Résumé de cours sur le JavaScript (+Bonus Python 3)

Premiers pas

Pour exécuter du code en JavaScript, on peut dans le cas d'une commande isolée utiliser la console du dev tool kit de Chrome (F12 puis onglet « console »)
La plus part du temps on voudra exécuter un programme et dans ce cas-là on sera obligé de passer par un fichier intermédiaire en HTML

page.html

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

hello.js

```
document.write("coucou les geeks!") ;
```

Ici *write* est une méthode qui agit sur l'objet « *document* » autrement dit la page qui est affichée dans la fenêtre chrome exécutant le fichier page.html
En JavaScript chaque commande se termine par un point-virgule.

Python : là c'est bien plus simple le programme sera : *print("coucou les geeks!")*

Commenter un programme

Très rapidement les programmes deviennent conséquents et il devient impératif de les commenter. Ça permet d'expliquer vos intention et votre logique à la personne qui va lire votre code :un camarde, le prof, ou tout simplement vous-même dans une semaine, alors que vous aurez oublié le contexte et votre raisonnement.
Pour que l'interpréteur ne mélange pas votre code et vos commentaires on utilisera les notations suivantes.

HTML : *< !--commentaire -->*
JavaScript : *// commentaire*
Python : *# commentaire*

ou s'il y en a plusieurs lignes :

```
/* commentaire 1
commentaire 2
commentaire 3 */
```

Variables

Elles sont les noms d'espaces mémoires dans lesquels on stockera de l'information.
Quand on fera une première assignation en JavaScript on écrira var devant le nom de la variable, par exemple : var nombre = 5
Si on veut changer la valeur de la variable nombre on écrira par ex : nombre = 17

En python pas besoin de « var »
Les variables peuvent être de plusieurs types : entier, réels, listes, chaines de caractère. Ça a une influence sur la mémoire prise par la variable et la manière dont les opérations vont agir. Par exemple :
Si on écrit *a= 5* et *b = 7* alors quand on écrit *c=a+b*, c vaudra 12
Si on écrit a= *"5"* et *b="7"* alors quand on écrit *c=a+b*, c vaudra *"57"*
+ est l'addition pour les nombres, mais pour les chaines de caractère ce symbole fait référence à la concaténation.
On pourra convertir d'un genre à l'autre réciproquement si alors :

parseInt(*"5"*) donnera 5
parseFloat(*"5"*) donnera 5.0
a.toString() donnera *"5"*

Quelques opérations :

Nom	Maths	JavaScript	Python
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`	`a / 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`

Listes

Une liste peut contenir des éléments hétéroclites liste1=[« truc »,5,[7,9],3.1,-1]
liste1[0] correspond à « truc », liste1[2] correspond à la liste [7,9]
liste1[2][1] correspond à 9 liste1[1 :3] donne une sous liste allant des elts 1 à

Chaines de caractère

Elle peut être considérée comme une liste de lettres.
Quand on veut créer des pages web on peut être amené à utiliser toute sorte de caractères spéciaux et d'effets de mise en page. Une partie sera gérée par les balises html comme
 (retour à la ligne) (qui correspond à \n en python) mais on doit être à même d'écrire et de récupérer toute sorte de chaines de caractères sans provoquer de plantage. \\ donne : \ \' donne : '

Python n'apprécie généralement pas les caractères accentués et a tendance à les modifier à moins qu'on n'écrive en début de programme : **#- *- coding: utf-8 -*-**

Booléens

On utilisera souvent des comparaisons, deux résultats seront possibles true(5>4) et false (x==3). Attention double égal pour parler d'une égalité de sens, triple égal pour une égalité exacte et un seul égal pour une assignation.

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`

Conditions

Exemple : JavaScript Python

```
x = 5;                                          x = 5
if (x <5 &&x>3))    {                           if x<5 and x>3 :
document.write("3<x<5") ; }                         print("3<x<5")
else {                                          else :
document.write("x>=5 ou x<=3") }                    print("x>=5 ou x<=3")
```

En JavaScript les blocs sont délimités à l'aide d'accolades en Python ils sont repérés grâce à l'indentation.

Boucles

Pour répéter une série d'action tant qu'une condition est réalisée on utilise **while**

Exemple : JavaScript Python

```
…                                               …
while (c < 10){                                 while (c < 10)
document.write("valeur de c"+ c + "<br />");        print(« valeur de c »,c)
c++;}                                               c+=1
```

Ce petit programme affiche toutes les valeurs de c à partir de sa valeur initiale jusqu'à 9

Comme on ne connait pas la valeur initiale de c(définie dans une partie non écrite du programme) on ne peut prévoir le nombre de fois où la boucle va tourner.

On a un peu plus de maitrise avec les boucles **for**

Exemple : JavaScript Python

```
var somme =0 ;                                  somme=0
for(var i =1 ; i<50 ;i++) {                     for(i in range(1,50) :
somme += i*i;}                                      somme+=i*i
document.write("la somme vaut"+ somme);         print(« la somme vaut »,i)
```

i prend la valeur 1 , puis augmente de 1, encore et encore jusqu'à valoir 49.

On peut se servir de boucles pour parcourir des listes :

JavaScript Python

```
var cars = ["BMW", "Volvo", "Saab"];            cars = ["BMW", "Volvo", "Saab"]
var i = 0;  var text = "";                      text = ""
for (;cars[i];) {                               for (voiture in cars) :
text += cars[i] + "<br>";       i++; }              text += voiture + "\n"
document.write(text)                            print(text)
```

Fonctions

Quand certaines actions sont utilisées de manière récurrente dans le code, plutôt que de recopier autant de fois les blocs d'instructions correspondants on utilisera des fonctions.

Exemple : JavaScript Python

```
function toCelsius(fahrenheit) {                def toCelsius(fahrenheit) :
var tempC = (5/9) * (fahrenheit-32);                tempC=(5/9)*(fahrenheit-32)
return tempC;}                                      return tempC

var C=toCelsius(77)                             C=toCelsius(77)
document.write(" 77°F correspond à  "+C+" C ")  print(" 77°F correspond à  ",C, " °C ")
```

Si on tape tempC après l'exécution de la fonction on aura un message d'erreur, cette variable n'existe que lors de l'exécution après elle est détruite (ce qui libère de la mémoire) c'est une variable locale certaines fonctions ne génèrent pas de résultat, certaines n'utilisent pas de paramètres.

Exemple : JavaScript Python

```
var nombre=17 ;                                 nombre=17

function quadruple() {                          def quadruple() :
nombre*=4 ;}                                        nombre*=4

quadruple() ;                                   quadruple()
```

ici nombre est une variable globale, elle est utilisée par la fonction quadruple() qui n'a pas d'argument ni de sortie mais elle existe en dehors de cette fonction.