université
PARIS-SACLAY

DIU Enseigner l'Informatique au Lycée
Bloc 1

Web : HTML & CSS

Consignes les exercices ou questions marqués d'un ⋆ peuvent être faits de manière débranchée.

1 UTF-8

⋆ Pour chacun des caractères suivants, dont le code décimal est donné, encoder le code de caractère en UTF-8 comme indiqué dans le cours et donner la séquence d'octets correspondants, en décimal et en hexadécimal (base 16).

En TP produire une séquence d'instruction Python qui permet de vérifier ses réponses.

1. caractère A (code 65)
2. caractère é (code 233)
3. caractère ⋆ (code 8902)

On rappelle que la méthode pour encoder un caractère en UTF-8 consiste à *(i)* traduire nombre en binaire, *(ii)* déterminer le nombre d'octets qui vont être nécessaires pour stocker tous les bits du nombre (en comptant les bits de contrôle) et *(iii)* traduire chaque groupe de 8 bits en un nombre (décimal ou hexadécimal). Une manière sûre de convertir un nombre en base 2 est de le décomposer en puissances décroissantes de 2 et de mettre à 1 le bit numéro i si 2^i est présent dans la décomposition.

Réponse:

1. $65 = 64 + 1 = 2^6 + 2^0 = 0100\,0001_2$. Comme $65 < 128$ le caractère est encodé comme en ASCII standard : 65_{10} ou 41_{16}
2. $233 = 128 + 64 + 32 + 8 + 1 = 2^7 + 2^6 + 2^5 + 2^3 + 2^0 = 1110\,1001_2$. On à donc besoin de 8 bits pour stocker ce nombre, on utilise donc l'encodage sur 2 octets car ce dernier permet de représenter des caractères allant jusqu'à 11 bits (les bits soulignés sont les bits du nombre à encoder, les bits non soulignés sont les bits de contrôle) : $110\underline{00011}_2\ 10\underline{101001}_2 = (128{+}64{+}3)_{10}\ (128{+}32{+}8{+}1)_{10} = 195_{10}\ 169_{10}$. Il est plus facile de convertir la base 2 en base 16 directement, sans passer par la base 10. Pour cela on groupe les bits par paquets de 4, qui donnent chacun un chiffre hexadécimal (de 0 à 9 suivis de A à F) : $1100\ 0011\ 1010\ 1001 = C3A9_{16}$
3. (on s'accroche et on rappelle aux étudiants qu'il est bon de connaître ses puissances de 2 jusqu'à $2^{16} = 65536$) $8902 = 8192 + 512 + 128 + 64 + 4 + 2 = 2^{13} + 2^9 + 2^7 + 2^6 + 2^2 + 2^1 = 100010\,11000110_2$. On a besoin de 14 bits pour représenter ce nombre, on utilise donc l'encodage sur 3 octets : $1110\underline{0010}\ 10\underline{001011}\ 10\underline{000110}_2 = (128{+}64{+}32{+}4)_{10}\ (128{+}16{+}2{+}1)_{10}\ (128{+}4{+}2)_{10} = 196\,151\,134$. En hexadécimal : $1110\ 0010\ 1000\ 1011\ 1000\ 0110_2 = E2\ 8B\ 86_{16}$

En Python :

```
s1 = b"\x41"
s1.decode()

s2 = b"\xc3\xa9"
s2.decode()

s3 = b"\xe2\x8b\x86"
s3.decode()
```

2 Que fait mon navigateur Web ?

NB Le programme Python de cet exercice ne fonctionnera que si la machine sur laquelle il s'exécute possède un accès direct à Internet (port 80 TCP en sortie non filtré), ce qui n'est pas le cas au PUIO (présence d'un proxy Web).

Le but de cet exercice est de simuler en Python la partie « réseau » d'un navigateur Web. On présente pour cela le module `socket` de Python, qui permet de créer des canaux TCP (*socket* en anglais). Les fonction nécessaires du module s'importent de la manière suivante :

```
from socket import gethostbyname, socket, AF_INET

#suite du programme
#...
```

Les fonctions s'utilisent de la manière suivante :

`gethostbyname(s)` prend une chaîne de caractères `s` représentant un nom de domaine et renvoie une chaîne de caractères représentant l'adresse IP correspondante.

`socket(AF_INET)` la fonction `socket` permet de renvoyer un objet représentant un canal de communication du type demandé (ici `AF_INET` pour une connection IP). D'autres paramètres permettent d'utiliser des protocoles différents et des protocoles autre que TCP (qui est la valeur par défaut).

`s.connect((a, p))` permet de connecter la *socket* `s` (créée par la fonction `socket` ci-dessus) à un serveur, représenté par le couple `(a, p)` où `a` est une chaîne représentant une adresse IP et `p` un entier représentant un numéro de port. Attention, `.connect` prend un seul argument qui est un couple, il faut donc 2 paires de parenthèses. La méthode ne renvoie pas de résultat, mais passe l'objet s en mode « connecté ».

`s.send(b)` envoie le `bytes` `b` au serveur auquel la *socket* `s` est connectée. Renvoie un entier qui est le nombre d'octet effectivement envoyés (normalement la longueur de `b`).

`s.recv(n)` renvoie un `bytes` d'au plus `n` octets qui est la réponse du serveur auquel la *socket* est connectée. Cette appel « bloque » le programme jusqu'à ce qu'il y ait des octets disponible (i.e. si le serveur ne répond jamais, le programme attend, jusqu'à ce qu'un *timeout* ait lieu et provoque une erreur).

`s.close()` termine la communication et ferme la *socket*. Cette dernière est fermée automatiquement lorsque le programme se termine, mais c'est une bonne pratique de couper la communication des que l'on a plus besoin de parler au serveur. Attention, une fois fermée, la *socket* n'est plus utilisable, il faut en créer une nouvelle.

Écrire un petit programme Python qui se connecte au site `idea.nguyen.vg` et récupère la page `test.html` (on pourra tester que cette page existe en navigant à l'adresse : `http://idea.nguyen.vg/test.html`).

Réponse:

```
from socket import gethostbyname, socket, AF_INET

host = "idea.nguyen.vg"  #Le nom de l'hôte que l'on veut contacter
a = gethostbyname(host)  #La fonction effectue la résolution DNS
s = socket(AF_INET)      #Création de la socket TCP/IP
s.connect((a, 80))       #On se connecte à l'adresse IP trouvée, sur le port 80

#On envoie la requête HTTP au serveur. Attention, si on change
#La variable host ci-dessus, il faut aussi changer la partie Host:
#de la requête

s.send(b"GET /test.html HTTP/1.1\r\nHost: idea.nguyen.vg\r\n\r\n")

data = s.recv(200)   #on lit ce que renvoie le serveur par paquet de
                     #200 octet
```

```
#Tant que le serveur nous envoie des données, on les affiche
while data:
      print(data.decode())
      data = s.recv(200)

s.close() #on ferme la connexion
```

3 Avec ou sans serveur...

Au PUIO, chaque utilisateur possède un répertoire `public_html` dans son répertoire personnel. Ce répertoire est connu du serveur Web Apache installé au PUIO. Si votre identifiant de connexion est `toto`, alors l'URL [1] :

`https://tp-ssh1.dep-informatique.u-psud.fr/~toto/`

pointe vers votre répertoire `public_html`.

1. Créer un fichier texte `test.txt` dans le répertoire `public_html`
2. Comment compléter l'URL ci-dessus pour accéder au fichier `test.txt` via le serveur Web ? Naviguer à cette URL avec le navigateur Web et s'assurer qu'on obtient bien le fichier.

> **Réponse:** `https://tp-ssh1.dep-informatique.u-psud.fr/~toto/test.txt`

3. Ouvrir (dans un autre onglet du navigateur) le fichier `test.txt` directement (par exemple depuis le menu `Fichier`→`Ouvrir`). Quelle est l'URL donnée dans la barre d'adresse du navigateur ?

> **Réponse:** `file:///home/tp.../toto/public_html/test.txt`

   (la partie `...` dépend de l'emplacement du répertoire utilisateur, et `toto` est à remplacer par le login de l'utilisateur).

4. Sous Unix, chaque fichier possède des permissions pour son propriétaire (le créateur du fichier), le groupe principal auquel le propriétaire appartient et tous les autres utilisateurs du système. Utiliser le navigateur de fichier et les propriétés des fichiers (bouton droit dans le navigateur sur l'icône `test.txt`) pour laisser uniquement les permissions en lecture et écriture au propriétaire et retirer toutes les permissions aux autres. (De manière équivalente, on pourra exécuter dans un terminal :

   `chmod 600 ~/public_html/test.txt`

   mais les détails de cette commande seront montrés lors du Bloc 3). Recharger les deux onglets (avec les deux URLs différentes). Que constate t-on ? Proposer une explication. On rappelle que sous Unix tout programme hérite des droits et permissions de l'utilisateur qui l'exécute.

> **Réponse:** Le fichier chargé avec l'URL `file:///` se charge sans problème. Le fichier chargé via le serveur Web donne une erreur 403 (permission denied). Le programme Apache, lancé par un utilisateur système autre que vous même n'a plus les droits pour lire le fichier. Il n'est donc pas en mesure d'envoyer son contenu au navigateur Web qui le demande et il répond avec le code d'erreur prévu par le protocole (ici 403).

5. Déplacer le fichier `test.txt` en dehors de `public_html`. Par quel type d'URL est-il toujours accessible ?

1. Ce serveur n'est visible que depuis les machines du PUIO, les pages ne sont pas accessibles depuis un réseau extérieur.

Réponse: Seule l'URL `file:///...` fonctionne. En effet, dans ce mode le navigateur ouvre le fichier en le lisant directement sur le disque de la machine (sans passer par le réseau). Il a donc l'accès au répertoire contenant le fichier. Avec l'URL `https://.../ toto/test.txt`, le serveur Web est configuré pour n'exposer **que** les fichiers du (sous-)répertoire `public_html`.

Remarque pour toute la suite du TP, on pourra choisir de charger les pages directement via l'URL `file:///` ou `https://tp-ssh1....` de manière indifférente.

4 Création de page Web

Le but de cet exercice est la création d'une page personnelle. Dans un premier temps on ne créera qu'une simple page HTML, sans information de styles. La feuille de style sera ajoutée dans l'exercice suivant. Dans tout cet exercice, si les supports de cours ne vous donnent pas assez d'indications, vous pouvez utiliser le site W3Schools et en particulier sa référence sur HTML qui explique au moyen d'exemples et de démo la spécification du langage HTML :

`http://www.w3schools.com/tags/default.asp`

Entre chaque question, on s'attachera à toujours créer une page *valide*. Pour savoir si une page est valide, on peut utiliser le validateur du W3C :

`http://validator.w3.org/#validate_by_upload`

Attention. La machine `tp-ssh1.dep-informatique.u-psud.fr` n'est pas accessible en HTTP depuis l'extérieur, il faut donc uploader votre fichier sur le site du validateur.

1. Créer une page vide, mais valide. Cette dernière doit contenir la balise `html`, la balises `head` et `body` et une balise `title` dans la section `head`. Où s'affiche le texte que vous avez mis dans la balise `title`? Utiliser le validateur pour vérifier votre document. Ce dernier est-il valide?

Réponse:

2. Le titre de la page s'affiche dans la barre de fenêtre du navigateur et dans les onglets si ceux-ci sont apparents. Le document demandé n'est **pas** valide. Il faut préciser deux choses :
   — une ligne `<!DOCTYPE html>` en tout début de fichier, qui indique que le document est au format HTML5
   — une ligne `<meta charset="utf-8" />` dans la partie `head` qui indique que le fichier est encodé en utf-8.

3. Ajouter des caractères accentués (çàé) dans la partie `body`. Les caractères s'affichent ils correctement? Changez maintenant l'attribut `charset` de la balise `meta` pour qu'il vaille `"latin-1"`. Rechargez la page. Les caractères s'affichent ils correctement? pourquoi? (à la fin de la question, effacez les caractères accentués et revenez au document de la question 1).

Réponse: Dans un premier temps, les caractères s'affichent correctement. Lorsque l'on change l'attribut `charset`, on annonce au navigateur que le contenu du fichier sera en `latin-1` (ascii 8 bits) ce qui n'est pas le cas. Firefox interprète donc les octets un par un au lieu de les grouper en séquences UTF-8 et les affiche individuellement.

4. Rajouter un titre de section en utilisant la balise `<h1>Page personnelle de ...</h1>` (insérez votre nom). La page est-elle valide?

Réponse: Le document est valide (mais on ne peut pas supprimer un Warning qui est lié à l'utilisation de HTML5, encore considéré comme expérimental).

5. Rajouter du texte (un court paragraphe vous décrivant ou un texte générique à la *lorem ipsum*) sous la balise `h1`. La page est-elle valide? placez ensuite votre texte dans une balise `<p> ... </p>` et re-validez.

Réponse: Mettre du texte dans le body n'est pas autorisé, il faut que le texte soit dans un élément p ou div.

6. Créez une table avec votre emploi du temps de la semaine, sous votre paragraphe. Une table se crée en utilisant la balise <table>...</table>. Dans cette balise, on peut utiliser des balises <tr> ...</tr> pour chaque lignes. Chaque balise <tr>...</tr> peut contenir une suite de balise <td>...</td> pour les colonnes. On souhaite que la table ait 5 lignes (une ligne de titre avec les jours de la semaine, lundi - vendredi et 4 lignes 2 pour les créneaux du matin et 2 pour les créneaux de l'après-midi). La table doit avoir 6 colonnes, une pour chaque jours de la semaine (sans les week-end) et une pour l'intitulé de chaque créneau.

Réponse: On peut mettre les étudiants sur la voie en donnant un exemple pour les table/tr/td (la première ligne par exemple).

7. Mettre les titre de colonne et de lignes en gras

Réponse: Il suffit de rajouter des balises autour du texte dans les bonnes cases.

8. rajouter sous la table une liste non-énumérée avec vos loisirs/activités extra-universitaires, précédé d'un titre de sous-section *Loisirs* (balise <h2>)
9. rajouter ensuite une liste énumérée de choses à faire (*todo-list*), précédé d'un titre de sous-section *Choses à faire*

Réponse: pas de difficulté

10. Rajouter une nouvelle sous-section intitulée « où me trouver ? ». Ajouter ensuite une balise <p> contenant un lien vers l'URL suivante :

    `https://goo.gl/maps/cKufC`

11. Rajouter enfin une sous-section « liens utiles » avec 5 ou 6 liens (ces derniers étant du texte, ils doivent êtres dans une balise p ou div
12. Enfin, une fois que votre document est valide, rajoutez l'image se trouvant à l'URL :

    `http://www.w3.org/html/logo/downloads/HTML5_Logo_64.png`

    (et revalidez une dernière fois).

Réponse: Il ne faut pas oublier l'attribut alt avec un texte décrivant l'image.

5 Redirections HTML

1. Il est possible, au moyen d'une balise meta de une redirections simplement en HTML (sans manipuler la configuration du serveur Web).
   — créer une page simple (mais valide) redir.html, similaire à la question 1 de l'exercice précédant.
   — Rajouter dans la section <head></head> une balise :

   `<meta http-equiv="refresh" content="`*X*`;url=`*nouvelle URL*`" />`

   où *X* est un nombre de secondes et *nouvelle URL* est l'URL de la page crée à l'exercice précédant.
   chargez votre page redir.html dans le navigateur Web. Que constatez vous ?

Réponse: Au bout de X secondes, le navigateur recharge automatiquement la nouvelle page.

2. Comment peut on utiliser cette technique pour forcer une page à se recharger toutes les 3 minutes ?

Réponse: Il suffit de mettre un nombre de secondes de 180 et de mettre l'URL de la page elle même dans l'attribut `content` de la balise `meta`.

3. Les URLs d'un site peuvent êtres longues (et donc mal commodes), pour plusieurs raisons :
   — parce qu'elles reflètent la structure d'un site complexe, par exemple

     `http://www.u-psud.fr/fr/formations/diplomes/licences/sciences-technologies-sante/informatique.html`

   — parce qu'elles encodent des données (par exemple une URL Google Map encode la position GPS des repères sur la carte)

   Pour palier à ce problème, certains sites (`bitly.com`, `tinyurl.com`, `goo.gl`) proposent de « stocker » un alias pour une URL longue sous forme courte. L'url de la licence informatique ci-dessus est par exemple accessible via : `http://goo.gl/hzNjDJ`. Comment pourriez vous mettre en place un tel service sur votre page au 336 ?

Réponse: Il suffit de créer un *répertoire* unique pour chaque URL, et d'y placer un fichier `index.html` contenant la balise `meta` redirigeant vers l'URL.

6 Le PHP du pauvre en Python

Le but de cet exercice est d'écrire un programme Python qui *génère* du HTML. Cet exercice permet donc :
— de souligner qu'écrire un programme qui génère du HTML *valide* est un exercice difficile
— d'illustrer le fonctionnement « calcul côté serveur », qui sera approfondit lors des blocs 3 et suivants.

1. Écrire un programme Python contenant une fonction `gen_table(n)` qui prend en argument un entier `n` supposé [2] être entre `1` et `9` et qui renvoie un tableau de chaînes de caractères représentant la table de multiplication de `n`. Par exemple, `gen_table(2)` renvoie le tableau :

   `["2 * 0 = 0", "2 * 1 = 2", "2 * 2 = 4", ..., "2 * 10 = 20"]`

2. Écrire une fonction `html_table(n)` qui appelle la fonction `gen_table(n)`. Une fois cette table créée, la fonction ouvre le fichier `resultat.html` en écriture, puis y écrit un document HTML dont le corps (document `body`) contient la table présentée en HTML. Par exemple, appeler `html_table(2)` doit produire un fichier `resultat.html` contenant :

```
<!DOCTYPE html>
<html>
 <head>
   <title>Table de multiplication</title>
 </head>
 <body>
   <ul>
         <li> 2 * 0 = 0 </li>
         <li> 2 * 1 = 2 </li>
         <li> 2 * 2 = 4 </li>
         ...
    </ul>
 </body>
</html>
```

   Valider le fichier `resultat.html` comme dans l'exercice 4. On rappelle les fonctions Python pour la manipulation de fichier :

```
#ouvre le fichier "monfichier.txt" en écriture seulement
#le crée s'il n'existe pas et écrase son contenu s'il existe

fichier = open ("monfichier.txt", "w")

#écrit une chaine caractères dans le fichier f, sans retour à la ligne
print("abc", end='', file=f)
```

2. votre fonction a le droit de faire ce qu'elle veut si `n` n'est pas dans les bornes.

```
print("def", end='', file=f)

#écrit une chaine caractères dans le fichier f, avec retour à la ligne
print("ghi", file=f)
print("jkl", end='', file=f)

#ferme le fichier
fichier.close()
```

Après exécution du programme ci-dessus, le fichier `monfichier.txt` contiendra :

```
abcdefghi
jkl
```

7 Sélecteurs CSS

1. ⋆ On considère le document représenté à la figure 1. Pour chacune des expressions de sélection CSS

```
<html>
  <head><title>Exercice 1</title></head>
  <body>
    <div id="i1">
      <ul id="i2" class="pair" >
        <li id="i3" class="impair" > texte </li>
        <li id="i4" class="pair" > <a id="i5" href="">texte </a> </li>
        <li id="i6" class="impair" > <b id="i7">  <a id="i8" href="">texte </a> </b></li>
        <li id="i9" class="pair" > <a id="i10" href="">texte </a> </li>
        <li id="i11" class="impair" > <b id="i12">  <a id="i13" href="">texte </a> </b> </li>
      </ul>
    </div>
    <div id="i14">
    <a id="i15" href=""> texte </a>
    <b id="i16"> <i id="i17">     <a id="i18" href=""> texte </a> </i> </b>

    </div>
  </body>
</html>
```

Figure 1 – Un document HTML simple

suivantes, dire en français quels éléments sont sélectionnés puis lister les valeurs des attributs `id` des éléments sélectionnés dans le document.

(a) `div`

(b) `div *`

(c) `div > * > *`

(d) `.impair a`

(e) `*.pair > *:hover`

(f) `div * b a`

Réponse:

(a) `div` : tous les éléments `div` du document (ici `i1` et `i14`)

(b) `div *` : tous les éléments se trouvant sous un `div` (à n'importe quelle profondeur. Ici tous les éléments ayant un `id` sauf `i1` et `i14`).

(c) `div > * > *` : tous les éléments se trouvant exactement deux niveaux sous un `div` (*i.e* qui sont à l'intérieur d'une balise qui est elle-même dans un `div`). Ici `i3`, `i4`, `i6`, `i9`, `i11` et `i17`

(d) `.impair a` : les balises `a` se trouvant sous les balises ayant la classe `impair`. Ici `i7`, `i8`, `i12`, `i13`.

(e) `*.pair > *:hover` : N'importe quel élément suvolé par le curseur de la souris et qui est exactement un niveau sous un élément dont la classe est `pair`. Ici `i5` ou `i10` (si la souris les survole).

(f) `div * b a` : Toutes les balises `a` se trouvant à n'importe quelle profondeur sous une balise `b` elle même se trouvant à profondeur 2 ou plus sous une balise `div`. Ici : `i8` et `i13`.

2. Testez vos réponses de la manière suivante. Créez un fichier `exo1.css`, contenant :

```
... { background: red;
      border: 1pt dashed black;
    }
```

où `...` est à remplacer par l'expression CSS de la question. Recopiez le document HTML de la figure 1 et ajoutez dans la section `head` l'élément :

```
<link rel="stylesheet" type="text/css" href="exo1.css"  />
```

Modifier le fichier CSS et recharger la page HTML pour vérifier que les éléments sélectionnés s'affichent en rouge.

8 Propriétés CSS

On souhaite maintenant écrire une feuille de style pour la page de garde faite à l'exercice 4. Dans tout ce qui suit, et sauf mention contraire, on écrira toutes les propriétés CSS dans un fichier CSS séparé, référencé par la page HTML. On n'utilisera pas l'attribut `style=". . . "` mais on fera au contraire un usage judicieux des attributs `class`, `id` et des sélecteurs CSS.

1. créer un fichier `exo2.css` et le référencer dans la page HTML

Réponse: Il suffit de rajouter la balise ci-dessous dans le `head` : `<link rel="stylesheet" type="text/css" href="style.css" title="Style" />`

2. Faites en sorte que tous les éléments de la page utilisent une police sans sérif (`font-family: sans`), en utilisant le sélecteur `*`. Faites de même en utilisant uniquement le sélecteur `body` puis le sélecteur `table`. Que constatez vous ? Que peut-on en déduire sur la transmission de la propriété `font-family` aux sous-éléments ?

Réponse: En utilisant `body`, le résultat est le même qu'avec `*`, en utilisant `table` seul la table est en police *sans*. On peut en déduire que l'attribut `font-family` est *hérité* et qu'il est propagé à toutes les balises se trouvant sous la balise touchée par le sélectionneur CSS.

3. Centrez le titre horizontalement (élément `h1`, propriété `text-align`), soulignez le et changez sa couleur (selon votre goût)

Réponse: voir le fichier exo2.css et exo2.html pour cette question et les suivantes.

4. Faites en sorte que la première lettre du paragraphe de description soit en gras et très grande. Vous pouvez mettre uniquement la première lettre dans un élément `<span id="p1">C</span>` (si C est la première lettre de votre paragraphe) puis assigner un style particulier à l'élément d'id `p1`. Utilisez une police de taille 200%.
5. Forcer la largeur de tous les paragraphes (éléments `p`) à etre de `40em`.
6. faites en sorte que le haut de la première lettre du paragraphe (l'élément `span` d'id `p1`) soit alignée avec le texte (et non pas le bas). Rajoutez la propriété `float:left;` à cet élément. Que constatez vous ?

Réponse: en faisant cela, le texte vient se coller le long de la lettre plus grande (effet de lettrine, ou « habillage de texte » sous Word).

7. faites en sorte que chaque case de la table fasse 10em de largeur, 3em de hauteur, et que le texte s'y trouvant soit centré horizontalement et verticalement.
8. faites en sorte que les lignes (élément `tr`) paires aient un fond gris foncé et les lignes impaires aient un fond clair (vous pouvez vous inspirer de l'exercice 1)
9. faites en sorte que la première case de la table (en haut à gauche) ait un fond blanc (en utilisant un id)
10. faites en sorte que lorsqu'on survole l'une des cases, sauf la première, le fond et le texte changent de couleur
11. faites en sorte que les liens hypertexte de la page ne soient pas soulignés et soient encadrés en bleu. Lorsqu'un lien est visité le cadre doit devenir rouge.
12. faites en sorte que le bloc contenant des listes non énumérées aient une largeur de 200pt et un fond gris clair
13. faites en sorte que la couleur des titres de sous-section soit la même que celle du titre de la page
14. faites en sorte que l'image HTML5 se trouvant en fin de fichier apparaisse toujours en bas à droite de la fenêtre du navigateur (même si on défile). Vous devrez modifier le type de position et les propriétés right et bottom.
15. redimentionnez la fenêtre du navigateur de manière à ce qu'apparaisse des barres de défilement. Changez la propriété `position` de l'image à `absolute`. Que se passe-t-il ?
16. Faites en sorte que votre texte de paragraphe introductif soit très long (au pire, copiez-collez le plusieurs fois). Forcer la hauteur de l'élément `p` correspondant à faire au maximum 100pt. Que se passe-til pour le texte ? Mettez le fond du paragraphe en jaune (composante rouge et verte plus forte que le bleu ou simplement `yellow`). Faites en sorte que le texte puisse défiler et ne dépasse pas de la boite.
17. faites en sorte que l'image HTML5 se retrouve dans le coin supérieur gauche de la page lorsqu'on la survole
18. faites en sorte que les textes *items* de listes énumérées (et seulement ceux-là) deviennent italiques quand ils sont survolés. Votre code CSS doit fonctionner même si une liste non-énumérée se trouve sous une liste énumérée. Par exemple dans le cas suivant :

    (a) A
    (b) — X
        — Y
    (c) B

    les textes A et B doivent passer en italique mais pas les X, Y

Réponse:

19. il faut utiliser `ol > li:hover` pour s'assurer que seul les fils direct d'un `ol` sont touchés

9 Problème de cascades

On considère le document ci-dessous :

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page de garde</title>
    <meta charset="utf-8"/>
    <style>
    body {
        font-size: 20pt;
    }

    .smaller {
     font-size:50%;
```

```
    }
    </style>
  </head>
  <body>A
    <div class="smaller"> B
      <div class="smaller"> C
       <div class="smaller"> D
        <div class="smaller"> E
         <div class="smaller"> F
          <div class="smaller"> G
          </div>
         </div>
        </div>
       </div>
      </div>
    </div>
  </body>
</html>
```

1. ⋆ Quelle est la taille (en points) de la lettre A? Quelle sont censé être les tailles pour les autres lettres (de B à G)?

> **Réponse:** 20pt pour A. Normalement 20 $\times$ 0.5 = 10pt pour B, 5pt pour C, 2.5pt pour D, 1.25pt pour E, 0.625pt pour F et 0.3125pt pour G.

2. Copiez le code HTML dans un fichier et observez la taille des caractères. Obtient on le résultat attendu ?

> **Réponse:** Non, à partir d'une certaine taille (la lettre C normalement) la taille arrête de décroître. Cela est du au fait que les navigateurs imposent par défaut une taille de police minimale pour éviter d'avoir du texte illisible. On peut ignorer cette limite sous Firefox en allant dans les Préférences $\rightarrow$ Contenu $\rightarrow$ Polices et couleurs $\rightarrow$ Avancé $\rightarrow$ Taille minimale de police : aucune.

3. Proposez un ajout à l'élément `<style></style>` pour que la taille arrête de décroître au delà de 3 niveaux d'imbrication d'éléments de classe `smaller`.

> **Réponse:** On peut ajouter :
>
> ```
> .smaller .smaller .smaller  font-size:100%;
> ```