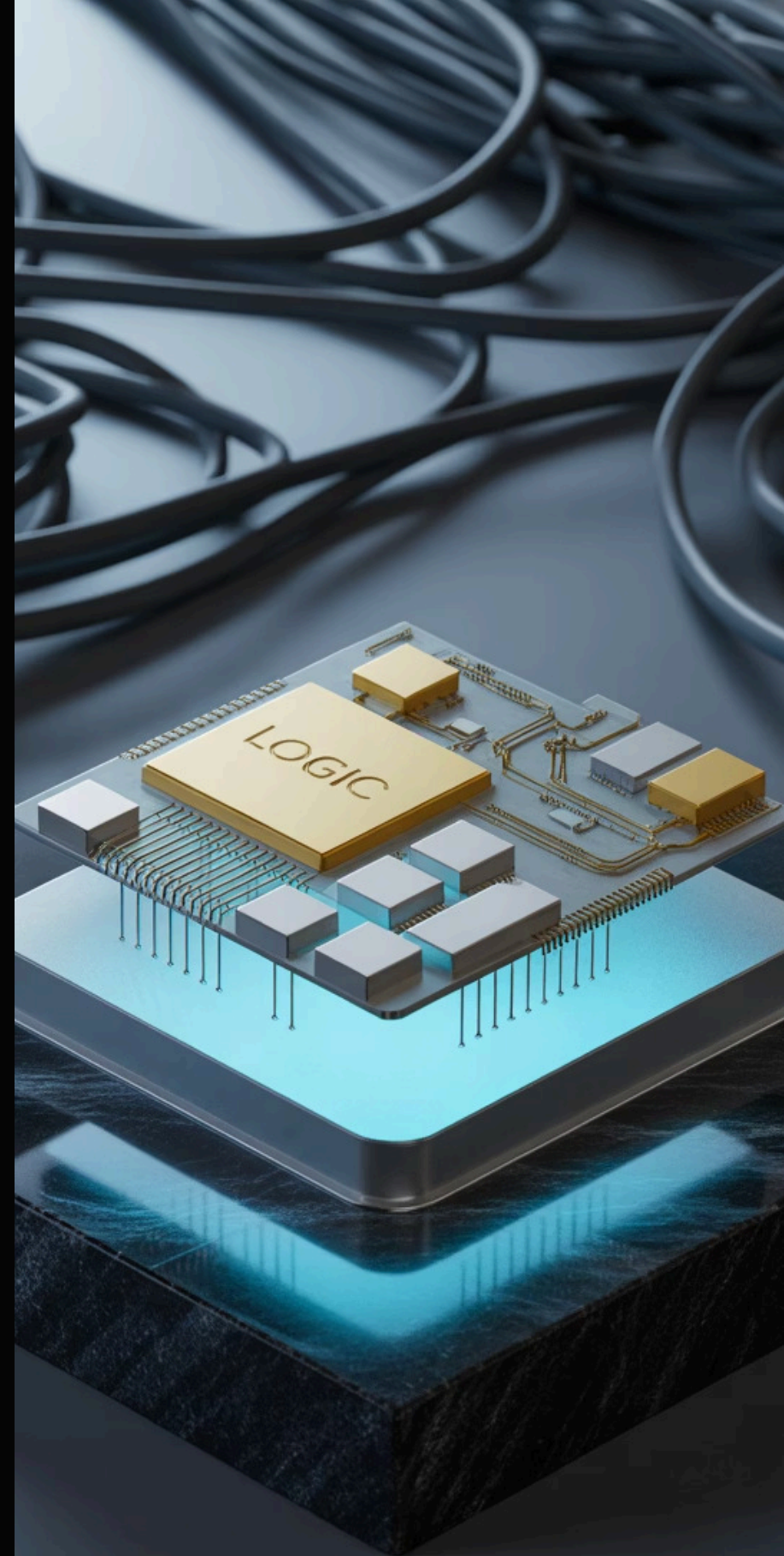
OR (ou)

Au moins une condition doit être vraie



NOT (non)

Inverse la valeur de vérité

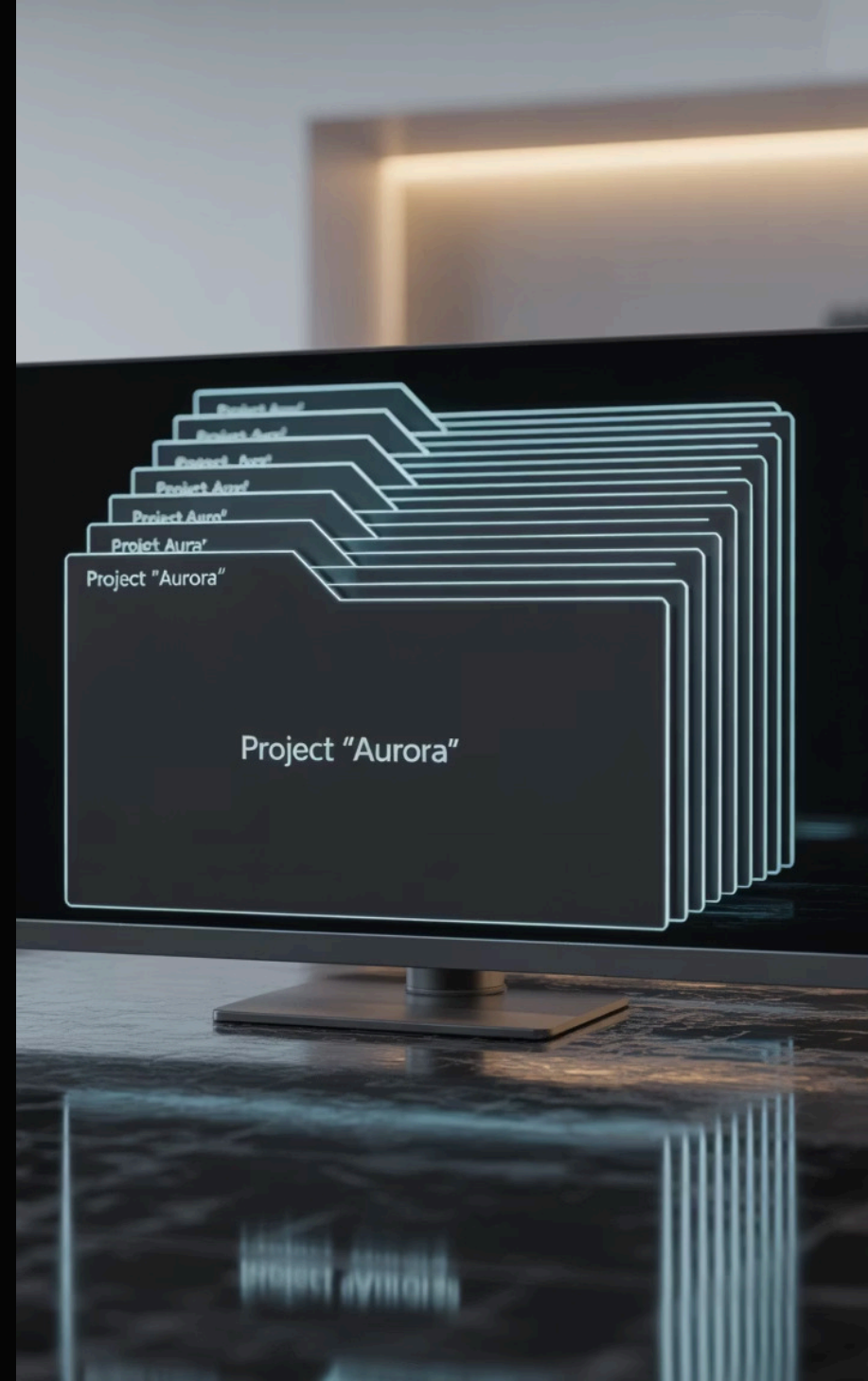


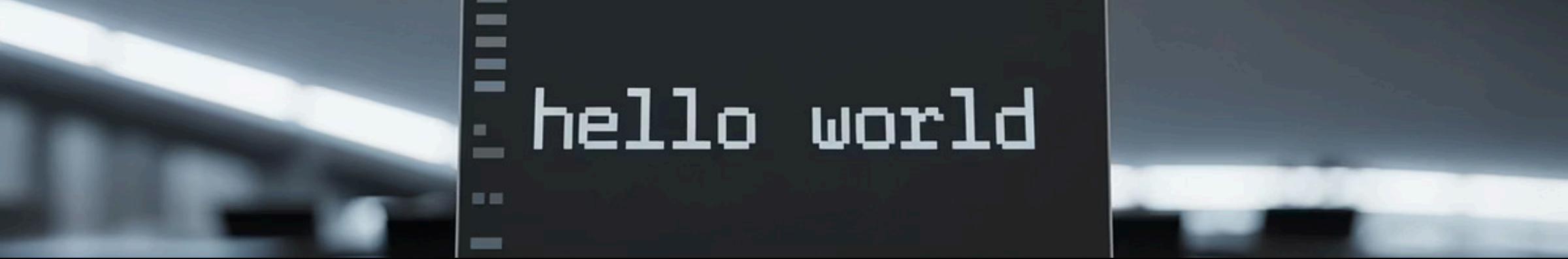
Conditions Imbriquées : L'Art de la Précision

Parfois, la logique métier nécessite des conditions dans des conditions. Voici un exemple de calcul de TVA selon le pays et le type de client :

```
pays = input("Pays : ")
montant_ht = float(input("Montant HT : "))
type_client = input("Type (entreprise/particulier) : ")

if pays.lower() == "france":
    tva = 0.20
    montant_ttc = montant_ht * (1 + tva)
    if type_client.lower() == "entreprise":
        print(f"TTC = {montant_ttc:.2f}€ (TVA récupérable)")
    else:
        print(f"TTC = {montant_ttc:.2f}€ (TVA payée)")
elif pays.lower() == "allemagne":
    tva = 0.19
    montant_ttc = montant_ht * (1 + tva)
    print(f"TTC (Allemagne) = {montant_ttc:.2f}€")
else:
    print("Export hors UE : pas de TVA")
```





```
hello world
```

L'Opérateur Ternaire : Concision

Syntaxe Classique

```
# Exemple de variable
benefice_net = 250

if benefice_net >= 0:
    statut_financier = "Bénéfice"
else:
    statut_financier = "Perte"
print(statut_financier)
```

Version Ternaire

```
# Exemples de variables
chiffre_affaire = 1200
depenses = 950

benefice_net = chiffre_affaire - depenses
statut_financier = "Bénéfice" if benefice_net >= 0 else "Perte"
print(f"Résultat : {statut_financier} ({benefice_net:.2f}€)")
```

L'opérateur ternaire suit la structure : `valeur_si_vrai if condition else valeur_si_faux`

Quand l'utiliser : Pour des affectations simples avec deux possibilités. Évitez-le pour des logiques complexes qui nuiraient à la lisibilité.

Match/Case : La Révolution Python 3.10

Depuis Python 3.10, une nouvelle syntaxe révolutionnaire est disponible : **match/case**. Elle remplace avantageusement les longues séries de if/elif pour comparer une variable à plusieurs valeurs.

Ancienne Méthode

```
choix_utilisateur = "facture"

if choix_utilisateur == "facture":
    print("Création facture")
elif choix_utilisateur == "devis":
    print("Création devis")
elif choix_utilisateur == "contrat":
    print("Rédaction contrat")
else:
    print("Opération inconnue")
```

Nouvelle Syntaxe

```
choix_utilisateur = "facture"

match choix_utilisateur:
    case "facture":
        print("Création facture")
    case "devis":
        print("Création devis")
    case "contrat":
        print("Rédaction contrat")
    case _: # cas par défaut
        print("Opération inconnue")
```

Avantages du Match/Case

- Plus lisible pour de nombreux cas
- Pas de "fallthrough" (traversée implicite)
- Support des patterns complexes

Patterns Avancés

```
match statut_commande:
    case "payé" | "validé":
        print("Paiement OK")
    case "en attente" | "en cours":
        print("En traitement")
    case _:
        print("Autre cas")
```


Quiz Interactif : Testez Vos Connaissances !

Question 1

Analysez ce code et prédisez le résultat :

```
x = 5
if x > 3 and x < 10 or x == 0:
    if x % 2 == 1:
        print("A")
    else:
        print("B")
else:
    print("C")
```

Question 2

Que se passe-t-il avec ce code et pourquoi ?

```
age = input("Votre âge : ")
if age >= 18:
    print("Majeur")
else:
    print("Mineur")
```

Question 3

Optimisez cette série de conditions :

```
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"
```

Comment pourriez-vous réécrire cela avec `match/case` et quels sont les avantages/inconvénients ?

Corrections du Quiz : Vérifiez vos réponses !

?

Question 1 : Analysez ce code et prédisez le résultat

Réponse : Le code affiche "A".

Explication : `x=5` satisfait `(x > 3 and x < 10)` qui est True, donc la condition globale est True. Ensuite, `5 % 2 == 1` est True (5 est impair), donc "A" s'affiche. Piège : l'ordre des opérateurs logiques (`and` a priorité sur `or`).

?

Question 2 : Que se passe-t-il avec ce code et pourquoi ?

Réponse : ERREUR !

Explication : `input()` renvoie toujours une chaîne de caractères. "18" (string) n'est pas `>= 18` (int). Il faut écrire : `age = int(input("Votre âge : "))`. C'est un piège classique quand on débute .

?

Question 3 : Optimisation avec `match/case`

Réponse :

```
match score // 10:
    case 10 | 9:
        grade = "A"
    case 8:
        grade = "B"
    case 7:
        grade = "C"
    case 6:
        grade = "D"
    case _:
        grade = "F"
```

Avantages : Plus lisible, pas de répétition de "score".

Inconvénients : Moins flexible pour des seuils non-multiples de 10.

Récapitulatif et Bonnes Pratiques

01

Maîtrisez la Syntaxe

N'oubliez jamais les **deux points (:)** après vos conditions et respectez l'**indentation** de 4 espaces.

02

Ordonnez Vos Tests

Traitez toujours les **cas spécifiques avant les cas généraux**. Python s'arrête au premier test réussi !

03


Choisissez le Bon Outil

if/elif pour la logique complexe, **match/case** pour comparer une variable à plusieurs valeurs, **ternaire** pour les affectations simples.

04

Testez Vos Conditions

Vérifiez tous les chemins possibles de votre code avec des **valeurs limites** et des cas d'erreur.

 **Prochaine étape :** Les structures répétitives (boucles) vous permettront de répéter des actions. Les conditions que vous maîtrisez maintenant seront essentielles pour contrôler ces répétitions !