ALGO 2 – INSTRUCTION CONDITIONNELLE IF

Une **instruction conditionnelle** permet d'exécuter des instructions uniquement sous une **certaine condition**.

I Une introduction aux instructions conditionnelles

Formellement, une liste d'instructions conditionnelles correspond à une liste d'instructions, données dans un certain **ordre** et telle que, chaque **instruction doit être exécutée si seulement si une certaine condition est vérifiée**. Si une certaine condition n'est pas respectée, on passe à l'instruction suivante et ainsi de suite.

Exercice 1 (....)

II Opérateurs de test

Ainsi, chaque instruction est donc conditionnelle à un test. Il faut donc comprendre comment effectuer des tests grâce à Python. Un test correspond en Python à un **booléen**, c'est-à-dire une variable Python qui vaut soit `True` (vrai) soit `False` (faux).

Entrée [1]:
```
x = 5
print(x > 2)
print(x = 3)
```

Out [1]:
```
True
False
```

On dispose des **opérateurs de test** suivants qui sont à connaître.

Test	Syntaxe Python	Exemples
Égalité	`==`	`1 == 1.0` renvoie `True` `2 == 3` renvoie `False`
Non-égalité	`!=`	`1 != 1.0` renvoie `False` `2 != 3` renvoie `True`
Inférieur strict	`<`	`1 < 2` renvoie `True` `10 < 3` renvoie `False`
Supérieur strict	`>`	`5 > 2` renvoie `True` `2 < 1` renvoie `False`
Inférieur ou égal	`<=`	`1 <= 2` renvoie `True` `3<= 2` renvoie `False`
Supérieur ou égal	`>=`	`3 >= 2` renvoie `True` `1 >= 2` renvoie `False`
Appartenance	`in`	`1 in [1,2,3]` renvoie `True` `1 in [5,6,7]` renvoie `False`

Attention à ne pas confondre `=` et `==` : l'instruction `A = 2` signifie qu'à la variable `A` est affectée la valeur `2` alors que l'instruction `A == 2` est un test d'égalité, renvoyant `True` (Vraie) si la variable `A` a pour valeur `2` et renvoyant `False` (Faux) si la variable `A` n'a pas pour valeur `2`.

Il existe d'autres mots pouvant servir dans les conditions.

Test	Syntaxe	Explications	Exemples
Et	`and`	Teste plusieurs conditions en même temps. Si toutes les conditions sont bonnes, le test est validé.	`1 == 1.0 and 2 == 3` renvoie `False`
Ou	`or`	Teste plusieurs conditions en même temps. Si au moins, l'une des conditions est bonne, le test est validé.	`1 == 1.0 or 2 == 3` renvoie `True`
Contraire	`not`	Teste l'inverse d'une condition	`not(1 == 1.0)` renvoie `False`

Exercice 2 Soit x un variable affectée d'un nombre réel.

1. Comment tester si x vaut 3 ?

Entrée [2]:
```
x == 3
```

2. Comment tester si x est différent de 5 ?

Entrée [3]:
```
x != 5
```

3. Comment tester si x est strictement supérieur à 10 ?

Entrée [4]:
```
x > 10
```

4. Comment tester si x est dans $[2, 7]$?

Entrée [5]:
```
x >= 2 and x<= 7
```

5. Comment tester si x est dans $]-\infty, -5[\cup[2, +\infty[$?

Entrée [6]:
```
x < -5 or x >=2
```

III Instruction if

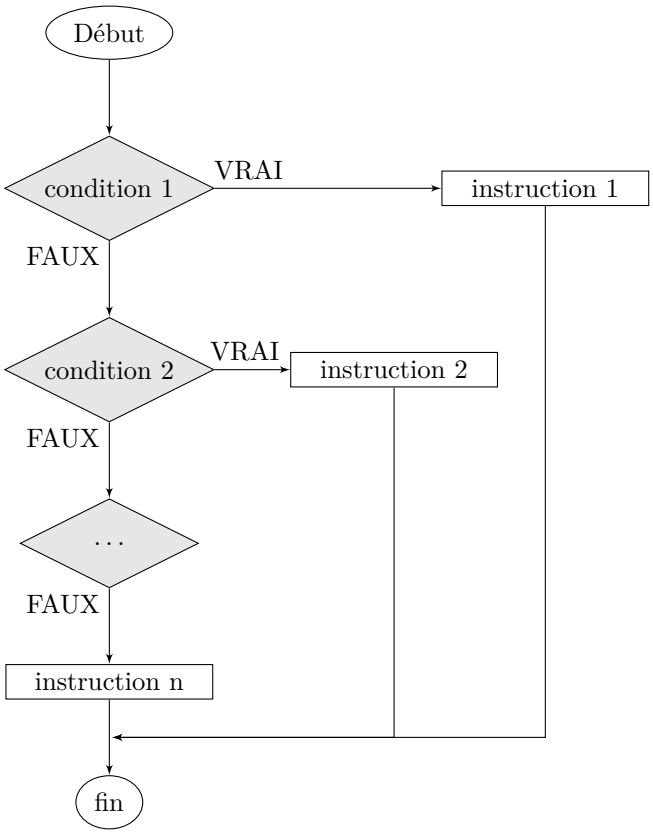
Nous allons voir comment écrire un programme Python comportant une instruction conditionnelle du type **si ... alors ...**

Algorithme

Si condition 1 **alors**
    Faire l'instruction 1
Sinon, si condition 2 **alors**
    Faire l'instruction 2
...
Sinon
    Faire l'instruction n
Fin du Si

Programme python

```
if condition 1 :
    instruction 1
elif condition 2 :
    instruction 2
elif condition 3 :
    instruction 3
...
else :
    instruction n
```



Entrée [7]:
```
# Exemple
x = 17  # on peut changer x ici

if x > 15:
    print("x est strictement supérieur à 15")
elif x >= 11:
    print(" x est entre 11 et 15")
else :
    print(" x est inférieur ou égal à 10")
```

Out [7]:
```
x est strictement supérieur à 15"
```

IV Lecture de programme

(i) Qu'affiche le programme suivant ?

```
x = 4
f = -5*x + 10
if f > 0 :
    print("f(x)>0")
else :
    print("f(x)<0")
```

```
Il affiche "f(x)<0"
```

(ii) Que renvoie le programme si `x=5` et `y=5` ? si `x=6` et `y=8` ?

```
N = x**2 + y**2
if N == 100 :
    print("Gagné")
else :
    print("Perdu")
```

```
Pour x=5 et y=5, le programme affiche Perdu.
Pour x=6 et y=8, le programme affiche Gagné.
```

(iii) Pour chaque valeur de `x` donnée, indiquer ce que le programme affiche.

```
#On suppose la valeur de x connue
if x > 0 :
    print("x est strictement positif")
elif x == 0 :
    print("x est nul")
else :
    print("x est strictement negatif")
```

x	Affichage
-4	x est strictement negatif
125	x est strictement positif
0	x est nul

(iv) Pour chaque valeur de `x` donnée, indiquer ce que le programme affiche.

```
#On suppose la valeur de x connue
import numpy as np
if x > 0 :
    print(np.log(x))
else :
    print(5*x+2)
```

x	Affichage
-3	-13
np.e	1
0	2

(v) Pour chaque valeur initiale de a donnée, indiquer la nouvelle valeur de a à la fin du programme.

```
#On suppose la valeur de a connue
if a != 1 :
    a = 1/(a-1)
else :
    print("Pas de division par 0")
```

a	Affichage
2	1
-4	-1/5
1	Pas de division par 0

V Exercices

Exercice 3 Écrire un programme qui, étant donné un réel a, remplace le contenu de la variable a par $\frac{1}{a}$ et affiche la nouvelle valeur lorsque a ne vaut pas 0, et sinon renvoie un message d'erreur.

Entrée [8]:
```
a = 4
if a != 0 :
    a = 1/a
    print(a)
else :
    print("Attention, pas de division par zéro !")
```

Out [8]:
```
0.25
```

Exercice 4 Écrire un programme qui, étant donné la valeur t de la température de l'eau, renvoie le message "Merci, sans façon" si la température est inférieure strictement à 18 degrés, qui renvoie le message "Demain, peut-être" si la température est comprise entre 18 (inclus) et 22 (exclu) degrés, et qui renvoie le message "Allez" si la température est supérieure ou égale à 22 degrés.

Entrée [9]:
```
t = 14
if t < 18 :
    print("Merci, sans façon")
elif t >= 18 and t<22 :
    print("Demain, peut-être")
else :
    print("Allez")
```

Out [9]:
```
Merci, sans façon
```

Exercice 5 On considère la fonction

$$f : x \mapsto \begin{cases} 2x-1 & \text{si } x \leqslant 2 \\ \ln(x) & \text{si } x > 2 \end{cases}$$

Écrire un programme qui, étant donné un réel x, affiche la valeur de $f(x)$. *On pensera à charger la bibiothèque numpy. On ne demande pas de définir une fonction.*

Entrée [10]:
```
import numpy as np
x = 2
if x <= 2 :
    y = 2*x-1
    print("La valeur de f(x) est", y)
else :
    y=np.log(x)
    print("La valeur de f(x) est", y)
```

Out [10]:
```
La valeur de f(x) est 3
```

Exercice 6 On considère un dé à six faces. Écrire un programme réalisant une simulation d'un lancer de dé. Celui-ci doit afficher "Gagné" si on obtient 1,3 ou 5 et "Perdu" sinon. *Pour simuler un lancer de dé, on chargera la bibliothèque numpy.random grâce à l'instruction* `import numpy.random as rd` *et on utilisera la commande* `rd.randint(a,b)` *qui renvoie un nombre entier aléatoire entre* `a` *et* `b-1`.

Entrée [11]:
```
import numpy.random as rd
valeurde = rd.randint(1,7) #on simule un lancer de dé à six faces

if valeurde == 1 or valeurde == 3 or valeurde == 5 :
    print("La valeur de dé vaut", valeurde, "On a gagné")
else :
    print("La valeur de dé vaut", valeurde, "On a perdu")
```

Out [11]:
```
La valeur de dé vaut 1 On a gagné
```

Exercice 7 Écrire un programme qui simule le lancer de deux dés et affiche "Gagné" si on obtient 7 en ajoutant les chiffres lus sur chaque dé et affiche "Perdu" sinon.

Entrée [12]:
```
import numpy.random as rd
de1 = rd.randint(1,7) #Simulation du premier dé
de2 = rd.randint(1,7) #Simulation du deuxième dé

if de1 + de2 == 7:
    print("Gagné")
else :
    print("Perdu")
```

Out [12]:
```
Gagné
```

Exercice 8 Écrire une suite d'instructions en langage Python qui, étant donnés des nombres a et b, affiche lequel est le plus petit des deux (par exemple « a est le plus petit ») ou s'il y a égalité.

Entrée [13]:
```
a = 1
b = 2

if a < b :
    print("a est le plus petit")
elif a == b :
    print("a et b sont égaux")
else :
    print("b est le plus petit")
```

Out [13]:
```
a est le plus petit
```

Exercice 9 Écrire un programme prenant en entrée trois nombres a, b, c (avec $a \neq 0$) et affichant les éventuelles solutions *réelles* de l'équation $ax^2 + bx + c = 0$.

Entrée [14]:
```
import numpy as np

#Equation 1
a = 2
b = -6
c = 4

Delta = b ** 2 - 4 * a * c
if Delta > 0 :
    x1= (-b - np.sqrt(Delta)) / (2 * a)
    x2= (-b + np.sqrt(Delta)) / (2 * a)
    print("L'équation admet deux solutions réelles données par", x1, "et", x2)
elif Delta == 0 :
    x0= (-b) / (2 * a)
    print("L'équation admet une solution réelle donnée par", x0)
else :
    print("L'équation n'admet pas de solution réelle")
```

Out [14]:
```
L'équation admet deux solutions réelles données par 1.0 et 2.0
```

Exercice 10 Il est possible de réaliser des **divisions euclidiennes** (divisions entières, avec reste) en Python à l'aide des commandes suivantes :

- `a // b` : quotient
- `a % b` : reste

Par exemple, dans la division euclidienne de 17 par 5, le quotient vaut 3 et le reste 2 car

$$17 = 5 \times 3 + 2.$$

1. En déduire une façon de tester si un entier n est pair.

Entrée [15]:
```
n = 4

if n % 2 == 0 :
    print("L'entier n qui vaut", n, "est pair")
else :
    print("L'entier n qui vaut", n, "est impair")
```

Out [15]:
```
L'entier n qui vaut 4 est pair
```

2. Écrire un programme qui, étant donnée une année, affiche `True` si celle-ci est bissextile et affiche `False` sinon. Une année est bissextile (366 jours) si l'un des deux cas suivants se produit : soit l'année est divisible par 4 et non divisible par 100, soit l'année est divisible par 400. Sinon, elle est non-bissextile (365 jours). Par exemple, les années 2020 et 2000 sont bissextiles, mais les années 2022 et 2100 ne le sont pas.

Entrée [16]:
```
annee = 2020

if (annee % 4 == 0 and annee % 100 != 0) or annee % 400 == 0 :
    print("L'annee", annee, "est bissextile")
else :
    print("L'annee", annee, "n'est pas bissextile")
```

Out [16]:
```
L'annee 2020 est bissextile
```