

SÉRIE N° 2

Structures conditionnelles en Python

if / elif / else et match-case

Consignes générales

- Utilisez des **noms de variables explicites** (par exemple `prix_vente`, `age`, `note`).
- Ajoutez des **commentaires** (`# ...`) pour expliquer les étapes importantes du programme.
- Affichez des **messages clairs** à l'utilisateur avant chaque saisie et pour chaque résultat.
- **Testez** chaque programme avec plusieurs valeurs, y compris les cas limites (valeurs égales, valeurs aux bornes, valeurs invalides).

PARTIE A : Exercices de la série

EXERCICE N° 1 – Minimum de trois nombres

Écrivez un programme Python qui demande à l'utilisateur de saisir trois nombres réels, supposés deux à deux différents, puis affiche le plus petit des trois.

EXERCICE N° 2 – Parité et divisibilité

1. Écrivez un programme qui demande un nombre entier et indique s'il est pair ou impair.
2. Écrivez un programme qui demande un nombre entier et indique s'il est divisible à la fois par 3 et par 10.

Exemple : le nombre 330 est divisible par 3 et par 10.

EXERCICE N° 3 – Catégories d'un club sportif

Un club sportif qui forme des jeunes de moins de 14 ans vous demande de créer un programme permettant d'informer les familles de la catégorie de leur enfant en fonction de son âge. Le club vous a communiqué les informations suivantes :

Catégorie	Âge
Poussin	de 6 à 7 ans
Pupille	de 8 à 9 ans
Minime	de 10 à 11 ans
Cadet	de 12 à 13 ans

Écrivez un programme qui demande l'âge d'un enfant, puis affiche sa catégorie. Si l'âge est inférieur à 6 ans ou supérieur à 13 ans, le programme affiche : « Âge non accepté par le club ».

EXERCICE N° 4 – Les saisons

Écrivez un programme Python qui demande à l'utilisateur le numéro d'un mois (de 1 à 12) et affiche la saison correspondante :

Numéro du mois	Saison à afficher
3, 4 ou 5	Printemps
6, 7 ou 8	Été
9, 10 ou 11	Automne
12, 1 ou 2	Hiver

Si le numéro saisi n'est pas compris entre 1 et 12, le programme affiche : « Mois invalide ».

EXERCICE N° 5 – Profit ou perte

Écrivez un programme Python qui demande le prix de fabrication et le prix de vente d'un produit, puis indique s'il y a un profit ou une perte, et affiche le montant correspondant :

- si le prix de fabrication est supérieur au prix de vente, il y a une **perte** (montant = prix de fabrication – prix de vente) ;
- si le prix de fabrication est inférieur au prix de vente, il y a un **profit** (montant = prix de vente – prix de fabrication) ;
- si les deux prix sont égaux, il n'y a ni profit ni perte.

EXERCICE N° 6 – Magasin de reprographie

Un magasin de reprographie facture chaque photocopie 0,5 DH pour les dix premières photocopies, puis 0,3 DH par photocopie au-delà de la dixième.

Écrivez un programme Python qui demande à l'utilisateur le nombre de photocopies effectuées, puis affiche le montant à payer.

Exemples :

- Si le nombre de photocopies est 8, le montant est : $8 \times 0,5 = 4$ DH.
- Si le nombre de photocopies est 22, le montant est : $10 \times 0,5 + 12 \times 0,3 = 8,6$ DH.

EXERCICE N° 7 – Réservation d'un billet d'avion

Écrivez un programme qui permet à un passager de réserver une place d'avion pour un vol Rabat – Paris. Le programme affiche d'abord le menu suivant :

```
Choisissez la classe :
P  Première Classe (1200 DH)
E  Classe Économique (300 DH)
Tapez la lettre de la classe voulue :
```

Le programme demande ensuite le poids du bagage à enregistrer (en kg), puis affiche le **total à payer**, calculé ainsi :

$\text{total} = \text{prix de la classe} + \text{frais de réservation (20 DH)} + \text{poids du bagage} \times 12$
DH

(le tarif du bagage est de 12 DH par kilogramme). Si la lettre saisie ne correspond à aucune classe, le programme affiche un message d'erreur.

Exemples :

- Classe E et bagage de 15 kg : $300 + 20 + 15 \times 12 = 500$ DH.

RAPPEL : règles de syntaxe

1. La structure if / elif / else

```
if condition1:
    instructions           # exécutées si condition1 est vraie
elif condition2:
    instructions           # exécutées si condition1 est fausse et condition2
                           # vraie
else:
    instructions           # exécutées si aucune condition n'est vraie
```

À retenir

- La ligne `if`, `elif` ou `else` se termine toujours par **deux-points** `:`.
- Les instructions du bloc sont **indentées** (4 espaces) ; l'indentation fait partie de la syntaxe.
- Les conditions sont testées **dans l'ordre** : dès qu'une condition est vraie, les suivantes sont ignorées.
- `elif` et `else` sont facultatifs. Un seul `else` est autorisé, toujours à la fin.
- `=` sert à **affecter** une valeur ; `==` sert à **comparer**.
- `input()` renvoie toujours du texte : convertir avec `int(...)` ou `float(...)` pour obtenir un nombre.

2. Opérateurs utiles

Opérateur	Signification	Exemple
<code>==</code> <code>!=</code>	égal à / différent de	<code>age == 10</code>
<code><</code> <code>></code> <code><=</code> <code>>=</code>	comparaisons d'ordre	<code>note >= 10</code>
<code>and</code>	ET logique (les deux conditions vraies)	<code>x > 0 and x < 10</code>
<code>or</code>	OU logique (au moins une condition vraie)	<code>m == 1 or m == 2</code>
<code>not</code>	négation	<code>not (x > 5)</code>
<code>%</code>	reste de la division entière	<code>10 % 3</code> donne 1
<code>//</code>	quotient de la division entière	<code>10 // 3</code> donne 3

3. La structure match-case (Python 3.10 et plus)

```
match variable:
    case valeur1:
        instructions
    case valeur2 | valeur3:      # plusieurs valeurs possibles
        instructions
    case n if condition:        # case avec une garde (condition)
        instructions
    case _:                     # cas par défaut
        instructions
```

À retenir

- `match variable`: se termine par deux-points ; chaque `case` est **indenté** sous le `match`, et ses instructions sont indentées sous le `case`.
- Chaque `case` compare la variable à une **valeur précise** : un nombre (`case 3:`) ou un texte entre guillemets (`case "P":`).
- Le symbole `|` regroupe plusieurs valeurs dans un même `case` : `case 3 | 4 | 5:`.
- `case _` : correspond à **tous les autres cas** (valeur par défaut). Il se place **toujours en dernier**.
- Les `case` sont testés **de haut en bas** ; seul le **premier** `case` qui correspond est exécuté. Il n'y a pas de `break`.
- Pour un **intervalle** de valeurs, on utilise une **garde** : `case age if 6 <= age <= 17:`.

PARTIE B : Exercices supplémentaires (if / elif / else)**EXERCICE N° 8 – Année bissextile**

Écrivez un programme Python qui demande à l'utilisateur une année (nombre entier positif) et indique si elle est bissextile ou non.

Une année est bissextile si elle est divisible par 4 sans être divisible par 100, **ou** si elle est divisible par 400.

Exemples :

- 2024 : année bissextile.
- 1900 : année non bissextile.
- 2000 : année bissextile.

Règles à utiliser

- Lire l'année avec `int(input(...))`.
- Utiliser l'opérateur modulo `%` : `annee % 4 == 0` signifie « `annee` est divisible par 4 ».
- Combiner les conditions avec `and` et `or`, en utilisant des **parenthèses** pour préciser l'ordre des opérations : `(a and b) or c`.
- Utiliser une structure `if ... else` pour afficher le résultat.

EXERCICE N° 9 – Mention selon la note

Écrivez un programme Python qui demande une note sur 20 (nombre réel) et affiche la mention correspondante :

Note	Mention
moins de 10	Ajourné
de 10 (inclus) à 12 (exclu)	Passable
de 12 (inclus) à 14 (exclu)	Assez bien
de 14 (inclus) à 16 (exclu)	Bien

Note	Mention
16 et plus	Très bien

Si la note n'est pas comprise entre 0 et 20, le programme affiche : « Note invalide ».

Exemples :

- 15,5 : Bien.
- 9,99 : Ajourné.
- 22 : Note invalide.

Règles à utiliser

- Lire la note avec `float(input(...))`.
- Vérifier **d'abord** la validité de la note : `if note < 0 or note > 20:`.
- Enchaîner ensuite les tranches avec `elif`. Les conditions sont testées dans l'ordre : après avoir testé `note < 10`, il est inutile de vérifier `note >= 10` dans le `elif` suivant.
- Python accepte la comparaison enchaînée : `10 <= note < 12` équivaut à `note >= 10 and note < 12`.
- Utiliser un `else` final pour la dernière tranche (16 et plus).

EXERCICE N° 10 – Nature d'un triangle

Écrivez un programme Python qui demande les longueurs de trois côtés `a`, `b` et `c` (nombres réels), puis :

1. vérifie si ces longueurs peuvent former un triangle : elles doivent être strictement positives et chaque côté doit être strictement inférieur à la somme des deux autres ;
2. si c'est le cas, affiche la nature du triangle : **équilatéral** (trois côtés égaux), **isocèle** (exactement deux côtés égaux) ou **scalène** (trois côtés différents) ;
3. sinon, affiche : « Ces longueurs ne forment pas un triangle ».

Exemples :

- 3, 3, 3 : triangle équilatéral.
- 5, 5, 8 : triangle isocèle.
- 3, 4, 5 : triangle scalène.
- 1, 2, 10 : ces longueurs ne forment pas un triangle.

Règles à utiliser

- Utiliser des **`if` imbriqués** : un premier `if` vérifie l'existence du triangle ; à l'intérieur de son bloc (indenté), un second ensemble `if / elif / else` détermine la nature du triangle.
- Condition d'existence : $a > 0$ and $b > 0$ and $c > 0$ and $a < b + c$ and $b < a + c$ and $c < a + b$.
- Tester le cas **équilatéral** ($a == b == c$) **avant** le cas isocèle ; sinon un triangle équilatéral serait classé isocèle.
- Cas isocèle : $a == b$ or $a == c$ or $b == c$.
- Le bloc imbriqué est décalé de **4 espaces supplémentaires** par rapport au `if` qui le contient.

EXERCICE N° 11 – Équation du second degré

Écrivez un programme Python qui demande les coefficients réels a , b et c de l'équation $ax^2 + bx + c = 0$ et affiche ses solutions réelles.

- Si $a = 0$, afficher : « Ce n'est pas une équation du second degré ».
- Sinon, calculer le discriminant $\Delta = b^2 - 4ac$:
 - si $\Delta > 0$: deux solutions distinctes $x_1 = (-b + \sqrt{\Delta}) / (2a)$ et $x_2 = (-b - \sqrt{\Delta}) / (2a)$;
 - si $\Delta = 0$: une solution double $x = -b / (2a)$;
 - si $\Delta < 0$: afficher « Aucune solution réelle ».

Exemples :

- $a = 1, b = -3, c = 2$: $x_1 = 2.0$ et $x_2 = 1.0$.
- $a = 1, b = 2, c = 1$: solution double $x = -1.0$.
- $a = 1, b = 1, c = 1$: aucune solution réelle.

Règles à utiliser

- Tester $a == 0$ **en premier** (avec un `if`), avant tout calcul comportant une division par a .
- Calculer le discriminant : `delta = b**2 - 4*a*c` (l'opérateur `**` désigne la puissance).
- Pour la racine carrée : écrire `import math` en début de programme puis `math.sqrt(delta)` (ou bien `delta ** 0.5`).
- Utiliser une structure à trois branches : `if delta > 0: / elif delta == 0: / else:.`
- Mettre le dénominateur entre parenthèses : `(-b + racine) / (2*a)`. Sans parenthèses, le résultat serait faux.

EXERCICE N° 12 – Remise dans un magasin

Un magasin accorde une remise selon le montant total des achats :

Montant des achats	Taux de remise
moins de 500 DH	0 %
de 500 DH (inclus) à 1000 DH (exclu)	5 %
1000 DH et plus	10 %

Les clients possédant une carte de fidélité bénéficient d'une remise supplémentaire de 2 %, qui s'ajoute au taux précédent.

Écrivez un programme qui demande le montant des achats et si le client possède une carte de fidélité (réponse « oui » ou « non »), puis affiche le taux de remise total, le montant de la remise et le net à payer.

Exemples :

- 800 DH avec carte : taux 7 %, remise 56 DH, net à payer 744 DH.
- 300 DH sans carte : taux 0 %, remise 0 DH, net à payer 300 DH.

Règles à utiliser

- Déterminer d'abord le taux selon la tranche avec `if / elif / else`, et le stocker dans une variable `taux`.
- Utiliser ensuite un **second `if` indépendant** (et non un `elif`) : `if carte == "oui": taux = taux + 2`. Cette condition ne dépend pas de la première structure.
- Un texte se compare avec `==` et s'écrit **entre guillemets** : `"oui"`. La réponse se lit avec `input()` seul, sans conversion.
- Formules : `remise = montant * taux / 100` et `net = montant - remise`.

PARTIE C : Exercices supplémentaires (match-case)

EXERCICE N° 13 – Jours de la semaine

Le numéro 1 correspond au lundi, 2 au mardi, ..., 7 au dimanche.

1. Écrivez un programme qui demande un numéro de jour (de 1 à 7) et affiche le nom du jour. Pour un autre numéro, afficher « Numéro invalide ».
2. Écrivez un second programme qui demande un numéro de jour et affiche « Jour de travail » (du lundi au vendredi) ou « Week-end » (samedi et dimanche). Pour un autre numéro, afficher « Numéro invalide ».

Exemples :

- 3 : Mercredi (question 1) ; Jour de travail (question 2).
- 7 : Dimanche (question 1) ; Week-end (question 2).
- 9 : Numéro invalide.

Règles à utiliser

- Lire le numéro avec `int(input(...))`, car les `case` comparent ici des nombres entiers.
- Question 1 : écrire `match jour:` avec un **`case` par valeur** (`case 1:`, `case 2:`, ...).
- Question 2 : regrouper les valeurs avec le symbole `|` : `case 1 | 2 | 3 | 4 | 5:` puis `case 6 | 7:`.
- Terminer chaque `match` par `case _:` pour traiter les numéros invalides.
- Il n'y a pas de `break` : seul le premier `case` correspondant est exécuté.

EXERCICE N° 14 – Calculatrice simple

Écrivez un programme Python qui demande deux nombres réels puis une opération (+, -, * ou /), et affiche le résultat de l'opération.

En cas de division par zéro, afficher « Division par zéro impossible ». Si l'opération saisie n'est pas l'une des quatre proposées, afficher « Opération inconnue ».

Exemples :

- 8, 2 et / : résultat 4.0.
- 5, 0 et / : Division par zéro impossible.
- 5, 3 et % : Opération inconnue.

Règles à utiliser

- Lire les nombres avec `float(input(...))` et l'opération avec `input(...)` seul (texte).
- Écrire `match operation:` avec un `case` par symbole, **entre guillemets** : `case "+":`, `case "-":`, `case "*":`, `case "/":`.
- Dans `case "/"`, utiliser un `if ... else` pour tester si le second nombre est égal à 0 avant de diviser.
- Terminer par `case _:` pour une opération inconnue.
- Chaque `case` contient son propre affichage du résultat.

EXERCICE N° 15 – Cafétéria

Une cafétéria propose les boissons suivantes :

Lettre	Boisson	Prix unitaire
C	Café	5 DH
T	Thé	4 DH
J	Jus	8 DH
E	Eau	3 DH

Écrivez un programme qui affiche ce menu, demande à l'utilisateur la lettre de la boisson choisie puis la quantité souhaitée, et affiche le total à payer. Si la lettre ne correspond à aucune boisson, afficher « Boisson inconnue » (et ne pas calculer de total).

Exemples :

- J et quantité 3 : total à payer 24 DH.
- X : Boisson inconnue.

Règles à utiliser

- Écrire `match choix` : avec un `case` par lettre, **entre guillemets** : `case "C":`, `case "T":`, ...
- Pour accepter aussi les minuscules, regrouper avec `|` : `case "C" | "c":`.
- **Initialiser** `prix = 0` avant le `match` ; chaque `case` affecte à `prix` le prix unitaire de la boisson.
- `case _` : affiche le message « Boisson inconnue » (le prix reste alors égal à 0).
- Après le `match`, utiliser `if prix > 0` : pour calculer `total = prix * quantite` et l'afficher.
- Lire la quantité avec `int(input(...))`.

EXERCICE N° 16 – Tarif d'un billet de cinéma

Un cinéma applique un tarif selon l'âge du spectateur :

Âge	Tarif
moins de 6 ans	Gratuit
de 6 à 17 ans	20 DH
de 18 à 59 ans	40 DH
60 ans et plus	25 DH

Écrivez un programme qui demande l'âge du spectateur et affiche son tarif. Si l'âge est négatif, afficher « Âge invalide ». **Vous devez utiliser `match-case`.**

Exemples :

- 4 : Gratuit.
- 25 : 40 DH.
- 70 : 25 DH.
- -3 : Âge invalide.

Règles à utiliser

- Un intervalle ne peut pas s'écrire directement dans un `case` : utiliser une **garde** avec la syntaxe `case age if condition:`.
- Exemple : `case age if 6 <= age <= 17` : s'exécute lorsque l'âge est compris entre 6 et 17.
- Les `case` sont testés **de haut en bas** : placer d'abord le cas invalide (`case age if age < 0`), puis les tranches dans l'ordre croissant.
- Terminer par `case _`, qui correspond ici aux personnes de 60 ans et plus.

EXERCICE N° 17 – Nombre de jours d'un mois

Écrivez un programme Python qui demande un numéro de mois (de 1 à 12) et affiche le nombre de jours de ce mois.

Pour le mois de février (2), le programme demande en plus l'année et affiche 29 jours si l'année est bissextile (voir l'exercice 8), sinon 28 jours. Pour un numéro de mois invalide, afficher « Mois invalide ».

Exemples :

- 7 : 31 jours.
- 2 puis 2024 : 29 jours.
- 2 puis 1900 : 28 jours.
- 13 : Mois invalide.

Règles à utiliser

- Regrouper les mois de 31 jours : `case 1 | 3 | 5 | 7 | 8 | 10 | 12:.`
- Regrouper les mois de 30 jours : `case 4 | 6 | 9 | 11:.`
- Dans `case 2:`, demander l'année (uniquement dans ce `case`) puis utiliser un `if ... else` avec la condition d'année bissextile de l'exercice 8.
- Terminer par `case _:` pour un mois invalide.

EXERCICE N° 18 – Bonus : réécriture avec match-case

Réécrivez l'exercice 4 (les saisons) et l'exercice 7 (réservation d'un billet d'avion) en utilisant `match-case` à la place de `if / elif / else`. Comparez ensuite les deux versions : dans quels cas `match-case` rend-il le programme plus lisible ?

Règles à utiliser

- Exercice 4 : utiliser `case 3 | 4 | 5:` pour le printemps, et de même pour les autres saisons ; `case _:` pour un mois invalide.
- Exercice 7 : utiliser `case "P" | "p":` et `case "E" | "e":` ; `case _:` pour une classe inconnue.
- `match-case` convient bien lorsqu'on compare une variable à des **valeurs précises** (numéros, lettres, mots).
- `if / elif / else` reste plus naturel pour des **intervalles** et des **conditions composées** (notes, âges, montants).