

# Conditions en Python

# Table des matières

<b>I. Contrôle de flux et les opérateurs en Python ?</b>	<b>3</b>
<b>II. Exercice : Quiz</b>	<b>6</b>
<b>III. Instructions et opérateurs en Python</b>	<b>7</b>
A. Instructions if, else, elif .....	7
B. Opérateurs booléens et logiques .....	11
<b>IV. Exercice : Quiz</b>	<b>12</b>
<b>V. Essentiel</b>	<b>13</b>
<b>VI. Auto-évaluation</b>	<b>14</b>
A. Exercice .....	14
B. Test .....	14
<b>Solutions des exercices</b>	<b>15</b>

# I. Contrôle de flux et les opérateurs en Python ?

Durée : 1 h

Environnement de travail : PC connecté à internet

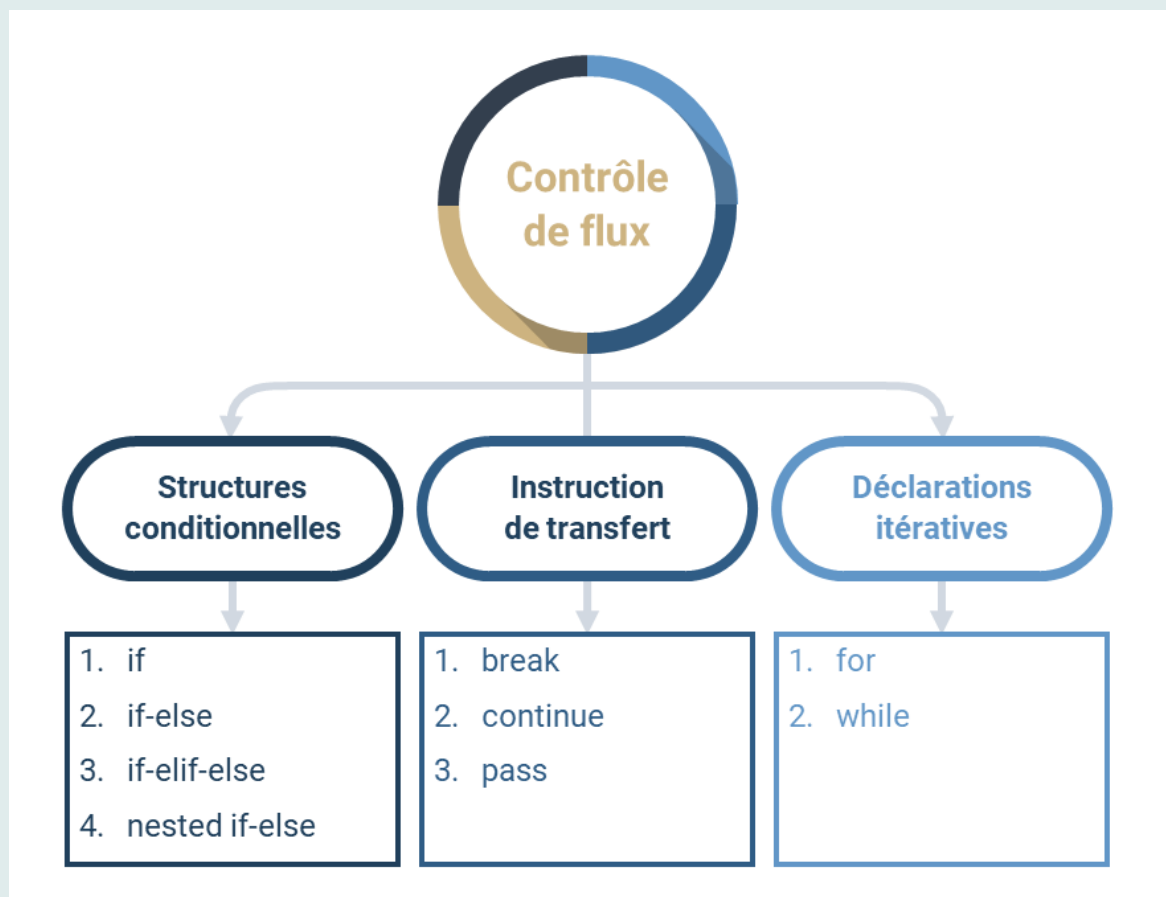
## Contexte

La prise de décision est aussi importante dans un langage de programmation que dans la vie en général. Dans un langage de programmation, elle est automatisée à l'aide d'instructions conditionnelles dans lesquelles Python évalue le code pour voir s'il répond aux conditions spécifiées. Les conditions sont évaluées et traitées comme étant vraies ou fausses. Si la condition s'avère être vraie, le programme est exécuté comme il se doit. Dans le cas contraire, l'instruction suivante est exécutée. Si vous maîtrisez un tant soit peu les instructions de base en Python, vous verrez qu'un programme n'est qu'une série d'instructions. Cependant, la vraie force de la programmation, ce n'est pas seulement de les exécuter l'une après l'autre. En évaluant des expressions à l'aide de structures conditionnelles, le programme peut décider de sauter des instructions, de les répéter ou de choisir une autre instruction à exécuter.

## Définition

En programmation Python, le contrôle de flux est **l'ordre dans lequel les instructions ou les blocs de code sont traités** au moment de l'exécution en fonction d'une condition. Les instructions de contrôle de flux sont divisées en trois catégories. Nous avons :

- Les expressions conditionnelles
- Les déclarations itératives
- Les instructions de transfert

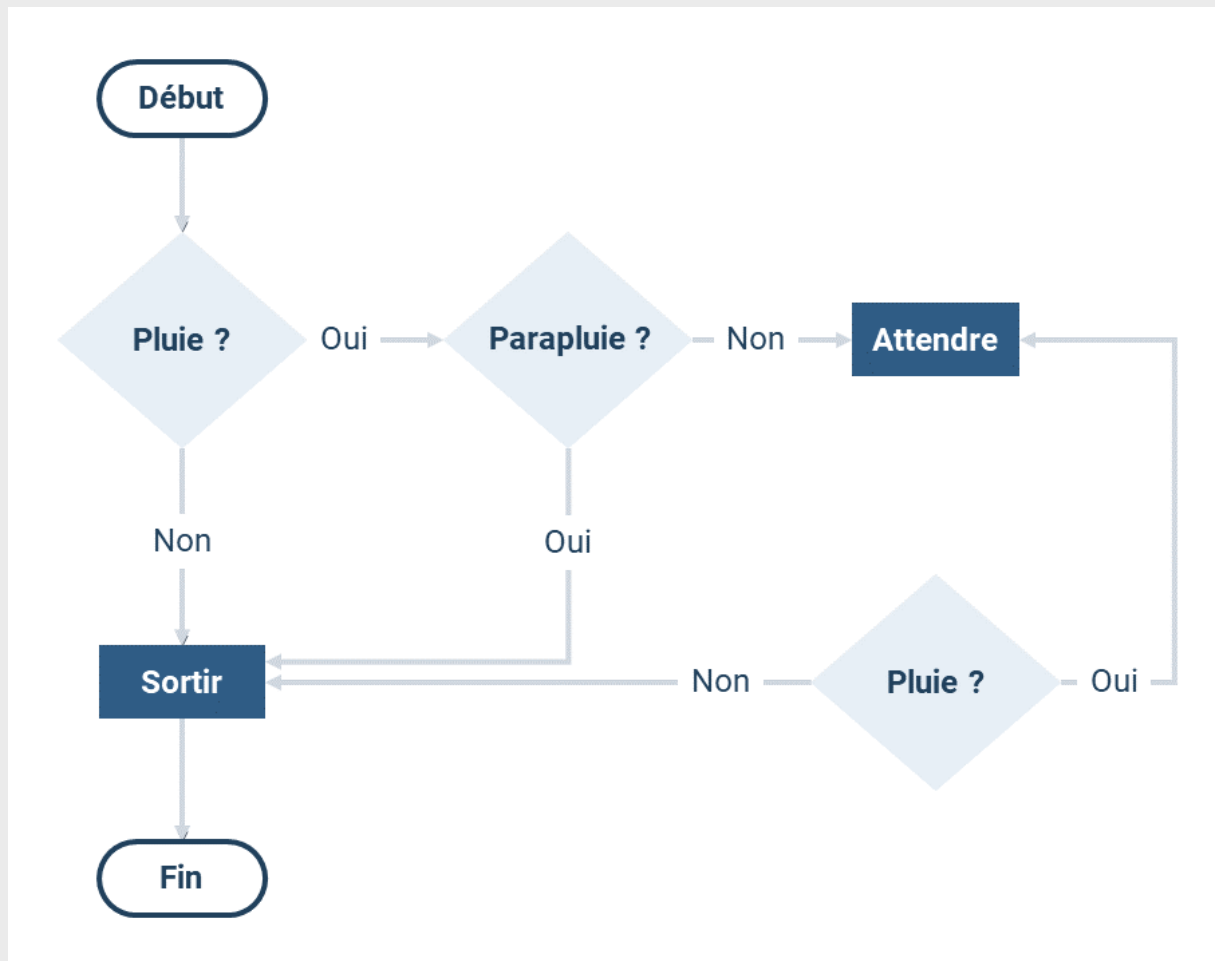


**Schéma : représentation du contrôle de flux**

**Exemple**

En pratique, vous ne voulez presque jamais que vos programmes commencent à partir de la première ligne de code et exécuter simplement chaque ligne jusqu'à la fin. Les instructions de contrôle de flux peuvent décider quelles instructions exécutées et à quelles conditions.

Cet organigramme indique ce qu'il faut faire en cas de pluie.



**Schéma : représentation des conditions en Python**

Dans un organigramme, il y a généralement plus d'un chemin à parcourir du début jusqu'à la fin. Il en va de même pour les lignes de code. Les étapes sur ce graphique sont représentées avec des diamants, tandis que les autres étapes sont représentées par des rectangles.

Les étapes de début et de fin sont représentées par des rectangles arrondis. Mais avant d'en savoir plus sur les instructions de contrôle de flux, vous devez d'abord apprendre à représenter ces options **oui** et **non**, et vous devez comprendre comment écrire ces embranchements sous forme de code Python. Pour représenter ces options, nous allons utiliser trois éléments : les **valeurs booléennes**, les **opérateurs de comparaison** et les **opérateurs booléens**.

## Valeurs booléennes

Les entiers, chiffres à virgule et les chaînes de caractères ont un nombre illimité de valeurs possibles, les valeurs booléennes n'ont que deux valeurs : **true** et **false**.

Lorsqu'il est tapé en code Python, les booléens True et False n'ont pas les guillemets que vous placez autour des chaînes, et elles commencent toujours par une majuscule, le reste du mot en minuscule.

Comme toutes les autres valeurs, les valeurs booléennes sont utilisées dans les expressions et peuvent être stockées dans les variables. Si vous n'utilisez pas de majuscule pour écrire **True** ou **False**, Python vous donnera un message d'erreur.

## Opérateurs de comparaison et d'identité

Un opérateur est un caractère ou une suite de caractères à laquelle la grammaire de Python donne une signification particulière. En Python la signification de l'opérateur n'est pas portée par l'opérateur lui-même, mais par le ou les objets sur lesquels il s'applique. Les opérateurs de comparaison comparent deux valeurs et évaluent le résultat à un opérateur booléen True ou False. Vous pouvez les utiliser sur tous les types de données et sur les variables.

	Opérateurs	Fonction
	<code>==</code>	Égalité
	<code>!=</code>	Inégalité
	<code>&lt;</code>	Moins que
+	<code>&gt;</code>	Supérieur à
	<code>&lt;=</code>	Moins ou égal à
	<code>&gt;=</code>	Supérieur ou égal à
	<code>is, is not</code>	Compare l'identité des objets

### Exemple

Comme vous pouvez vous y attendre, « `==` » est évalué à **True** lorsque les valeurs des deux côtés sont identiques, et « `!=` » est évalué à **True** lorsque les deux valeurs sont différentes. Les opérateurs « `==` » et « `!=` » peuvent fonctionner avec tout type de données. Nous allons utiliser une suite d'instructions pour appliquer l'utilisation de ces opérateurs en python.

```

1 # a et b sont des variables
2 a = 10
3 b = 12
4
5 Si ( a == b) Alors
6     Afficher('la variable a est égale à la variable b')
7 Sinon
8     Afficher('la variable a est différent de la variable b')
9 Finsi

```

Nous déclarons deux variables **a** et **b**, toutes deux entières. Ensuite, nous vérifions si la variable **a** est égale (mathématiquement) à la variable **b**. 10 étant différent de 12, la condition (**10 == 12**) renvoie **False**. L'instruction suivante est alors exécutée et on obtient l'affichage de : « **la variable a est différente de la variable b** ». Dans le cas où la condition (**a != b**) sera évaluée, nous obtiendrons l'affichage « **la variable a est égale à la variable b** », ce qui n'est évidemment pas le cas.

Notez que l'opérateur de comparaison « == » est différent de celui d'affectation « = ». Le premier est utilisé pour évaluer l'égalité dans une condition tandis que l'autre sert à attribuer des valeurs à des variables.

```
1 # a est une variable
2 a = 5
3
4 Si ( a > 5) Alors
5     Afficher('la variable a est supérieur à 5')
6 Finsi
7
8 Si ( a >= 5) Alors
9     Afficher('la variable a est supérieur ou égale à 5')
10 Finsi
```

Dans ce bloc de code, nous avons évalué deux conditions. La première vérifie si la variable **a** qui est égale à 5 est supérieure au chiffre 5. Cette première condition est évaluée comme **False**. L'instruction à l'intérieur de la condition ne s'affiche donc pas. Par contre, au niveau de la deuxième condition, nous évaluons non seulement la supériorité, mais également l'égalité. La condition avec l'opérateur « >= » (**Supérieur ou égal à**) renvoie donc **True**. Par conséquent, il est affiché « **la variable a est supérieur ou égale à 5** ».

```
1 # a et b sont des tableaux de données
2 a = ['c', 'd']
3 b = ['c', 'd']
4
5 a == b      # renvoie True.
6 a is a     # renvoie True.
7 a is b     # renvoie False.
8 a is not b  # renvoie True.
```

Les variables **a** et **b** sont des tableaux de caractères. Lorsque nous évaluons la condition (**a == b**), nous essayons de comparer chaque entité de chaque tableau pour voir s'ils sont identiques. (**a == b**) renvoie alors **True** puisque 'c' == 'c' et 'd' == 'd'.

La condition (**a is a**) renvoie **True** et celle (**a is b**) renvoie **False** parce que a et b sont deux variables différentes. Elles sont considérées comme des entités qui n'ont rien avoir l'un avec l'autre même s'ils ont des valeurs identiques. L'opérateur de comparaison **is not** utilisé dans la dernière condition représente l'opposé de **is**. Par conséquent, la dernière condition renvoie **True**.

## Exercice : Quiz

[solution n°1 p.17]

### Question 1

Comment évaluer la différence entre deux variables ?

- ☐ a != b
- ☐ a == b

### Question 2

Comment évaluer l'égalité entre deux variables ?

- ☐ a = b
- ☐ a == b

Question 3

Quelles sont les deux valeurs booléennes qui existent en Python ?

- ☐ True
- ☐ AND
- ☐ OR
- ☐ NOT
- ☐ False

Question 4

Quels sont les trois opérateurs booléens ?

- ☐ True
- ☐ False
- ☐ AND
- ☐ OR
- ☐ NOT

Question 5

Quels sont les opérateurs de comparaison ?

- ☐ ==
- ☐ =
- ☐ >=

### III. Instructions et opérateurs en Python

#### A. Instructions if, else, elif

Si vous voulez pouvoir écrire des applications véritablement utiles, il vous faut des techniques permettant d'aiguiller le déroulement du programme dans différentes directions, en fonction des circonstances rencontrées. Pour ce faire, nous devons disposer d'instructions capables de tester une certaine condition et de modifier le comportement du programme en conséquence.

Dans la partie précédente, nous avons utilisé des instructions **Si**, **Sinon** qui nous permettaient d'évaluer des conditions dans notre programme. Dans cette partie, nous allons voir leurs équivalents en Python. Il s'agit des instructions **If**, **else**, et **elif**.

Supposons que vous ayez du code qui vérifie si le nom de quelqu'un est Alice. Toutes les instructions de contrôle de flux se terminent par deux points et sont suivies d'un nouveau bloc de code (la clause). Cette clause **if** est le bloc qui imprime ('Bonjour, Alice').

Instruction **if** :

Le type d'instruction de contrôle de flux le plus courant est l'instruction if. La clause de l'instruction (c'est-à-dire le bloc suivant l'instruction if) s'exécutera si la condition de l'instruction est True sinon la clause est ignorée puisqu'elle est False. En langage simple, une déclaration **if** pourrait se lire comme suit : « ***Si** cette condition est True, exécutez le code dans la clause.* »

En Python, une instruction **if** se compose de la façon suivante :

- Le mot-clé **if**,
- Une condition (c'est-à-dire une expression qui a la valeur True ou False),
- Deux points,
- À partir de la ligne suivante, un bloc de code **indenté** (appelé la clause **if**).

```
1 name = "Alice"
2 if name == "Alice":
3     print("Bonjour Alice")
```

Instruction **else** :

Revenons à l'exemple d'Alice, regardons un code qui utilise une instruction else pour offrir un message d'accueil différent si le nom de la personne n'est pas Alice. Une clause **if** peut éventuellement être suivie d'une instruction **else**. La clause **else** est exécutée uniquement lorsque la condition de l'instruction **if** est False. Une instruction **else** pourrait être lue comme suit : « *si cette condition est vraie, exécutez ce code, **sinon** exécutez ce code.* »

Une instruction else n'a pas de condition et elle se compose toujours des éléments suivants :

- Le mot-clé **else**
- Deux points
- À partir de la ligne suivante, un bloc de code en retrait (appelé la clause **else**)



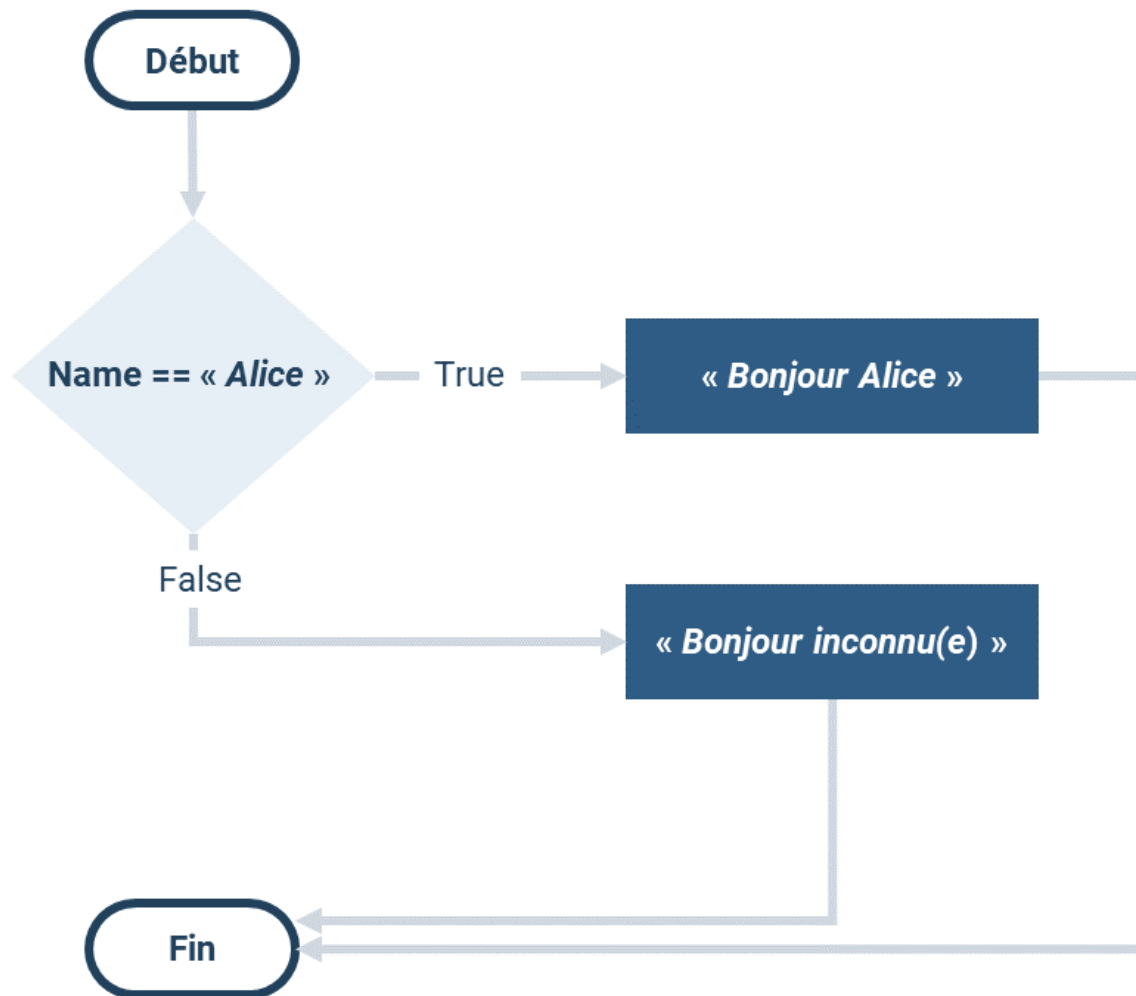


Schéma : représentation de l'instruction else

```

1 name = "Alice"
2 if name == "Alice":
3     print("Bonjour Alice")
4 else:
5     print("Bonjour inconnu(e)")
  
```

#### Instruction **elif** :

Cette fois, vous vérifiez l'âge de la personne et le programme lui dira quelque chose de différent s'il a moins de 12 ans. Bien qu'une seule des clauses if ou else soit exécutée, vous pouvez avoir un cas où vous souhaitez exécuter l'une des nombreuses clauses possibles. La déclaration **elif** est une instruction qui suit toujours une instruction **if** ou une autre instruction **elif**. Elle fournit une autre condition qui n'est vérifiée que si l'une des conditions précédentes est fausse. Une instruction **elif** pourrait être lue comme suit : « *si cette condition est vraie, exécutez ce code, **sinon si** une autre condition est vraie, exécutez ce code.* »

Dans le code, une instruction **elif** se compose toujours des éléments suivants :

- Le mot-clé **elif**,
- Une condition (c'est-à-dire une expression qui a la valeur True ou False),
- Deux points,
- À partir de la ligne suivante, un bloc de code en retrait (appelé clause **elif**).

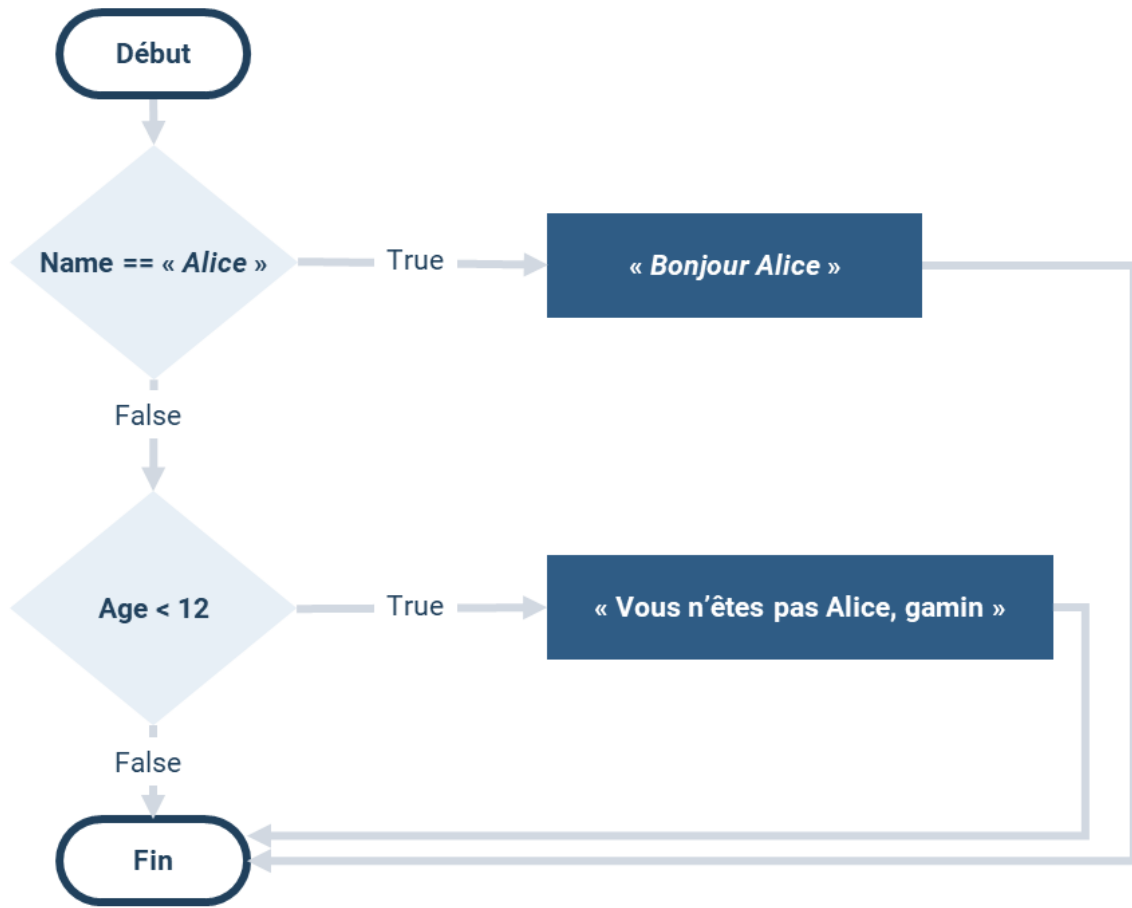


Schéma : représentation de l'instruction elif

```

1 name = "Ali"
2 age = 10
3 if name == "Alice":
4     print("Bonjour Alice")
5 elif age < 12:
6     print("Vous n'êtes pas Alice, gamin")
  
```

Notez que lorsqu'il existe une chaîne d'instructions **elif**, une seule ou aucune des instructions seront exécutées. Une fois que l'une des conditions est **True**, les autres clauses elif sont automatiquement ignorées.

#### Complément

Parfois, il y a plus de deux possibilités et nous avons besoin de plus de deux succursales. Une façon d'exprimer un calcul comme celui-ci est une **condition enchaînée** :

```

1 if x < y:
2     # INSTRUCTIONS A
3 elif x > y:
4     # INSTRUCTIONS B
5 else:
6     # INSTRUCTIONS C
  
```

Il existe plusieurs autres manières d'utiliser les structures conditionnelles pour obtenir du code complexe et plus utile.

**Exemple**

Créez une condition **if** qui imprime "Bien joué" si la note est supérieure à 15 et une condition **else** "Tu devras recopier 3 fois toutes les décimales de pi !"

**Une solution :**

```
1 note = 17
2 if note >= 15 :
3     print("Bien joué !")
4 else :
5     print("Tu devras recopier 3 fois toutes les décimales de pi !")
```

**B. Opérateurs booléens et logiques**

Il existe trois opérateurs booléens : AND, OR et NOT. Ils sont utilisés pour comparer les valeurs booléennes. Comme les opérateurs de comparaison, ils évaluent ces expressions à une valeur booléenne **True** ou **False**.

**AND**

Les opérateurs AND et OR prennent toujours deux valeurs booléennes, ils sont donc considérés comme des opérateurs binaires. L'opérateur AND évalue une expression comme True si les deux valeurs booléennes sont True ; sinon, il l'évalue comme False.

Expression	Évalue comme
True and True	True
True and False	False
False and True	False
False and False	False

**OR**

D'un autre côté, l'opérateur OR évalue une expression à True si l'une des deux valeurs booléennes est True. Si les deux sont fausses, elle est évaluée comme False.

+

Expression	Évalue comme
True or True	True
True or False	True
False or True	True
False or False	False

## NOT

Contrairement à AND et OR, l'opérateur NOT opère sur une seule valeur booléenne. L'opérateur NOT évalue simplement la valeur booléenne opposée. Tout comme l'utilisation de doubles négatifs à l'écrit ou à l'oral, vous pouvez imbriquer les opérateurs NOT.

Expression	Évalue comme
Not True	True
Not False	False

### Complément Mélanger les opérateurs booléens et les opérateurs de comparaison

Étant donné que les opérateurs de comparaison évaluent les valeurs booléennes, vous pouvez les utiliser dans les expressions avec les opérateurs booléens. Vous pouvez également utiliser plusieurs opérateurs booléens dans une expression, ainsi que les opérateurs de comparaison.

### Exemple

```

1 True and True      # retourne True
2 True and False     # retourne False
3 True or True       # retourne True
4 True or False      # retourne True
5 False or False     # retourne False
6 not True           # retourne False
7 not False          # retourne True
8
9 2+2 == 4 and not 2+2 == 5 and 2*2 == 2+2 # retourne True

```

## Exercice : Quiz

[solution n°2 p.18]

### Question 1

À quoi est évaluée une instruction de contrôle de flux ?

- ☐ True
- ☐ False
- ☐ AND
- ☐ NOT

### Question 2

Évaluez l'expression suivante (5 > 4) and (3 == 5)

- ☐ False
- ☐ True

### Question 3

Évaluez l'expression suivante not (5 > 4)

- ☐ False
- ☐ True

## Question 4

Évaluez l'expression suivante (5 > 4) or (3 == 5)

- ☐ False
- ☐ True

## Question 5

Quels sont les opérateurs de comparaison dans cette liste ?

- ☐ ==
- ☐ !=
- ☐ >
- ☐ <=
- ☐ =
- ☐ AND
- ☐ IS
- ☐ IS NOT
- ☐ OR

## Question 6

Quel signe de ponctuation doit impérativement accompagner une condition if ?

- ☐ Une virgule (,)
- ☐ Un point-virgule (;)
- ☐ Deux points (:)
- ☐ Un point d'exclamation (!)

## Question 7

Comment reconnaît-on toutes les instructions d'une structure en python ?

- ☐ Ils sont mis dans des accolades
- ☐ Ils sont mis dans des crochets ou des parenthèses
- ☐ On les reconnaît grâce à des tabulations

## V. Essentiel

En Python, ce qui fait un programme complexe et utile, c'est une suite d'instructions ordonnées et conditionnées. Le déroulement d'un programme est subordonné à des instructions qui seront exécutées et d'autres pas. Pour mettre en place cette logique en programmation, ce sont les structures conditionnelles qui sont utilisées. Les structures conditionnelles sont un moyen de vérifier la logique et le flux de votre code à l'aide de conditions. Dans votre vie quotidienne, vous prenez constamment des décisions en fonction des conditions, c'est exactement la même chose en programmation. Les instructions sont exécutées suivant des conditions tout au long du programme.

Un exemple simple : votre décision d'aller à la plage dépend du climat. Le climat est donc une condition.

Les structures conditionnelles en Python sont **if**, **else**, et **elif**. Chacune de ces structures a un rôle bien défini dans l'exécution d'un programme. L'instruction **if** vérifie si la condition qu'elle compose est vraie avant d'exécuter le code à l'intérieur. Le code à l'intérieur de la clause **else** s'exécute lorsque l'instruction conditionnelle qu'elle succède est fausse. Quant à **elif**, elle vient toujours après une instruction **if** et vérifie une autre condition.

Les structures conditionnelles peuvent être utilisées avec des opérateurs logiques. Ces opérateurs logiques permettent d'évaluer plusieurs conditions à la fois.

## VI. Auto-évaluation

### A. Exercice

Les instructions conditionnelles aident les mathématiciens et les programmeurs informatiques à prendre des décisions en fonction de l'état d'une situation. Bien qu'ils varient en termes d'utilisation et de complexité, les professionnels utilisent généralement des instructions conditionnelles pour tester des hypothèses et établir des règles à suivre par les programmes. Ainsi, à la fin de ce cours, il vous est demandé de donner une explication de vos connaissances sur les déclarations conditionnelles.

#### Question 1

[solution n°3 p.20]

Quand utilise-t-on des instructions conditionnelles ?

#### Question 2

[solution n°4 p.20]

Donnez quelques exemples concrets de déclaration conditionnelle en géométrie.

### B. Test

#### Exercice 1 : Quiz

[solution n°5 p.20]

##### Question 1

Pour évaluer une égalité en Python, on écrit "=" entre les variables à comparer.

- ☐ Vrai
- ☐ Faux

##### Question 2

L'instruction elif suit toujours une instruction if.

- ☐ Vrai
- ☐ Faux

##### Question 3

L'instruction elif peut être utilisée plus d'une fois dans un code python.

- ☐ Vrai
- ☐ Faux

##### Question 4

Avec une condition, que l'instruction soit vérifiée ou pas, le programme est exécuté.

- ☐ Vrai
- ☐ Faux

##### Question 5

Les expressions booléennes ne peuvent être représentées que par True et False.

- ☐ Vrai
- ☐ Faux

## Solutions des exercices






**Exercice p. 6 Solution n°1****Question 1**

Comment évaluer la différence entre deux variables ?

☒ a != b

☐ a == b


 En python, la différence est évaluée grâce au symbole “!=”. C’est une autre écriture du symbole “<>” qui n’est pas utilisée.

**Question 2**

Comment évaluer l’égalité entre deux variables ?

☐ a = b

☒ a == b

 Pour évaluer l’égalité entre deux variables, on utilise le symbole “==” car le symbole “=” utilisé une seule fois sert à attribuer une valeur à une variable.

**Question 3**

Quelles sont les deux valeurs booléennes qui existent en Python ?

☒ True

☐ AND

☐ OR

☐ NOT

☒ False

 En Python, comme pour d’autres langages de programmation, une expression booléenne est soit vraie ou fausse et ne peut avoir une autre valeur d’où les valeurs **True** et **False**.

**Question 4**

Quels sont les trois opérateurs booléens ?


☐ True

☐ False

☒ AND

☒ OR

☒ NOT

 Les opérateurs booléens servent à combiner des conditions afin de les préciser. On utilisera donc dans ce cas les opérateurs **AND** dans le cas où les deux conditions sont vérifiées, **OR** lorsque l’une des conditions est vérifiée et **NOT** lorsqu’aucune des deux conditions ne doit être vérifiée. True et False ne sont que les valeurs qui résultent.


### Question 5

Quels sont les opérateurs de comparaison ?

☒ ==

☐ =

☒ >=

 Le symbole “=” n’est pas un opérateur de comparaison, mais plutôt un opérateur d’affectation.

### Exercice p. 12 Solution n°2

#### Question 1


À quoi est évaluée une instruction de contrôle de flux ?

☒ True

☒ False

☐ AND

☐ NOT

 Avec **True**, le programme exécute les instructions présentes dans la structure de contrôle. Avec **False**, les instructions après la structure de contrôle sont exécutées.

#### Question 2

Évaluez l'expression suivante (5 > 4) and (3 == 5)

☒ False

☐ True


 Ici, on évalue d’abord les expressions dans les parenthèses

#### Question 3

Évaluez l'expression suivante not (5 > 4)

☒ False

☐ True


 Ici, on évalue d’abord les expressions dans les parenthèses

#### Question 4

Évaluez l'expression suivante (5 > 4) or (3 == 5)

☐ False


☒ True

 Ici, on évalue d’abord les expressions dans les parenthèses

**Question 5**

Quels sont les opérateurs de comparaison dans cette liste ?


- ☒ ==
- ☒ !=
- ☒ >
- ☒ <=
- ☐ =
- ☐ AND
- ☒ IS
- ☒ IS NOT
- ☐ OR

 Les symboles "=", "AND et OR" sont respectivement des opérateurs d'affectation et booléens.

**Question 6**

Quel signe de ponctuation doit impérativement accompagner une condition if ?


- ☐ Une virgule (,)
- ☐ Un point-virgule (;)
- ☒ Deux points (:)
- ☐ Un point d'exclamation (!)

 En python, les structures conditionnelles sont toutes suivies de deux points (:) pour signifier le début des instructions de la condition.

**Question 7**

Comment reconnaît-on toutes les instructions d'une structure en python ?

- ☐ Ils sont mis dans des accolades
- ☐ Ils sont mis dans des crochets ou des parenthèses
- ☒ On les reconnaît grâce à des tabulations

 En python, selon l'outil utilisé pour écrire du code, on remarque des tabulations qui marquent le début de chaque instruction

p. 14 Solution n°3

Une instruction conditionnelle est un ensemble de règles considérées comme valides si certaines conditions sont remplies. Elles commencent par une hypothèse et se terminent par une conclusion. Il se peut que vous utilisiez des déclarations conditionnelles dans votre vie quotidienne, car elles vous permettent de formuler des affirmations de manière logique. Sur le lieu de travail, les gens n'utilisent les déclarations conditionnelles que dans des domaines spécifiques. Elles s'appliquent aux emplois qui requièrent certaines formes de mathématiques, notamment la géométrie et la programmation informatique. Elles servent des objectifs différents selon la façon dont elles sont utilisées, mais le concept est le même dans toutes les disciplines.

p. 14 Solution n°4

En géométrie, on utilise généralement les déclarations conditionnelles pour déterminer si ce que nous pensons d'un objet ou d'une formule est un fait concret.

**Exemple :** dave pense que son polygone est un triangle. Il se souvient que si un polygone à trois côtés, alors c'est un triangle. Son polygone à trois côtés, il peut donc confirmer qu'il s'agit d'un triangle.


Vous pouvez échanger le début de la phrase, ou l'hypothèse, avec la conclusion pour créer l'inverse d'une déclaration conditionnelle. Un inverse est parfois vrai et parfois faux, mais il est toujours considéré comme une déclaration conditionnelle dans les deux sens.

**Exemple :** jo sait que si une forme a trois côtés, c'est un polygone. Elle permute l'hypothèse et la conclusion pour créer l'inverse : si une forme est un polygone, elle a trois côtés. Bien que l'inverse soit faux, c'est toujours considéré comme une instruction conditionnelle.

Exercice p. 14 Solution n°5


Question 1

Pour évaluer une égalité en Python, on écrit “=” entre les variables à comparer.

- ☐ Vrai
- ☒ Faux
-  Pour faire l'égalité entre deux variables en python, on utilise le signe “==”

Question 2


L'instruction elif suit toujours une instruction if.

- ☒ Vrai
- ☐ Faux
-  L'instruction elif permet de définir une autre condition si la première condition n'est pas vérifiée.

Question 3

L'instruction elif peut être utilisée plus d'une fois dans un code python.

- ☒ Vrai
- ☐ Faux


-  L'instruction elif peut être utilisée après une instruction if ou une première condition elif si vous prévoyez d'ajouter plusieurs conditions.

#### Question 4

---

Avec une condition, que l'instruction soit vérifiée ou pas, le programme est exécuté.

- ☐ Vrai
- ☒ Faux


-  La condition en Python permet de changer le fonctionnement du programme si une condition n'est pas respectée.

#### Question 5

---

Les expressions booléennes ne peuvent être représentées que par True et False.

- ☒ Vrai
- ☐ Faux

-  Une expression booléenne est soit vraie ou fausse et ne peut prendre aucune autre forme.