SNT — Le Web

Polycopié élève · parties W1 à W6

Julien Kergot

Partie W1 — Comment fonctionne le Web ?

Adresse, navigateur, serveur : le trajet d'une page jusqu'à ton écran

Tu ouvres un site, tu cliques sur un lien, une page apparaît. Derrière ce geste très simple, plusieurs ordinateurs se parlent. Dans ce thème, tu vas comprendre comment une page arrive jusqu'à toi, **fabriquer toi-même un petit site**, puis découvrir comment un moteur de recherche choisit l'ordre des résultats.

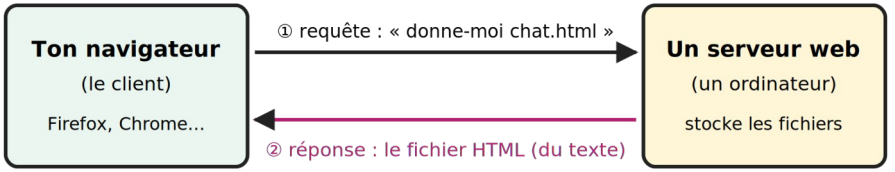
Cours · Internet et Web : deux choses différentes

- **Internet** est le réseau mondial d'ordinateurs et de câbles. Il transporte toutes sortes de données : courriers électroniques, appels vidéo, jeux en ligne, pages web…
- Le **Web** est un *service* qui fonctionne sur Internet : un immense ensemble de **pages reliées entre elles par des liens**. Il a été inventé au CERN (Suisse) à partir de 1989 par Tim Berners-Lee ; le premier site a été ouvert au public en 1991.

> **À retenir**
>
> Le Web n'est qu'une partie d'Internet. Un courrier électronique utilise Internet, mais pas le Web.

Cours · Le client et le serveur

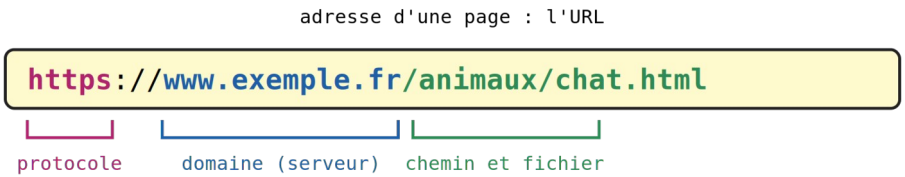
Une page web est un **fichier** rangé sur un ordinateur qu'on appelle un **serveur web**. Ton navigateur (Firefox, Chrome, Safari…) est le **client** : il demande le fichier, le reçoit, puis l'affiche.



Le trajet d'une page : le navigateur demande, le serveur répond.

- La demande s'appelle une **requête** ; le langage de communication est le protocole **HTTP** (*HyperText Transfer Protocol*).
- Avec **HTTPS**, les échanges sont en plus **chiffrés** (voir le thème Cryptage) : personne ne peut lire la requête ni la réponse en chemin.
- Le fichier reçu est écrit en **HTML** (*HyperText Markup Language*) : c'est du **texte**, avec des marques qui décrivent la page.

Cours · L'adresse d'une page : l'URL



Les trois parties d'une URL.

Chaque page a une adresse unique : son **URL**. Le **nom de domaine** désigne le serveur. Le **chemin** indique où se trouve le fichier sur ce serveur.

> **Attention**
>
> Dans une URL, le nom de domaine se lit **entre `://` et le premier `/`**, en regardant sa fin : `www.exemple.fr` appartient au site `exemple.fr`. Une adresse qui *ressemble* à celle d'un site connu peut cacher un autre domaine.

Exercices · Je comprends le trajet d'une page

Exercice 1 — Internet ou Web ?

Pour chaque élément, dis s'il appartient plutôt à **Internet** ou au **Web** :

1. un câble posé au fond de l'océan ;
2. une page de Wikipédia ;
3. un lien hypertexte ;
4. l'envoi d'un courrier électronique.

Réponses dans ton compte rendu (exercice 1)

Exercice 2 — Décomposer une URL

Pour chaque adresse, relève le protocole, le nom de domaine et le chemin :

1. `https://fr.wikipedia.org/wiki/Chat`
2. `http://monsite.fr/photos/vacances/plage.html`
3. Dans la deuxième adresse, quel est le nom du fichier ? Pourquoi vaut-il mieux ne jamais mettre d'espace dans un nom de fichier ?

Réponses dans ton compte rendu (exercice 2)

Exercice 3 — Le trajet d'une page

Remets dans l'ordre les quatre étapes qui permettent d'afficher une page :

1. Le serveur renvoie le fichier HTML.
2. Le navigateur lit le HTML et affiche la page.
3. Je tape l'URL dans le navigateur.
4. Le navigateur envoie une requête au serveur.
5. Où est rangé le fichier HTML avant d'arriver sur mon écran ?
6. Que se passe-t-il, à ton avis, si le serveur est éteint ?

Réponses dans ton compte rendu (exercice 3)

Exercice 4 — Regarder le code source

Ouvre un site de ton choix (par exemple Wikipédia). Fais un clic droit sur la page, puis choisis **Afficher le code source** (ou tape `Ctrl + U`).

1. Quelle est la première ligne du code ?
2. Cherche la balise `<title>` (avec `Ctrl + F`). Où voit-on ce texte sur la page ouverte ?
3. Cherche `<a` : à quoi correspond ce début de balise sur la page ?

Réponses dans ton compte rendu (exercice 4)

Exercice 5 — Une adresse qui trompe

Tu reçois un message qui t'invite à te connecter à ta banque en cliquant sur :

console

```
https://www.banque.fr.verification-compte.com/connexion
```

1. Quel est le **vrai nom de domaine** de ce site ?
2. Le site appartient-il à la banque ? Que fais-tu ?
3. Le cadenas (https) prouve-t-il que le site est honnête ? Explique.

Réponses dans ton compte rendu (exercice 5)

Partie W2 — Le langage HTML

Titres, listes, tableaux, liens et images : écrire sa première page

Une page web est un **fichier texte** dont le nom se termine par `.html`. Le texte est accompagné de **balises** qui disent au navigateur ce que représente chaque morceau : un titre, un paragraphe, une liste, une image… Tout au long de cette partie, tu construis la page d'un chat nommé Tigrou.

Deux façons de travailler

Dans le navigateur : l'Atelier HTML et CSS affiche le résultat dès que tu écris, sans enregistrer de fichier.

Avec un éditeur de texte (Notepad++, VS Code…) : tu enregistres un fichier `.html`, tu l'ouvres avec le navigateur, puis tu actualises la page (F5) après chaque modification. C'est ainsi que tu travailleras pour le mini-site.

Cours · Le squelette d'une page

tigrou.html

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Tigrou le chat</title>
  </head>
  <body>
    <h1>Tigrou le chat</h1>
    <p>Tigrou est un chat roux qui adore dormir au soleil.</p>
  </body>
</html>
```

- `<!DOCTYPE html>` : indique que c'est une page HTML5.
- `<html lang="fr">` : la racine de la page. `lang="fr"` annonce que le texte est en français (utile aux lecteurs d'écran et aux traducteurs).
- `<head>` : informations **non affichées** sur la page : `<meta charset="UTF-8">` (le codage, pour que les accents s'affichent) et `<title>` (le titre de l'onglet).
- `<body>` : tout ce qui est **visible** sur la page.

Les balises

La plupart des balises vont par deux : une balise **ouvrante** `<p>` et une balise **fermante** `</p>`. Le texte concerné est placé entre les deux. Certaines balises ont des **attributs** : `<html lang="fr">`, où `lang` est le nom de l'attribut et `"fr"` sa valeur.

Les erreurs classiques

- Oublier de **fermer** une balise.
- Oublier un guillemet dans un attribut.
- Mettre un **espace** ou un accent dans un nom de fichier (`mon chat.jpg`).
- Oublier l'extension : le fichier doit s'appeler `page.html`, pas `page` ni `page.txt`.
- Modifier le fichier **sans l'enregistrer** : le navigateur affiche toujours l'ancienne version.

Cours · Titres et paragraphes

- `<h1>` : le titre principal (**un seul** par page). `<h2>`, `<h3>` : titres de plus en plus secondaires, jusqu'à `<h6>`.
- `<p>` : un paragraphe. Le navigateur ignore les retours à la ligne du fichier : pour séparer deux paragraphes, on ouvre un nouveau `<p>`.

Exercice 6 — Ma première page

Dans l'atelier (ou dans ton éditeur), écris le squelette ci-dessus, en remplaçant le titre et le texte par ton sujet.

1. Quel texte apparaît dans l'onglet du navigateur ?
2. Quel texte apparaît dans la page ?
3. À quoi sert la ligne `<meta charset="UTF-8">` ? (Pense aux accents.)

Fichier : dans ton dossier minisite (fichier .html)

Exercice 7 — Titres et paragraphes

Écris une page avec un titre `<h1>`, puis **trois** parties : chacune a un titre `<h2>` et au moins un paragraphe `<p>`.

1. Compare l'aspect de `<h1>` et de `<h2>`.
2. Ajoute un `<h3>` sous un `<h2>`. Que constates-tu ?

Fichier : dans ton dossier minisite (fichier .html)

Exercice 8 — Que va afficher cette page ?

Sans l'ordinateur, décris ce qu'affichera le navigateur pour ce code. Vérifie ensuite dans l'atelier.

lire.html

```
<body>
  <h1>Mon jeu préféré</h1>
  <h2>Pourquoi je l'aime</h2>
  <p>On joue à plusieurs.</p>
  <ul>
    <li>Les parties sont courtes</li>
    <li>Il y a des défis</li>
  </ul>
  <h2>Mes niveaux</h2>
  <ol>
    <li>Facile</li>
    <li>Difficile</li>
  </ol>
</body>
```

1. Combien y a-t-il de titres, et lesquels ?
2. Quelle liste est numérotée ? Quelle liste est à puces ?

Réponses dans ton compte rendu (exercice 8)

Cours · Listes, tableaux, liens, images

- `<ul>` crée une liste **à puces** et `<ol>` une liste **numérotée**. Chaque élément est écrit dans une balise `<li>`.
- `<table>` crée un tableau. Chaque ligne est dans `<tr>` ; chaque case est dans `<td>` (ou `<th>` pour une case d'en-tête).
- `<a href="adresse">texte du lien</a>` crée un **lien hypertexte**.
- `<img src="fichier.jpg" alt="description" width="300">` insère une image.

balises.html

```
<ul>
  <li>Dormir</li>
  <li>Manger</li>
</ul>

<table>
  <tr><th>Prénom</th><th>Espèce</th></tr>
  <tr><td>Tigrou</td><td>chat</td></tr>
</table>

<a href="https://fr.wikipedia.org/wiki/Chat">En savoir plus sur les chats</a>

<img src="eiffel.jpg" alt="La tour Eiffel au bout d'une rue de Paris" width="300">
```

Accessibilité

- L'attribut **`alt`** décrit l'image avec des mots. Il est lu par les lecteurs d'écran (pour une personne qui ne voit pas l'image) et s'affiche si l'image ne se charge pas.
- Le texte d'un lien doit être **clair tout seul** : « En savoir plus sur les chats » est meilleur que « cliquez ici ».

Exercice 9 — Des listes 🌶

Dans ta page, ajoute :

1. une liste à puces de quatre choses que tu aimes ;
2. une liste numérotée de tes trois préférées, de la première à la troisième.

Fichier : dans ton dossier minisite (fichier .html)

Exercice 10 — Un tableau 🌶🌶

Ajoute un tableau de trois colonnes (**Prénom**, **Espèce**, **Caractère**) et trois lignes de personnages d'un animal ou d'un jeu de ton choix.

1. Utilise `<th>` pour la ligne d'en-tête. Quelle différence avec `<td>` ?
2. Pourquoi dit-on qu'un tableau sert à présenter des **données**, et pas à faire la mise en page ?

Fichier : dans ton dossier minisite (fichier .html)

Exercice 11 — Des liens 🌶🌶

Dans le dossier `minisite`, crée une deuxième page `chat.html`. Puis :

1. Dans `index.html`, écris un lien vers `chat.html`.

2. Dans `chat.html`, écris un lien pour revenir à `index.html`.
3. Écris un lien vers une page de Wikipédia sur ton sujet.
4. Quelle différence y a-t-il entre l'adresse du premier lien et celle du troisième ?

Fichier : dans ton dossier minisite (fichier .html)

Exercice 12 — Une image 🌶🌶

Les photos `eiffel.jpg`, `ardeche.jpg` et `gargouille.jpg` sont dans Web.zip. Copie-en une dans ton dossier `minisite`, puis :

1. Insère l'image avec une description (`alt`) et une largeur de 300.
2. Remplace volontairement le nom du fichier par un nom qui n'existe pas. Que voit-on ?
3. À quoi sert l'attribut `alt` ? Écris une bonne description pour la photo `gargouille.jpg`.

Fichier : dans ton dossier minisite (fichier .html)

Exercice 13 — Chasse aux erreurs 🌶🌶🌶

Cette page contient au moins six erreurs. Recopie-la dans l'atelier (ou charge le modèle « Page à corriger ») et corrige-la.

mal.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tigrou le chat</title>
  </head>
  <body>
    <h1>Tigrou le chat<h1>
    <p>Tigrou aime :
    <li>dormir</li>
    <li>manger</li>
    <img src="mon chat.jpg" width="200">
    <a href="https://fr.wikipedia.org/wiki/Chat>En savoir
plus</a>
  </body>
</html>
```

1. Liste les erreurs que tu as trouvées.
2. Quelles erreurs empêchent un lecteur d'écran, ou un étranger qui traduit la page, de bien la lire ?

Fichier : dans ton dossier minisite (fichier .html)

Partie W3 — Le langage CSS

Couleurs, polices, marges : donner un style à ses pages

Le HTML décrit **ce que contient** une page. Le **CSS** (*Cascading Style Sheets*, feuilles de style en cascade) décrit **son apparence** : couleurs, polices, tailles, marges. On sépare les deux : le HTML dans un fichier `.html`, le CSS dans un fichier `.css`.

Cours · Une règle CSS

style.css

```
h2 {
  color: orange;
  text-align: center;
  font-size: 30px;
}
```

- h2 est le **sélecteur** : il dit à quelles balises la règle s'applique.
- `color`, `text-align`, `font-size` sont des **propriétés**.
- `orange`, `center`, `30px` sont les **valeurs** de ces propriétés.
- Chaque ligne se termine par un **point-virgule** ; toutes les lignes sont entre **accolades** { }.

Relier le fichier CSS à la page

Dans la partie <head> de **chaque** page HTML, on ajoute la ligne suivante. Le fichier `style.css` doit être dans le même dossier.

tigrou.html

```
<link rel="stylesheet" href="style.css">
```

Dans l'atelier

L'Atelier HTML et CSS a un onglet `style.css` : la ligne `<link>` est déjà en place, le style s'applique tout de suite.

Les erreurs classiques

- Oublier le **point-virgule** à la fin d'une ligne : les lignes suivantes ne sont plus prises en compte.
- Écrire `:` à la place de `;` (ou l'inverse).
- Se tromper dans le nom du fichier du `<link>` : le style disparaît.
- Ne pas **actualiser** (F5) la page après avoir enregistré le CSS.

Exercices · Premier style

Exercice 14 — Mon premier fichier CSS

Crée un fichier `style.css` dans ton dossier `minisite` avec la règle suivante, relie-le à ta page (voir ci-dessus), puis actualise la page.

style.css

```
body {
  margin: 20px;
  font-family: Verdana;
  background: yellow;
  color: blue;
}
```

1. Qu'observes-tu ?
2. Modifie la règle pour avoir un fond rose, un texte blanc, la police Arial et une marge de 50 pixels.
3. À ton avis, que signifient `margin`, `font-family` et `background` ?

Fichier : dans ton dossier minisite (fichiers .html et .css)

Exercice 15 — Lire une règle

regle.css

```
h3 {
  color: yellow;
  font-weight: bold;
  font-size: 20px;
  background: black;
}
```

1. Quel est le sélecteur ?
2. Quelles sont les quatre propriétés ? Quelle est la valeur de chacune ?
3. Quels éléments de la page seront modifiés ?
4. Écris la règle qui mettrait les titres `<h1>` en gris, centrés et en gras.

Réponses dans ton compte rendu (exercice 15)

Exercice 16 — Titres, listes et tableaux

Ajoute les règles suivantes à ton fichier, **une par une**, en actualisant la page à chaque fois pour observer le changement :

style.css

```
h1 {
  color: grey;
  text-align: center;
  font-size: 40px;
}

ul {
  color: red;
  list-style-type: circle;
}

ol {
  font-variant: small-caps;
}

table {
  border: 2px dotted black;
}

td {
  border: 2px outset silver;
```

```
  padding: 5px;
}
```

1. Décris l'effet de chaque règle.
2. Pourquoi le texte des deux listes ne change-t-il pas de la même façon ?
3. Que fait `padding` ? En quoi diffère-t-il de `margin` ?

Fichier : dans ton dossier minisite (fichiers .html et .css)

Cours · Les couleurs

Une couleur peut s'écrire avec son **nom** (`pink`), avec ses trois composantes `rgb(255, 105, 180)`, ou en **hexadécimal** `#FF69B4`. C'est le même codage rouge–vert–bleu que dans le thème Photographie numérique : FF = 255 pour le rouge, 69 = 105 pour le vert, B4 = 180 pour le bleu.

Pour trouver le code d'une couleur, utilise le Mélangeur RVB du thème Photo : il affiche le code hexadécimal.

Exercice 17 — Couleurs en hexadécimal 🌶🌶

1. Avec le mélangeur RVB, trouve le code hexadécimal du rouge pur, du noir et du blanc.
2. Quelle couleur obtient-on avec `#0000FF` ?
3. Cherche le code hexadécimal d'un orange. Écris une règle `h1 { color: ...; }` qui l'utilise.

Fichier : dans ton dossier minisite (fichiers .html et .css)

Cours · Les classes et le survol

Pour mettre en forme **certains** éléments seulement, on leur donne une **classe**, puis on écrit une règle dont le sélecteur commence par un point.

page.html

```
<p class="important">Ceci est important !</p>
```

style.css

```
.important {
  color: green;
  font-weight: bold;
}
```

Le CSS peut aussi réagir à la souris : `a:hover` décrit un lien **au moment où la souris le survole**.

style.css

```
a {
  color: rgb(102, 102, 255);
  text-decoration: underline;
}

a:hover {
  color: #009999;
  cursor: pointer;
}
```

Exercice 18 — Classes et liens qui réagissent 🌶🌶

1. Ajoute à ta page deux paragraphes avec la classe `important`, et une règle `.important` qui les met en gras et en vert foncé.
2. Ajoute les règles `a` et `a:hover` ci-dessus. Que se passe-t-il quand tu survoles un lien ?
3. Pourquoi vaut-il mieux garder les liens **soulignés** ?

Fichier : dans ton dossier minisite (fichiers .html et .css)

Cours · Lisible ou pas ?

Choisir des couleurs, c'est aussi **se faire lire**. Le **contraste** entre la couleur du texte et celle du fond se mesure par un nombre : plus il est grand, plus le texte est facile à lire. Le Testeur de contraste le calcule.

Repères

- Un contraste d'au moins **4,5** est recommandé pour un texte courant (norme d'accessibilité WCAG).
- Un texte sur une **image** ou un dégradé est souvent plus difficile à lire que sur un fond uni.

- Ne jamais transmettre une information **uniquement par la couleur** (« les mots en rouge sont importants »).

Exercice 19 — Lisible ou pas ? 🌶🌶

1. Avec le testeur de contraste, essaie le texte blanc sur le fond rose `#FF69B4` de l'exercice « Mon premier fichier CSS ». Quel est le contraste ?
2. Quelle couleur de texte donnerait un bon contraste sur ce rose ?
3. Trouve trois couples de couleurs (texte, fond) lisibles, et trois couples illisibles.

Réponses dans ton compte rendu (exercice 19)

Exercice 20 — Je donne un style à ma page 🌶🌶🌶

Choisis une ambiance pour ta page (ses couleurs, ses polices), puis écris ta feuille de style. Elle doit contenir :

- au moins **six** règles différentes ;
- une règle pour `body`, une règle pour les titres, une règle avec une **classe**, et une règle `a:hover` ;
- des couleurs dont le contraste est correct (vérifie-le avec le testeur) ;
- une marge ou un espacement (`margin` ou `padding`).

1. Décris en deux phrases l'ambiance choisie.
2. Quel est le plus faible contraste de ta page ? Est-il suffisant ?

Fichier : dans ton dossier `minisite` (fichiers `.html` et `.css`)

Partie W4 — Mon mini-site

Le projet : un petit site de plusieurs pages, mis en forme, que tu vérifies toi-même

Tu sais écrire une page, la relier à une autre et la mettre en forme. Il reste à **assembler** ces connaissances dans un vrai petit site. Tu choisis le sujet : un animal, un sport, un jeu, un groupe de musique, une ville, un réseau social et ses risques… à condition de bien le connaître ou de pouvoir t'informer.

Cours · Le cahier des charges

Ton mini-site contient **au minimum** :

- **trois pages HTML**, reliées entre elles par des liens, dont une page d'accueil nommée `index.html` ;
- sur la page d'accueil : ton **prénom**, ton **nom**, ta **classe** et un **sommaire** avec des liens vers les autres pages ;
- au moins **une image** (avec une description `alt`), **une liste à puces**, **une liste numérotée** et **un tableau** ;
- au moins **un lien vers un site extérieur** ;
- **un seul fichier CSS** qui met en forme toutes les pages ;
- sur **chaque** page : un titre dans l'onglet (`<title>`), un seul `<h1>` et un **menu** pour passer d'une page à l'autre.

Règles du jeu

- Un dossier `minisite`, **sans espace ni accent** dans aucun nom de fichier.
- Les images sont rangées dans le dossier, avec leur **source** indiquée sur la page (auteur ou site d'origine).
- Tu n'écris **aucune information personnelle** (adresse, téléphone) et tu ne publies **aucune photo d'une personne** sans son accord.
- Tu peux t'aider d'une intelligence artificielle pour rédiger un texte ou obtenir des idées, mais tu relis, tu vérifies et tu indiques que le texte vient d'une IA.

Cours · Un site est un graphe

Tes pages sont reliées par des liens : si tu dessines une **flèche** pour chaque lien, de la page où il est écrit vers la page qu'il désigne, tu obtiens un **graphe orienté**, comme au thème Localisation, mais avec des flèches. Les pages sont les **sommets**, les liens sont les **arcs**. Cette idée sera reprise dans la partie W6.

Deux questions à te poser : peut-on arriver sur **chaque** page en partant de l'accueil ? peut-on **revenir** à l'accueil depuis chaque page ?

Exercice 21 — Le plan de mon site 🌶🌶

Avant d'écrire du code, prépare ton projet sur papier :

1. Choisis ton sujet et donne un titre à chacune de tes trois pages.
2. Dessine les trois pages comme des cercles, et une flèche pour chaque lien prévu.
3. Peut-on aller de l'accueil à chaque page ? Peut-on revenir à l'accueil depuis chaque page ?

Réponses dans ton compte rendu (exercice 21)

Cours · Vérifier son travail

Le Vérificateur de mini-site lit tes fichiers **dans ton navigateur** (rien n'est envoyé sur Internet) et te dit ce qui manque par rapport au cahier des charges, avec des conseils d'accessibilité. Tu peux t'en servir autant de fois que tu veux, avant de me rendre ton travail.

Exercice 22 — Réparer un site incomplet 🌶🌶

Dans Web.zip, le dossier `exemple_incomplet` contient un petit site de deux pages. Ouvre-le dans le vérificateur (bouton « Charger l'exemple incomplet »).

1. Quels points sont signalés ?
2. Corrige-les un par un dans les fichiers, puis recharge-les dans le vérificateur.
3. Que faut-il ajouter pour atteindre trois pages ?

Fichier : dans ton dossier minisite (fichiers .html et .css)

Exercice 23 — Relire le site d'un camarade 🌶🌶

Échange ton dossier avec un camarade. Ouvre son site dans le navigateur et réponds, **sans toucher à ses fichiers** :

1. Peux-tu comprendre le sujet en lisant la page d'accueil en 30 secondes ?
2. Tous les liens fonctionnent-ils ? Y a-t-il un menu sur chaque page ?
3. Le texte est-il lisible (taille, contraste) ?
4. Donne-lui une chose réussie et un conseil précis pour améliorer son site.

Fichier : dans ton dossier minisite (fichiers .html et .css)

Cours · Ce qui sera regardé

Domaine	Ce que je regarde
Contenu	Le sujet est présenté clairement ; les textes sont personnels et les sources sont indiquées.
HTML	Structure correcte (titres, listes, tableau, liens, images avec `alt`) ; pages reliées entre elles.
CSS	Un fichier unique ; une vraie mise en forme ; des couleurs dont le contraste est suffisant.
Autonomie	Tu as vérifié ton site avec le vérificateur et corrigé ce qu'il signalait.

Exercice 24 — Mon mini-site 🌶🌶🌶🌶

Réalise ton mini-site en respectant le cahier des charges. Quand le vérificateur n'indique plus aucun point à corriger :

1. range tous les fichiers dans le dossier `minisite` ;
2. fais un clic droit sur le dossier, puis **Compresser dans un fichier ZIP** ;
3. nomme l'archive `NOM_Prenom_minisite.zip` et dépose-la sur Pronote.

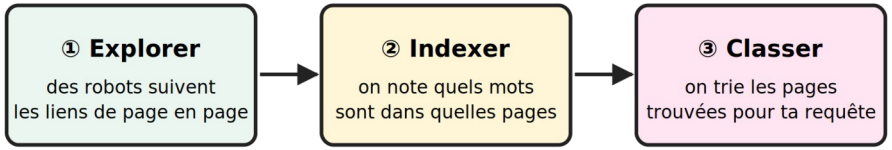
Fichier : NOM_Prenom_minisite.zip

Partie W5 — Le moteur de recherche

Explorer, indexer, classer : comment trouver une page parmi des milliards

Le Web compte un nombre gigantesque de pages. Pourtant, un moteur de recherche te donne des résultats en une fraction de seconde. Il ne parcourt pas le Web au moment de ta recherche : il a **préparé son travail à l'avance**.

Cours · Les trois étapes



Les étapes ① et ② se font à l'avance ; l'étape ③ se fait à chaque recherche.

Explorer, indexer, classer : les trois étapes d'un moteur de recherche.

1. **Explorer.** Des programmes appelés **robots d'exploration** partent de quelques pages, suivent leurs liens, puis les liens des pages découvertes, et ainsi de suite. Une page vers laquelle **aucun lien** ne pointe n'est jamais découverte.
2. **Indexer.** Pour chaque page trouvée, le moteur note les mots qu'elle contient. Il construit un **index inversé** : pour chaque mot, la liste des pages où il apparaît, comme l'index à la fin d'un livre.
3. **Classer.** Quand tu fais une recherche, le moteur retrouve dans l'index les pages qui contiennent tes mots, puis les **trie** : le nombre d'apparitions des mots, l'endroit où ils se trouvent (titre ou texte), et la **popularité** de la page (voir la partie W6).

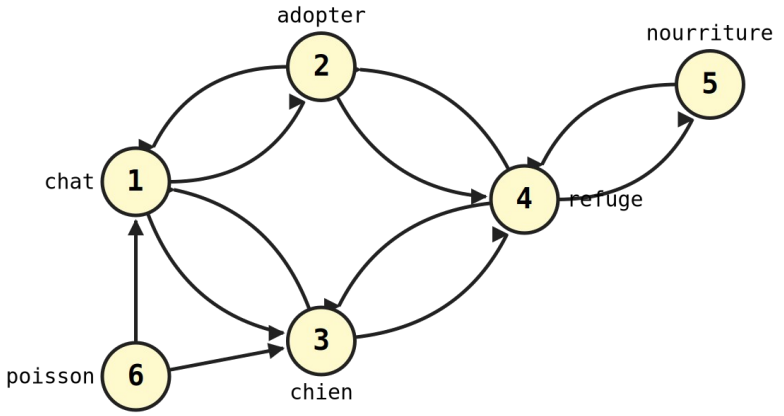
Un fichier pour les robots

Un site peut indiquer aux robots ce qu'il ne souhaite pas voir explorer, dans un fichier nommé `robots.txt`. C'est une demande de politesse : les robots sérieux la respectent.

Cours · Un moteur en miniature

Pour comprendre, on utilise un mini-Web de **six pages** sur les animaux. Voici leurs textes, puis le graphe de leurs liens.

N°	Titre	Texte de la page
1	chat	Le chat est un animal domestique. Le chat dort beaucoup et chasse la nuit.
2	adopter	Pour adopter un chat, il faut un refuge ou un éleveur. Un chat adopté demande des soins.
3	chien	Le chien est un animal fidèle. Le chien aime courir et jouer dehors.
4	refuge	Un refuge accueille des chats et des chiens abandonnés. On peut adopter un animal au refuge.
5	nourriture	Croquettes pour chat et pour chien. Choisis une nourriture adaptée à ton animal.
6	poisson	Le poisson rouge vit dans un aquarium. Ce n'est pas un chat, ni un chien.



Les liens entre les six pages : une flèche de a vers b signifie que la page a contient un lien vers la page b.

Tu peux interroger ce mini-Web avec le Moteur de recherche en miniature.

Exercices · Explorer et indexer

Exercice 25 — Le robot explore

Un robot démarre sur la page 6 et suit tous les liens.

1. Quelles pages découvre-t-il après **un** clic ? après **deux** clics ?
2. Quelle page n'est atteinte depuis aucune autre ?

3. Que se passerait-il pour cette page si le robot démarrait ailleurs ? Que peut faire son auteur pour qu'elle soit trouvée ?

Réponses dans ton compte rendu (exercice 25)

Exercice 26 — Construire l'index 🌶🌶

À partir des textes du tableau, complète l'index inversé pour les quatre mots suivants : indique pour chacun les numéros des pages qui le contiennent (le mot exactement écrit ainsi).

1. `chat` ; `chien` ; `refuge` ; `adopter`.
2. La page 4 parle de chats et de chiens. Pourquoi n'apparaît-elle ni pour `chat` ni pour `chien` ?
3. Que devrait faire un moteur réel pour que `chats` et `chat` soient réunis ?

Réponses dans ton compte rendu (exercice 26)

Exercice 27 — Interroger le mini-moteur 🌶🌶

Dans le Moteur de recherche en miniature, garde les réglages d'origine et lance les recherches suivantes.

1. `chat` : dans quel ordre apparaissent les pages ? Pourquoi les pages 1 et 2 sont-elles à égalité d'occurrences, et comment sont-elles départagées ?
2. `chat chien` : quelles pages sont trouvées ? Pourquoi la page 3 ne l'est-elle pas ?
3. `refuge adopter` : quelle page arrive en premier ? Pourquoi ?

Réponses dans ton compte rendu (exercice 27)

Cours · Un index en Python

Un **dictionnaire** Python convient très bien pour un index : la **clé** est le mot, la **valeur** est la liste des pages. Pour découper un texte en mots, on utilise `.split()`.

console

```
>>> "le chat dort".split()
['le', 'chat', 'dort']
```

Pour construire l'index

1. On parcourt les pages une par une.
2. Pour chaque mot de la page : si le mot n'est pas encore dans l'index, on crée sa liste vide.
3. Si la page n'est pas encore dans la liste de ce mot, on l'ajoute avec `append`.

Exercice 28 — Mon index en Python 🌶🌶🌶

Copie le programme, complète les deux lignes indiquées, puis enregistre-le sous le nom demandé.

index.py

```
pages = {
    1: "le chat est un animal domestique",
    2: "adopter un chat demande des soins",
    3: "le chien est un animal fidele",
}

index = {}
for numero in pages:
    for mot in pages[numero].split():
        if mot not in index:
            index[mot] = ...                    # à compléter : une liste vide
        if numero not in index[mot]:
            index[mot].append(...)              # à compléter : le numéro de la page

print(index)
```

1. Exécute le programme. Quelles sont les pages qui contiennent le mot `animal` ? le mot `chat` ?
2. À la fin du programme, ajoute des lignes pour afficher les pages du mot `chien`, et un message « aucun résultat » si le mot n'est pas dans l'index. Teste avec `poisson`.
3. Un texte contient « Chat » avec une majuscule. L'index le rapproche-t-il de `chat` ? Quelle méthode de texte permet de tout passer en minuscules ?

Fichier : NOM_Prenom_ex28.py

Cours · Bien chercher

Les moteurs comprennent quelques outils pour affiner une recherche (la plupart des moteurs en acceptent au moins une partie) :

Ce que tu écris	Effet
`"chat noir"` entre guillemets	cherche exactement cette expression
`chat -noir`	exclut les pages qui contiennent `noir`
`chat site:wikipedia.org`	limite la recherche à un site

Exercice 29 — Bien chercher 🌶🌶

Dans le moteur de ton choix :

1. Compare `chat noir` et `"chat noir"` : qu'est-ce qui change dans les résultats ?
2. Cherche `jaguar` : quels sens différents trouves-tu ? Comment affiner pour obtenir uniquement l'animal ?
3. Limite une recherche à un site avec `site:`, par exemple `climat site:education.gouv.fr`. À quoi cela peut-il servir ?

Réponses dans ton compte rendu (exercice 29)

Exercice 30 — Qui paie ? Les résultats personnalisés 🌶🌶

Fais une recherche d'un produit (par exemple « chaussures de sport ») sur un moteur de recherche, puis compare avec ton voisin.

1. Repère les résultats marqués comme **annonce** ou **sponsorisé**. Où sont-ils placés ?
2. Pourquoi ces résultats sont-ils placés à cet endroit ? Qui les finance ?
3. Ton voisin et toi obtenez-vous exactement les mêmes résultats ? Propose deux raisons possibles.

Réponses dans ton compte rendu (exercice 30)

Partie W6 — PageRank : le surfeur aléatoire

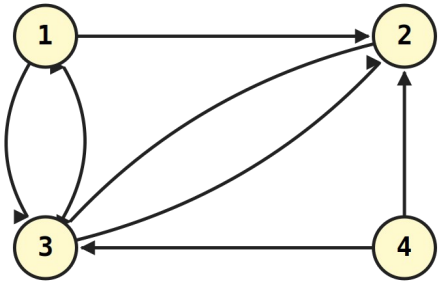
Comment classer les pages ? Un graphe, des liens et beaucoup de hasard

Quand tu cherches un mot, des centaines de pages peuvent convenir. Laquelle montrer en premier ? Au début des années 2000, les créateurs de Google ont eu une idée : **une page est importante si beaucoup de pages importantes pointent vers elle**. Ce principe s'appelle le **PageRank**, du nom de son inventeur Larry Page. Ce n'est qu'**un critère parmi d'autres** dans le classement d'un moteur de recherche.

Cours · Le Web est un graphe orienté

Dans le thème Localisation, un graphe relie des sommets par des **arêtes**, sans direction. Pour le Web, les liens ont un **sens** : la page *a* peut contenir un lien vers la page *b* sans que *b* en ait un vers *a*. On dessine donc une **flèche** de *a* vers *b*.

- Les pages sont les **sommets** ; les liens sont les **arcs** (des flèches).
- Les liens **sortants** d'une page sont ceux qui partent d'elle ; les liens **entrants** sont ceux qui arrivent sur elle.
- Pour décrire le graphe en Python, on utilise un **dictionnaire** : la clé est une page, la valeur est la **liste des pages vers lesquelles elle pointe**.



Un graphe de quatre pages. La flèche de 1 vers 2 signifie que la page 1 contient un lien vers la page 2. Il n'y a pas de flèche de 2 vers 4 : la page 2 n'a pas de lien vers la page 4.

liens.py

```
liens = {1: [2, 3], 2: [3], 3: [1, 2], 4: [2, 3]}
```

Exercice 31 — Lire un graphe orienté 🌶

Observe le graphe ci-dessus.

1. Écris pour chaque page la liste des pages vers lesquelles elle pointe. Compare avec le dictionnaire `liens`.
2. Combien de liens **entrants** reçoit chaque page ?
3. Quelle page ne reçoit aucun lien ? Peut-on y arriver en cliquant sur des liens ?

Réponses dans ton compte rendu (exercice 31)

Cours · Le surfeur aléatoire

Pour mesurer l'importance des pages, on imagine un **surfeur aléatoire** : il choisit une page au hasard, puis, **toujours au hasard**, il suit l'un des liens de la page où il se trouve vers une autre page. Il répète cela sans arrêt, en comptant combien de fois il passe sur chaque page.

> **Le score d'une page**
>
> Le **score** d'une page est la part des visites qu'elle reçoit : **score (en %) = nombre de visites ÷ nombre total de pas × 100**. Plus le surfeur marche longtemps, plus ces scores se stabilisent : c'est une estimation du PageRank.

Pour t'entraîner sans programmer, tu peux regarder le surfeur marcher dans l'outil Surfeur aléatoire.

Exercice 32 — Le surfeur à la main 🌶🌶

Pour le graphe ci-dessus, simule un surfeur avec une pièce : à chaque page qui a **deux** liens, pile choisit le premier lien de la liste, face choisit le second. (Une page avec un seul lien ne laisse pas le choix.)

1. Choisis une page de départ et simule **15 pas** en notant les pages visitées.
2. Compte les passages sur chaque page. Propose un classement des quatre pages.
3. Compare ton classement avec celui de deux camarades. Ce classement est-il toujours le même ?

Réponses dans ton compte rendu (exercice 32)

Cours · Le surfeur en Python

On écrit une fonction qui fait faire **un pas** au surfeur : à partir de la page où il est, elle renvoie une page choisie au hasard parmi ses liens sortants. La fonction `random.choice(liste)` choisit un élément au hasard dans une liste.

Exercice 33 — Un pas du surfeur 🌶🌶

Copie le programme (ou ouvre `pagerank_depart.py` de Web.zip) et complète la ligne indiquée. Enregistre-le sous le nom demandé.

pagerank.py

```
import random

liens = {1: [2, 3], 2: [3], 3: [1, 2], 4: [2, 3]}

def suivant(site):
    return random.choice(...)                # à compléter :
une page parmi les liens de `site`

site = 1
for i in range(5):
    site = suivant(site)
    print(site)
```

1. Exécute le programme plusieurs fois. Quelles pages peuvent suivre la page 1 ? et la page 2 ?
2. Que fait la boucle `for i in range(5)` ?

Fichier : NOM_Prenom_ex33.py

Exercice 34 — Compter les visites 🌶🌶🌶

On complète le programme pour compter les visites. Le dictionnaire `visites` associe à chaque page son nombre de passages. Complète les deux lignes indiquées.

pagerank.py

```
import random
```

```python
liens = {1: [2, 3], 2: [3], 3: [1, 2], 4: [2, 3]}

def suivant(site):
    return random.choice(liens[site])

nombre = 1000
site = 1
visites = {}
for page in liens:
    visites[page] = 0

for i in range(nombre):
    site = suivant(site)
    visites[site] = visites[site] + ...        # à compléter

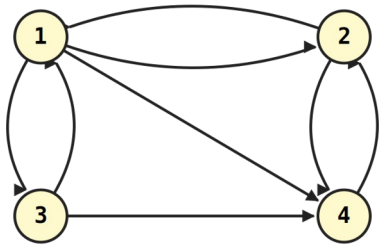
for page in visites:
    print(page, visites[page], round(100 * visites[page]
/ ..., 1), "%")    # à compléter
```

1. Exécute le programme avec `nombre = 100`, puis `1000`, puis `10000`. Note les pourcentages obtenus.
2. Relance plusieurs fois avec `nombre = 100`, puis avec `nombre = 10000`. Les résultats changent-ils beaucoup ? Que peux-tu en conclure ?
3. Quelle page obtient le meilleur score ? Quelle page a un score voisin de 0 ? Explique pourquoi.

Fichier : NOM_Prenom_ex34.py

Exercice 35 — Un autre graphe 🌶🌶🌶

Voici un autre graphe de quatre pages.



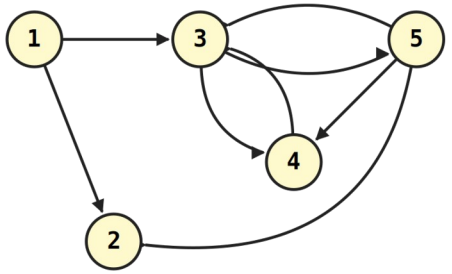
Le graphe de l'exercice.

1. Écris son dictionnaire `liens`.
2. Dans ton programme, **change uniquement `liens`** et relance avec `nombre = 10000`. Quelles sont les parts des quatre pages ?
3. Quelle page est la mieux classée ? Retrouve-t-on le même résultat dans l'outil Surfeur aléatoire ?

Fichier : NOM_Prenom_ex35.py

Cours · Quand le surfeur est coincé

Le Web réel est plein de pages sans lien sortant, ou de groupes de pages qui se renvoient les uns aux autres sans jamais mener ailleurs. Le surfeur aléatoire peut alors **rester bloqué**.



Un graphe avec une page sans lien sortant.

Exercice 36 — Le surfeur est coincé 🌶🌶

Dans ce graphe, la page 2 n'a aucun lien sortant : `liens = {1: [2, 3], 2: [], 3: [4, 5], 4: [3], 5: [3, 4, 2]}`.

1. Que risque-t-il d'arriver au surfeur quand il arrive en page 2 ?
2. Lance ton programme avec ce dictionnaire, jusqu'à ce que le surfeur passe par la page 2. Que se passe-t-il ?
3. Que vaut `liens[2]` ? Pourquoi `random.choice` ne peut-il pas fonctionner ?

Réponses dans ton compte rendu (exercice 36)

Exercice 37 — Se téléporter quand on est bloqué 🌶🌶🌶

Pour résoudre le problème, on décide que **si le surfeur est bloqué, il se téléporte sur une page choisie au hasard parmi toutes les pages**. Voici la nouvelle fonction. Complète-la, puis estime les scores du graphe de la page 2 sans lien.

pagerank.py

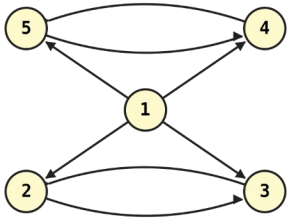
```
def suivant(site):
    if liens[site] == ...:                  # à compléter : la
page n'a aucun lien sortant
        return random.choice(list(liens))
    return random.choice(liens[site])
```

1. Que renvoie `list(liens)` ?
2. Lance le programme avec `nombre = 10000`. Quels scores obtiens-tu ?

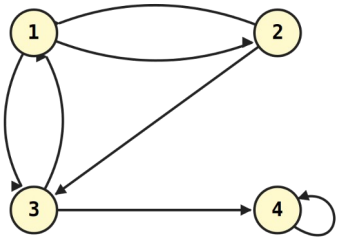
Fichier : NOM_Prenom_ex37.py

Cours · Les pièges

Se téléporter *seulement* quand on est bloqué ne suffit pas : certains groupes de pages forment un **piège** dont on ne sort plus, même si chaque page a des liens sortants.



Deux groupes de pages, {2, 3} et {4, 5}, qui se renvoient les uns aux autres.



La page 4 ne renvoie que vers elle-même.

Exercice 38 — Les pièges 🌶🌶🌶

Estime les scores de ces deux graphes avec le programme de l'exercice précédent (téléportation seulement si bloqué) et `nombre = 10000`.

Graphe de la première figure (page 1 vers 2, 3, 4 et 5 ; 2 ↔ 3 ; 4 ↔ 5) : `liens = {1: [2, 3, 4, 5], 2: [3], 3: [2], 4: [5], 5: [4]}`.

Graphe de la seconde figure : `liens = {1: [2, 3], 2: [1, 3], 3: [1, 4], 4: [4]}`.

1. Pour le premier graphe, lance le programme **plusieurs fois**, avec la page 1 pour départ. Que remarques-tu sur les scores des pages 2 et 3, et ceux des pages 4 et 5 ?
2. Pour le second graphe, quels scores obtiens-tu ? Pourquoi toutes les visites finissent-elles par aller sur la page 4 ?
3. Pourquoi la téléportation « seulement si bloqué » ne règle-t-elle pas ces deux problèmes ?

Fichier : NOM_Prenom_ex38.py

Cours · Le surfeur raisonnable

Pour sortir des pièges, l'algorithme évolue : à chaque pas, avec une petite probabilité (15 % dans la version d'origine de Google), le surfeur **se téléporte** sur une page quelconque, au lieu de suivre un lien. C'est le **surfeur raisonnable** : il lui arrive de taper une nouvelle adresse au lieu de cliquer. Pour tirer un événement avec une probabilité de 15 %, on utilise `random.random()`, qui renvoie un nombre au hasard entre 0 et 1 : le nombre est inférieur à 0,15 dans 15 % des cas.

Exercice 39 — Le surfeur raisonnable 🌶🌶🌶🌶

Modifie la fonction `suivant` pour que le surfeur se téléporte si la page n'a aucun lien, **ou** avec une probabilité de 0,15 à chaque pas.

pagerank.py

```
def suivant(site):
    if liens[site] == [] or random.random() < ...:     # à
compléter : la probabilité
        return random.choice(list(liens))
    return random.choice(liens[site])
```

1. Complète la fonction puis estime les scores des deux graphes de l'exercice précédent avec `nombre = 10000`.
2. Que deviennent les scores des pages qui étaient piégées ? Et celui de la page 1 du premier graphe, qui ne recevait aucun lien ?
3. Pourquoi un surfeur qui se téléporte de temps en temps donne-t-il un classement plus **fiable** ?

Fichier : NOM_Prenom_ex39.py

Exercice 40 — Que vaut le mini-Web ? 🌶🌶🌶

Dans l'outil Surfeur aléatoire, choisis le graphe « mini-Web » (les six pages de la partie W5) et règle la téléportation sur 15 %.

1. Quelle page a le meilleur PageRank ? Pourquoi, à ton avis ?
2. La page 6 (poisson) n'est pointée par aucune page. Quel est son score ?
3. Ajoute un lien de la page 4 vers la page 6. Que devient son score ? Que se passe-t-il si, à la place, les pages 1, 2 et 3 pointent chacune vers la page 6 ?

Réponses dans ton compte rendu (exercice 40)

Cours · Ce que PageRank ne dit pas

Le PageRank mesure la **popularité** d'une page à partir des liens, **pas la vérité de son contenu**. Des personnes qui veulent faire monter leurs pages dans les résultats (on parle de *référencement*) créent parfois de faux liens ; les moteurs luttent contre ces manipulations.

Exercice 41 — Popularité et vérité 🌶🌶🌶

1. Explique en deux phrases le principe du surfeur aléatoire.
2. Une page contenant une information fausse peut-elle être très bien classée ? Explique.
3. Pourquoi un moteur de recherche n'utilise-t-il pas **uniquement** le PageRank pour classer les résultats ?
4. Que ferais-tu, avant de faire confiance à un résultat, pour vérifier une information ?

Réponses dans ton compte rendu (exercice 41)