Comme mentionne dans l'introduction,nous avons choisi de travailler avec scrapy. Scrapy est un framwork est spécialement conçu pour le web scraping et gère nativement les requêtes asynchrones. Plus besoin de se prendre la tête avec la programmation asynchrone,Scrapy s'occupe de tout ! Il suffit de définir nos "spiders" (nos robots collecteurs) et le framework optimise automatiquement les performances.Dans le cadre de ce tutoriel nous n'aborderont qu'une partie des possibilité qu'offre Scrapy,pour plus de details veillez vous referer sur la documentation officielle [lien https://docs.scrapy.org/en/latest/].
** Installation de scrapy

Veillez vous referer a [https://docs.scrapy.org/en/2.11/intro/install.html].Une derniere remarque,a l'heure ou nous ecrivons ces lignes [23 octobre 2024],la version 3.13 de Python est disponible cependant elle ne supporte pas tout a fait certains packages notament streamlit ,aussi nous vous recommandons d'utuliser la version 3.12 de Python et d'utuliser un environnement virtuelle pour tout vos projets ayant trait a ce tutoriel.
** Presentation d'un exemple de spider

** Question

Qu'est-ce qu'un Crawler/Spider dans Scrapy ?

**

** definition

Un **crawler** ou un **spider** en Scrapy est un programme informatique conçu pour naviguer automatiquement sur le World Wide Web, à la manière d'une araignée qui tisse sa toile. Il suit les liens hypertextes d'une page à l'autre, télécharge le contenu de ces pages et extrait les données qui nous intéressent.

**

Afin de vous montrer ce que Scrapy apporte, nous allons vous présenter un exemple d'un crawler utilisant la manière la plus simple de scraper un site web.
Nous nous donnons pour tache de récupérer les citations présent dans le site web [https://quotes.toscrape.com/] [image de la page d'accueil].

La premiere des choses a faire c'est d'inspecter le site web et esseyer de voir est ce que visuellement il y'a des choses qui sont redondant, des schemas qui se succedent en boucle.Dans notre cas nous remarquons que tous citations sont dans un rectangle avec la même mise en forme, seul le texte change,ect.C'est bon signe ! cela suggère que les developpeurs ont esseyer de respecter un modele de base pour construire tous les citations.

Ensuite nous inspectons le code source de la page web a l'aide d'outils mis à disposition par un navigateur, pour notre cas ce serra Google Chrome mais n'importe quel navigateur moderne a en son sein des outils qui conviennent pour la tâche.

On fait un clic droit a n'importe quel endroit du site web puis sélectionner <<inspecter>> nous obtenons la figure ??.Nous obtenons entre autre l'architecture html de la page web .Sur l'icone qui ressemble a un pointeur de souris (encadre rectangulaire de bordure rouge sur la figure ??) cliquez y et faites deviller votre souris sur les differents blocs qui contiennent les citations. Vous obtenez la figure ?? avec un infobule qui donne des details sur les balises qui contiennent nos fameux rectangles.En regardant le code source,on remarque que chaque rectangle est dans une balise et que toutes les citations sont dans une balise parent qui est un En cliquant sur l'icone precedente nous

pouvons constater que les tags ainsi que le nom des auteurs suivent aussi un schema bien precis par exemple chaque rectangle a un enfant .
** Question
Maintenant que nous savons tout cela a quoi cela va nous servir
**
Le travail precedent nous permet de savoir ou chercher pour trouver les informations dont nous souhaitons extraire. Une recherche (Crtl+f) dans le code source permet de remarquer que seul nos rectangles sont des div avec la class 'quote'.Pour comprendre le code qui va suivre il est imperatif de comprendre un minimum l'utulisation des selecteur css pour cela nous vous recommandons l'article [https://www.codeur.com/tuto/css/selecteurs-css/].Ces selecteurs vont nous permettre de cibler de façon precis le texte que nous souhaitons extraire.
Maintenant place au code << https://docs.scrapy.org/en/latest/intro/overview.html>>
Mettez ceci dans un fichier texte, nommez-le comme vous voulez nous on va le nommer `quotes_spider.py` et pour executer ce script utiliser l'invite de commande windows ou le terminal de votre IDE si vous coder avec un IDE ,positionnez vous dans le dossier mere du script et saisissez scrapy runspider quotes_spider.py -o quotes.jsonl
Une fois cette opération terminée, vous aurez dans le `quotes.jsonl` fichier une liste des citations au format JSON Lines, contenant le texte et l'auteur, qui ressemblera à ceci :
figure ???

** Question
Que vient-il de se passer ?
**

L'exploration a commencé par faire des requêtes aux URL définies dans l' `start_urls` attribut (dans ce cas, uniquement l'URL des citations dans la catégorie *humour*) et a appelé la méthode de rappel par défaut `parse`, en passant l'objet de réponse comme argument. Dans le `parse` rappel, nous parcourons les éléments de citation à l'aide d'un sélecteur CSS, produisons un dictionnaire Python avec le texte de citation extrait et l'auteur, recherchons un lien vers la page suivante et planifions une autre requête en utilisant la même `parse` méthode que le rappel.

Vous remarquerez ici l'un des principaux avantages de Scrapy : les requêtes sont planifiées et traitées de manière asynchrone . Cela signifie que Scrapy n'a pas besoin d'attendre qu'une requête soit terminée et traitée, il peut envoyer une autre requête ou faire d'autres choses en attendant. Cela signifie également que d'autres requêtes peuvent continuer même si une requête échoue ou qu'une erreur se produit lors de son traitement.

Bien que cela vous permette d'effectuer des crawls très rapides (en envoyant plusieurs requêtes simultanées en même temps, de manière tolérante aux pannes), Scrapy vous permet également de contrôler la politesse du crawl via quelques paramètres . Vous pouvez faire des choses comme définir un délai de téléchargement entre chaque requête, limiter le nombre de requêtes simultanées par domaine ou par

IP, et même utiliser une extension de limitation automatique qui essaie de les comprendre automatiquement.

** Note

Il s'agit d'utiliser des exportations de flux pour générer le fichier JSON. Vous pouvez facilement modifier le format d'exportation (XML ou CSV, par exemple) ou le backend de stockage (FTP ou Amazon S3 , par exemple). Vous pouvez également écrire un pipeline d'éléments pour stocker les éléments dans une base de données.

**

Créer un projet
** Information
Dans la suite nous utiliseront comme IDE VScode .Toute fois vous etez libre de vous servir de votre IDE préfère du moment ou vous avez accès a un terminal
**
Avant de commencer à scraper, vous devrez configurer un nouveau projet Scrapy. Entrez un répertoire dans lequel vous souhaitez stocker votre code , ouvrez ce dossier dans votre IDE puis creez votre environement virtuelle nommee env et installez y scrapy et activez l'environement vurtuelle .puis dans le terminale de l'ide tapez scrapy startproject Auchan .
Cela cree repertoire Auchan dans lequel nous avons : figure ??