Comparaisons, booléens, tests
Cours

Contenus	Capacités attendues
Constructions élémentaires	Être capable de : ▪ Comprendre les 3 opérations booléennes (conjonction, disjonction, négation) et leurs interactions ; ▪ Ecrire un programme avec plusieurs conditions, y compris des conditions imbriquées.

Une part importante de la conception d'un programme consiste à imaginer les différents cas de figure possibles, notamment selon les entrées fournies par l'utilisateur ou les valeurs des différentes variables, et à s'assurer que le programme est adapté à chacun de ces cas. Un outil technique clé pour traduire cela dans l'écriture du programme est l'instruction de *branchement* qui rassemble plusieurs blocs de code alternatifs, chacun associé à une condition logique, et qui à chaque exécution sélectionne au plus l'un de ces blocs.

1. Conditions et branchement

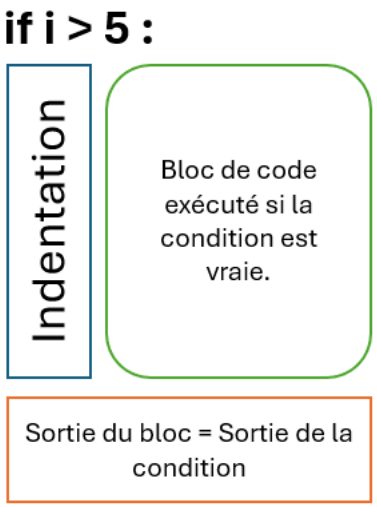
L'instruction conditionnelle **if** permet de soumettre l'exécution d'une instruction ou d'un bloc de code à une condition.

```
if n > 0 :
    print("Le nombre", n, "est positif.")
```

Une telle instruction est composée d'abord du mot clé **if** et d'une condition (ici n > 0), puis comme pour l'instruction **for** du symbole « : » suivi d'une instruction ou d'un bloc de code en retrait. Cette dernière instruction n'est exécutée que si la condition s'avère être vraie.

Les conditions peuvent être exprimée en termes de comparaison numériques entre deux expressions avec les symboles suivants :

>	Plus grand que	>=	Supérieur ou égal à	==	Égal à
<	Plus petit que	<=	Inférieur ou égal à	!=	Différent de

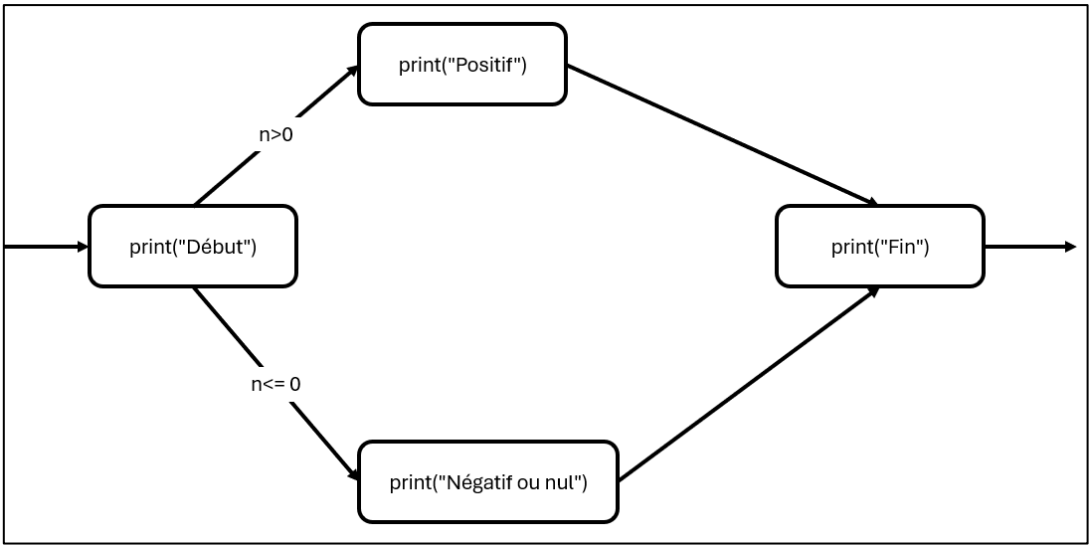


En plus d'un bloc à n'exécuter que lorsque sa condition est vérifiée, une instruction **if** peut contenir un bloc alternatif, à n'exécuter que dans le cas contraire, introduit avec le mot-clé **else**. L'instruction complète est donc constituée de deux *branches*, dont une seule sera choisie lors de chaque exécution.

```
print(" Début" )

if n > 0 :
   print("Positif" )
else :
   print("Négatif ou nul")

print("Fin" )
```

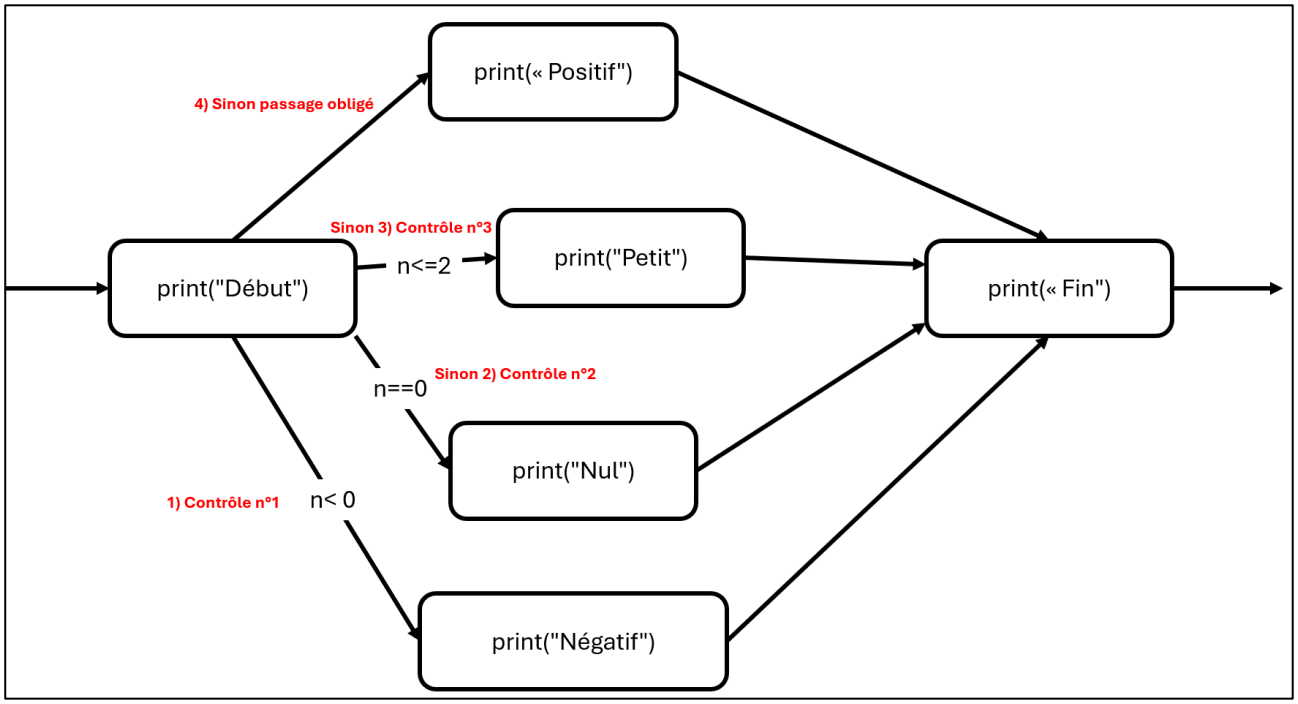


Le bloc introduit par **else** obéit aux mêmes conventions syntaxiques que les autres, à savoir être introduit par le symbole « : » et être constitué d'une séquence d'instructions en retrait.

Il est enfin possible de proposer trois branches ou plus. Après une première branche conditionnelle **if**, chaque branche suivante peut être ajoutée avec sa propre condition grâce au mot clé **elif** (contraction de else if), l'ensemble pouvant être à nouveau complété d'une ultime branche **else** sélectionnée lorsque toutes les autres ont été écartées.

```
print("Début" )
if n < 0 :
   print("Négatif" )
elif n == 0 :
   print("Nul" )
elif n <= 2 :
   print("Petit" )
else :
   print("Grand" )

print("Fin" )
```



L'ordre dans lequel apparaissent les conditions compte : seule la première branche dont la condition est vraie est exécutée. Python ne considère donc la deuxième branche que si la première n'a pas été sélectionnée, la troisième que si ni la première ni la deuxième n'ont été retenues, ainsi de suite.

Ainsi, dans le programme précédent, si la variable n a la valeur -3 alors la branche introduite par elif n <= 2 : ne sera pas prise en compte, bien que -3 soit inférieur à 2, car la première branche aura déjà été choisie.

S'entraîner sur les exercices 1,2,3 et 4.

Erreur

Les symboles = et == sont différents. Le premier est l'opérateur d'affectation d'une nouvelle valeur à une variable, tandis que le deuxième désigne l'égalité mathématique entre deux éléments. Utiliser l'affectation à la place du test d'égalité produit une erreur.

```
>>> if a = 1: print("a vaut 1")
```

```
Traceback (most recent call last):
  File "<string>", line 1
    if a = 1: print("a vaut 1")
       ^^^^^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Utiliser le test d'égalité à la place de l'affectation n'empêche pas l'exécution du programme, mais implique à priori un résultat erroné. Ainsi le programme suivant n'affiche pas 9 mais 1.

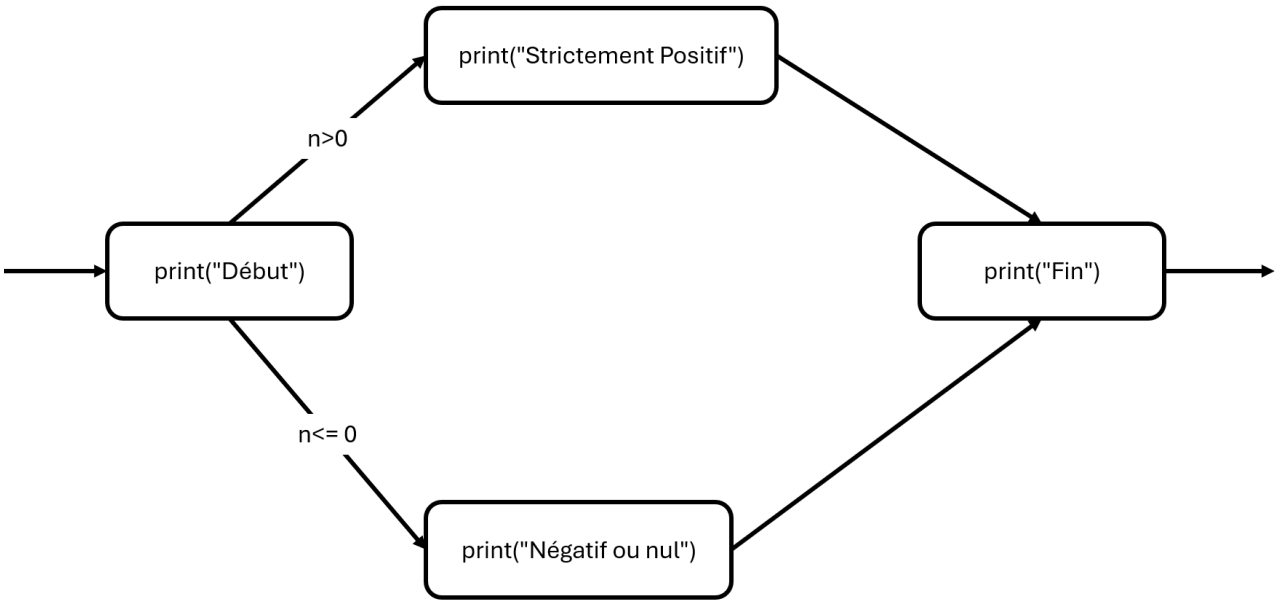
```
a = 1
for _ in range(4) :
  a == a +2
print(a)
```

2. Après le branchement : jonction

L'exécution d'une instruction **if** / **elif** / **else** consiste en l'exécution d'au plus l'un des blocs alternatifs, en fonction de la véracité des conditions.
Après cela l'exécution du programme se poursuit avec l'instruction suivante, et ceci quelle que soit la branche qui a été sélectionnée, et y compris si aucune branche ne l'a été. On dit que les différentes branches d'un branchement se rejoignent après l'instruction de branchement.
Ainsi le programme :

```
print("Début")
if n > 0 :
   print("Strictement positif")
else :
   print("Négatif ou nul")
print("Fin")
```

Après l'exécution de la branche sélectionnée le programme continuera quoi qu'il arrive avec l'instruction print("Fin"). On peut se représenter les différents scénarios d'exécutions possibles avec le diagramme suivant, appelé *graphe de flot de contrôle*.

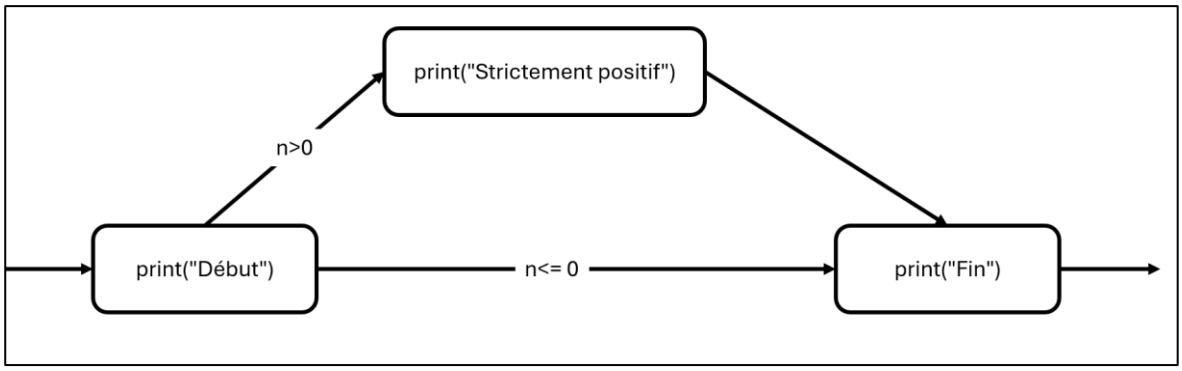


Dans ce diagramme, les cadres contiennent des instruction ou blocs d'instructions séquentielles, et les flèches indiquent les différents enchaînements possibles entre les blocs. Le branchement lui-même est représenté par les deux flèches partant de l'instruction print("Début") vers chacune des deux branches. La jonction après le branchement correspond, aux flèches partant de chacune des deux branches et aboutissant à la même instruction suivante print("Fin").

Dans le cas d'une instruction conditionnelle **if** sans branche alternative **else**, comme illustré par le programme ci-dessous :

```
print("Début")
if n > 0 :
   print("Strictement positif")
print("Fin")
```

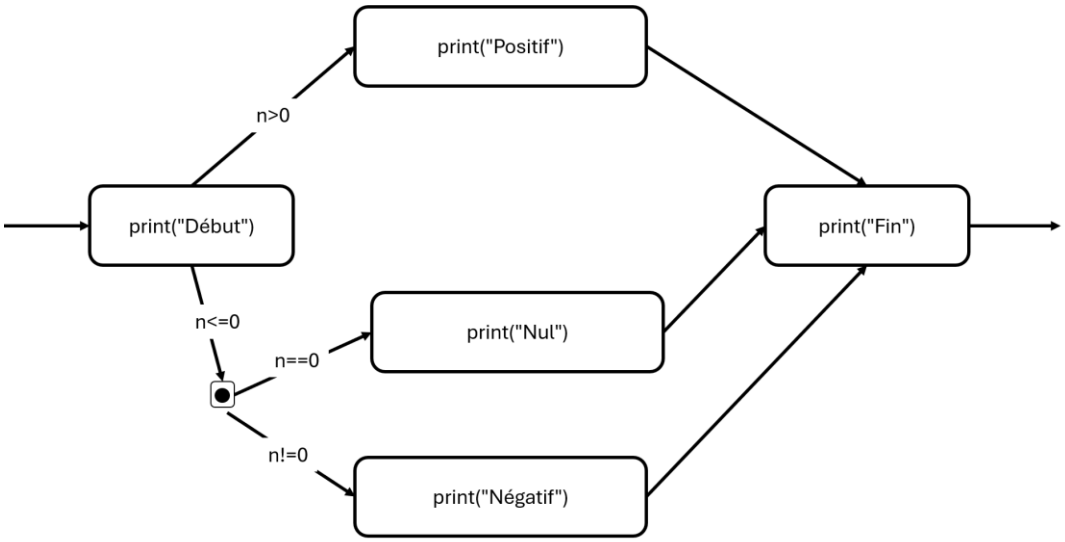
Le diagramme représente la possibilité d'un passage direct à l'instruction suivante :



La présence d'une séquence de branches alternatives **elif**, par exemple avec le programme suivant :

```
print("Début")
if n > 0 :
   print("Positif")
elif n == 0 :
   print("Nul")
else :
   print("Négatif")
print("Fin")
```

Peut se traduire par de nouveaux branchements ayant lieu uniquement lorsque le premier test a échoué :



3. Expressions booléennes

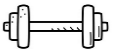
Les comparaisons et tests d'égalité sont des expressions Python ordinaires, qui comme les expressions arithmétiques produisent un résultat. Ce résultat est l'une des deux valeurs dites *booléennes*, l'une représentant le *vrai* (notée **True**) et l'autre le *faux* (notée **False**).

Comme dans le cas de la boucle **for**, c'est l'indentation qui détermine quelles instructions font partie du bloc conditionnel et quelles instructions font partie de la suite du programme qui est toujours exécutée.

```
if n > 0:
    print("Strictement")
    print("Positif")

else:
    print("Négatif")
    print("ou nul")

print("Suite du programme ...")
```

 **Activité :** Ecrire l'affichage si n vaut -1puis l'affichage si n vaut 1 (sans utiliser l'ordinateur).

```
1  if n > 0:
2      print("Strictement")
3      print("Positif")
4
5  print("Je ne suis plus dans le if")
6
7  else:
8      print("Négatif")
9      print("ou nul")
10
11 print("Suite du programme ...")
12
```

Dans le programme ci-dessus, l'arrêt de l'indentation à la ligne 5 marque la fin de l'instruction de condition. Donc l'instruction **else** n'a plus de sens, car elle n'est rattachée à aucune condition **if**.
On obtient alors l'erreur suivante :

```
Traceback (most recent call last):
  File "<string>", line 7
    else:
    ^^^^
SyntaxError: invalid syntax
```

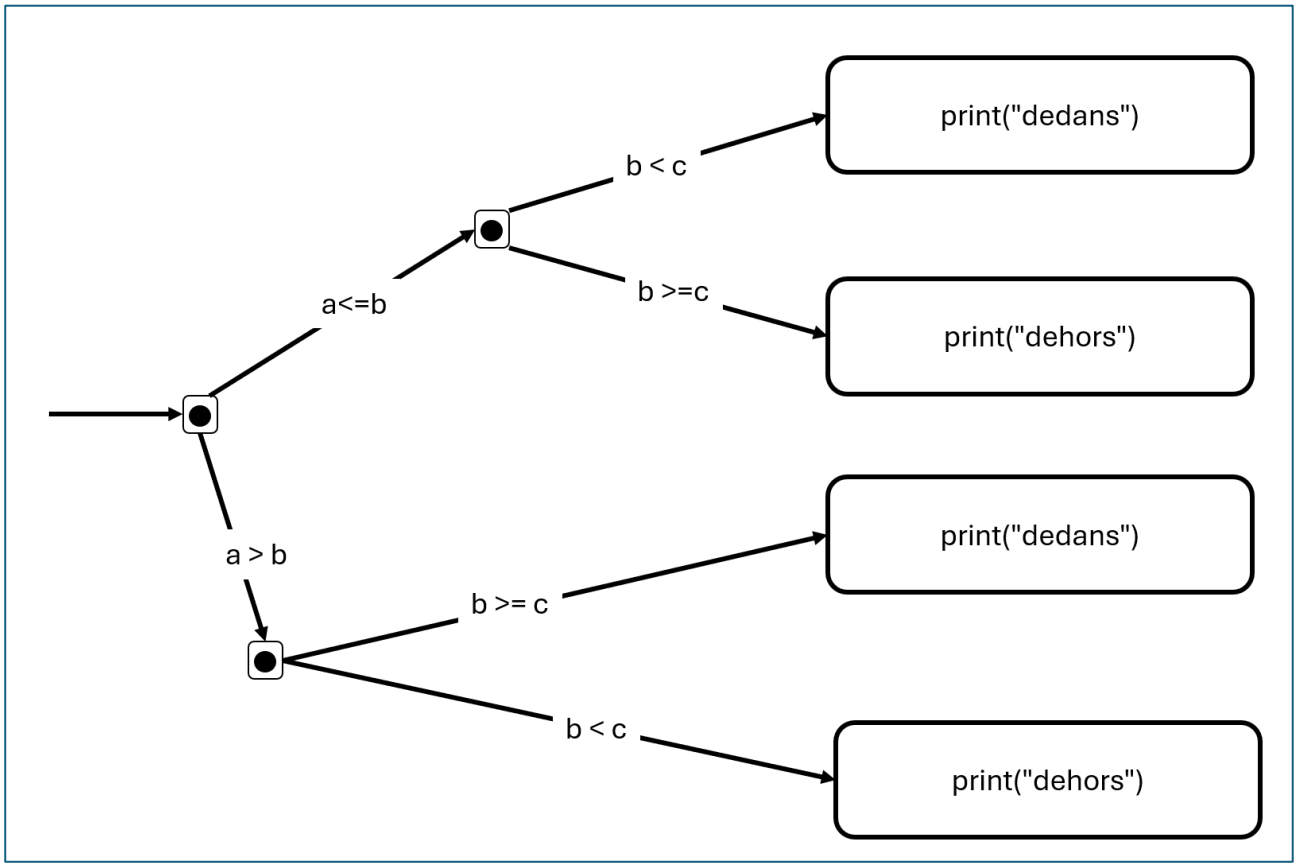
Une suite d'instructions **if** ne définit pas une alternative entre plusieurs branches, mais simplement plusieurs conditions qui seront testées et exécutées indépendamment l'une de l'autre.

4. Conditionnelles imbriquées

Chaque branche d'une instruction conditionnelle est un bloc arbitraire, pouvant lui-même contenir d'autres instructions conditionnelles point il est ainsi possible de définir des cascades de conditions logiques où formes les plus variées menant à de multiples branches.

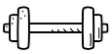
Comparaison entre **elif** et **else** : **if** :

```
if a <= b :
   if b < c : print("dedans")
   else : print("dehors")
else :
   if b >= c : print("dedans")
   else : print("dehors")
```



Le mot clé **elif** a le même effet qu'une combinaison de **else** et **if**, et le programme précédent est équivalent à la variante suivante :

```
if a <= b :
   if b < c : print("dedans")
   else : print("dehors")
elif b >= c: print("dedans")
else : print("dehors")
```

 **S’entraîner sur les exercices 5,6,7.**

 **Préparer une carte mentale de la leçon.**