Initiation à la programmation en Python et en JavaScript

Cours introductif pour un démarrage rapide

Stéphane Perret

Version 3.000

Table des matières

1 Introduction

1.1 À propos de Python

Je cite le livre «Apprendre à programmer avec Python»qui est librement téléchargable à partir de `http://inforef.be/swi/`.

Python est un langage portable, dynamique, extensible, gratuit, qui permet (sans l'imposer) une approche modulaire et orientée objet de la programmation. Python est développé depuis 1989 par Guido van Rossum et de nombreux contributeurs bénévoles.

Voici un aperçu de quelques caractéristiques du langage Python.

1. Python est gratuit et portable : il existe sur Linux, Unix, Mac et Windows.
2. La syntaxe de Python permet d'écrire des programmes compacts et lisibles.
3. Python est un langage qui continue à évoluer, soutenu par une communauté d'utilisateurs dont la plupart sont des supporters du logiciel libre.
4. Python est un langage de choix pour traiter le XML.

1.2 À propos de JavaScript

JavaScript est utilisé dans des millions de pages web afin d'améliorer leur conception. Il s'agit d'une couche de programmation supplémentaire qui vient s'ajouter au langage HTML [1]. Le code HTML est le langage de base que toute page Internet se doit d'utiliser : en plus de son rôle proche d'un traitement de texte, ce langage permet de surfer grâce aux liens hypertextes. Quant à JavaScript, il a été conçu pour donner plus d'interactivité aux pages HTML. Le mot script indique qu'il s'agit d'un langage de programmation simplifié qui s'exécute en local sur l'ordinateur qui est en train de lire la page web. Ce langage, comme l'HTML, ne nécessite l'achat d'aucune licence pour pouvoir l'utiliser.

Initialement, JavaScript a été développé par Netscape, mais maintenant la plupart des explorateurs permettant de naviguer sur Internet sont compatibles avec JavaScript.

Voici un aperçu de quelques caractéristiques du langage JavaScript.

1. JavaScript est déjà installé sur la grande majorité des navigateurs Internet.
2. JavaScript est gratuit et portable : il existe sur Linux, Unix, Mac et Windows.
3. JavaScript livre aux concepteurs de pages web un outil de programmation avec une syntaxe élémentaire.
4. JavaScript permet l'utilisation de textes dynamiques dans une page web : on peut par exemple afficher la date d'aujourd'hui.
5. JavaScript peut réagir à un événement particulier. Par exemple quand une page a terminé son chargement ou quand un utilisateur clique sur un élément HTML.

1.3 JavaScript et Java sont deux langages différents

Il est important de bien préciser que même si les noms sont très proches, Java et JavaScript sont deux langages bien distincts. Java est développé par Sun Microsystems et est un langage de programmation bien plus puissant et complexe que JavaScript. Java peut se comparer au langage `C++`.

1. HyperText Markup Language

2 Le programme «Hello World !»

Il s'agit du premier programme qu'un utilisateur crée lors d'une prise de contact avec un langage de programmation inconnu. Il consiste à afficher le texte «Hello World !».

2.1 La version JavaScript

Voici les étapes à respecter, sous Windows, afin de pouvoir faire fonctionner ce programme.

1. Ouvrir un éditeur de texte tel que `Notepad` ou `Notepad++`. Il est important de ne pas utiliser de fonctions avancées de la mise en page et de toujours sauvegarder le document en mode texte.

2. Créer les deux fichiers suivants dans le même répertoire.

   Le premier sera un fichier `HTML`, le deuxième sera le code JavaScript.

   Fichier `page.html`

   ```
   <html>
   <body>
   <script src="hello.js"></script>
   </body>
   </html>
   ```

   Fichier `hello.js`

   ```
   document.write("Hello World!")
   ```

3. Double-cliquer sur l'icône du fichier `page.html` afin d'exécuter l'explorateur Internet par défaut. Le script `hello.js` sera ainsi automatiquement interprété.

La balise `<html> ... </html>` indique à l'explorateur Internet que le fichier est écrit en langage `HTML`. Quant à la balise `<body> ... </body>`, elle indique qu'il s'agit du corps du document `HTML`.

Enfin, la balise `<script src="hello.js"></script>` indique à l'explorateur Internet qu'il va falloir interpréter en langage JavaScript le contenu du ficher `hello.js`.

Dans le fichier `hello.js`, la commande JavaScript `document.write` permet d'écrire `Hello World !` sur la page `page.HTML` affichée par l'explorateur Internet.

Les commentaires en JavaScript et en HTML

Lorsqu'on programme, il faut toujours bien commenter ce que le programme fait afin qu'il soit rapidement compréhensible. Des entreprises comme *Google* n'engagent que des programmeurs qui ont un code clair et lisible (donc bien documenté). Or, un commentaire est facile à mettre : tout le texte qui suit `//` n'est pas interprété par JavaScript.

Néanmoins, `//` ne fonctionne que pour une ligne. Si on veut écrire un commentaire de plusieurs lignes, on peut aussi utiliser `/*` pour commencer le commentaire et `*/` pour le finir. Voici les deux façons d'écrire un commentaire sur plusieurs lignes.

```
// Ceci est un commentaire
// sur plusieurs lignes
```

```
/* Ceci est un commentaire
sur plusieurs lignes */
```

Attention à ne pas confondre avec les balises de commentaires du langage qui sont les suivantes et qui fonctionnent indépendamment du nombre de lignes.

```
<!-- Ceci est un commentaire en HTML -->
```

2.2 La version Python

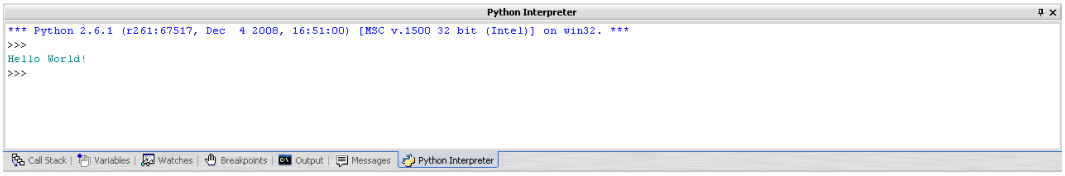
Voici les étapes à respecter, sous Windows, afin de pouvoir faire fonctionner ce programme.

1. Installer Python. Sous Windows, on utilisera `PortablePython`. On peut aussi utiliser `MovableIDLE`. Ces deux environnements ont l'énorme avantage d'être portables, c'est-à-dire qu'ils peuvent être installés sur une clé USB et ainsi permettre leur exécution sur n'importe quel PC.
2. Ouvrir l'éditeur `PyScripter-Portable.exe`.
3. Taper le texte suivant dans la fenêtre principale.

```
print("Hello World!")
```

4. Sauvegarder le document sous le nom `hello.py` et cliquer sur l'icône verte en forme de bouton *play* (tout à droite sur l'image ci-dessous).

La commande `print` permet d'écrire `Hello World !`, en-dessous des symboles `>>>` qui signalent l'interprétation du script (programme), dans la fenêtre appelée `Python Interpreter` en bas de l'écran.



Alternative. Lorsqu'on désire tester une commande en `Python`, on peut directement la taper dans la fenêtre `Python Interpreter`. Le résultat est ainsi instantané.

Les commentaires en Python

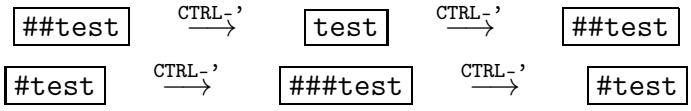
Dans `Python`, un commentaire est facile à mettre : tout le texte qui suit `#` n'est pas interprété.

Le programme `PyScripter` permet de mettre plusieurs lignes en commentaires. Pour cela, il suffit de les sélectionner et d'utiliser le raccourci clavier `CTRL-'` (la touche à côté du 0). Ce raccourci agit comme un interrupteur et commente/décommente les lignes sélectionnées. Il est réglé pour ajouter ou retirer une paire `##` et n'interfère ainsi pas avec les commentaires simples `#`.

```
# Ceci est un commentaire
# sur plusieurs lignes
```

```
## Ceci est un commentaire
## sur plusieurs lignes
```

L'avantage de la deuxième version est qu'une pression sur `CTRL-'` permet d'enlever le commentaire et une deuxième pression permet de le remettre.



3 Les variables en informatique

Les variables sont les éléments clés d'un langage de programmation. On conçoit une variable comme un tiroir contenant une information. La valeur de la variable peut être appelée à changer. Pour se référer à la variable, on y donne un nom.

Les noms de variables obéissent aux règles usuelles suivantes.

1. Leur nom doit commencer par une lettre, aucune lettre accentuée ou caractère spécial n'est permis, sauf «_».
2. Il faut faire attention aux majuscules et aux minuscules. Par exemple `X` et `x` sont deux noms de variables différents (cette contrainte est générale à toute programmation HTML).

En langage JavaScript et en Python, une variable est automatiquement créée quand on lui assigne une valeur. Par exemple, la commande

```
nomVariable = valeurVariable
```

assigne la valeur `valeurVariable` dans le *tiroir* nommé `nomVariable`.

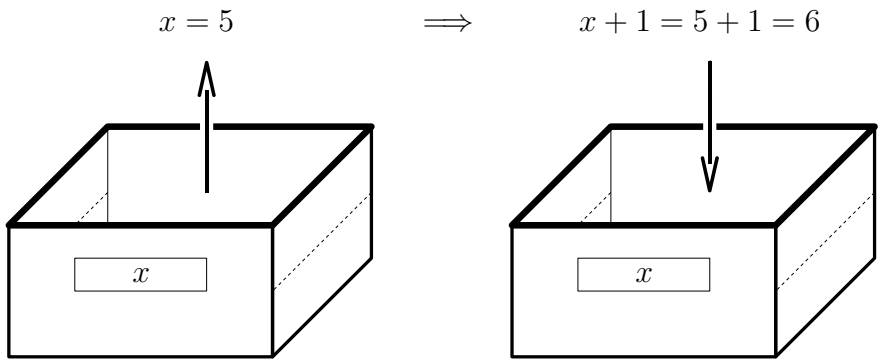
3.1 L'assignation de variables en informatique

Attention, contrairement au symbole = en mathématique, dans un langage de programmation le symbole `=` ne doit se lire que dans un sens. Ci-dessus, il faut comprendre que le tiroir appelé `nomVariable` contient l'objet `valeurVariable`. Il faut bien faire la différence entre le tiroir (le nom de la variable) et son contenu (la valeur de la variable).

Ainsi, si dans le contenu, on retrouve le nom d'une variable, il faut penser qu'il s'agit du contenu de cette variable. Voici un exemple qui illustre bien ce phénomène.

```
x = 5
x = x + 1
```

La première ligne crée la variable `x` (si elle n'est pas déjà créée) et y assigne la valeur `5`. Puis la deuxième ligne prend la valeur qui se trouve dans la variable `x`, y ajoute 1 et assigne le résultat dans la variable `x`. Voici l'image mentale de la deuxième ligne qu'il faut avoir :



Remarque

Les assignations de variables se font dans la mémoire de l'ordinateur. À moins d'utiliser une commande d'impression (`document.write` en JavaScript ou `print` en Python), on ne voit pas les assignations se produire. En Python, il y a exception dans la fenêtre `Python Interpreter`, où la commande `x` permet d'afficher la valeur de `x`.

3.2 Variables numériques

Il y a deux types de variables numériques en JavaScript et en Python : les entiers et les nombres à virgule flottante. En Python, il y a un troisième type : les entiers longs (qui peuvent être arbitrairement grands).

Fonctions de conversion

Les fonctions `parseInt` et `parseFloat` sont des fonctions JavaScript qui permettent de convertir une variable en un nombre entier ou à virgule flottante respectivement. Leurs équivalents Python sont `int` et `float`.

Rappels sur la division entière (ou euclidienne)

Lorsqu'on effectue la division entière de deux nombres entiers a et b, on obtient un reste r et un quotient q tels que $a = q \cdot b + r$ et $0 \leqslant r < |b|$.

Par exemple, la division de 7 par 2 donne un quotient de 3 et un reste de 1.

3.2.1 Opérateurs mathématiques

Pour Python, il faut utiliser la commande `import math` afin d'accéder à toutes les commandes ci-dessous qui commencent par `math.` (en minuscule). Pour JavaScript, il suffit d'inscrire `Math.` (avec la majuscule) avant ces commandes.

Nom	Maths	JavaScript	Python 2.6
addition	$a + b$	`a + b`	`a + b`
soustraction	$a - b$	`a - b`	`a - b`
multiplication	$a \cdot b$	`a * b`	`a * b`
division	$\frac{a}{b}$	`a / b`	`float(a) / float(b)`
puissance	a^b	`Math.pow(a,b)`	`math.pow(a,b)` ou `a**b`
partie entière	$\text{ent}(a)$	`Math.floor(a)`	`math.floor(a)`
division entière			
· quotient	$\lfloor \frac{a}{b} \rfloor$	`Math.floor(a/b)`	`a / b`
· reste	$a \pmod b$	`a % b`	`a % b`

3.2.2 Fonctions mathématiques

JavaScript		Python	
`Math.abs(x)`	`Math.acos(x)`	`abs(x)`	`math.acos(x)`
`Math.asin(x)`	`Math.atan(x)`	`math.asin(x)`	`math.atan(x)`
`Math.ceil(x)`	`Math.cos(x)`	`math.ceil(x)`	`math.cos(x)`
`Math.exp(x)`	`Math.floor(x)`	`math.exp(x)`	`math.floor(x)`
`Math.log(x)`	`Math.pow(x,y)`	`math.log(x)`	`math.pow(x,y)`
`Math.round(x)`	`Math.sin(x)`	`math.round(x)`	`math.sin(x)`
`Math.sqrt(x)`	`Math.tan(x)`	`math.sqrt(x)`	`math.tan(x)`

3.2.3 Constantes mathématiques

JavaScript		Python	
`Math.PI`	`Math.E`	`math.pi`	`math.e`

3.3 Les chaînes de caractères

On peut assigner à une variable une chaîne de caractère. Il s'agit d'un texte entouré de guillemets (" ou ', sans mélange).

```
texte = "c'est génial !"
```

```
texte = 'c\'est génial !'
```

En JavaScript, on doit utiliser le caractère spécial \ pour afficher des caractères qui prêtent à confusion (\' pour l'apostrophe et \\ pour afficher \). En Python, il faut aussi utiliser \' pour l'apostrophe.

3.3.1 Concaténation de chaînes de caractères

L'opérateur + permet aussi de concaténer deux chaînes de caractères. Par exemple `"bon"+"jour"` donne `"bonjour"`.

En JavaScript, remarquons que `21+" pommes"` donne `"21 pommes"`, car elle transforme le nombre `21` en chaîne de caractères.

Par contre en Python, une telle opération provoque une erreur. Il faut d'abord transformer le nombre `21` en chaîne de caractères à l'aide de la fonction `str`. La commande sera ainsi `str(21) + "pommes"`.

3.3.2 Affichage des caractères accentués et spéciaux

En Python, il faut commencer le programme par la ligne suivante, afin que les caractères accentués et spéciaux soient bien reconnus.

```
# -*- coding: cp1252 -*-
```

3.3.3 Les sauts à la lignes et les espaces

En JavaScript, on effectue un saut à la ligne en utilisant `<br />` (ou `<br>` qui n'est pas le dernier standard mais qui marche encore). En JavaScript, lorsqu'il y a plusieurs espaces à la suite, seul une espace est reportée (en fait cela provient du langage `HTML`), mais les espaces insécables s'ajoutent en utilisant ` ` (une autre commande `HTML`).

En Python, on effectue un saut à la ligne à l'affichage en utilisant `\n`. On peut effectuer un saut à la ligne dans le code en utilisant \. La commande `\n\` permet de faire un saut de ligne à l'affichage et dans le code. De plus, les espaces dans les chaînes de caractères sont tous reportés à l'affichage. En Python, le code suivant

```
Salut = "Ceci est une chaîne plutôt longue\n\
contenant plusieurs lignes \
de texte (Ceci fonctionne\n de la même façon en C/C++).\n\
    Notez que les blancs en début\n de ligne sont significatifs.\n"
print(Salut)
```

donne ceci à l'affichage

```
Ceci est une chaîne plutôt longue
contenant plusieurs lignes de texte (Ceci fonctionne
 de la même façon en C/C++).
    Notez que les blancs en début
 de ligne sont significatifs.
```

3.4 Les listes (aussi appelées tableaux)

On peut assigner à une variable une liste d'éléments. Il s'agit d'une énumération d'éléments séparés par une virgules et encadrée par des crochets.

```
liste = ["Alain", "Olivier", "Patrick", "Paul"]
```

On peut afficher le premier élément de la liste à l'aide de la commande

JavaScript `document.write(liste[0])` Python `print(liste[0])`

Python permet de créer des listes de nombres grâce à la commande `range`. Voici trois exemples de tels listes.

```
liste1 = range(1,15)
liste2 = range(20,100,10)
liste3 = range(99,70,-2)
```

Voici les définitions équivalentes de ces listes qui s'arrêtent avant la deuxième borne.

```
liste1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
liste2 = [20, 30, 40, 50, 60, 70, 80, 90]
liste3 = [99, 97, 95, 93, 91, 89, 87, 85, 83, 81, 79, 77, 75, 73, 71]
```

En JavaScript, il faut programmer soi-même la fonction `range`. De plus, avant de modifier une liste, appelée `liste`, élément par élément (plutôt qu'avec les crochets), il faut taper la ligne `liste = Array(n)` où `n` est le nombre d'élément de la liste.

3.5 Opérateurs de comparaison et variables booléennes

Un opérateur de comparaison associe à une proposition une valeur de vérité : (`true` pour vrai et `false` pour faux). Cette valeur de vérité peut être assignée à une variable, dite *booléenne*. L'opérateur de négation renverse les valeurs de vérité (vrai devient faux et vice-versa).

Nom	JavaScript	Python
est égal à	`a == b`	`a == b`
n'est pas égal à	`a != b`	`a != b`
est plus grand que	`a > b`	`a > b`
est plus petit que	`a < b`	`a < b`
est plus grand ou égal à	`a >= b`	`a >= b`
est plus petit ou égal à	`a <= b`	`a <= b`
ET logique	`a && b`	`a and b`
OU logique	`a || b`	`a or b`
négation (NOT)	`!a`	`not a`

Le «ou exclusif»

En JavaScript le XOR (ou exclusif) paraît manquant, mais on peut utiliser `^` qui livre 1 (au lieu de *true*, mais qui est considéré comme tel) ou 0 (au lieu de *false*, mais qui est considéré comme tel).

En Python, le XOR (ou exclusif) est donné par l'opérateur `^`.

Cas particuliers

Pour JavaScript, le test `2 == "2"` donne *vrai*. Par contre `2 === "2"` donne *faux*.

Pour Python, le test `1 == 1.0` donne *vrai*. Par contre `1 is 1.0` donne *faux*.

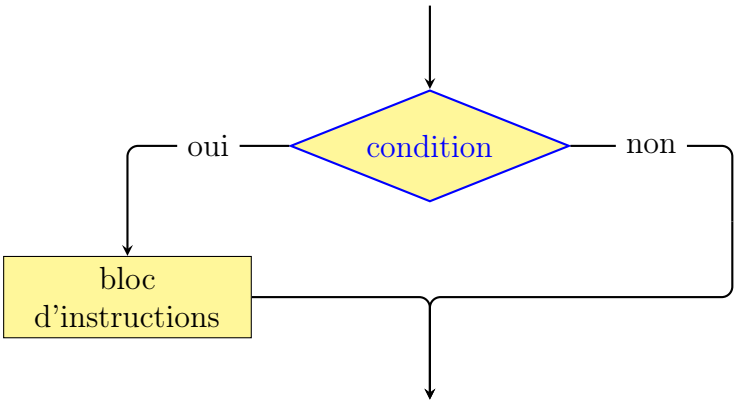
4 Les structures de contrôles

En programmation, on a besoin d'effectuer différentes actions selon l'état en cours de l'exécution du programme. Les structures de contrôles sont là pour résoudre ces difficultés.

4.1 La commande «si . . ., alors . . .»

4.1.1 Version simple

si (condition) alors (bloc d'instructions).



Code JavaScript

```
if (condition)              //parenthèses obligatoires
    {
    //code qui sera exécuté si la condition est vraie
    //pas besoin d'indenter (décaler à droite), mais
    //les accolades sont nécessaires s'il y a
    //plusieurs lignes de code
    }
```

Code Python

```
if condition :            ##double-points obligatoire
    ##code qui sera exécuté si la condition est vraie
    ##ce code doit être indenté (décalé à droite)
    ##dès que l'indentation stoppe, on sort du "if"
```

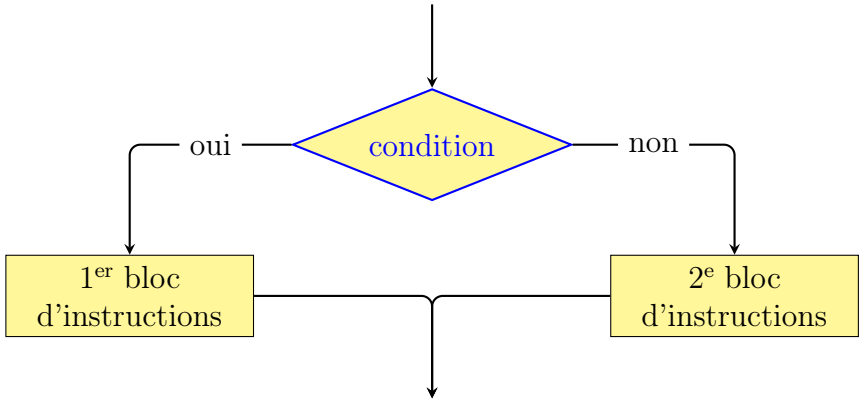
Exemple

```
x = 5
if (x == 5)
    {
    document.write("x vaut 5. ")
    document.write("Si, si ! ")
    }
document.write("Hors du if")
```

```
x = 5
if x == 5 :

    print("x vaut 5. ")
    print("Si, si ! ")

print("Hors du if")
```

4.1.2 Version usuelle

si (condition) alors (1er bloc d'instructions), sinon (2e bloc d'instructions).



Code JavaScript

```
if (condition)                  //parenthèses obligatoires
    {
    //code qui sera exécuté si la condition est vraie
    }
else
    {
    //code qui sera exécuté si la condition est fausse
    }
```

Code Python

```
if condition :                 ##double-points obligatoire
    ##code qui sera exécuté si la condition est vraie
else :
    ##code qui sera exécuté si la condition est fausse
```

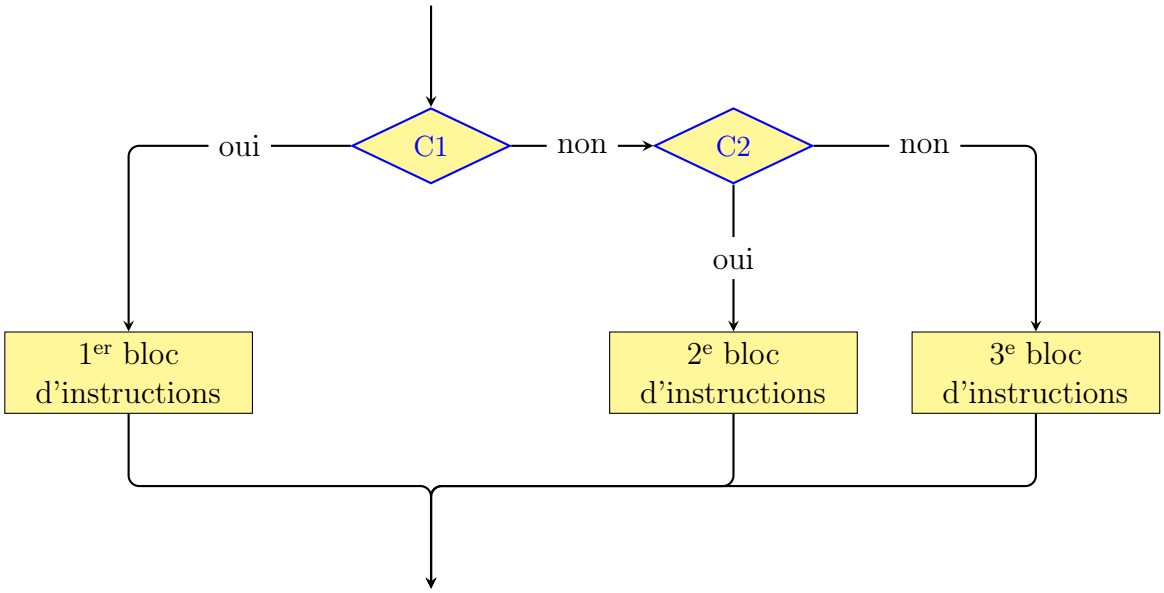
Exemple

```
x = 5
if (x == 5)
    {
    document.write("x vaut 5. ")
    }
else
    {
    document.write("x ne vaut pas 5. ")
    }
document.write("Hors du if")
```

```
x = 5
if x == 5 :

    print("x vaut 5. ")

else :

    print("x ne vaut pas 5. ")

print("Hors du if")
```

4.1.3 Version sophistiquée

Si (C1), alors (1er bloc d'instructions), sinon si (C2), alors (2e bloc d'instructions), sinon (3e bloc d'instructions). On peut bien sûr ajouter d'autres conditions «sinon si», ce qui ne change pas le schéma (à part le prolonger sur la droite).



Code JavaScript

```
if (condition1)
    {
    //code qui sera exécuté si la condition1 est vraie
    }
else if (condition2)
    {
    //code qui sera exécuté si la condition1 est fausse
    //et si la condition2 est vraie
    }
else
    {
    //code qui sera exécuté lorsque toutes les conditions
    //précédentes sont fausses
    }
```

Code Python

```
if condition1 :
    ##code qui sera exécuté si la condition1 est vraie
elif condition2 :
    ##code qui sera exécuté si la condition1 est fausse
    ##et si la condition2 est vraie
else :
    ##code qui sera exécuté lorsque toutes les conditions
    ##précédentes sont fausses
```

Exemple en JavaScript

```
x = 5
if (x == 5)
    {
    document.write("x vaut 5. ")
    }
else if (x == 6)
    {
    document.write("x vaut 6. ")
    }
else
    {
    document.write("x ne vaut ni 5, ni 6. ")
    }
document.write("Hors du if")
```

Exemple en Python

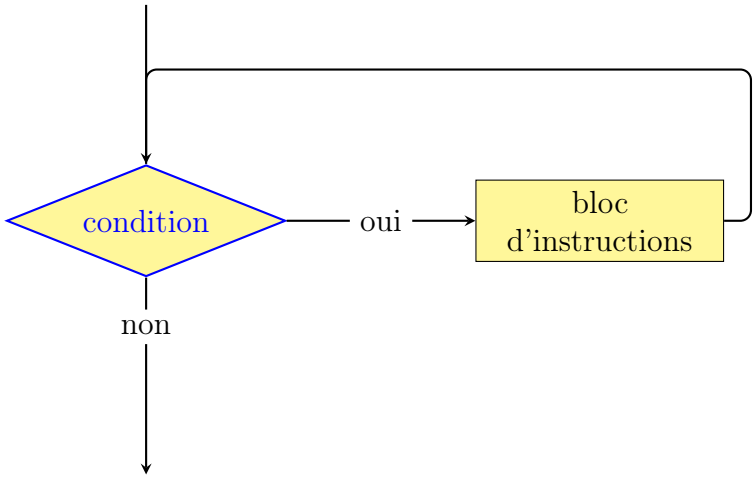
```
x = 5
if x == 5 :

    print("x vaut 5. ")

elif x == 6 :

    print("x vaut 6. ")

else :

    print("x ne vaut ni 5, ni 6.")

print("Hors du if")
```

4.2 Les boucles

Les boucles servent à exécuter plusieurs instructions un certains nombres de fois, prédéfini ou pas.

4.2.1 La commande while

While signifie *tant que*. Ainsi la commande suivante exécute le code tant que la condition est vraie. Si au départ, la condition est fausse, alors le code n'est pas exécuté.



Code JavaScript

```
while (condition)                    //parenthèses obligatoires
    {
    //code qui sera exécuté tant que la condition est vraie
    }
```

Code Python

```
while condition :                  ##double-points obligatoire
    ##code qui sera exécuté tant que la condition est vraie
    ##ce code doit être indenté (décalé à droite)
    ##dès que l'indentation stoppe, on sort du "while"
```

Exemple

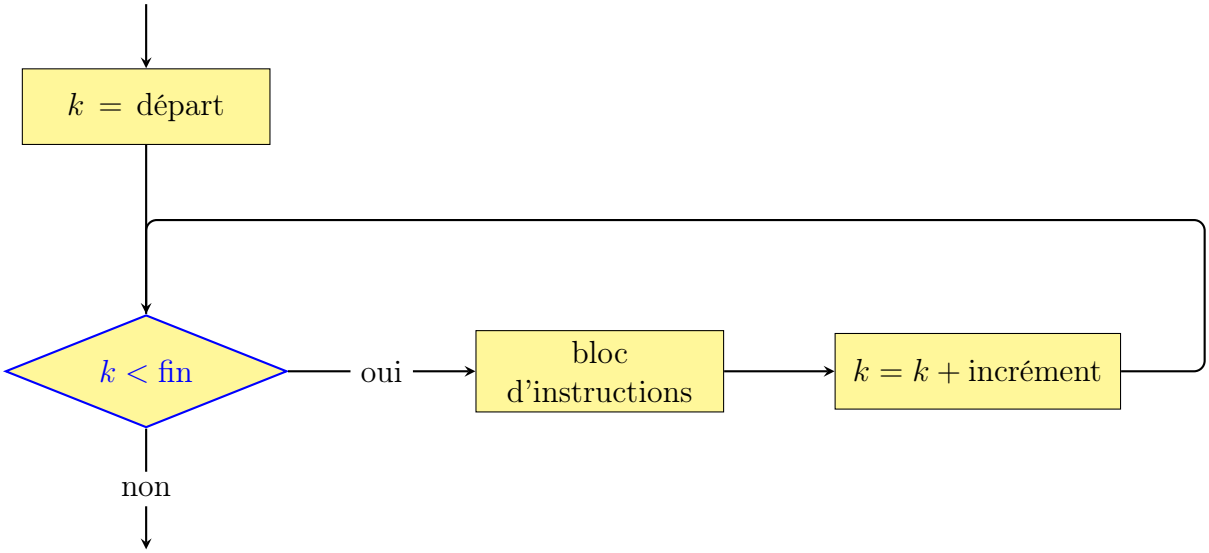
Dans cet exemple, on construit la somme des nombres de 1 à 10, 10 non compris.

```
somme = 0
k = 1
while (k < 10)
    {
    somme = somme + k
    k = k + 1
    }
document.write("la somme vaut " + somme)
```

```
somme = 0
k = 1
while k < 10 :

    somme = somme + k
    k = k + 1

print("la somme vaut", somme)
```

4.2.2 La commande for

For signifie *pour*. Cette commande dépend d'une variable que l'on initialise au début de la commande. Ensuite, tant qu'une certaine condition est vraie, on exécute le code, puis la variable subit une incrémentation.



Code JavaScript

```
// dans cette boucle for, la variable utilisée s'appellera k
for (k = valeur de départ; k < valeur de fin; k = k + incrémentation)
    {
    //code qui sera exécuté tant que
    //la variable k n'a pas atteint la valeur de fin
    //la valeur de k peut être utilisée dans ce code
    //à chaque passage, la valeur de k augmente de l'incrément
    };
```

Code Python

```
## dans cette boucle for, la variable utilisée s'appellera k
for k in range(valeur de départ, valeur de fin, incrémentation) :
    ##le code indenté sera exécuté tant que
    ##la variable k n'a pas atteint la valeur de fin
    ##la valeur de k peut être utilisée dans ce code
    ##à chaque passage, la valeur de k augmente de l'incrément
```

Exemple

Voici une autre façon de construire la somme des nombres de 1 à 10, 10 non compris.

```
somme = 0
for (k = 1; k < 10; k = k + 1)
    {
    somme = somme + k
    }
document.write("la somme vaut " + somme)
```

```
somme = 0
for k in range(1,10,1) :

    somme = somme + k

print("la somme vaut", somme)
```

5 Les fonctions

Une fonction est un morceau de code, éventuellement dépendant de certaines variables, qui peut être exécutée directement à l'aide de son nom. Cela permet

1. de ne pas retaper plusieurs fois des bouts de code qui seraient très semblables ;
2. de décomposer clairement le programme principal en plusieurs parties.

Les fonctions peuvent être mises à disposition d'autres programmes, voire même d'autres programmeurs. Leur nom doit commencer par une lettre, aucune lettre accentuée ou caractère spécial n'est permis, sauf «_». Attention à bien distinguer les majuscules des minuscules.

Code JavaScript

```
function nomDeLaFonction(argument1, argument2, ...)
    {
    //lignes de code
    return sortie  //commande optionnelle :
                   //le contenu de la variable sortie
                   //peut directement être utilisé
    }
```

Code Python

```
def nomDeLaFonction(argument1, argument2, ...) :
    ##lignes de code qui doivent être indentées
    return sortie  ##commande optionnelle :
                   ##le contenu de la variable sortie
                   ##peut directement être utilisé
```

Si la fonction est sans argument, elle doit tout de même comporter des parenthèses.

Un premier exemple

Voici une fonction sans argument qui affiche un message d'alerte.

```
function danger()
    {
    document.write("fonction danger exécutée")
    }
```

```
def danger() :

    print("fonction danger exécutée")

```

On constate que les parenthèses doivent être présentes, même s'il n'y a pas d'argument à l'intérieur.

La seule manière d'utiliser cette fonction est d'y faire appel directement afin d'afficher le message à l'écran.

`danger()` va imprimer le message `"fonction danger exécutée"` à l'écran.

Un deuxième exemple

Voici une fonction qui qui renvoie le triple du nombre passé en argument.

```
function triple(x)
    {
    t=3*x
    return t
    }
```

```
def triple(x) :

    t=3*x
    return t
```

Dans le code ci-dessus, la variable `t` est une *variable locale* : elle est créée durant l'exécution de la fonction et détruite dès que la fonction a fini son exécution.

Voici deux utilisations de la fonction `triple` si `x` est une variable à laquelle une valeur est déjà assignée :

1. `resultat = triple(x)` va stocker le triple du contenu de la variable `x` dans la variable `resultat`.
2. `document.write(triple(x))` (JavaScript) ou `print(triple(x))` (Python) va imprimer le triple du contenu de la variable `x` à l'écran.

Règle de base concernant les arguments des fonctions

Sauf si on fait de la programmation avancée, on évite de modifier les arguments passés en paramètres dans une fonction. Il peut ainsi être utile d'assigner les valeurs des paramètres dans des *variables locales*.

6 Importation de fonctions ou modules externes

On peut mettre les scripts dans un fichier à part de la manière suivante. L'avantage de cette manière de procéder est que le script peut être utilisé dans d'autres scripts de manière simple et efficace. Cela simplifie la lecture des scripts et cela permet aussi de ne modifier qu'un seul fichier dans le cas où le code est amélioré !

Code JavaScript

Le fichier original est un fichier `HTML`. Les scripts sont stockés directement en JavaScript à part : cela permet d'éviter d'utiliser les balises `HTML` chaque fois qu'on programme en JavaScript.

Voici le fichier `externe.js` :

```
document.write("ce script est externe")
```

Voici le code HTML :

```
<html>
<body>
<script src="externe.js"></script>
</body>
</html>
```

Code Python

Pour signaler à Python d'utiliser le code se trouvant dans le fichiers `externe.py`, il suffit d'insérer la ligne suivante au début du programme principal.

```
from externe import *
```

7 Interaction avec l'utilisateur

Un programme requiert bien souvent une information de la part de l'utilisateur. Le code suivant (JavaScript à gauche et Python à droite) pose une question à l'utilisateur. La réponse à cette question sera stockée comme une chaîne de caractères dans la variable `reponse`.

```
reponse = prompt("question")
```

```
reponse = raw_input("phrase")
```

Si on désire transformer la réponse en un entier, on utilise les commandes

`parseInt(reponse)` (en JavaScript) et `int(reponse)` (en Python).

Si on désire transformer la réponse en un nombre à virgule flottante, on utilise les commandes

`parseFloat(reponse)` (en JavaScript) et `float(reponse)` (en Python).

Exemple

```
texte1 = prompt("entrer un 1er nombre")
texte2 = prompt("entrer un 2ème nombre")

// si on additionne les textes, on les mets bout à bout.
document.write("concaténation : ")
document.write( texte1 + texte2 )

// saut à la ligne
document.write("<br />")

// si on additionne les textes transformés en nombres
// à virgule flottante, alors on additionne les deux nombres.
// Ne fonctionne que si texte1 et texte2 sont des nombres
document.write("addition : ")
document.write( parseFloat(texte1) + parseFloat(texte2) )
```

```
texte1 = raw_input("entrer un 1er nombre")     ##sans accents
texte2 = raw_input("entrer un 2eme nombre")    ##sans accents

## si on additionne les textes, on les mets bout à bout.
print("concaténation : ")
print(texte1 + texte2)

## saut à la ligne
print("\n")

## si on additionne les textes transformés en nombres
## à virgule flottante, alors on additionne les deux nombres.
## Ne fonctionne que si texte1 et texte2 sont des nombres
print("addition : ")
print( float(texte1) + float(texte2) )
```

8 Comment trouver une erreur dans le code JavaScript

Selon les explorateurs Internet, il se peut qu'en cas d'erreur dans le code JavaScript, la page `HTML` ne soit simplement pas affichée (ou que partiellement). Voici un script externe qui peut s'avérer très utile pour traquer une majorité d'erreurs au cas où l'explorateur resterait muet.

```
// onerror est une commande qui permet de récupérer les erreurs
onerror=messageErreur

function messageErreur(msg,url,line)
{
txt = "Il y a une erreur sur cette page :\n\n"  //la commande \n
txt = txt + "Erreur: " + msg + "\n"             //permet d'effectuer
txt = txt + "URL: " + url + "\n"                //un saut de ligne
txt = txt + "Ligne: " + line + "\n\n"           //dans la fenêtre de
                                                //dialogue
alert(txt)
return true // afin de dire à l'explorateur qu'on gère l'erreur
}
```

Il est conseillé d'enregistrer ce script dans un fichier `msgerreur.js` que l'on peut appeler par la ligne de code HTML suivante (à insérer avant tout script dans une page HTML) :

```
<script src="msgerreur.js"></script>
```

9 Comment récupérer une erreur en Python

Alors qu'en JavaScript, les commandes `parseInt` et `parseFloat` ne créent pas d'erreurs fatales lorsqu'on essaie de transformer du texte qui n'est pas un nombre en un nombre (en fait elles renvoient `NaN`, qui signifie *not a number* (pas un nombre)), leurs équivalents en Python déclarent une erreur et stoppent l'exécution du programme.

Afin d'éviter ce soucis, on utilise les commandes `try` et `except` dont l'utilisation est éclaircie ci-dessous.

```
texte = raw_input("entrer un nombre")

try :
    nombre = float(texte)
except :
    print("la conversion a échouée")
else :
    print("la conversion a réussie")
```

10 JavaScript et HTLM : faire un cycle d'images

Voici un module externe appelé `changeImage.js` contenant une fonction qui permet d'afficher une image dans une page `HTML` et de changer cette image à chaque clic de souris.

```
// Exemple d'utilisation de la fonction change(name,tableauImages)
//
// <img src="img/image1.png"
//      name="cycle1"
//      onclick='javascript:change("cycle1",["img/image1.png",
//                                           "img/image2.png",
//                                           "img/image3.png",
//                                           "img/image4.png"])' />
//
// La source src DOIT se trouver dans le tableau tableauImages
// (en fait les images vont faire un cycle à partir de src
// quelque soit la position de l'image src dans le tableauImage).
// Le tableau tableauImages doit être au moins de longueur 2.
// Les images DOIVENT toutes être dans un sous-répertoire appelé "img".
// La valeur name DOIT être la même que celle de la balise <img>.

function change(nom,tableauImages)
{
    ////////////////////////////////////////////////////////////////////
    // extraction du nom de l'image affichée à partir du chemin complet //
    ////////////////////////////////////////////////////////////////////
    // extraction du chemin complet

    var path        = document.images[nom].src

    // transformation du chemin en un tableau en cassant le chemin complet
    // là où se situent les '/'

    var tableau     = path.split('/')

    // en prenant la dernière cellule du tableau et en y ajoutant le chemin "img/",
    // le nom de l'image devient celui qui se trouve dans tableauImages
    // (il faut donc que l'image src de la balise <img> se trouve dans ce tableau)

    var nomFichier  = "img/" + tableau[tableau.length-1]

    ////////////////////////////////////////////////////////////////////
    // Recherche de l'image actuellement affichée dans le tableau        //
    // et remplacement de cette dernière par l'image suivante du tableau //
    // (de manière cyclique, d'où l'utilisation du modulo "%")           //
    ////////////////////////////////////////////////////////////////////
    changement = false   // deviendra true dès que le changement a été fait
    longueurTableau = tableauImages.length

    // on parcourt le tableau en recherchant l'image actuellement affichée
    for (k = 0; k < longueurTableau; k=k+1)
    {
        if ( changement == false && nomFichier == tableauImages[k] )
        {
            document.images[nom].src = tableauImages[ (k+1) % longueurTableau ]
            changement = true
        }
    }
}
```