1 Quelques rappels

1. Rappeler ce qu'est le WWW, en quoi cela consiste.
2. Quelles sont les éléments qui sont compris dans cette architecture ?
3. Sur quoi s'appuient cette infrastructure pour faite communiquer des machines entre elles ?
4. Quels sont, pour un client et un serveur, les logiciels et/ou technologies qui doivent être présents qu'un échange WEB soit possible ? Faire un schéma.
5. En quoi consiste le fait de réaliser une requête WEB ? Donner un exemple.
6. Connaissez-vous une méthode simple pour savoir si la machine sur laquelle vous travaillez possède un serveur WEB actif ?
7. Pour un serveur WEB, où sont en général stockées les ressources mises accessibles ?
8. Connaissez-vous les grandes étapes du développement des technologies WEB ?
9. Quand vous-êtes sur une page WEB, savez-vous comment accéder au code HTML ?

2 Les bases du langage HTML

2.1 Norme HTML et validation

Le format HTML est standardisé par le W3C. Ce dernier possède un validateur qui permet de vérifier si un fichier est conforme au standard. Il est accessible à l'adresse :

`https://validator.w3.org`

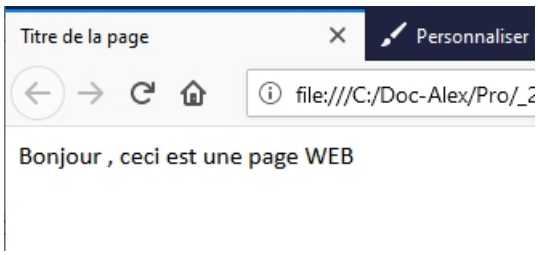
2.2 Le code minimal...

Le format HTML est un langage qui utilise des balises « <> » qui permet de décrire le contenu et la structure d'un document (ou d'une page) WEB. Il faut bien distinguer le code source HTML de la page, et son rendu (affichage) dans un navigateur. Un navigateur WEB est donc un interprétateur de ficher HTML.
La structure est données par les balises HTML, qui ne sont pas affichées dans le rendu, mais servent au concepteur de la page pour structurer les informations.
Voici un premier exemple. A gauche, on a le code minimal d'une page web, qui est un fichier créé avec un éditeur comme NotePad++, et enregistré au format « html », comme par exemple « page1.html ».
A droite, on a ouvert le fichier avec un navigateur. On remarque que « Titre de la page » apparait comme nom de l'onglet, et que « Bonjour , ceci est une page WEB » apparait dans le corps de la page.

```
<!DOCTYPE html>
<html lang = "fr">
<head>
<title>Titre de la page</title>
<meta charset = "utf-8">
</head>

<body>
Bonjour , ceci est une page WEB

</body>
</html>
```



On retiendra que :

- les balises « <html> ... </html> marquent le début et la fin du code ;
- les balises « <head> ... </head> marquent le début et la fin du de l'en-tête de la page ;
- les balises « <body> ... </body> marquent le début et la fin du corps de texte de la page.
- chaque balise ouverte doit être fermée dans un bon ordre d'écriture, avec le contenu entre chaque.
- Les seules balises vides autorisées sont **area, br, embed, img, meta, source, wbrn, base, col, hr, input, param, track**.

► Copier-coller ce code dans un fichier, puis afficher dans un navigateur. Enfin faire des modifications et des actualisations...

2.3 Structurer le texte

Les balises **<h1> ... </h1>**, **<h2> ... </h2>** permettent d'indiquer les titres et les sous-titres.

```
<h1 id=titre1>Titre de niveau 1</h1>
        <h2>Titre de niveau 2<h2>
        ...
```

```
        <p>Cette balise permet de commencer un nouveau paragraphe</p>
        <br>permet de retourner à la ligne
```

Le code **id = titre1** à la ligne 1 permet d'identifier ce titre en particulier. Il peut servir pour établir un lien hypertexte vers lui à partir d'un autre emplacement de la page, comme le montre un des exemples qui suit.

► Copier-coller ce code dans un fichier, puis afficher dans un navigateur. Enfin faire des modifications, par exemple en ajoutant des balises <h3> et des actualisations...

2.4 Ajouter un hyperlien

La balise **<a>...** permet de créer un lien hypertexte. La valeur pour **href** est la ressource cible.

```
<a href="nom_du_fichier.html">ceci est un hyperlien vers une page locale</a>
<a href="http://www.wikipedia.fr">Ceci est un lien externe</a>
<a href="#titre1">Ceci est un lien vers l'endroit titre1 de la page</a>
```

- Si c'est un fichier dans le même dossier, on écrit son nom sans oublier l'extension, **html** souvent ;
- si c'est une page extérieure, on donne son URL complète, qui commence en général par **http ://** ou **https ://**

► Copier-coller ce code dans un fichier, puis afficher dans un navigateur. Enfin faire des modifications, par exemple en ajoutant des balises <h3> et des actualisations...

2.5 Modifier l'apparence du texte

```
<b>Texte en gras</b>  <br>
<u>Texte souligné</u>  <br>
<i>Texte en italic</i>  <br>
<center>Ce texte sera centré</center>  <br>
La notation mathématique que "x puissance 3" est : x<sup>3</sup>
```

► Copier-coller ce code dans un fichier, puis afficher dans un navigateur. Enfin faire des modifications, et vérifier que vos liens fonctionnent correctement. Tester également les balises **mark, small, code, sub et sup**.

2.6 Dresser une liste

La balise <ul>... permet de faire une **liste avec puces**, alors que la balise <ol>... permet de faire une **liste numérotée**. Chaque **nouvel item** est indiqué par <li>.... Il est possible d'imbriquer les listes les unes dans les autres comme ci-dessous. A droite on voit le résultat dans la page un navigateur.

```
<br><br> Voici des <u>ingrédients</u> :
<ul>
    <li>Lait</li>
    <li>Fromage</li>
        <ol>
            <li>Camenbert</li>
            <li>Feta</li>
        </ol>
</ul>
```

Voici des ingrédients :

- Lait
- Fromages
    1. Camenbert
    2. Feta

► Copier et coller le code dans votre page HTML, puis vérifier que son exécution correspond à ce qui est donné à côté.

► Dans la liste des « ingrédients », ajouter "Sucre", et dans la liste des fromages, ajouter « Tomme de Savoie ».

2.7 Insérer une image

```
<img src="nom_du_fichier.jpg" width="400"   height="300">
```

La valeur de **src** est le nom du fichier, qui en général est placé dans le même dossier. Si le fichier image est sur un autre ordinateur, on utilise alors l'URL de cette image.
Les valeurs de **width** et **hight** permettent de dimensionner l'image. Elles ne sont pas nécessaires.

► Télécharger une image dans votre dossier, puis insérer cette image dans votre page WEB.

2.8 Insérer un tableau

Les tableaux ne sont pas toujours bien mis en valeur en HTML. Cependant pour présenter certaines données, il sont parfois nécessaires. Ils sont créés avec la balise **<table>**. On peut préciser l'épaisseur des bordures avec **border = ...** .
Il est possible de fusionner des cellules, horizontalement ou verticalement, avec rowspan et colspan. Il est également possible d'ajouter des en-tête avec les balises **<th> </th>** voir sur internet pour plus d'informations.
Voici un exemple, avec le code à gauche et le résultat dans un navigateur.

```
<table border = 1>
<tr>
  <td>Ligne 1 Colonne 1</td>
  <td>Ligne 1 Colonne 2</td>
  <td>Ligne 1 Colonne 3</td>
</tr>
<tr>
  <td>Ligne 2 Colonne 1</td>
  <td>Ligne 2 Colonne 2</td>
  <td>Ligne 2 Colonne 3</td>
</tr>
</table>
```

Ligne 1 Colonne 1	Ligne 1 Colonne 2	Ligne 1 Colonne 3
Ligne 2 Colonne 1	Ligne 2 Colonne 2	Ligne 2 Colonne 3

► Tester l'exemple qui est proposé, puis le compléter avec une troisième ligne qui contiendra ce que vous voulez.
► Pour aller plus loin (non exigible) reproduire en HTML le tableau qui suit :

Une table test avec des cellules fusionnées

	Moyenne		Yeux rouges
	hauteur	poids	
Mâles	1.9	0.003	40%
Femelles	1.7	0.002	43%

3 Les feuilles de style CSS

Le langage « CSS », qui signifie **Cascading Style Sheets**, ou feuilles de styles en cascade, est un langage permettant de définir les propriétés graphiques des éléments HTML constituant une page WEB. Les feuilles de styles permettent de regrouper tous les styles de toutes les pages dans des fichiers à part.
Certes les choix de styles peuvent se faire directement dans chaque page, voire même dans chaque balise, comme sur l'exemple ci-dessous qui donne un titre écrit en rouge et souligné.

```
<h1 STYLE = "color : red ; text-decoration: underline">Titre avec style</h1>
```

Voici un autre exemple :

```
<p> On écrit un <span STYLE = "border : 1pt solid black">paragraphe</span>
de texte, mais cette fois on <span STYLE = "color : gray">modifie</span> le style graphique
<span STYLE = "border : 1pt solid black>des éléments.</span></p>
```

Mais pour uniformiser toutes les pages web, il vaut mieux directement les regrouper dans un ou plusieurs fichiers.
Dans ce genre de fichiers, on spécifie les couleurs des fonds d'écran, la taille et la couleur des polices, l'apparence des liens et bien d'autres choses.
L'objectif est de vous faire découvrir comment mettre en place une feuille de style, et modifier l'apparence de quelques éléments de vos pages WEB.

3.1 Créer le fichier CSS

Avec **Notepad++**, créer un nouveau fichier dans le même dossier que vos pages WEB, que vous nommerez :

styles.css

3.2 Faire le lien de la page vers le fichier CSS

Il faut indiquer à chaque page WEB quelle feuille de styles utiliser. Pour faire ce lien, on rajoute de l'en-tête de la page la balise **LINK** qui suit :

```
<head>
<title>Titre de la page</title>

<LINK HREF="styles.css" REL = "stylesheet" media="all" type="text/css">

</head>
```

3.3 Éditer le fichier CSS

Avec **Notepad++**, éditer le fichier **styles.css**. On va pour illustrer changer l'aspect des titres de second rang écrits avec la balise **<h2>**. Pour cela, recopier le code dans le fichier, enregistrer ce fichier puis recharger la page pour voir les changements.

```
h2 {
        color : red ;
}
```

On retiendra que le modèle est :

```
sélecteur {
propriété: valeur;
}
```

3.4 Des styles à tester

Voici des exemples de styles, que vous pouvez utiliser pour donner du style à vos pages WEB!

```
html, body {
 margin: 0;
 padding: 0;
 }

body {
 background-color: white;
 font-family: Verdana, sans-serif;
 font-size: 100%;  }

h1 {
 font-size: 200%;
 color: navy;
 text-align: center; }

h2 {
 font-size: 150%;
 color: red;
 padding-left: 15px; }

p,ul,li,td {
 color: black;  }

a:link {
 color: green;
 text-decoration: underline; }
 }

a:visited {
 color: gray;  }


a:hover {
 color: red;
 text-decoration: none; }

a:active, a:focus {
 color: red; }
```

Sur internet, vous pourrez trouver beaucoup de prolongements aux CSS... Même si la connaissance des feuilles de styles n'est pas un impératif du programme, il est nécessaire de le reconnaitre dans l'écriture d'une page WEB.

4 Mettre en place un serveur WEB sur une machine

Comme pour la partie précédente sur les CSS, le fait de savoir transformer une machine en un serveur WEB ne fait pas partie des attendus du programme de NSI en première. Cependant il serait dommage de ne pas le faire, car il est attendu que les élèves comprennent les interactions client-serveur dans le cadre d'une requête **HTTP**!!
La méthode décrite dans cette partie décrit l'installation d'un petit serveur WEB en Python, avec un simple script. Pour fonctionner, il suffit que **Python** dans une version 3 minimum soit installée. Normalement cela fonctionne aussi bien sur **Windows** que sur *Linux*. Cela évite de se lancer dans l'installation d'un serveur WEB plus professionnel, comme **Apache2** ou d'autres. Si vous utiliser *Windows*, l'installation d'un serveur WEB, d'une système de base de donnée et du langage PHP se fait assez facilement avec *EasyPHP*... Ne pas hésiter à me demander des explications, mais en cours le sujet ne sera pas fondamentalement abordé.
Suivre les instructions pas à pas :

1. Dans votre navigateur WEB, saisir l'URL http ://localhost. L'adresse **localhost** désigne la machine que vous utilisez. C'est normalement équivalent à http ://127.0.0.1. Si votre machine est équipée d'un serveur WEB, elle doit répondre et donner la page d'index qui se trouve à la racine des ressources WEB de votre machine. Si votre machine n'est pas équipée d'un serveur WEB, alors au bout d'un certain temps le navigateur doit indiquer un message du genre « Délai d'attente dépassé... », ce qui indique que votre machine ne répond pas aux requêtes HTTP.

2. Créer sur le bureau (Desktop) un dossier que vous pouvez nommer ServeurWEB.

3. Créer, avec IDLE, un nouveau fichier Python, un nouveau fichier que vous pouvez appeler ServeurWEB.py, que vous enregistrez dans le dossier de que vous venez de créer.

4. Copier et coller le code ci-dessous, puis enregistrer le fichier. La compréhension intégrale de chaque ligne de code n'est pas utile. En, quelques commentaires seront faits pour comprendre quelques détails.

```
#!/usr/bin/python

import http.server

PORT = 8888
server_address = ("", PORT)

server = http.server.HTTPServer
handler = http.server.CGIHTTPRequestHandler
handler.cgi_directories = [""]
print("Serveur actif sur le port :", PORT)

httpd = server(server_address, handler)
httpd.serve_forever()
```

5. Exécuter ce code. Il est nécessaire de laisser cette fenêtre *Python Shell* (console) active tout le temps que l'on souhaite utiliser le serveur. Le premier message affiché est : « Serveur actif sur le port : 8888 ». Pour couper le serveur, il suffit de fermer cette fenêtre.

6. copier et coller les fichiers WEB réalisés aux parties 2 et 3, notamment le fichier index.html et aussi les autres fichiers (images...) dans le dossier ServeurWEB. Ce dossier contient les ressources WEB lises à disposition par les requêtes HTTP.

7. Dans le navigateur WEB, saisir la requête (adresse) : http ://localhost :8888/index.html . Si tout fonctionne bien, votre page d'index s'affiche dans le navigateur.

8. Trouver l'adresse IP de votre machine, puis sur un autre ordinateur (ou smartphone) connecté au même réseau (wifi ou filaire), saisir l'adresse http ://monIP :8888/index.html . Normalement votre page s'affiche. Votre machine est un serveur WEB actif! Pour rappel, la commande WIN pour trouver son IP est ipconfig, et sur Linux c'est ifconfig.

9. Retourner sur la fenêtre « Shell Python » du fichier ServeurWEB, vous pouvez voir à chaque connexion une ligne qui indique l'IP du client.

10. Modifier la page WEB, puis faire actualiser sur le navigateur... si ça marche vous êtes prêt pour la suite!

5 La programmation côté CLIENT

Les balises HTML présentées plus haut permettent de présenter des documents structurés, illustrés, pratiques avec avec hyperliens, mais STATIQUES. Le contenu ne peut pas être changé par le lecteur, il n'y a **aucune interaction** avec le client hormis les hyperliens. Des technologies ont été développées pour rendre DYNAMIQUE le contenu de la page. La principale est le langage **JavaScript**, qui à l'origine s'appelait **LiveScript**. On peut citer aussi les **formulaires** en HTML qui permettent de définir dans la page des zones interactives (zones de saisie de texte, boutons radio, zone de sélection pour l'essentiel). **Ces technologies WEB permettent de créer de réelles Interfaces Homme-Machine (IHM), c'est à dire des outils qui peuvent être utilisées côté client par des novices en informatique**.

5.1 Définition de JavaScript et un exemple

JavaScript c'est quoi ?

JavaScript est un langage de programmation dont le script est directement écrit sur la page WEB demandée par le client. C'est le navigateur de la personne qui télécharge la page qui **repère** les scripts et les **exécute**. Si le navigateur n'est pas compatible avec JavaScript (pour les TRÈS anciens navigateurs), le code est **ignoré** grâce aux balises « <!– » et « //–> », et le reste de la page s'affiche normalement.

L'intérêt de JavaScript est de pouvoir rendre une page WEB **dynamique**, c'est à dire que l'utilisateur peut avoir des **interactions** avec la page WEB.

L'écriture de JavaScript est plus proche de Java que de Python, en particulier les blocs d'instructions s'écrivent non pas avec des indentations mais avec des **accolades**.

▶ Observer le code HTML suivant. Après analyse, copier et coller le code dans un fichier nommé exempleJS.html, et utiliser le navigateur pour son exécution.

```
<!DOCTYPE html>
<html lang = "fr">

<head>
<title>ExempleJS</title>
<meta charset = "utf-8">
<script language = "JavaScript">
<!--
function message() {
window.alert('Bonjour') }
//-->
</script>
</head>

<body>
<h1>Un exemple d'utilisation de JavaScript</h1>
<script language = "JavaScript"><!--
for (var i = 1 ; i <=5 ; i++){
document.write("Bonjour ! passage : " + i + "<br>")
}
//-->
</script>
<br> Ceci est un bouton :
<form><input type = "button"  value = "Cliquer !" onclick = "message()"></form>
</body>
</html>
```

On retiendra que **JavaScript** permet à l'utilisateur d'interagir avec TOUS les éléments (on parle d'**objets**) de la page. Le script est directement inséré dans la page WEB, et il est exécuté côté client dans le navigateur. Les scripts JavaScript sont encadrés par les balises `<script>...</script>` qui marquent le début et la fin. Le script peut être dans le header, on général on regroupe ici les fonctions, ou sur la page comme la boucle qui affiche ici 5 fois bonjour.

Sur la page, les **formulaires**, repérables avec les balises `<form>...</form>` permettent de créer des **zones d'interaction** avec les utilisateurs : les différents objets appartenant à la famille des formulaires sont : les **boutons cliquables**, les **zones de saisie**, les **boutons radios** et les **zones de sélection multiples** pour les principaux. Sur l'exemple on utilise un bouton : `<input type = "button" onclick = "message()" value = "Cliquer!">` de sorte que lorsqu'on clique dessus, on appelle la fonction `message()` définie dans le header. Voilà pour l'exemple.

5.2 Découvrir le DOM en JavaScript

Qu'est-ce que le DOM ?
Le Document Object Model (DOM) (en français on dirait « Schéma Relationnel des Objets ») est une interface de programmation pour les documents HTML (et aussi XML).Il fournit une représentation structurée du document sous forme d'un arbre et définit la façon dont la structure peut être manipulée par les programmes, en termes de style et de contenu. Cette représentation du document permet de le voir comme un groupe structuré de nœuds et d'objets possédant différentes propriétés et méthodes. **Fondamentalement, il relie les pages WEB aux scripts ou langages de programmation** : la page WEB est un ensemble d'objets qui peuvent être manipulés à l'aide d'un langage de script comme JavaScript. Vous pouvez accéder à une référence sur la DOM en cliquant sur le lien :

`https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model`

Les principaux objets de la DOM sont `window` , `document`. Il y a en beaucoup : il est inutile de les apprendre tous, il suffit d'identifier l'objet qui nous intéresse, et de se documenter sur la manière de l'utiliser. Chaque objet possède ses **propriétés** et ses **méthodes**.

Par exemple, l'objet `window` possède :

- la méthode `window.alert()` qui permet d'afficher une fenêtre d'alerte ;
- la méthode `window.confirm()` qui permet comme son nom l'indique ouvre une fenêtre qui permet à l'utilisateur de confirmer ou infirmer un choix ;
- la méthode `window.prompt()` qui permet d'ouvrir une fenêtre dans laquelle l'utilisateur pourra saisir une valeur, un peu comme un « input » en Python.

L'objet `document` possède deux méthodes intéressantes qui sont :

- `document.forms` qui renvoie la liste de tous les formulaires de la page ;
- `document.images` qui renvoie la liste de toutes les images de la page.

1. Tester, dans le fichier "exempleJS.html", les méthodes citées de l'objet **window**.
2. Visiter le site indiquer ci-dessus, et en particulier rendez-vous sur les objets `window` , `document`, `form`, `images`.

5.3 Le projet « imc.html »

Pour comprendre la DOM, voici le premier projet JavaScript qui sera traité : une page WEB nommée `imc.html` qui permet à un utilisateur de :

1. saisir son poids ;
2. saisir sa taille ;
3. cliquer sur un bouton puis voir son IMC (indice de masse corporelle) s'afficher, avec un commentaire adapté.

Le code ci-dessous met en place une solution possible pour le projet imc.html. Analyser chaque partie du code, et compléter les fonction imc(). Pour information, les 2 premières lignes de cette fonction récupèrent les valeurs saisies dans le formulaire... Il faut que lorsqu'on clique sur le bouton « Calculer ! », l'IMC et le commentaire s'affichent dans les deux derniers formulaires... Bon courage !

```
<!DOCTYPE html>
<html lang = "fr">

<head>
<title>IMC</title>
<meta charset = "utf-8">
<script language = "JavaScript">
function imc() {
var p = document.forms[0].poids.value ;
var t = document.forms[0].taille.value ;
var imc = ...
}
</script>
</head>

<body>
<h1>Utilisation de JavaScript : calculer son IMC ! </h1>
<br><br>
<form>
<table border = 0>
<tr>
<td>Donner votre poids en kg :</td>
<td><input type = "text" name = "poids"></input></td>
</tr>
<tr>
<td>Donner votre taille en mètres :</td>
<td><input type = "text" name = "taille"></input></td>
</tr>
</table>
<input type = "button" onclick = "imc()" value = "Calculer !">
</form>

<br><br>
Votre IMC est :
<form>
<input type = "text" name = "resultat"></input>
</form>
<br>
Commentaire sur votre IMC :
<form>
<input type = "text" name = "comm"></input>
</form>

</body>
</html>
```

Voici la synthèse sur le fichier **imc.html**. Quelques remarques :

- quand on déclare une variable en JavaScript, il faut écrire var maVariable = ... ;
- j'ai ajouté un bouton **Annuler**, qui appelle la fonction **init()** et ainsi effacer les valeurs déjà saisies ;
- dans la balise **<body >**, j'ai ajouter **onLoad = "init()**. C'est un élément de programmation événementiel : à chaque chargement de la page, la fonction **init()** est appelée ;
- j'ai écrit convenablement les tests des instruction conditionnelles pour afficher le bon commentaire : il y a des différences avec Python !

A vous de tester et de vérifier que vous avez bien compris tous les éléments de ce programme WEB !

```
<!DOCTYPE html>
<html lang = "fr">

<head>
<title>IMC</title>
<meta charset = "utf-8">
<script language = "JavaScript">
function imc() {
var p = document.forms[0].poids.value ;
var t = document.forms[0].taille.value ;
var imc = p/t/t ;
imc = imc * 10 ;
imc = Math.round(imc) ;
imc = imc / 10 ;
document.forms[1].resultat.value = imc ;
if (imc<18.5) {
document.forms[2].comm.value = "Poids insuffisant" ; }
if (18.5<=imc && imc<25){
document.forms[2].comm.value = "Bien" ;}
if (25<=imc && imc<30){
document.forms[2].comm.value = "Surpoids" ;}
if (imc >= 30){
document.forms[2].comm.value = "Obésité" ;}
}
function init() {
document.forms[0].poids.value = "" ;
document.forms[0].taille.value = "" ;
document.forms[1].resultat.value = "" ;
document.forms[2].comm.value = "" ;
}
</script>
</head>

<body onload = "init()">
<h1>Utilisation de JavaScript : calculer son IMC ! </h1>
<br><br>
<form>
<table border = 0>
<tr>
<td>Donner votre poids en kg :</td>
<td><input type = "text" name = "poids"></input></td>
```

```
</tr>
<tr>
<td>Donner votre taille en mètres :</td>
<td><input type = "text" name = "taille"></input></td>
</tr>
</table>
<input type = "button" onclick = "imc()" value = "Calculer !">
</form>

<br><br>
Votre IMC est :
<form>
<input type = "text" name = "resultat"></input>
</form>
<br>
Commentaire sur votre IMC :
<form>
<input type = "text" name = "comm"></input>
</form>

<br>
<form>
<input type = "button" value = "Annuler" onclick = "init()"/>
</form>

</body>
</html>
```

5.4 Le projet « convertisseur degré décimal (DD) - dégré :minutes :secondes (DMS)

Dans un premier temps, rendez-vous sur le site :

`https://www.sunearthtools.com/dp/tools/conversion.php?lang=fr`

Vous pouvez voir qu'il existe deux manières pour donner une position (latitude / longitude) pour se repérer sur la Terre :

- Le degré décimal (DD) ;
- Le Degré :Minutes :Secondes DD :MM :SS).

1. Documentez-vous sur la méthode pour passer de l'un à l'autre ;
2. Proposer, sur une page HTML avec JavaScript, un outil qui permet de passer de l'un à l'autre. Vous pourrez appeler votre page convertisseur.html.

6 La programmation WEB côté SERVEUR

Dans la partie précédente, nous avons vu le principe de la programmation WEB côté CLIENT : c'est la navigateur du client qui non seulement affiche la page mais aussi exécute les codes JavaScript présents sur la page. Lorsque le client se déconnecte du serveur, il ne reste pas de trace de ce qu'à fait le client (sauf des cookies!!). Parfois il est utile que le serveur récupère et enregistre des informations, qui pourront alors être stockées et exploitées par le serveur. Ce travail côté serveur se fait en général avec des langages de programmation comme **PHP**, **Python**, **Java** et bien d'autres. Pour éviter de nous disperser avec un autre langage, c'est en Python que nous allons écrire notre premier script côté serveur.

6.1 Un exemple de programmation côté SERVEUR

Dans cet exemple de programmation côté SERVEUR, nous allons proposer un formulaire qui va demander une valeur, puis transmettre une requête pour enregistrer la valeur dans un fichier « texte » sur le serveur avec un script Python.

1. Créer sur le bureau (Desktop) un dossier que vous pouvez nommer ServeurWEB.
2. Créer, avec IDLE, un nouveau fichier Python, un nouveau fichier que vous pouvez appeler ServeurWEB.py, que vous enregistrez dans le dossier de que vous venez de créer, comme dans la partie 4. Copier et coller le code ci-dessous, puis enregistrer le fichier. La compréhension intégrale de chaque ligne de code n'est pas utile : il suffit juste de savoir que lorsqu'il est exécuté, l'ordinateur répond aux requêtes WEB sur le port 8888.

```
#!/usr/bin/python

import http.server

PORT = 8888
server_address = ("", PORT)

server = http.server.HTTPServer
handler = http.server.CGIHTTPRequestHandler
handler.cgi_directories = ["/cgi-bin"]
print("Serveur actif sur le port :", PORT)

httpd = server(server_address, handler)
httpd.serve_forever()
```

3. Exécuter ce code. Il est nécessaire de laisser cette fenêtre *Python Shell* (console) active tout le temps que l'on souhaite utiliser le serveur. Le premier message affiché est : « Serveur actif sur le port : 8888 ». Pour couper le serveur, il suffit de fermer cette fenêtre.
4. Nous allons maintenant écrire le formulaire. Dans le dossier serveurWEB, créer un fichier appelé formulaire.html, puis copier et analyser le code ci-dessous.

```
<html>
  <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>Formulaire</title>
  </head>

  <body>
```

```
    <h1>Formulaire de saisie d'une valeur</h1>
    <p>Compléter le formulaire en donnant une valeur :</p>
    <br>
    <form action = "cgi-bin/traitement.py" method = "get">
    <input type = "text" name = "valeur" value = "écrire">
    <input type = "submit" value = "Envoyer">
    </form>
  </body>
</html>
```

5. Dans le navigateur WEB, saisir la requête (adresse) : http ://localhost :8888/formulaire.html . Si tout fonctionne bien, votre formulaire s'affiche dans le navigateur.

6. Nous allons maintenant créer le fichier d'exploitation du formulaire : c'est ce que l'on appelle un script **CGI** : c'est le script qui est exécuté, dont la référence est donnée dans le formulaire par **action = ...**, et sur le formulaire on voit bien que c'est **cgi-bin/traitement.py**. Dans le dossier serveurWEB, créer un dossier appelé cgi-bin, puis dans ce dossier recopier dans un fichier appelé traitement.py le code ci-dessous. Analyser ce code.

```
#!/usr/bin/python3

import cgi

# on récupère les valeurs du formulaire
formulaire = cgi.FieldStorage()

valeur = formulaire.getvalue('valeur')

print("Content-type: text/html; charset=utf-8\n")
print("<html><head><title>Traitement</title></head><body>")
print("<h1>La valeur que vous avez saisie est : </h1><br>")
print(valeur)

fichier = open("data.txt", "a")

fichier.write(valeur + "\n")
fichier.close()

print("<br><br><a href = '../formulaire.html'>Ajouter une autre valeur</a>")

print("</body></html>")
```

Attention!!! sur Linux, pour que ce code soit exécuté, il faut que le droit d'exécution lui soit accordé. Pour cela, avec la console, dans le dossier cgi-bin, il faut saisir la commande suivante :

sudo chmod 777 traitement.py

Si cette commande est correctement exécutée, quand on fait un ls, on voit que le nom du fichier est écrit en vert, ce qui signifie qu'il peut être exécuter à travers la console Python qui devrait démarrer automatiquement.

7. Dans le formulaire, saisir une valeur, cliquer sur le bouton **submit**, puis vérifier que dans le dossier cgi-bin, le fichier data.txt est bien créé, avec votre donnée enregistrée dedans.