

Cours 3

ALTERNATIVE DE BASE, OPÉRATEURS DE COMPARAISON ET
OPÉRATEURS LOGIQUES

Alternative de base

Définition

En programmation, une **structure d'alternative**, également connue sous le nom de **structure conditionnelle**, est un mécanisme de contrôle du flux d'exécution qui permet à un programme de prendre des décisions en fonction de certaines conditions.

Les structures d'alternative les plus courantes sont les instructions *if*, *else if* (ou *elif* selon le langage) et *else*. Ces instructions permettent de définir des chemins d'exécution différents en fonction des conditions évaluées.

Le principe de base est simple :

- Le programme **évalue une condition**.
- Si la condition est **vraie**, il exécute un bloc d'instructions.
- Sinon, il passe à une **alternative** (souvent avec un bloc *else*).

Syntaxe

Voici la syntaxe générale de la structure conditionnelle en Python :

```
if condition:
    # Bloc d'instructions à exécuter si la condition est vraie
else:
    # Bloc d'instructions à exécuter si aucune condition précédente n'est vraie
```

Schéma du SI

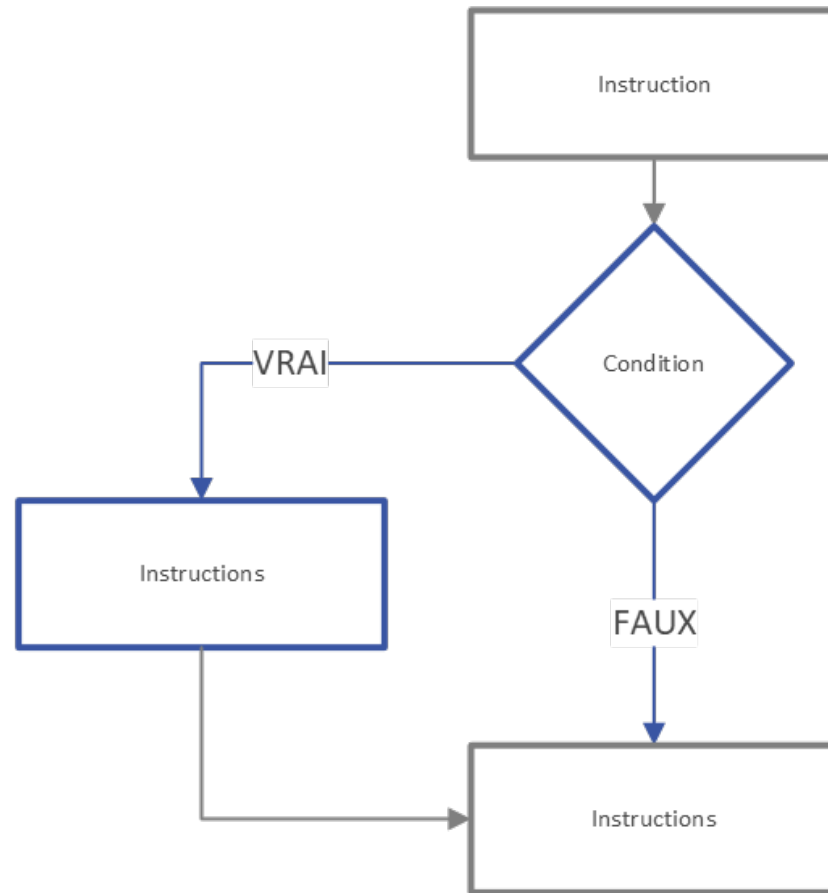
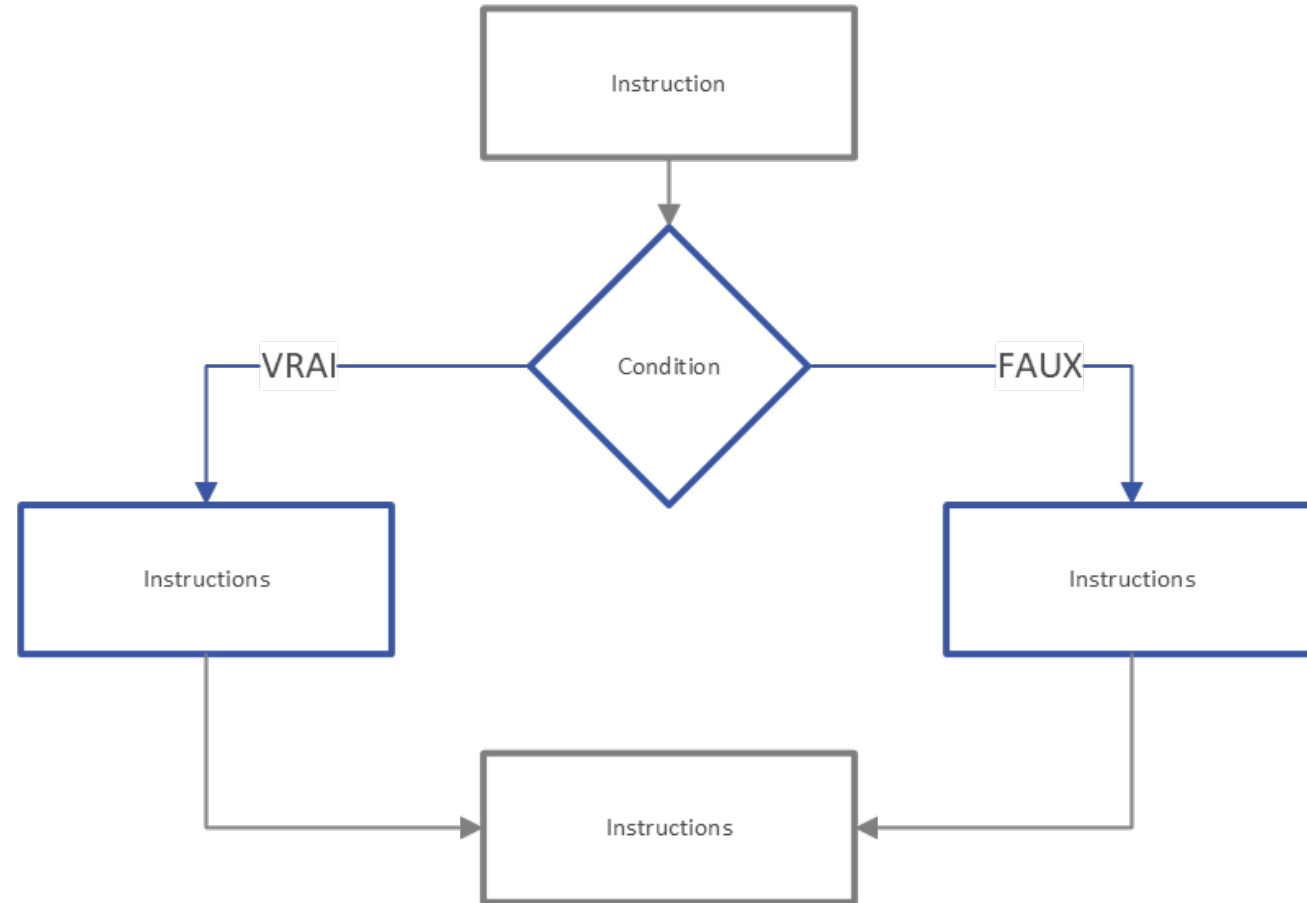


Schéma du SI-SINON



Exemple

Code

```
nombre : int = int(input(« Veuillez entrer un nombre »))

if nombre >= 0:
    print("Le nombre entré")
    print("est positif ou zéro")
else:
    print("Le nombre est négatif")
```

Résultat (si l'utilisateur entre 5)

```
Le nombre entré
est positif ou zéro
```

Bloc d'instructions

- Plus d'une instruction peut être exécutée lorsque la condition est vraie ou fausse.
- En Python, pour démarquer un bloc d'instructions, on utilise l'indentation.
- Toutes les instructions qui sont alignées ensembles font partie du même bloc de code.
- Exemple précédent :

```
nombre : int = int(input(« Veuillez entrer un nombre »))

if nombre >= 0:
    print("Le nombre entré")
    print("est positif ou zéro")
else:
    print("Le nombre est négatif")
```

Évaluation des conditions

Les conditions dans un *if* sont des expressions booléennes qui retournent soit *True* soit *False*. Les opérateurs couramment utilisés sont :

- **Opérateurs de comparaison** : `==`, `!=`, `<`, `>`, `<=`, `>=`
- **Opérateurs logiques** : `and`, `or`, `not`

Exemple avec plusieurs conditions

Code

```
age : int = 20
nationalite : str = "française"

if age >= 18 and nationalite == "française":
    print("Vous pouvez voter")
else:
    print("Vous ne pouvez pas voter")
```

Résultat

Vous pouvez voter

Cas particulier : une condition unique

Code

```
temperature : int = 35  
  
if temperature > 30:  
    print("Il fait très chaud aujourd'hui")
```

Résultat

```
Il fait très chaud aujourd'hui
```

Opérateurs de comparaison

Définition

Les **opérateurs de comparaison** sont utilisés pour comparer deux valeurs et déterminer leur relation.

Permet de répondre à des questions de type :

- Est-ce que deux valeurs sont égales ?
- Est-ce que l'une est supérieure/inférieures à l'autre ?
- Est-ce que deux valeurs sont différentes ?

Exemples

- Est-ce que 10 est égal à 5 ?
- Est-ce que 15 est inférieur à 20 ?
- Est-ce 8 est différent de 8 ?

Les opérateurs

Opérateur	Description	Exemple (a = 10, b = 5)	Résultat
==	Égal à	a == b	False
!=	Différent de	a != b	True
>	Supérieur à	a > b	True
<	Inférieur à	a < b	False
>=	Supérieur ou égal à	a >= b	True
<=	Inférieur ou égal à	a <= b	False

Exemple 1

Code

```
note : int = 15
if note != 20:
    print("Vous n'avez pas la note maximale.")
else:
    print("Félicitations, vous avez la note maximale !")
```

Résultat

Vous n'avez pas la note maximale.

Exemple 2

Code

```
vitesse : int = 60
if vitesse < 80:
    print("Vous respectez la limite de vitesse.")
else:
    print("Attention, vous roulez trop vite !")
```

Résultat

Vous respectez la limite de vitesse.

Opérateurs logiques

Définition

Les **opérateurs logiques**, aussi souvent appelés opérateurs "booléens", permettent de combiner plusieurs conditions et de prendre des décisions plus complexes en fonction de plusieurs critères.

Ils sont utilisés dans des **expressions booléennes** (retournant soit True, soit False) et sont souvent employés dans les structures de contrôle comme if, while, et les boucles.

En Python, les trois opérateurs logiques principaux sont :

- and
- or
- not

L'opérateur « and »

L'opérateur **and** permet de vérifier si **toutes les conditions** dans une expression sont vraies.

L'opérateur retourne *True* si toutes les conditions sont vérifiées, sinon il retourne *False*.

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

Exemple « and »

Code

```
identifiant_valide : bool = True
mot_de_passe_valide : bool = False

if identifiant_valide and mot_de_passe_valide:
    print("Accès autorisé.")
else:
    print("Accès refusé.")
```

Résultat

Accès refusé.

L'opérateur « or »

L'opérateur **or** permet de vérifier si **au moins une condition** dans une expression est vraie.

L'opérateur retourne *True* si au moins une condition est vérifiée, sinon il retourne *False*.

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

Exemple « or »

Code

```
age : int = 16
autorisation_parentale : bool = True

if age >= 18 or autorisation_parentale:
    print("Vous pouvez participer à la compétition.")
else:
    print("Vous ne pouvez pas participer.")
```

Résultat

Vous pouvez participer à la compétition.

L'opérateur « not »

L'opérateur **not** est utilisé pour **inverser** une valeur booléenne.

Si l'expression initiale retourne *True*, alors **not** la transforme en *False*, et vice-versa.

a	not a
False	True
True	False

Exemple « not »

Code

```
carte_acces_valide : bool = False

if not carte_acces_valide:
    print("Carte invalide, accès refusé.")
else:
    print("Accès autorisé.")
```

Résultat

Carte invalide, accès refusé.

Résumé

Opérateur	Description	Exemple (a = 10, b = 5)	Résultat
and	Vrai si les deux conditions sont vraies	(a > 5 and b > 1)	True
or	Vrai si au moins une des conditions est vraie	(a > 15 or b > 1)	True
not	Inverse la valeur booléenne	not (a > 5)	False

Utilisation combinée

On peut combiner plusieurs opérateurs pour formuler des conditions plus complexes.

Code

```
age = 17
diplome = True
lettre_recommandation = False

if (age >= 18 and diplome) or lettre_recommandation:
    print("Admission acceptée.")
else:
    print("Admission refusée.")
```

Résultat

Admission refusée.

Priorités d'évaluation

Voici une table de priorité d'évaluation des divers opérateurs Python vus jusqu'à présent:

Opérateur	Description
()	Parenthèses
*, /, //, %	Opérateurs arithmétiques – multiplication et division
+, -	Opérateurs arithmétiques – addition et soustraction
==, !=, >, >=, <, <=	Opérateurs de comparaison
not	Opérateur logique – négation
and	Opérateur logique – conjonction
or	Opérateur logique – disjonction

Exercices

Déterminer si les expressions suivantes retournent True ou False.

Expression	Résultat
4 > 2 and 3 < 5	
7 == 7 or 5 != 5	
not (10 <= 10)	
("chat" != "chien") and (2 + 2 == 4)	
(8 >= 8) and (3 > 5)	
(5 < 3) or (7 > 6 and 1 == 1)	
not (5 != 5 or 2 < 1)	
(6 <= 6 or 4 > 9) and (2 + 3 == 5)	

Résumé

- Opérateurs de comparaison
 - Comparer deux valeurs et déterminer leur relation.
 - ==, !=, >, <, >=, <=
- Opérateurs logiques
 - Combiner plusieurs conditions; prendre des décisions plus complexes en fonction de plusieurs critères.
 - and, or, not
- L'alternative de base
 - Mécanisme de contrôle du flux d'exécution qui permet à un programme de prendre des décisions en fonction de certaines conditions.
 - if, else, elif

Laboratoire #2
