1. Modèle (Model)

- **Rôle :** Gérer les données et la logique métier.
- **Utilité :**
    - Représente les données de l'application et les règles pour y accéder ou les modifier.
    - Communique avec la base de données pour lire, écrire ou mettre à jour les informations.
    - N'interagit pas directement avec la vue ou l'utilisateur.

2. Vue (View)

- **Rôle :** Gérer l'interface utilisateur.
- **Utilité :**
    - Affiche les données issues du modèle dans un format compréhensible pour l'utilisateur.
    - Récupère les entrées utilisateur et les transmet au contrôleur.
    - Découplée de la logique métier, elle se concentre sur la présentation.

3. Contrôleur (Controller)

- **Rôle :** Agir comme un intermédiaire entre le modèle et la vue.
- **Utilité :**
    - Reçoit les actions et les requêtes de l'utilisateur via la vue.
    - Interagit avec le modèle pour récupérer ou manipuler les données nécessaires.
    - Renvoie les données au modèle ou met à jour la vue en conséquence.

Elasticité dans AWS :

Cela désigne la capacité d'AWS à ajuster automatiquement les ressources en fonction des besoins réels. Vous utilisez exactement ce dont vous avez besoin, et AWS adapte les ressources à la demande, tout en appliquant le modèle **"Pay-As-You-Go"**.

- **Scalabilité** est un terme connexe, mais il se concentre sur la capacité à augmenter ou réduire les ressources.
- **Elasticité**, en revanche, met l'accent sur l'automatisation et l'adaptation dynamique, ce qui correspond mieux à votre description.

1. SaaS (Software as a Service)

- **Définition :** Modèle où des applications logicielles complètes sont fournies sur Internet. Les utilisateurs accèdent aux logiciels via un navigateur sans avoir besoin de les installer ou de les maintenir.
- **Exemples :** Gmail, Microsoft 365, Dropbox.

- **Utilisation :** Les utilisateurs finaux se concentrent uniquement sur l'utilisation de l'application, sans se soucier de l'infrastructure ou de la maintenance.
- **Avantage :** Pas de gestion matérielle ou logicielle.

---

2. PaaS (Platform as a Service)

- **Définition :** Plateforme cloud qui fournit un environnement complet pour développer, tester et déployer des applications. Les développeurs se concentrent uniquement sur le code, tandis que le fournisseur gère l'infrastructure.
- **Exemples :** AWS Elastic Beanstalk, Google App Engine, Heroku.
- **Utilisation :** Idéal pour les développeurs qui souhaitent déployer rapidement des applications sans gérer les serveurs.
- **Avantage :** Simplifie le développement et le déploiement, en réduisant le temps et les efforts.

---

3. IaaS (Infrastructure as a Service)

- **Définition :** Fournit des ressources informatiques virtualisées telles que serveurs, stockage et réseaux sur une base à la demande. Les utilisateurs gèrent eux-mêmes leurs systèmes d'exploitation et applications.
- **Exemples :** AWS EC2, Microsoft Azure, Google Compute Engine.
- **Utilisation :** Idéal pour les entreprises qui souhaitent plus de contrôle sur leur infrastructure informatique sans avoir à investir dans du matériel physique.
- **Avantage :** Flexibilité et contrôle total sur l'infrastructure.

1. Intégration Continue (CI - Continuous Integration)

Définition :
Pratique consistant à intégrer fréquemment (plusieurs fois par jour) le code des développeurs dans une branche principale partagée, en automatisant les tests et la validation.

Objectif :
Détecter rapidement les bugs et garantir que le code ajouté fonctionne correctement avec le reste de la base.

Étapes principales :

- Fusion du code dans un dépôt partagé (ex : GitHub, GitLab).
- Exécution automatique de tests unitaires et d'intégration.
- Identification des erreurs dès qu'elles apparaissent.

Exemple d'outil : Jenkins, GitLab CI, Travis CI.

---

2. Livraison Continue (CD - Continuous Delivery)

Définition :
Étape suivant l'intégration continue, où le code validé est automatiquement préparé pour le déploiement en production. La livraison en production nécessite encore une validation humaine.

Objectif :
Assurer que le code est toujours dans un état "livrable" en production.

Étapes principales :

- Automatisation des builds.
- Tests fonctionnels, d'acceptation et de performance.
- Préparation du déploiement avec validation manuelle optionnelle.

Exemple d'outil : AWS CodePipeline, GitLab CI/CD.

---

3. Déploiement Continu (CD - Continuous Deployment)

Définition :
Prolongement de la livraison continue, où tout changement validé passe automatiquement en production sans intervention humaine.

Objectif :
Automatiser entièrement le pipeline pour que les changements atteignent l'utilisateur final dès qu'ils sont prêts.

Étapes principales :

- Automatisation complète des tests et validations.
- Mise à jour instantanée des environnements de production.

Exemple d'outil : Spinnaker, GitHub Actions.

La **programmation orientée objet (POO)** repose sur des concepts clés qui permettent de structurer le code sous forme d'objets, représentant des entités du monde réel ou abstraites.

1. Encapsulation

- **Définition :** Regrouper les données (attributs) et les méthodes (fonctions) dans une seule unité appelée objet.
- **Avantage :** Protège les données internes d'un objet en contrôlant leur accès via des méthodes publiques (getters et setters).
- **Exemple :** Les attributs privés en Python (_attribut) accessibles uniquement via des méthodes.

---

2. Héritage

- **Définition :** Permet à une classe (classe fille) de réutiliser ou étendre les propriétés et méthodes d'une autre classe (classe parent).
- **Avantage :** Encourage la réutilisation du code et la réduction des duplications.
- **Exemple :** Une classe Voiture peut être héritée par VoitureElectrique pour ajouter des spécificités.

---

3. Polymorphisme

- **Définition :** Capacité d'une méthode ou d'un opérateur à se comporter différemment en fonction des objets.
- **Avantage :** Rend le code plus flexible et extensible.
- **Exemple :** Une méthode deplacer() peut avoir différentes implémentations pour Voiture et Avion.

---

4. Abstraction

- **Définition :** Masquer les détails complexes d'implémentation et fournir une interface simplifiée pour interagir avec les objets.
- **Avantage :** Facilite la compréhension et la maintenance du code.
- **Exemple :** Une classe abstraite Animal peut définir une méthode parler() que les sous-classes Chien ou Chat implémentent différemment.

---

5. Interface

- **Définition :** Une interface définit un contrat que les classes doivent respecter. Elle contient uniquement des déclarations de méthodes (sans implémentation).
- **Avantage :** Permet de garantir que des classes non liées respectent les mêmes comportements.
- **Exemple :** Une interface Voler peut être implémentée par des classes Avion et Oiseau, assurant que toutes deux ont une méthode voler().
- **Utilisation :** Les interfaces favorisent le polymorphisme et permettent de découpler l'implémentation des comportements.

---

6. Modularité

- **Définition :** Permet de diviser une application en modules indépendants (objets et classes).
- **Avantage :** Facilite la maintenance et le développement collaboratif.

---

7. Réutilisation du code

- **Définition :** Grâce à l'héritage, aux interfaces, et à la modularité, les composants existants peuvent être réutilisés dans de nouveaux projets.
- **Avantage :** Réduit les efforts de développement et les erreurs.

---

8. Relation entre objets

- Les objets communiquent entre eux via des **messages** (méthodes). Cela permet d'établir des relations complexes comme l'association, l'agrégation ou la composition.

La surcharge consiste à définir plusieurs méthodes avec le même nom dans une classe, mais avec des signatures différentes (c'est-à-dire un nombre ou un type d'arguments différents).

Caractéristiques :

- Implémentée dans la même classe.
- Permet une meilleure lisibilité du code en utilisant un seul nom de méthode pour des comportements similaires.
- Ne dépend pas du type de retour (uniquement des paramètres).

2. Redéfinition (Overriding)

Définition :

La redéfinition consiste à modifier le comportement d'une méthode héritée d'une classe parent dans une classe enfant.

Caractéristiques :

- Implémentée dans une **classe dérivée** (enfant).
- Nécessite que la méthode ait le **même nom**, les **mêmes paramètres**, et le **même type de retour**.
- Permet d'adapter ou de spécialiser le comportement de la classe parent.
- Utilise souvent l'annotation @Override en Java pour indiquer explicitement une redéfinition.

Définition :

La gestion des exceptions permet de capturer et de traiter des erreurs qui se produisent à l'exécution, évitant ainsi l'interruption brutale d'un programme.

Caractéristiques :

- *Les exceptions peuvent être **vérifiées (checked)** ou **non vérifiées (unchecked)**.*
- *Utilise les blocs try, catch, finally, et throw pour capturer et traiter les exceptions.*
- *Les classes peuvent redéfinir ou étendre la gestion des exceptions à travers l'héritage.*

DevOps est une culture, une méthodologie et un ensemble de pratiques qui combinent le développement logiciel (**Dev**) et les opérations informatiques (**Ops**). L'objectif est de favoriser une collaboration fluide entre les équipes de développement et d'exploitation pour accélérer le cycle de vie des logiciels tout en garantissant la qualité et la fiabilité.

Le cloud computing (informatique en nuage) désigne l'accès à des ressources informatiques (serveurs, stockage, bases de données, applications, etc.) via Internet, sans avoir besoin d'infrastructure matérielle locale.

Questions Générales :

1. **Qu'est-ce que JavaScript et à quoi sert-il principalement ?**
   JavaScript est un langage de programmation léger, interprété, utilisé principalement pour créer des interactions dynamiques sur les pages web. Il permet de manipuler le DOM, de gérer les événements, et d'améliorer l'expérience utilisateur.
2. **Quelle est la différence entre JavaScript et Java ?**
   - **JavaScript :** Langage de script exécuté côté client ou serveur, dynamiquement typé.
   - **Java :** Langage compilé orienté objet, statiquement typé, utilisé pour des applications plus complexes.
3. **Quelles sont les principales fonctionnalités de JavaScript côté client ?**
   - Manipulation du DOM.
   - Gestion des événements utilisateur.
   - Communication avec des serveurs via AJAX ou Fetch.
   - Animation et mise à jour des éléments HTML/CSS.
4. **Quelle est la différence entre le JavaScript côté client et côté serveur ?**
   - **Côté client :** Exécuté dans le navigateur pour manipuler l'interface utilisateur.
   - **Côté serveur :** Exécuté avec des outils comme Node.js pour gérer les requêtes, les bases de données, et les API.
5. **Expliquez la différence entre un langage statiquement typé et dynamiquement typé. Où se situe JavaScript ?**
   - **Statique :** Le type des variables est défini à la compilation (ex : Java).
   - **Dynamique :** Le type est déterminé à l'exécution (ex : JavaScript).
     JavaScript est un langage dynamiquement typé.
6. **Qu'est-ce que l'Event Loop en JavaScript et pourquoi est-il important ?**
   L'Event Loop est un mécanisme qui permet à JavaScript d'exécuter des tâches asynchrones (promesses, callbacks) tout en maintenant une seule thread. Il gère les files d'attente et exécute les tâches dans l'ordre approprié.

Questions Techniques :

Bases du langage :

7. **Quelle est la différence entre var, let, et const ?**
    - var : Déclaration fonctionnelle, sujet au hoisting.
    - let : Portée au bloc, pas de hoisting.
    - const : Comme let, mais pour des valeurs constantes.
8. **Expliquez la différence entre les types de données primitifs et les objets en JavaScript.**
    - **Primitifs :** String, Number, Boolean, Null, Undefined, Symbol (stockés par valeur).
    - **Objets :** Structures complexes (Array, Object) stockées par référence.
9. **Que sont les fonctions anonymes et quand les utilise-t-on ?**
   Ce sont des fonctions sans nom, souvent utilisées comme callbacks ou dans des expressions fonctionnelles.
10. **Qu'est-ce que le hoisting en JavaScript ?**
    Le hoisting déplace les déclarations de variables et de fonctions en haut de leur portée avant l'exécution.
11. **Quelle est la différence entre == et === ?**
    - == : Compare les valeurs, fait une conversion de type si nécessaire.
    - === : Compare les valeurs **et** les types.

Fonctionnalités avancées :

12. **Qu'est-ce qu'une promesse en JavaScript et comment fonctionne-t-elle ?**
    Une promesse représente une opération asynchrone qui retourne un résultat futur (résolu ou rejeté).
    Syntaxe :

javascript

Copier le code

new Promise((resolve, reject) => { ... });

13. **Expliquez l'asynchronicité avec async et await.**
    - async : Marque une fonction comme asynchrone.
    - await : Suspend l'exécution jusqu'à ce que la promesse soit résolue.
14. **Quelle est la différence entre call(), apply() et bind() ?**
    - call() : Appelle une fonction avec un contexte (this) donné et des arguments séparés.
    - apply() : Similaire à call, mais les arguments sont passés sous forme de tableau.

- bind() : Renvoie une nouvelle fonction liée à un contexte donné.

15. **Qu'est-ce que la fermeture (closure) ?**
    Une fermeture est une fonction qui conserve l'accès à ses variables locales même après la fin de son exécution.

16. **Expliquez la différence entre prototype et classe.**
    - **Prototype :** Modèle pour créer des objets en JavaScript (ancien paradigme).
    - **Classe :** Syntaxe moderne pour définir des objets et des héritages (introduit avec ES6).

DOM et manipulation d'éléments :

17. **Comment accédez-vous à un élément HTML dans le DOM ?**
    Via document.getElementById, querySelector, ou getElementsByClassName.

18. **Quelles sont les différences entre getElementById et querySelector ?**
    - getElementById : Sélectionne un élément par son ID (unique).
    - querySelector : Sélectionne un élément avec un sélecteur CSS.

19. **Comment ajouter ou modifier des classes CSS ?**
    Avec classList:

javascript

Copier le code

element.classList.add('new-class');

element.classList.remove('old-class');

20. **Comment gérez-vous les événements DOM avec addEventListener ?**
    En attachant une fonction à un événement :

javascript

Copier le code

element.addEventListener('click', function() { ... });

Fonctionnalités modernes :

21. **Qu'est-ce que ES6 ?**
    Une version majeure de JavaScript introduisant des fonctionnalités modernes comme let, const, classes, et les templates de chaînes.

22. **Quelle est la différence entre Map et Object ?**
    - Map : Optimisé pour stocker des clés/valeurs.
    - Object : Structure de base pour les données non ordonnées.

23. **Qu'est-ce que les Arrow Functions ?**
    Une syntaxe concise pour écrire des fonctions. Elles ne lient pas leur propre this.

javascript

Copier le code

const func = () => { ... };

24. **Comment fonctionne le destructuring ?**
    Permet d'extraire des valeurs d'objets ou de tableaux :

javascript

Copier le code

const [a, b] = [1, 2];

const {x, y} = {x: 10, y: 20};

25. **Qu'est-ce que import et export ?**
    Utilisés pour modulariser le code :

javascript

Copier le code

export const func = () => {};

import { func } from './file.js';

Docker : Questions et Réponses

1. **Qu'est-ce que Docker et pourquoi l'utiliser ?**
   Docker est une plateforme de conteneurisation qui permet de regrouper une application et ses dépendances dans un conteneur. Cela garantit que l'application s'exécute de manière cohérente dans tous les environnements.
2. **Quelle est la différence entre un conteneur et une machine virtuelle ?**
   - Conteneur : Léger, partage le noyau de l'hôte, démarrage rapide.
   - Machine virtuelle : Plus lourd, émule un système complet avec son propre OS.
3. **Que fait un fichier Dockerfile ?**
   Un Dockerfile est un script contenant des instructions pour construire une image Docker (base, installation des dépendances, configuration).
4. **Comment fonctionnent les volumes dans Docker ?**
   Les volumes permettent de persister les données générées par un conteneur, même après son arrêt.
5. **Qu'est-ce qu'une image Docker ?**
   Une image est un modèle immuable contenant tout ce dont un conteneur a besoin pour fonctionner (code, dépendances, configuration).

6. **Comment gérez-vous la sécurité dans Docker ?**
    - Utiliser des images officielles.
    - Gérer les permissions avec des utilisateurs non-root.
    - Mettre à jour régulièrement les images et dépendances.

7. Aspect Dockerfile Docker Compose

But Créer une image Docker. Orchestrer plusieurs conteneurs.

Fichier associé `Dockerfile` `docker-compose.yml`

Complexité Configure un conteneur unique. Configure et connecte plusieurs conteneurs.

Utilisation	Définir l'environnement d'exécution d'une application.	Lancer des environnements complexes (multi-conteneurs).
Format	Basé sur des instructions Docker.	Basé sur YAML.
Exemples courants	Installer des dépendances pour un conteneur unique.	Configurer une application web avec une base de données.

Kubernetes : Questions et Réponses

1. **Qu'est-ce que Kubernetes et pourquoi est-il utilisé ?**
   Kubernetes est une plateforme d'orchestration de conteneurs qui automatise leur déploiement, mise à l'échelle et gestion.
2. **Qu'est-ce qu'un Pod dans Kubernetes ?**
   Un Pod est l'unité de base de Kubernetes, regroupant un ou plusieurs conteneurs qui partagent les mêmes ressources réseau et stockage.
3. **Quelle est la différence entre un Deployment et un Service ?**
    - Deployment : Définit comment déployer et gérer un Pod.
    - Service : Expose les Pods pour permettre leur accès réseau (interne ou externe).
4. **Comment Kubernetes gère-t-il la scalabilité ?**
   Kubernetes ajuste dynamiquement le nombre de Pods en fonction de la charge grâce à l'autoscaling horizontal.
5. **Qu'est-ce qu'un ConfigMap et un Secret dans Kubernetes ?**
    - ConfigMap : Stocke des configurations non sensibles pour les applications.
    - Secret : Stocke des données sensibles comme des mots de passe ou des clés.
6. **Qu'est-ce que le kube-scheduler ?**
   C'est un composant de Kubernetes qui décide sur quel nœud (machine) chaque Pod doit être déployé.

Jenkins : Questions et Réponses

1. **Qu'est-ce que Jenkins et pourquoi l'utiliser ?**
   Jenkins est un outil d'intégration et de déploiement continu (CI/CD) qui automatise les tâches comme la construction, le test, et le déploiement du code.
2. **Comment configurez-vous un pipeline Jenkins ?**
   En utilisant un fichier Jenkinsfile, qui définit les étapes (stages) et actions (steps) du pipeline.
3. **Qu'est-ce qu'un agent dans Jenkins ?**
   Un agent est une machine (ou conteneur) qui exécute les tâches du pipeline, comme les tests ou les builds.
4. **Comment intégrer Jenkins avec Git ?**
   En configurant un webhook dans le dépôt Git pour déclencher automatiquement des pipelines dans Jenkins après chaque modification.
5. **Comment gérez-vous la sécurité dans Jenkins ?**
   - Restreindre les accès avec des rôles et permissions.
   - Utiliser un HTTPS sécurisé.
   - Garder Jenkins à jour.
6. **Quelle est la différence entre un freestyle job et un pipeline dans Jenkins ?**
   - Freestyle : Configuration manuelle, moins flexible.
   - Pipeline : Défini dans un fichier, plus adapté pour les workflows complexes.

GitHub : Questions et Réponses

1. **Qu'est-ce que GitHub et pourquoi l'utiliser ?**
   GitHub est une plateforme de gestion de code source basée sur Git, qui permet la collaboration et le suivi des versions.
2. **Quelle est la différence entre un fork et un clone dans GitHub ?**
   - Fork : Copie d'un dépôt pour le modifier sans affecter l'original.
   - Clone : Copie locale d'un dépôt GitHub pour travailler dessus.
3. **Comment fonctionnent les pull requests dans GitHub ?**
   Une pull request propose des changements à un dépôt principal. Les mainteneurs peuvent les examiner et les fusionner ou les rejeter.
4. **Qu'est-ce qu'une GitHub Action ?**
   Une GitHub Action est un outil d'automatisation (CI/CD) permettant d'exécuter des workflows comme des tests ou des déploiements.
5. **Comment sécuriser un dépôt GitHub ?**
   - Utiliser des branches protégées.
   - Activer l'authentification à deux facteurs.
   - Restreindre les accès par équipes et rôles.

6. **Qu'est-ce que GitHub Pages ?**
   Un service permettant d'héberger des sites web directement depuis un dépôt GitHub.

GitLab : Questions et Réponses

1. **Qu'est-ce que GitLab et comment diffère-t-il de GitHub ?**
   GitLab est une plateforme de gestion de code source qui inclut des outils CI/CD natifs, tandis que GitHub nécessite des intégrations externes pour certaines fonctionnalités.
2. **Qu'est-ce qu'un runner dans GitLab CI/CD ?**
   Un runner exécute les jobs définis dans le pipeline CI/CD. Il peut être partagé ou spécifique à un projet.
3. **Comment configurez-vous un pipeline dans GitLab ?**
   Avec un fichier .gitlab-ci.yml, qui définit les étapes, scripts et variables pour le pipeline.
4. **Quelle est la différence entre GitLab Community Edition (CE) et Enterprise Edition (EE) ?**
   - CE : Version gratuite avec les fonctionnalités de base.
   - EE : Version payante avec des fonctionnalités avancées pour les grandes entreprises.
5. **Comment gérer les permissions des utilisateurs dans GitLab ?**
   En utilisant les rôles prédéfinis comme développeur, mainteneur, ou propriétaire, associés à des niveaux d'accès spécifiques.
6. **Comment GitLab CI/CD facilite-t-il le déploiement ?**
   En automatisant les builds, les tests, et les déploiements via des runners intégrés.

Git : Questions et Réponses

1. **Qu'est-ce que Git et pourquoi est-il utilisé ?**
   Git est un système de gestion de version distribué qui permet de suivre les modifications du code et de collaborer efficacement.
2. **Quelle est la différence entre un commit et un push ?**
   - Commit : Enregistre des changements localement dans la branche.
   - Push : Envoie les changements vers le dépôt distant.
3. **Comment gérez-vous les conflits de fusion dans Git ?**
   En résolvant les conflits manuellement dans les fichiers concernés, puis en effectuant un commit pour valider les modifications.
4. **Quelle est la différence entre git fetch et git pull ?**
   - git fetch : Récupère les modifications depuis le dépôt distant sans les fusionner.
   - git pull : Récupère et fusionne directement les modifications.

5. **Comment utiliser des branches dans Git ?**
   Les branches permettent de travailler sur des fonctionnalités ou correctifs séparément sans affecter la branche principale. On peut ensuite les fusionner via git merge ou git rebase.
6. **Comment annuler un commit ?**
   - git revert : Crée un commit inversé pour annuler les modifications.
   - git reset : Supprime un ou plusieurs commits de l'historique local.

React : Questions et Réponses

1. **Qu'est-ce qu'un composant dans React ?**
   Un composant est une unité réutilisable d'interface utilisateur. Il peut être fonctionnel ou basé sur une classe et gère son propre état ou reçoit des données via des props.
2. **Quelle est la différence entre un composant fonctionnel et un composant basé sur une classe ?**
   - Fonctionnel : Une simple fonction JavaScript, sans this, introduit avec les hooks pour gérer l'état.
   - Classe : Défini avec class, gère l'état via this.state et utilise lifecycle methods.
3. **À quoi sert le Hook useEffect ?**
   Il permet d'exécuter des effets secondaires (comme les appels API, mise à jour DOM) dans un composant fonctionnel.
4. **Qu'est-ce que le Virtual DOM dans React ?**
   Une copie légère du DOM réel qui permet de minimiser les manipulations DOM réelles et d'améliorer les performances.
5. **Qu'est-ce que Redux et pourquoi est-il utilisé avec React ?**
   Redux est une bibliothèque de gestion d'état centralisée. Il est utilisé avec React pour gérer efficacement les états complexes dans des applications volumineuses.
6. **Comment gérer la montée des états dans React ?**
   L'état doit être élevé au plus petit composant commun qui a besoin d'y accéder, puis partagé via des props.

Angular : Questions et Réponses

1. **Qu'est-ce qu'un module dans Angular ?**
   Un module est une unité de structure logique qui regroupe des composants, directives, services et pipes pour une partie spécifique de l'application.
2. **Quelle est la différence entre un service et un composant dans Angular ?**
   - Service : Contient la logique métier et les fonctionnalités partagées.
   - Composant : Gère l'interface utilisateur et les interactions.

3. **À quoi sert le décorateur @Input() ?**
   Il permet de transmettre des données d'un composant parent à un composant enfant.
4. **Qu'est-ce qu'une directive en Angular ?**
   Une directive est utilisée pour manipuler le DOM. Par exemple :
   - Structurelle (*ngIf, *ngFor) : Ajoute ou supprime des éléments.
   - Attribut ([ngClass]) : Modifie les propriétés d'un élément.
5. **Quelle est la différence entre Reactive Forms et Template-Driven Forms ?**
   - **Reactive Forms** : Utilisent des structures de données dans le TypeScript pour contrôler les formulaires.
   - **Template-Driven Forms** : Contrôlés directement via le HTML.
6. **Comment gérez-vous les appels HTTP dans Angular ?**
   Avec le service HttpClient fourni par Angular, en utilisant des observables (.get(), .post(), etc.).

Django : Questions et Réponses

1. **Qu'est-ce que le modèle MVT dans Django ?**
   MVT (Model-View-Template) est l'architecture de Django :
   - Model : Gère les données et leur interaction avec la base.
   - View : Contient la logique métier et envoie les données au template.
   - Template : Génère le HTML affiché à l'utilisateur.
2. **À quoi sert le fichier settings.py ?**
   Il contient toutes les configurations globales de l'application, comme les bases de données, les chemins statiques, et les middlewares.
3. **Comment fonctionne le système ORM de Django ?**
   L'ORM (Object-Relational Mapping) permet d'interagir avec la base de données en manipulant des objets Python au lieu d'utiliser des requêtes SQL.
4. **Comment définir une relation "one-to-many" dans un modèle Django ?**
   En utilisant ForeignKey :

python

Copier le code

```
class Article(models.Model):
    author = models.ForeignKey(User, on_delete=models.CASCADE)
```

5. **Qu'est-ce qu'un middleware dans Django ?**
   Un middleware est une fonction ou une classe qui agit sur les requêtes et réponses HTTP, par exemple pour l'authentification ou la gestion des cookies.
6. **Comment gérez-vous les migrations dans Django ?**
   Avec les commandes :

- python manage.py makemigrations : Prépare les migrations.
- python manage.py migrate : Applique les migrations à la base.

Spring Boot : Questions et Réponses

1. **Qu'est-ce que Spring Boot et pourquoi l'utiliser ?**
   Spring Boot est un framework basé sur Spring qui simplifie le développement en intégrant des configurations par défaut et des outils comme les starters.
2. **Comment fonctionnent les annotations dans Spring Boot ?**
   - @RestController : Déclare une classe comme contrôleur REST.
   - @Autowired : Injecte une dépendance automatiquement.
   - @Entity : Marque une classe comme une entité persistante.
3. **Comment Spring Boot gère-t-il les dépendances ?**
   Avec **Maven** ou **Gradle**, utilisant des starters comme spring-boot-starter-web pour simplifier l'ajout des bibliothèques nécessaires.
4. **Qu'est-ce que Spring Data JPA ?**
   Une abstraction de Spring Boot pour interagir facilement avec des bases de données relationnelles en utilisant des repositories.
5. **Comment gérer les propriétés de configuration dans Spring Boot ?**
   Avec les fichiers application.properties ou application.yml.
6. **Qu'est-ce que le mécanisme de gestion des exceptions globales dans Spring Boot ?**
   Utilisation de @ControllerAdvice et @ExceptionHandler pour capturer et traiter les exceptions.

SQL : Questions et Réponses

1. **Quelle est la différence entre INNER JOIN et LEFT JOIN ?**
   - INNER JOIN : Retourne uniquement les lignes correspondantes.
   - LEFT JOIN : Retourne toutes les lignes de la table de gauche, même si aucune correspondance n’existe.
2. **Qu'est-ce qu'une clé primaire ?**
   Une colonne ou un ensemble de colonnes qui identifie de manière unique chaque ligne dans une table.
3. **Qu'est-ce qu'une requête de groupe dans SQL ?**
   Une requête qui utilise GROUP BY pour regrouper les données selon des critères spécifiques, souvent avec des fonctions d'agrégation comme SUM ou AVG.
4. **Comment optimiser une requête SQL ?**
   - Ajouter des index.

- Éviter les requêtes imbriquées complexes.
- Utiliser les limites (LIMIT) pour réduire les résultats.

5. **Quelle est la différence entre DELETE et TRUNCATE ?**
   - DELETE : Supprime les lignes avec des conditions, déclenche les triggers.
   - TRUNCATE : Supprime toutes les lignes sans déclencher les triggers.
6. **Qu'est-ce qu'une vue en SQL ?**
   Une vue est une requête stockée qui agit comme une table virtuelle.

NoSQL : Questions et Réponses

1. **Qu'est-ce que NoSQL ?**
   NoSQL désigne des bases de données non relationnelles conçues pour gérer de grandes quantités de données non structurées ou semi-structurées.
2. **Quand utiliser NoSQL au lieu de SQL ?**
   - Données non structurées ou en évolution rapide.
   - Besoin de haute scalabilité et faible latence.
3. **Quelles sont les catégories principales des bases NoSQL ?**
   - Document (ex : MongoDB).
   - Clé-valeur (ex : Redis).
   - Colonne (ex : Cassandra).
   - Graphes (ex : Neo4j).
4. **Quelle est la différence entre MongoDB et une base SQL ?**
   - MongoDB stocke des documents JSON-like, sans schéma fixe.
   - SQL utilise des tables avec schémas fixes.
5. **Comment gérez-vous les relations dans une base NoSQL ?**
   Avec l'imbrication de documents ou par des références (relations manuelles).
6. **Qu'est-ce qu'un index dans MongoDB ?**
   Un index améliore la vitesse des requêtes en facilitant la recherche des documents.

HTML5 : Questions et Réponses

1. **Quelles sont les nouvelles balises introduites par HTML5 ?**
   - Structure : <header>, <footer>, <section>, <article>.
   - Médias : <audio>, <video>.
   - Formulaire : <datalist>, <output>.

2. **Qu'est-ce qu'une balise sémantique ?**
   Une balise qui donne un sens clair à son contenu (ex : <article> indique un article).
3. **À quoi sert l'attribut data-* ?**
   Il stocke des données personnalisées associées à un élément HTML.
4. **Qu'est-ce que le localStorage ?**
   Une fonctionnalité qui permet de stocker des données localement dans le navigateur sans expiration.
5. **Quelle est la différence entre <script> avec defer et async ?**
   - async : Télécharge et exécute le script dès qu'il est prêt.
   - defer : Exécute le script après le chargement complet du DOM.
6. **Qu'est-ce que la compatibilité rétrograde d'HTML5 ?**
   Les navigateurs modernes ignorent les balises non reconnues, assurant la compatibilité.

CSS3 : Questions et Réponses

1. **Quelles sont les nouveautés introduites par CSS3 ?**
   - Animations (@keyframes).
   - Grilles CSS (display: grid).
   - Transitions (transition).
2. **Qu'est-ce qu'un Flexbox ?**
   Un modèle de mise en page pour aligner et distribuer les éléments dans un conteneur.
3. **Qu'est-ce que la pseudo-classe :nth-child() ?**
   Elle sélectionne un enfant spécifique dans un conteneur selon sa position.
4. **Qu'est-ce qu'une media query ?**
   Une règle CSS pour adapter le style en fonction de la taille ou des capacités de l'écran.
5. **Quelle est la différence entre relative, absolute et fixed ?**
   - relative : Position par rapport à son emplacement normal.
   - absolute : Par rapport à l'ancêtre positionné.
   - fixed : Par rapport à la fenêtre du navigateur.
6. **Comment créer une animation CSS simple ?**
   En utilisant @keyframes et animation:

css

Copier le code

```
@keyframes example { from { opacity: 0; } to { opacity: 1; } }

div { animation: example 2s; }
```

Tailwind CSS : Questions et Réponses

1. **Qu'est-ce que Tailwind CSS ?**
   Tailwind CSS est un framework utilitaire qui permet de concevoir des interfaces rapidement en utilisant des classes prédéfinies.
2. **Quelle est la différence entre Tailwind CSS et CSS classique ?**
   Tailwind CSS utilise des classes utilitaires prêtes à l'emploi, contrairement à CSS classique qui nécessite d'écrire des styles personnalisés.
3. **Comment personnaliser une configuration Tailwind CSS ?**
   En modifiant le fichier tailwind.config.js pour ajouter des couleurs, des polices, ou des espacements.
4. **Comment gérer le responsive design avec Tailwind CSS ?**
   En utilisant des préfixes comme sm:, md:, ou lg: pour appliquer des styles spécifiques aux tailles d'écran.
5. **Quels sont les avantages de Tailwind CSS ?**
   - Rapidité de développement.
   - Pas besoin de créer de fichiers CSS séparés.
   - Design réactif intégré.
6. **Qu'est-ce que le mode JIT (Just-In-Time) de Tailwind CSS ?**
   Un mode qui génère les classes CSS à la demande, réduisant ainsi la taille finale des fichiers CSS.

7. 1. Problème : Vérifier si une chaîne est une permutation d'un palindrome

8. Énoncé :

9. Écrire une fonction qui vérifie si une chaîne donnée peut être réarrangée pour former un palindrome. Un palindrome a au maximum un caractère avec une fréquence impaire.

10. Solution :

```
11.
12. from collections import Counter
13.
14. def is_permutation_of_palindrome(s):
15.     # Nettoyer la chaîne (supprimer les espaces et convertir en minuscules)
16.     s = s.replace(" ", "").lower()
17.     char_count = Counter(s)
18.
19.     # Vérifier le nombre de caractères avec une fréquence impaire
20.     odd_count = sum(1 for count in char_count.values() if count % 2 != 0)
21.     return odd_count <= 1
22.
23. # Exemple
24. print(is_permutation_of_palindrome("Tact Coa"))  # True (e.g., "taco cat", "atco cta")
```

---

25. 2. Problème : Trouver la sous-chaîne palindrome la plus longue

26. Énoncé :

27. Écrire une fonction qui retourne le plus long palindrome dans une chaîne donnée.

28. Solution :

```
def longest_palindrome(s):
    def expand_around_center(left, right):
        while left >= 0 and right < len(s) and s[left] == s[right]:
            left -= 1
            right += 1
        return s[left + 1:right]

    longest = ""
    for i in range(len(s)):
        # Vérifier les deux types de centres
        odd_palindrome = expand_around_center(i, i)
        even_palindrome = expand_around_center(i, i + 1)
        longest = max(longest, odd_palindrome, even_palindrome, key=len)

    return longest

# Exemple
print(longest_palindrome("babad"))  # "bab" ou "aba"
```

47.3. Problème : Trouver le mot unique

48. Énoncé :

49. Étant donné un tableau où tous les mots apparaissent deux fois sauf un, trouvez le mot unique.

50. Solution :

```
from collections import Counter

def find_unique_word(words):
    word_count = Counter(words)
    for word, count in word_count.items():
        if count == 1:
            return word
    return None

# Exemple
words = ["apple", "banana", "apple", "orange", "banana"]
print(find_unique_word(words))  # "orange"
```

51.4. Problème : Trouver la sous-séquence croissante la plus longue

52. Énoncé :

53. Écrire une fonction qui retourne la longueur de la plus longue sous-séquence strictement croissante dans un tableau d'entiers.

54. Solution :

```
55.      def length_of_lis(nums):
56.          if not nums:
57.              return 0
58.
59.          dp = [1] * len(nums)  # dp[i] stocke la longueur de la sous-
    séquence croissante se terminant à i
60.          for i in range(1, len(nums)):
61.              for j in range(i):
62.                  if nums[i] > nums[j]:
63.                      dp[i] = max(dp[i], dp[j] + 1)
```

```
64.
65.         return max(dp)
66.
67.     # Exemple
68.     print(length_of_lis([10, 9, 2, 5, 3, 7, 101, 18]))  # 4 (2, 3, 7,
   101)
```
69. ---

70. 5. Problème : Vérifier si une chaîne est une rotation d'une autre

71. Énoncé :

72. Écrire une fonction qui vérifie si une chaîne est une rotation d'une autre. Par exemple, "waterbottle" est une rotation de "erbottlewat".

73. Solution :

```
74. def is_rotation(s1, s2):
75.     if len(s1) != len(s2):
76.         return False
77.     return s2 in (s1 + s1)
78.
79. # Exemple
80. print(is_rotation("waterbottle", "erbottlewat"))  # True
81. print(is_rotation("hello", "world"))  # False
```

82. 6. Problème : Trouver les deux nombres avec une somme donnée

83. Énoncé :

84. Étant donné un tableau d'entiers et une cible, trouver deux nombres qui additionnent pour donner cette cible.

85. Solution :

```
def two_sum(nums, target):
    num_map = {}  # Dictionnaire pour stocker les valeurs et leurs indices
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_map:
            return [num_map[complement], i]
        num_map[num] = i
    return []


# Exemple
nums = [2, 7, 11, 15]
target = 9
print(two_sum(nums, target))  # [0, 1]
```

1. Maven

Maven est un outil de gestion de projets et de construction (build automation) principalement utilisé pour les projets Java. Il facilite la gestion des dépendances, l'automatisation des compilations, des tests, et du déploiement.

Caractéristiques :

- **Gestion des dépendances** : Maven permet de télécharger automatiquement les bibliothèques nécessaires à votre projet à partir de repositories centralisés comme Maven Central.
- **Cycle de vie de construction** : Il définit un cycle de vie complet pour la construction du projet, y compris la compilation, les tests, l'assemblage et le déploiement.
- **Fichier de configuration :** pom.xml (Project Object Model), qui contient des informations sur le projet, ses dépendances, ses plugins, etc.
- **Convention plutôt que configuration** : Maven a des conventions prédéfinies, ce qui facilite la configuration du projet.

Exemple de commande Maven :

bash

Copier le code

```
mvn clean install
```

---

2. Gradle

Gradle est un autre outil de construction open-source qui est plus flexible et plus rapide que Maven. Il est souvent utilisé dans des projets Java, mais prend également en charge d'autres langages comme Groovy, Kotlin, et C/C++.

Caractéristiques :

- **Utilisation de Groovy et Kotlin :** Gradle permet d'écrire des scripts de construction avec des langages de programmation Groovy ou Kotlin, offrant ainsi plus de flexibilité et de personnalisation par rapport à Maven.
- **Performances** : Gradle utilise un système de cache intelligent pour améliorer les performances. Il ne reconstruit que ce qui a changé, réduisant ainsi le temps de construction.
- **Polyvalence** : Bien qu'il soit principalement utilisé avec Java, il est compatible avec d'autres langages et frameworks.
- **Configuration fluide** : Gradle permet de personnaliser presque tous les aspects de la construction grâce à ses scripts de build.

Exemple de commande Gradle :

bash

Copier le code

```
gradle build
```

---

3. Apache (Apache Software Foundation)

Apache fait référence à une organisation (Apache Software Foundation) qui soutient un grand nombre de projets open-source. Cependant, dans le contexte des outils de développement, il existe plusieurs projets qui utilisent le nom "Apache", comme **Apache Ant** et **Apache Maven**.

Caractéristiques :

- **Apache Ant :** Un autre outil de construction (build tool) pour Java, moins conventionnel que Maven. Il offre une approche plus flexible mais nécessite davantage de configuration.
- **Apache HTTP Server :** Serveur web open-source populaire pour héberger des applications web, mais non lié directement à la gestion de projets ou de constructions de code.
- **Apache Kafka** : Système de gestion de flux de données. Il n'est pas un outil de construction, mais il appartient à la même organisation Apache.

Exemple de projet Apache utilisé dans les builds :

- **Apache Ant** est souvent utilisé dans les projets Java pour gérer les tâches de construction, mais il est plus manuel que Maven et Gradle.

Différences entre Maven et Gradle :

1. **Configuration :**
    - **Maven** : Utilise un fichier XML (pom.xml), avec une approche "convention over configuration".
    - **Gradle** : Utilise des fichiers de script Groovy ou Kotlin (build.gradle), offrant plus de flexibilité et une syntaxe plus concise.
2. **Performances :**
    - **Maven** : Plus lent comparé à Gradle, surtout pour des projets très larges, car il recompte tout depuis le début.
    - **Gradle** : Plus rapide grâce à l'optimisation et le cache intelligent.
3. **Extensibilité :**
    - **Maven** : Moins flexible. Il suit des conventions strictes.
    - **Gradle** : Hautement configurable et extensible, avec la possibilité d’ajouter facilement des plugins personnalisés.
4. **Dépendances et gestion des versions :**
    - **Maven** : Gestion très simplifiée des dépendances via pom.xml.
    - **Gradle** : Utilise le même système de gestion de dépendances, mais permet une meilleure gestion des versions et des configurations via le script build.gradle.
5. **Plugins :**

- **Maven** : Large écosystème de plugins, mais configuration plus rigide.
- **Gradle** : Plugins flexibles et adaptés aux besoins spécifiques.

6. **Utilisation :**
   - **Maven** : Utilisé principalement pour les projets Java et les projets d'entreprise.
   - **Gradle** : Utilisé pour les projets modernes, principalement Java, mais aussi pour Android, C/C++, Python, etc.

Un **design pattern** est une solution réutilisable à un problème récurrent dans le développement logiciel. C'est une sorte de **modèle conceptuel** qui guide les développeurs pour résoudre des problèmes courants de manière standardisée

Un **token** est une chaîne de caractères générée par un serveur pour identifier un utilisateur ou une session de manière sécurisée. Il est souvent utilisé pour l'authentification et l'autorisation dans les applications web.

Un **JSON Web Token (JWT)** est un standard (RFC 7519) pour créer des tokens sécurisés et portables. Il est souvent utilisé pour l'authentification et le transfert de données entre des parties.

. Qu'est-ce que Laravel et pourquoi l'utiliser ?

- **Question :** Expliquez Laravel et ses avantages par rapport à d'autres frameworks PHP.
- **Réponse :**
  Laravel est un framework PHP open-source basé sur le modèle MVC (Model-View-Controller). Il offre une syntaxe élégante et des fonctionnalités modernes comme l'ORM Eloquent, la gestion des routes, les migrations, et l'intégration facile avec des outils tiers.
- **Qu'est-ce que l'ORM Eloquent ?**
- **Question :** Expliquez comment fonctionne Eloquent dans Laravel.
- **Réponse :**
  Eloquent est l'ORM (Object-Relational Mapping) intégré à Laravel. Il permet d'interagir avec une base de données en manipulant des objets au lieu d'écrire des requêtes SQL complexes. Chaque table de la base correspond à un **modèle** Eloquent.
- **3. Comment fonctionne le moteur de templates Blade ?**
- **Question :** Expliquez le rôle et les avantages du moteur de templates Blade.
- **Réponse :**
  Blade est le moteur de templates de Laravel, qui permet de créer des vues dynamiques tout en utilisant une syntaxe propre.

- **4. Comment fonctionne le système de routage dans Laravel ?**
- **Question :** Expliquez la gestion des routes dans Laravel.
- **Réponse :**
  Les routes définissent les chemins HTTP de l'application et leur logique associée. Elles sont configurées dans les fichiers comme routes/web.php ou routes/api.php.

Qu'est-ce que la migration de base de données dans Laravel ?

- **Question :** Comment les migrations gèrent-elles les bases de données ?
- **Réponse :**
  Les migrations permettent de versionner les schémas de base de données. Cela facilite la création, modification ou suppression de tables via des scripts PHP, sans manipuler directement la base.

Comment Laravel gère-t-il l'authentification ?

- **Question :** Expliquez le système d'authentification de Laravel.
- **Réponse :**
  Laravel propose un système d'authentification intégré via des middleware comme auth et des packages comme laravel/breeze ou laravel/sanctum.

Voici trois problèmes algorithmiques populaires souvent posés en entretiens techniques, avec leurs solutions en Python :

---

1. Vérifier si un nombre est premier

Problème :

Un nombre est dit premier s'il est supérieur à 1 et n'a aucun diviseur autre que 1 et lui-même. Écrire une fonction pour vérifier si un nombre donné est premier.

Solution :

python

Copier le code

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):  # Vérifie jusqu'à la racine carrée de n
        if n % i == 0:
            return False
    return True
```

```
# Exemple d'utilisation
print(is_prime(7))  # True
print(is_prime(10)) # False
```

2. Trouver la sous-suite maximale de somme maximale (Kadane's Algorithm)

Problème :

Étant donné un tableau d'entiers, trouvez la sous-suite contiguë ayant la somme maximale.

Solution :

python

Copier le code

```
def max_subarray_sum(nums):
    max_sum = nums[0]
    current_sum = nums[0]

    for i in range(1, len(nums)):
        current_sum = max(nums[i], current_sum + nums[i])
        max_sum = max(max_sum, current_sum)

    return max_sum
```

---

3. Vérifier si une chaîne est un palindrome

Problème :

Écrire une fonction pour vérifier si une chaîne est un palindrome (se lit de la même manière dans les deux sens).

```
def is_palindrome(s):
    s = ''.join(c.lower() for c in s if c.isalnum())  # Supprime les espaces et caractères spéciaux
    return s == s[::-1]  # Compare la chaîne à son inverse

# Exemple d'utilisation
print(is_palindrome("A man, a plan, a canal: Panama"))  # True
print(is_palindrome("hello"))                           # False
```

Bonjour, je m'appelle Yosra Omrane, et je suis en dernière année d'ingénierie logicielle à l'Institut Supérieur d'Informatique et de Mathématiques de Monastir. Mon parcours m'a permis d'acquérir des compétences solides en développement logiciel, apprentissage automatique, et en DevOps.

Lors de mes stages, j'ai travaillé sur des projets variés comme le développement d'un système de recommandation intelligent pour l'irrigation et un chatbot multilingue basé sur des techniques NLP. Je suis également passionnée par les compétitions, comme la Nuit de l'Info et IEEEXtreme, où j'ai développé mes compétences en résolution de problèmes et en travail d'équipe.

Je suis ici aujourd'hui pour mettre à profit mes compétences dans un environnement qui valorise l'innovation et pour continuer à apprendre et relever des défis techniques.

J'ai choisi cette société car elle est reconnue pour ses avancées dans le domaine de developeement logiciel et son engagement envers l'innovation technologique. Votre projet de Application web de branding et de lancement de marque m'inspire pour travailler avec vous. De plus, je suis attiré par votre environnement collaboratif et votre culture axée sur l'apprentissage continu, qui sont en parfaite adéquation avec mes objectifs professionnels et personnels.

Tout d'abord, je tiens à vous remercier de m'avoir offert cette opportunité d'entretien pour mieux discuter de ma candidature

```
def is_palindrome(s):
    s = ''.join(c.lower() for c in s if c.isalnum())  # Normaliser la chaîne
    return s == s[::-1]

print(is_palindrome("A man, a plan, a canal: Panama"))  # True

def length_of_lis(nums):
    dp = [1] * len(nums)
    for i in range(len(nums)):
        for j in range(i):
            if nums[i] > nums[j]:
                dp[i] = max(dp[i], dp[j] + 1)
    return max(dp, default=0)

print(length_of_lis([10, 9, 2, 5, 3, 7, 101, 18]))  # 4
```

```
def two_sum(nums, target):
    num_map = {}
    for i, num in enumerate(nums):
        complement = target - num
        if complement in num_map:
            return [num_map[complement], i]
        num_map[num] = i
    return []

print(two_sum([2, 7, 11, 15], 9))  # [0, 1]



def length_of_longest_substring(s):
    char_set = set()
    left, max_length = 0, 0
    for right in range(len(s)):
        while s[right] in char_set:
            char_set.remove(s[left])
            left += 1
        char_set.add(s[right])
        max_length = max(max_length, right - left + 1)
    return max_length

print(length_of_longest_substring("abcabcbb"))  # 3

def are_anagrams(s1, s2):
    return sorted(s1) == sorted(s2)

print(are_anagrams("listen", "silent"))  # True
```

```
def first_non_repeated(s):
    char_count = Counter(s)
    for char in s:
        if char_count[char] == 1:
            return char
    return None

print(first_non_repeated("swiss"))  # 'w'

def max_subarray_sum(nums):
    max_sum = nums[0]
    current_sum = nums[0]
    for i in range(1, len(nums)):
        current_sum = max(nums[i], current_sum + nums[i])
        max_sum = max(max_sum, current_sum)
    return max_sum

print(max_subarray_sum([-2, 1, -3, 4, -1, 2, 1, -5, 4]))  # 6

def majority_element(nums):
    counts = Counter(nums)
    for num, count in counts.items():
        if count > len(nums) // 2:
            return num
    return None

print(majority_element([3, 3, 4, 2, 3, 3, 3]))  # 3

def three_sum(nums):
    nums.sort()
    result = []
```

```
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        left, right = i + 1, len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total == 0:
                result.append([nums[i], nums[left], nums[right]])
                left += 1
                right -= 1
            elif total < 0:
                left += 1
            else:
                right -= 1
    return result

print(three_sum([-1, 0, 1, 2, -1, -4]))  # [[-1, -1, 2], [-1, 0, 1]]

from collections import Counter

def common_chars_unique(words):
    # Initialiser avec les caractères du premier mot
    common_count = Counter(words[0])

    # Intersection des compteurs pour chaque mot
    for word in words[1:]:
        common_count &= Counter(word)  # Intersection des fréquences

    # Retourner uniquement les caractères uniques (les clés du compteur final)
    return list(common_count.keys())
```

```
# Exemple d'utilisation
words = ["bella", "label", "roller"]
print(common_chars_unique(words))  # ['e', 'l']


def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True


def primes_up_to_n(N):
    primes = []
    for i in range(2, N + 1):
        if is_prime(i):
            primes.append(i)
    return primes


# Exemple d'utilisation
print(primes_up_to_n(10))  # [2, 3, 5, 7]
print(primes_up_to_n(30))  # [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

1. Programmation

1. **Quel est le résultat de l'exécution du code suivant ?**

```
python
Copier le code
x = [1, 2, 3]
y = x
y[0] = 10
print(x)
```

Réponse : b) [10, 2, 3]

2. **Quel type de structure de données est la meilleure pour stocker une collection unique d'éléments sans ordre particulier ?**
   Réponse : c) Set

3. **Quelle est la complexité temporelle de la recherche d'un élément dans une liste non triée en Python ?**
   Réponse : c) O(n)

4. **Quelle est la sortie du code suivant en Python ?**

python

Copier le code

```
def func(a, b):
    return a + b
print(func(2, "3"))
```

Réponse : c) Erreur de type

5. **Quel est l'opérateur utilisé pour l'exponentiation en Python ?**
   Réponse : b) **

---

2. Structures de données

6. **Quelle structure de données est utilisée par Python pour implémenter une pile ?**
   Réponse : a) Liste

7. **Quel est le principal inconvénient d'une liste par rapport à un tableau fixe ?**
   Réponse : c) Taille dynamique mais temps d'accès non constant

8. **Que fait la méthode pop() sur une liste en Python ?**
   Réponse : b) Retire et retourne le dernier élément de la liste

9. **Dans quelle situation l'utilisation d'un tableau est préférable à une liste en Python ?**
   Réponse : b) Lorsqu'il faut manipuler des données numériques de manière efficace

10. **Que permet d'accomplir un dictionnaire en Python ?**
    Réponse : a) Permet de stocker des paires clé-valeur

---

3. Algorithmes

11. **Quelle est la complexité temporelle de l'algorithme de recherche binaire ?**
    Réponse : b) O(log n)

12. **Quel algorithme est le plus efficace pour trier un tableau d'éléments ?**
    Réponse : c) Tri rapide (QuickSort)

13. **Quel est l'objectif principal de l'algorithme de Dijkstra ?**
    Réponse : a) Trouver le chemin le plus court entre deux sommets dans un graphe

14. **Lequel des suivants n'est pas un algorithme de tri ?**
    Réponse : c) Hashing

15. **Laquelle des structures de données suivantes est utilisée par l'algorithme de Dijkstra pour stocker les distances ?**
    Réponse : b) File de priorité

---

4. Systèmes et Réseaux

16. **Que signifie HTTP dans les systèmes de communication ?**
    Réponse : a) HyperText Transfer Protocol

17. **Qu'est-ce qu'un DNS dans les réseaux informatiques ?**
    Réponse : c) Un système qui traduit les noms de domaine en adresses IP

18. **Quel est le rôle d'un serveur proxy ?**
    Réponse : d) Servir de point de relais pour la communication entre le client et le serveur

19. **Quelle est la principale différence entre TCP et UDP ?**
    Réponse : c) UDP ne garantit pas la livraison des données, tandis que TCP le garantit

20. **Qu'est-ce que la méthode GET dans HTTP ?**
    Réponse : b) Récupère des données à partir d'un serveur

---

5. Bases de données SQL

21. **Quelle est la commande SQL pour récupérer toutes les lignes d'une table ?**
    Réponse : a) SELECT * FROM table_name

22. **Qu'est-ce qu'une clé primaire dans une table ?**
    Réponse : a) Un champ unique qui identifie chaque enregistrement

23. **Que fait la commande SQL ALTER ?**
    Réponse : b) Modifie une table existante

24. **Quelle commande SQL permet de supprimer une ligne dans une table ?**
    Réponse : a) DELETE

25. **Quel est le type de jointure qui retourne toutes les lignes de la table de gauche et les lignes correspondantes de la table de droite ?**
    Réponse : b) LEFT JOIN

---

6. Conception de logiciels

26. **Quel modèle de conception est utilisé pour garantir qu'une classe n'a qu'une seule instance dans un programme ?**
    Réponse : b) Singleton

27. **Lequel des suivants est un exemple de principe SOLID ?**
    Réponse : d) Toutes les réponses ci-dessus

28. **Quelle méthode de conception est utilisée pour séparer l'interface de l'implémentation ?**
    Réponse : b) Bridge

29. **Dans quel design pattern les objets se tiennent au courant des changements d'un autre objet ?**
    Réponse : c) Observer

30. **Le pattern Factory est utilisé pour :**
    Réponse : a) Créer des objets sans spécifier la classe concrète

---

7. Tests et Débogage

31. **Quel est l'objectif principal des tests unitaires ?**
    Réponse : b) Tester une unité de code de manière isolée

32. **Quel est l'outil Python pour réaliser des tests unitaires ?**
    Réponse : d) Les deux a et b (unittest et pytest)

33. **Qu'est-ce qu'un test de régression ?**
    Réponse : b) Un test pour s'assurer que les anciennes fonctionnalités ne sont pas affectées par les nouvelles modifications

34. **Quel est le but principal du débogage ?**
    Réponse : b) Identifier et corriger les erreurs dans le programme

35. **Quelle méthode est utilisée pour exécuter un test dans unittest ?**
    Réponse : b) run()

---

8. Développement Web

36. **Quel est l'objectif principal d'un framework web comme Django ou Flask ?**
    Réponse : b) Fournir des outils pour créer des applications web rapidement

37. **Quel est le but principal d'une API REST ?**
    Réponse : a) Permettre la communication entre différents systèmes via HTTP

38. **Que signifie l'acronyme CRUD ?**
    Réponse : a) Create, Read, Update, Delete

39. **Dans une architecture MVC, que représente "V" ?**
    Réponse : a) Vue (View)

40. **Quel est le rôle principal d'une session en développement web ?**
    Réponse : b) Suivre les utilisateurs pendant une session de navigation

41. **1. Programmation**

42. **Quel est le résultat de ce code Python ?**

43. python

44. Copier le code

45. a = 5
46. b = 2
47. print(a / b)
48. **Réponse :** b) 2.5
49. **Comment pouvez-vous vérifier si une chaîne de caractères est vide en Python ?**
    Réponse : d) Toutes les réponses ci-dessus
50. **Quel est le but du mot-clé self en Python ?**
    Réponse : b) Représente l'instance de la classe
51. ---
52. **2. Structures de données**
53. **Quelle structure de données est la meilleure pour effectuer des recherches rapides ?**
    Réponse : b) Dictionnaire
54. **Quels types de données peuvent être stockés dans une base de données NoSQL ?**
    Réponse : c) Des données non structurées, semi-structurées ou structurées
55. **Quel est l'inconvénient principal des listes en Python par rapport aux tuples ?**
    Réponse : c) Les listes sont mutables, tandis que les tuples sont immutables
56. ---
57. **3. Algorithmes**
58. **Quel est l'algorithme de tri le plus performant dans le pire des cas ?**
    Réponse : c) Tri par fusion (MergeSort)
59. **Quel est le temps d'exécution moyen de l'algorithme de tri rapide (QuickSort) ?**
    Réponse : a) O(n log n)
60. **Dans l'algorithme de Dijkstra, quelle structure de données est utilisée pour stocker les distances minimales ?**
    Réponse : c) File de priorité
61. **Quel est l'objectif principal de l'algorithme de recherche binaire ?**
    Réponse : c) Trouver un élément dans une liste triée
62. ---
63. **4. IA et Machine Learning**
64. **Quel est le rôle de l'algorithme de régression linéaire en apprentissage supervisé ?**
    Réponse : b) Prédire une valeur continue à partir de données d'entrée
65. **Dans un réseau de neurones, quel est le rôle de la fonction d'activation ?**
    Réponse : c) Introduire des non-linéarités pour permettre au réseau d'apprendre des patterns complexes

66. **Qu'est-ce que l'overfitting en apprentissage automatique ?**
    Réponse : b) Le modèle s'adapte trop aux données d'entraînement et ne généralise pas bien aux nouvelles données

67. ---

68. 5. Cloud Computing

69. **Quel service AWS permet de déployer des applications web sans gérer les serveurs ?**
    Réponse : d) Elastic Beanstalk

70. **Qu'est-ce qu'un conteneur dans le contexte du cloud computing ?**
    Réponse : b) Un environnement isolé qui exécute des applications

71. **Quel est le rôle de l'auto-scaling dans le cloud ?**
    Réponse : b) Ajouter des instances automatiquement en fonction de la charge de travail

72. ---

73. 6. DevOps

74. **Quelle est la principale fonction de Jenkins ?**
    Réponse : b) Automatiser les tests et les déploiements via des pipelines CI/CD

75. **Que signifie le terme "Infrastructure as Code" (IaC) ?**
    Réponse : b) Utiliser du code pour gérer et provisionner des ressources d'infrastructure

76. **Quel est le rôle de Docker dans un environnement DevOps ?**
    Réponse : b) Conteneuriser les applications pour les rendre portables et facilement déployables

77. ---

78. 7. SQL et Bases de données

79. **Quelle commande SQL est utilisée pour modifier les valeurs dans une table ?**
    Réponse : b) UPDATE

80. **Quel est le type de jointure qui retourne toutes les lignes des deux tables, même s'il n'y a pas de correspondance ?**
    Réponse : c) FULL OUTER JOIN

81. **Que fait une commande SQL GROUP BY ?**
    Réponse : b) Regroupe les résultats selon une ou plusieurs colonnes

82. ---

83. 8. NoSQL

84. **Quel type de données est couramment stocké dans les bases de données NoSQL ?**
    Réponse : c) Des données non structurées, semi-structurées ou structurées

85. **Quel est le but de MongoDB ?**
    Réponse : b) Stocker des données sous forme de documents JSON

86. **Dans une base de données NoSQL, qu'est-ce qu'un "shard" ?**
Réponse : b) Un fragment de données réparti sur plusieurs serveurs pour améliorer la performance

87. ---

88. 9. Test et Débogage

89. **Quel est l'objectif des tests unitaires ?**
Réponse : b) Vérifier les composants individuels du code

90. **Quel est l'outil pour faire des tests unitaires en Python ?**
Réponse : d) Les deux a et b (unittest et pytest)

91. **Qu'est-ce qu'un test de régression ?**
Réponse : b) Un test pour s'assurer que les anciennes fonctionnalités ne sont pas affectées par les modifications récentes

92. ---

93. 10. Web et Réseaux

94. **Quel est le rôle du serveur web dans une application web ?**
Réponse : c) Exécuter le code d'application sur un serveur distant

95. **Qu'est-ce qu'une API RESTful ?**
Réponse : c) Un ensemble de règles permettant de construire des services web interopérables via HTTP

96. **Quel est le rôle principal de HTTP ?**
Réponse : a) Transférer des fichiers entre un serveur et un client

97. ---

98. Résumé des réponses :

99. **1. b) 2.5**

100. **2. d) Toutes les réponses ci-dessus**

101. **3. b) Représente l'instance de la classe**

102. **4. b) Dictionnaire**

103. **5. c) Des données non structurées, semi-structurées ou structurées**

104. **6. c) Les listes sont mutables, tandis que les tuples sont immutables**

105. **7. c) Tri par fusion (MergeSort)**

106. **8. a) O(n log n)**

107. **9. c) File de priorité**

108. **10. c) Trouver un élément dans une liste triée**

109. **11. b) Prédire une valeur continue à partir de données d'entrée**

12. c) Introduire des non-linéarités pour permettre au réseau d'apprendre des patterns complexes

13. b) Le modèle s'adapte trop aux données d'entraînement et ne généralise pas bien aux nouvelles données

14. d) Elastic Beanstalk

15. b) Un environnement isolé qui exécute des applications

16. b) Ajouter des instances automatiquement en fonction de la charge de travail

17. b) Automatiser les tests et les déploiements via des pipelines CI/CD

18. b) Utiliser du code pour gérer et provisionner des ressources d'infrastructure

19. b) Conteneuriser les applications pour les rendre portables et facilement déployables

20. b) UPDATE

21. c) FULL OUTER JOIN

22. b) Regroupe les résultats selon une ou plusieurs colonnes

23. c) Des données non structurées, semi-structurées ou structurées

24. b) Stocker des données sous forme de documents JSON

25. b) Un fragment de données réparti sur plusieurs serveurs pour améliorer la performance

26. b) Vérifier les composants individuels du code

27. d) Les deux a et b (unittest et pytest)

28. b) Un test pour s'assurer que les anciennes fonctionnalités ne sont pas affectées par les modifications récentes

29. c) Exécuter le code d'application sur un serveur distant

30. c) Un ensemble de règles permettant de construire des services web interopérables via HTTP

31. a) Transférer des fichiers entre un serveur et un client