Chapitre 3

Le Web

Objectifs du chapitre

Thème : Le Web.
Comprendre comment fonctionne le Web : pages écrites en HTML et mises en forme en CSS, adresses URL, dialogue HTTP entre client et serveur, fonctionnement des moteurs de recherche et réglages de sécurité et de confidentialité du navigateur.
Organisation : programme officiel, cours, méthodes, ateliers Python, exercices corrigés, QCM et fiche de synthèse.

Programme officiel (BO)

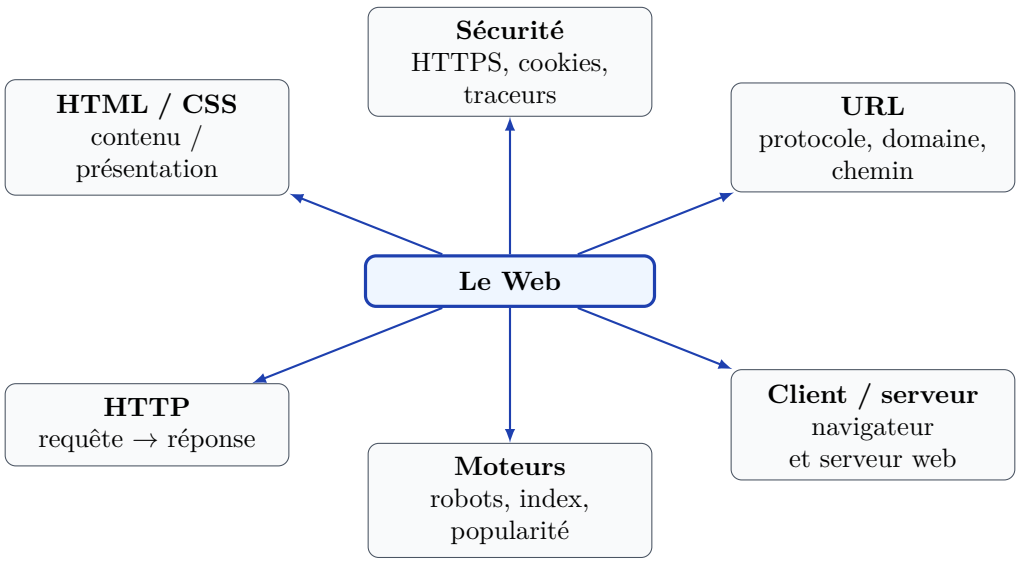
Thème « Le Web » – Contenus et capacités attendues

- **Repères historiques** : Connaître les étapes du développement du Web.
- **Notions juridiques** : Connaître certaines notions juridiques (licence, droit d'auteur, droit d'usage, valeur d'un bien).
- **Hypertexte** : Maîtriser les renvois d'un texte à différents contenus.
- **Langages HTML et CSS** : Distinguer ce qui relève du contenu d'une page et de son style de présentation. Étudier et modifier une page HTML simple.
- **URL** : Décomposer l'URL d'une page. Reconnaître les pages sécurisées.
- **Requête HTTP** : Décomposer le contenu d'une requête HTTP et identifier les paramètres passés.
- **Modèle client/serveur** : Inspecter le code d'une page hébergée par un serveur et distinguer ce qui est exécuté par le client et par le serveur.
- **Moteurs de recherche : principes et usages** : Mener une analyse critique des résultats fournis par un moteur de recherche. Comprendre les enjeux de la publication d'informations.
- **Paramètres de sécurité d'un navigateur** : Maîtriser les réglages les plus importants concernant la gestion des cookies, la sécurité et la confidentialité d'un navigateur. Sécuriser sa navigation en ligne et analyser les pages et fichiers.

Source : BO spécial n°1 du 22 janvier 2019, programme de Sciences numériques et technologie de Seconde générale et technologique.

3.1 Carte mentale du chapitre

Carte mentale



3.2 Repères historiques et notions juridiques

Le **Web** (toile) est un système qui donne accès à des pages et des documents reliés entre eux par des **liens hypertextes**. Il fonctionne **au-dessus d'Internet** : Internet transporte les paquets, le Web est une application qui l'utilise.

Date	Repère
1965	Ted Nelson invente le mot **hypertexte** : un texte qui contient des renvois vers d'autres textes.
1989	Au **CERN** (Genève), Tim Berners-Lee propose un système de documents reliés : il invente le **HTML**, les **URL** et le protocole **HTTP**.
1991	Le premier site web est mis en ligne ; en 1993, le CERN place la technologie du Web dans le domaine public.
1993	Mosaic, premier navigateur grand public affichant textes et images sur la même page.
1994	Création du **W3C**, qui définit les standards du Web (HTML, CSS).
1998	Création de Google ; les moteurs de recherche deviennent la porte d'entrée du Web.
2004 et après	**Web 2.0** : les utilisateurs publient eux-mêmes (blogs, Wikipédia, réseaux sociaux) ; essor du Web sur mobile.

Définition – Droit d'auteur et licences

- Le **droit d'auteur** protège automatiquement une œuvre (texte, photo, musique, programme) dès sa création : on ne peut pas la copier ni la diffuser sans l'accord de son auteur.
- Une **licence** est un contrat par lequel l'auteur précise les **droits d'usage** qu'il accorde : copier, modifier, utiliser commercialement…
- Les licences **Creative Commons** permettent de partager une œuvre en fixant des conditions : **BY** (citer l'auteur), **NC** (pas d'usage commercial), **ND** (pas de modification), **SA** (partager dans les mêmes conditions).
- Un contenu gratuit sur le Web n'est pas pour autant libre de droits : une œuvre a une **valeur**, même si son coût de copie est presque nul.

3.3 Hypertexte, HTML et CSS

Définition – HTML et CSS

— Le **HTML** (*HyperText Markup Language*) décrit le **contenu** et la **structure** d'une page : titres, paragraphes, liens, images. Il utilise des **balises** comme `<p> ... </p>`.
— Le **CSS** (*Cascading Style Sheets*) décrit la **présentation** : couleurs, polices, tailles, mise en page. Une même page HTML peut changer complètement d'apparence en changeant de feuille de style.

Voici une page HTML simple. Le navigateur lit ce code et affiche la page.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Mon club de robotique</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <h1>Club de robotique</h1>
    <p>Rendez-vous le <strong>mercredi</strong> en salle B12.</p>
    <a href="https://fr.wikipedia.org/wiki/Robotique">En savoir plus</a>
    <img src="robot.png" alt="Notre robot">
  </body>
</html>
```

Balise	Rôle
`<h1>` ... `<h6>`	titres, du plus important au moins important
`<p>`	paragraphe
`<strong>`, `<em>`	texte important (gras), texte mis en valeur (italique)
`<a href="...">`	**lien hypertexte** vers une autre page ou une autre ressource
`<img src="..." alt="...">`	image ; `alt` décrit l'image pour les personnes malvoyantes
`<ul>`, `<li>`	liste à puces et élément de liste

La feuille de style `style.css` associée met en forme les titres et les paragraphes :

```
h1 {
  color: darkblue;
  text-align: center;
}
p {
  font-size: 18px;
}
```

Méthode – Distinguer contenu et présentation

Pour savoir si une information relève du HTML ou du CSS, on se demande : « si je change de feuille de style, cette information reste-t-elle ? ». Le texte d'un titre est du **contenu** (HTML) ; sa couleur bleue est de la **présentation** (CSS). Dans un navigateur, le raccourci **Ctrl + U** affiche le code source d'une page ; la touche **F12** ouvre l'inspecteur, qui permet de modifier le HTML et le CSS en direct (sans toucher au serveur).

3.4 URL et requêtes HTTP

Définition – URL

Une **URL** (*Uniform Resource Locator*) est l'adresse d'une ressource sur le Web. Par exemple :
`https://www.exemple.fr/sciences/snt/cours.html?chapitre=3#exercices`
— `https://` : le **protocole** ;
— `www.exemple.fr` : le **nom de domaine** du serveur (traduit en adresse IP par le DNS) ;
— `/sciences/snt/cours.html` : le **chemin** de la ressource sur le serveur ;
— `?chapitre=3` : les **paramètres**, sous la forme `nom=valeur`, séparés par `&` ;
— `#exercices` : l'**ancre**, qui désigne un endroit précis de la page.

Animation interactive « Décomposer une URL » : à manipuler sur https://utspe.com/snt/chapitre-3.

Définition – Protocole HTTP

Le protocole **HTTP** (*HyperText Transfer Protocol*) fixe les règles du dialogue entre le navigateur (le **client**) et le **serveur web** :
1. le client envoie une **requête** : une méthode (`GET` pour obtenir une page, `POST` pour envoyer un formulaire), le chemin de la ressource et des en-têtes ;
2. le serveur renvoie une **réponse** : un **code de statut**, des en-têtes, puis le contenu (HTML, image…).

Exemple de requête HTTP envoyée par le navigateur :

```
GET /recherche?q=robot&page=2 HTTP/1.1
Host: www.exemple.fr
User-Agent: Mozilla/5.0 (Windows NT 10.0) Firefox/131.0
Accept-Language: fr-FR
Cookie: session=8f3a2c
```

Et la réponse du serveur :

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5120
Set-Cookie: session=8f3a2c

<!DOCTYPE html>
<html> ... </html>
```

Code	Signification
200	OK : la ressource est envoyée
301	la ressource a été déplacée définitivement (redirection)
403	accès interdit
404	ressource introuvable
500	erreur interne du serveur

Animation interactive « Le dialogue HTTP pas à pas » : à manipuler sur https://utspe.com/snt/chapitre-3.

Propriété – Pages sécurisées : HTTPS

Avec **HTTPS**, les échanges entre le navigateur et le serveur sont **chiffrés** et le serveur prouve son identité grâce à un **certificat**. Le navigateur l'indique par un **cadenas** devant l'adresse. Avec HTTP simple, une personne sur le même réseau Wi-Fi peut lire les données échangées.

Attention

Le cadenas garantit que la communication est chiffrée et que l'on parle bien au serveur du nom de domaine affiché, **pas** que le site est honnête : un site d'hameçonnage peut très bien être en HTTPS. Il faut toujours vérifier le nom de domaine.

3.5 Le modèle client/serveur

Définition – Client et serveur

— Le **client** est le programme qui demande une ressource : le navigateur web (Firefox, Chrome, Safari…).
— Le **serveur** est un ordinateur, souvent dans un centre de données, qui attend les requêtes et y répond. Un même serveur répond à des milliers de clients.

	Exécuté par le serveur	**Exécuté par le client (navigateur)**
Exemples de langages	PHP, Python, Java…	HTML, CSS (affichage), JavaScript
Rôle	fabriquer la page à la demande : consulter une base de données, vérifier un mot de passe	afficher la page, réagir aux clics, animer, vérifier un formulaire avant envoi
Visible avec Ctrl + U ?	non : on ne voit que le résultat (le HTML produit)	oui : le code est envoyé au navigateur

Exemple – Une page dynamique

Sur un site de vente, la page « Mon panier » est différente pour chaque client : le programme exécuté sur le serveur lit le contenu du panier dans une base de données et **génère** le HTML correspondant. Le navigateur reçoit ce HTML et l'affiche ; un script JavaScript peut ensuite recalculer le total quand on change une quantité, sans recharger la page.

3.6 Les moteurs de recherche

Propriété – Fonctionnement d'un moteur de recherche

1. **Exploration** : des robots (*crawlers*) parcourent le Web en suivant les liens et téléchargent les pages.
2. **Indexation** : pour chaque mot, le moteur enregistre la liste des pages qui le contiennent ; c'est l'**index**, comme à la fin d'un livre.
3. **Classement** : pour une requête, le moteur cherche dans l'index les pages qui contiennent les mots, puis les classe selon leur pertinence et leur **popularité**.
4. **Affichage** des résultats, souvent précédés de liens commerciaux (publicités).

Définition – Popularité d'une page (PageRank)

On représente le Web par un **graphe** : chaque page est un sommet et chaque lien est une flèche. Une page est d'autant plus **populaire** que des pages elles-mêmes populaires pointent vers elle. C'est l'idée de l'algorithme **PageRank** de Google (1998) : un internaute qui clique au hasard sur les liens passe plus souvent sur les pages populaires.

Attention

Les résultats d'un moteur de recherche ne sont **ni neutres ni complets** : ils dépendent de l'algorithme de classement, de la publicité, de la position de l'utilisateur et de son historique. Il faut croiser les sources, vérifier l'auteur et la date d'une information, et comparer plusieurs moteurs.

Publier sur le Web, c'est rendre une information accessible au monde entier, pour longtemps : elle peut être copiée, indexée et retrouvée des années plus tard.

3.7 Sécurité et confidentialité dans le navigateur

Définition – Cookies et traceurs

— Un **cookie** est un petit fichier texte déposé par un site dans le navigateur. Il sert à garder une session ouverte, mémoriser un panier ou une langue.
— Les **cookies tiers** sont déposés par d'autres sites que celui visité (régies publicitaires, réseaux sociaux) : ils servent de **traceurs** pour suivre l'internaute de site en site et construire son profil.
— Le **RGPD** impose de demander le consentement de l'utilisateur pour les cookies non indispensables : refuser doit être aussi simple qu'accepter.

Méthode – Réglages essentiels du navigateur

— bloquer les cookies tiers et activer la protection contre le pistage ;
— utiliser la **navigation privée** : elle n'enregistre pas l'historique sur l'ordinateur, mais ne rend pas anonyme vis-à-vis des sites ou du fournisseur d'accès ;
— effacer régulièrement l'historique et les cookies ; gérer les autorisations (position, caméra, micro, notifications) site par site ;
— activer le mode « HTTPS uniquement » et maintenir le navigateur à jour ;
— ne pas ouvrir un fichier téléchargé sans vérifier son origine ; se méfier des extensions inconnues.

3.8 Ateliers Python

Les ateliers se font sur ordinateur, dans un éditeur Python ou directement sur `https://utspe.com/snt/chapitre-3`. Pour chacun : recopier le programme de départ, suivre les étapes dans l'ordre et exécuter le programme après chaque modification.

Atelier 1 – Décomposer une URL chaînes, find(), split()

Objectif : Écrire un programme qui extrait le protocole, le nom de domaine et le chemin d'une URL.

1. `url.split("://")` coupe l'URL en deux morceaux : avant et après `://`. Exécuter le programme et observer.
2. Dans `reste`, la position du premier `/` est donnée par `reste.find("/")`. Compléter pour obtenir le domaine (avant ce `/`) et le chemin (à partir de ce `/`).

3. Afficher `Page securisee` si le protocole est `https`, sinon `Page NON securisee`.
4. Tester avec `http://www.site-ancien.fr/index.html`.

Programme de départ :

```
url = "https://fr.wikipedia.org/wiki/World_Wide_Web"

morceaux = url.split("://")
print(morceaux)
protocole = morceaux[0]
reste = morceaux[1]

position = reste.find("/")
domaine = ""       # a completer
chemin = ""        # a completer

print("Protocole :", protocole)
print("Domaine   :", domaine)
print("Chemin    :", chemin)
```

Coup de pouce : Les tranches de chaînes : `reste[:position]` donne les caractères avant la position, `reste[position:]` ceux à partir de la position.

Corrigé rédigé

```
url = "https://fr.wikipedia.org/wiki/World_Wide_Web"

morceaux = url.split("://")
protocole = morceaux[0]
reste = morceaux[1]

position = reste.find("/")
domaine = reste[:position]
chemin = reste[position:]

print("Protocole :", protocole)
print("Domaine   :", domaine)
print("Chemin    :", chemin)
if protocole == "https":
    print("Page securisee")
else:
    print("Page NON securisee")
```

Atelier 2 – Lire les paramètres d'une requête

split(), boucle for, dictionnaire

Objectif : Récupérer les paramètres passés dans une URL, comme le fait un serveur web qui reçoit une requête GET.

1. Les paramètres sont après le `?`. Exécuter le programme pour obtenir la chaîne `q=robot&page=2&lang=fr`.
2. Découper cette chaîne avec `split("&")` pour obtenir la liste des couples `nom=valeur`.
3. Pour chaque couple, le découper avec `split("=")` et ranger la valeur dans le dictionnaire `parametres`.
4. Afficher `Recherche de : robot (page 2)` à partir du dictionnaire.

Programme de départ :

```
url = "https://www.exemple.fr/recherche?q=robot&page=2&lang=fr"
```

```

chaine = url.split("?")[1]
print(chaine)

parametres = {}
# a completer : remplir le dictionnaire

print(parametres)
```

Coup de pouce : `for couple in chaine.split("&"):` puis `nom, valeur = couple.split("=")` et `parametres[nom] = valeur`.

Corrigé rédigé

```
url = "https://www.exemple.fr/recherche?q=robot&page=2&lang=fr"

chaine = url.split("?")[1]
parametres = {}
for couple in chaine.split("&"):
    nom, valeur = couple.split("=")
    parametres[nom] = valeur

print(parametres)
print(f"Recherche de : {parametres['q']} (page {parametres['page']})")
```

Atelier 3 – Un serveur qui fabrique une page HTML

f-strings, listes, boucle for

Objectif : Simuler un programme exécuté côté serveur qui génère le code HTML d'une page à partir de données.

1. Exécuter le programme : il produit le début d'une page HTML avec un titre.
2. Compléter la boucle pour ajouter une ligne `<li>...</li>` par activité de la liste.
3. Ajouter un lien hypertexte `<a href="https://utspe.com">Cours de SNT</a>` à la fin.
4. Ajouter une activité à la liste et vérifier que la page générée change : c'est le principe d'une page dynamique.

Programme de départ :

```
club = "Club de robotique"
activites = ["Construire un robot", "Programmer en Python", "Participer a la coupe"]

page = "<html>\n<body>\n"
page = page + f"<h1>{club}</h1>\n"
page = page + "<ul>\n"
# a completer : une ligne <li> par activite
page = page + "</ul>\n"
page = page + "</body>\n</html>"
print(page)
```

Coup de pouce : `for a in activites:` puis `page = page + f"<li>{a}</li>\n"`.

Corrigé rédigé

```
club = "Club de robotique"
activites = ["Construire un robot", "Programmer en Python", "Participer a la coupe"]

page = "<html>\n<body>\n"
```

```
page = page + f"<h1>{club}</h1>\n"
page = page + "<ul>\n"
for a in activites:
    page = page + f"  <li>{a}</li>\n"
page = page + "</ul>\n"
page = page + '<a href="https://utspe.com">Cours de SNT</a>\n'
page = page + "</body>\n</html>"
print(page)
```

Atelier 4 – Construire l'index d'un moteur de recherche

dictionnaires, listes, split()

Objectif : Indexer quelques pages : pour chaque mot, la liste des pages qui le contiennent. Puis répondre à une requête.

1. Exécuter le programme : il parcourt les mots de chaque page.
2. Compléter pour que `index[mot]` soit la liste des pages contenant `mot` (créer la liste si le mot est nouveau).
3. Afficher les pages qui contiennent le mot `python`, puis le mot `web`.
4. Défi : afficher les pages qui contiennent à la fois `python` et `web`.

Programme de départ :

```
pages = {
    "page1": "python est un langage de programmation",
    "page2": "le web relie des pages par des liens",
    "page3": "programmer une page web en python",
}

index = {}
for nom, texte in pages.items():
    for mot in texte.split():
        print(nom, mot)
        # a completer

print(index)
```

Coup de pouce : `if mot not in index: index[mot] = []` puis `if nom not in index[mot]: index[mot].append(nom)`. Pour le défi, parcourir `index["python"]` et garder les pages qui sont aussi dans `index["web"]`.

Corrigé rédigé

```
pages = {
    "page1": "python est un langage de programmation",
    "page2": "le web relie des pages par des liens",
    "page3": "programmer une page web en python",
}

index = {}
for nom, texte in pages.items():
    for mot in texte.split():
        if mot not in index:
            index[mot] = []
        if nom not in index[mot]:
            index[mot].append(nom)

print("python :", index["python"])
```

```
print("web    :", index["web"])
les_deux = []
for p in index["python"]:
    if p in index["web"]:
        les_deux.append(p)
print("python ET web :", les_deux)
```

Atelier 5 – La popularité d'une page

dictionnaires, boucles imbriquées

Objectif : Programmer une version simplifiée de l'algorithme PageRank sur un petit graphe de quatre pages.

1. `liens["A"] = ["B", "C"]` signifie que la page A contient des liens vers B et C. Dessiner le graphe sur une feuille.
2. Au départ, chaque page a une popularité de 1. À chaque tour, chaque page **partage** sa popularité à parts égales entre les pages vers lesquelles elle pointe. Exécuter un tour.
3. Compléter la boucle pour faire 20 tours et afficher les popularités arrondies à 0,01.
4. Quelle page est la plus populaire ? Ajouter un lien de D vers A et observer.

Programme de départ :

```
liens = {"A": ["B", "C"], "B": ["C"], "C": ["A"], "D": ["C"]}
popularite = {"A": 1, "B": 1, "C": 1, "D": 1}

nouvelle = {"A": 0, "B": 0, "C": 0, "D": 0}
for page in liens:
    part = popularite[page] / len(liens[page])
    for cible in liens[page]:
        nouvelle[cible] = nouvelle[cible] + part
popularite = nouvelle
print(popularite)
```

Coup de pouce : Mettre le calcul d'un tour dans une boucle `for tour in range(20):`, en recréant le dictionnaire `nouvelle` à chaque tour. Pour l'affichage : `round(popularite[p], 2)`.

Corrigé rédigé

```
liens = {"A": ["B", "C"], "B": ["C"], "C": ["A"], "D": ["C"]}
popularite = {"A": 1, "B": 1, "C": 1, "D": 1}

for tour in range(20):
    nouvelle = {"A": 0, "B": 0, "C": 0, "D": 0}
    for page in liens:
        part = popularite[page] / len(liens[page])
        for cible in liens[page]:
            nouvelle[cible] = nouvelle[cible] + part
    popularite = nouvelle

for p in popularite:
    print(p, round(popularite[p], 2))
```

3.9 Exercices corrigés

Exercice

Exercice 1 – Contenu ou présentation ? Pour chaque élément, dire s'il relève du HTML (contenu) ou du CSS (présentation) : le texte d'un titre ; la couleur de fond de la page ; un lien vers Wikipédia ; la police des paragraphes ; le texte alternatif d'une image ; la largeur d'une colonne.

Corrigé rédigé

HTML : le texte du titre, le lien vers Wikipédia, le texte alternatif de l'image. CSS : la couleur de fond, la police des paragraphes, la largeur d'une colonne.

Exercice

Exercice 2 – Décomposer une URL. On considère l'URL `https://www.meteo.fr/previsions/ville.php?nom=Lyon&jours=3#carte`.
1. Donner le protocole, le nom de domaine, le chemin, les paramètres et l'ancre.
2. La page est-elle sécurisée ?
3. Quelle URL faut-il taper pour obtenir les prévisions de Marseille sur 7 jours ?

Corrigé rédigé

1. Protocole `https` ; domaine `www.meteo.fr` ; chemin `/previsions/ville.php` ; paramètres `nom=Lyon` et `jours=3` ; ancre `#carte`.
2. Oui : le protocole est HTTPS.
3. `https://www.meteo.fr/previsions/ville.php?nom=Marseille&jours=7`.

Exercice

Exercice 3 – Lire une requête HTTP. Un navigateur envoie la requête suivante :

```
GET /catalogue/livres?auteur=Hugo&tri=date HTTP/1.1
Host: www.librairie.fr
```

1. Quelle méthode est utilisée ? Quelle ressource est demandée, sur quel serveur ?
2. Quels paramètres sont transmis ?
3. Le serveur répond `HTTP/1.1 404 Not Found`. Que s'est-il passé ?

Corrigé rédigé

1. Méthode GET ; ressource `/catalogue/livres` sur le serveur `www.librairie.fr`.
2. `auteur` vaut `Hugo` et `tri` vaut `date`.
3. Le code 404 signifie que la ressource demandée n'existe pas sur ce serveur (adresse erronée ou page supprimée).

Exercice

Exercice 4 – Client ou serveur ? Sur un site de réservation de billets de train, dire si chaque action est exécutée par le client (navigateur) ou par le serveur : 1. vérifier qu'il reste des places ; 2. afficher un calendrier qui s'ouvre quand on clique sur la date ; 3. enregistrer la réservation ; 4. vérifier que le champ e-mail contient un @ avant l'envoi ; 5. envoyer l'e-mail de confirmation.

Corrigé rédigé

Serveur : 1, 3 et 5 (il faut accéder à la base de données des trains et des réservations). Client : 2 et 4 (scripts JavaScript exécutés par le navigateur, sans échange avec le serveur).

Exercice

Exercice 5 – Indexation à la main. Trois documents : D1 « le chat dort », D2 « le chien et le chat jouent », D3 « le chien dort ».
1. Construire l'index des mots `chat`, `chien`, `dort`.
2. Quels documents répondent à la requête « chat dort » (les deux mots présents) ? À « chien » ?
3. Écrire un programme qui construit l'index complet.

Corrigé rédigé

1. chat : D1, D2 ; chien : D2, D3 ; dort : D1, D3.
2. « chat dort » : seulement D1 (intersection de {D1, D2} et {D1, D3}). « chien » : D2 et D3.

```
docs = {"D1": "le chat dort", "D2": "le chien et le chat jouent", "D3": "le chien dort"}
index = {}
for nom, texte in docs.items():
    for mot in texte.split():
        if mot not in index:
            index[mot] = []
        if nom not in index[mot]:
            index[mot].append(nom)
for mot in ["chat", "chien", "dort"]:
    print(mot, ":", index[mot])
```

Exercice

Exercice 6 – Popularité. Quatre pages : A pointe vers B ; B pointe vers A et C ; C pointe vers A ; D pointe vers A.
1. Dessiner le graphe. Sans calcul, quelle page semble la plus populaire ? Pourquoi D est-elle la moins populaire ?
2. Vérifier avec le programme de l'atelier 5 (20 tours).

Corrigé rédigé

1. A reçoit des liens de B, C et D : elle semble la plus populaire. Aucune page ne pointe vers D : sa popularité devient nulle dès le premier tour.
2. Le programme donne environ A : 1,6 ; B : 1,6 ; C : 0,8 ; D : 0. A et B s'échangent leur popularité ; B est populaire car A, très populaire, ne pointe que vers elle.

```
liens = {"A": ["B"], "B": ["A", "C"], "C": ["A"], "D": ["A"]}
popularite = {"A": 1, "B": 1, "C": 1, "D": 1}
for tour in range(20):
    nouvelle = {"A": 0, "B": 0, "C": 0, "D": 0}
    for page in liens:
        part = popularite[page] / len(liens[page])
        for cible in liens[page]:
            nouvelle[cible] = nouvelle[cible] + part
    popularite = nouvelle
for p in popularite:
    print(p, round(popularite[p], 2))
```

Exercice

Exercice 7 – Licences Creative Commons. Une photo est publiée sous licence **CC BY-NC**.
1. Peut-on l'utiliser dans un exposé au lycée ? À quelle condition ?
2. Une entreprise peut-elle l'utiliser sur une affiche publicitaire ?
3. Une photo trouvée sur un site sans aucune mention de licence est-elle libre de droits ?

Corrigé rédigé

1. Oui, à condition de citer l'auteur (BY).
2. Non, l'usage commercial est interdit (NC), sauf accord de l'auteur.
3. Non : sans licence, le droit d'auteur s'applique, et toute réutilisation nécessite l'accord de l'auteur.

Exercice

Exercice 8 – Cookies et navigation privée. Vrai ou faux ? Justifier.
1. La navigation privée empêche le fournisseur d'accès de savoir quels sites je visite.
2. Un cookie tiers peut permettre à une régie publicitaire de me suivre sur plusieurs sites.
3. Refuser tous les cookies empêche de rester connecté à son compte sur un site.
4. Un site en HTTPS est forcément digne de confiance.

Corrigé rédigé

1. Faux : la navigation privée n'efface les traces que sur l'ordinateur.
2. Vrai : c'est le principe du pistage publicitaire.
3. Vrai en général : le cookie de session sert justement à rester connecté.
4. Faux : HTTPS chiffre la communication mais ne garantit pas l'honnêteté du site.

3.10 QCM – Teste tes connaissances

Pour chaque question, une seule réponse est correcte.

Q1. Le Web a été inventé…

a) en 1969 aux États-Unis par l'armée
b) en 1989 au CERN par Tim Berners-Lee
c) en 1998 par Google
d) en 2004 avec les réseaux sociaux

Q2. Le langage CSS sert à…

a) structurer le contenu d'une page
b) décrire la présentation d'une page
c) transmettre les pages
d) exécuter des programmes sur le serveur

Q3. Quelle balise crée un lien hypertexte ?

a) <link>
b) <a href="...">
c) <p>
d) <img src="...">

Q4. Dans https ://www.lycee.fr/snt/cours.html ?ch=3, le nom de domaine est…

a) https
b) www.lycee.fr
c) /snt/cours.html
d) ch=3

Q5. Dans l'URL précédente, ch=3 est…

a) un paramètre
b) une ancre
c) un protocole
d) un cookie

Q6. Le code de réponse HTTP 404 signifie…

a) tout s'est bien passé
b) accès interdit
c) ressource introuvable
d) erreur du serveur

Q7. La méthode HTTP utilisée pour demander une page est…

a) GET
b) SEND
c) PUT
d) ASK

Q8. Le cadenas devant l'adresse indique que…

a) le site est honnête
b) la communication est chiffrée
c) le site n'utilise pas de cookies
d) le site est hébergé en France

Q9. Lequel de ces langages est exécuté par le navigateur ?

a) PHP
b) SQL
c) JavaScript
d) aucun

Q10. Que produit ce programme ?

```
liens = {"A": ["B", "C"], "B": ["A"], "C": ["A"]}
print(len(liens["A"]))
```

a) 1
b) 2
c) 3
d) A

Q11. L'index d'un moteur de recherche associe…

a) à chaque page son adresse IP
b) à chaque mot la liste des pages qui le contiennent
c) à chaque internaute son historique
d) à chaque site son propriétaire

Q12. Un cookie tiers est…

a) un cookie déposé par un autre site que celui visité
b) un virus
c) un cookie indispensable à la connexion
d) un mot de passe enregistré

Corrigé rédigé

Réponses du QCM.
Q1. b) Tim Berners-Lee invente HTML, URL et HTTP au CERN en 1989 ; le premier site est en ligne en 1991.
Q2. b) HTML décrit le contenu et la structure, CSS la présentation.
Q3. b) La balise `<a>` avec l'attribut `href` crée un lien hypertexte.
Q4. b) Le nom de domaine se trouve entre :// et le premier /.
Q5. a) Les paramètres suivent le ? et ont la forme nom=valeur.
Q6. c) 200 : OK ; 403 : interdit ; 404 : introuvable ; 500 : erreur du serveur.
Q7. a) GET demande une ressource ; POST sert à envoyer les données d'un formulaire.
Q8. b) HTTPS chiffre les échanges ; cela ne garantit pas l'honnêteté du site.
Q9. c) JavaScript s'exécute côté client ; PHP et SQL s'exécutent côté serveur.
Q10. b) liens["A"] est la liste ["B", "C"], qui contient 2 éléments : la page A a 2 liens sortants.
Q11. b) L'index permet de trouver très vite les pages contenant les mots d'une requête.
Q12. a) Les cookies tiers, souvent publicitaires, permettent de suivre l'internaute de site en site.

3.11 Fiche de synthèse

Fiche de synthèse

— **Web et Internet :** Le Web (1989, CERN, Tim Berners-Lee) est une application d'Internet : des pages reliées par des liens hypertextes. Une œuvre publiée reste protégée par le droit d'auteur ; une licence précise les droits d'usage.

— **HTML et CSS :** HTML décrit le contenu et la structure avec des balises (`<h1>`, `<p>`, `<a href>`, `<img>`) ; CSS décrit la présentation.

— **URL :** protocole ://nom-de-domaine/chemin ?paramètres#ancre. HTTPS (cadenas) : communication chiffrée et serveur authentifié.

— **HTTP :** Le client envoie une requête (GET, POST, chemin, en-têtes), le serveur répond par un code (200, 404…) et le contenu.

— **Client/serveur :** Le serveur fabrique les pages (PHP, Python) ; le navigateur les affiche et exécute HTML, CSS et JavaScript.

— **Moteurs et sécurité :** Exploration, indexation, classement (popularité). Esprit critique sur les résultats. Bloquer les cookies tiers, navigation privée, HTTPS uniquement.