Jour 1 - Fondamentaux et introduction à la POO

Niveau 0 - Bas

1. **Variables et types de données** :
    - Déclarer trois variables (texte, entier, booléen) et afficher leurs valeurs.
    - Créer une boucle qui affiche les nombres pairs entre 1 et 20.
2. **Conditions et fonctions** :
    - Créer une fonction qui retourne "Mineur" ou "Majeur" selon l'âge saisi.
    - Implémenter un formulaire HTML pour saisir un prénom et afficher un message de bienvenue.
3. **Gestion des fichiers** :
    - Créer un script pour enregistrer des informations utilisateur (nom, âge, email) dans un fichier texte.
    - Lire le fichier pour afficher les données sous forme de tableau HTML dynamique.
4. **Système de commentaires** :
    - Implémenter une page avec un formulaire pour saisir des commentaires.
    - Stocker les commentaires dans un fichier JSON et les lire pour les afficher en liste sur la même page.
5. **Exercice supplémentaire - Gestion des sessions** :
    - Créer une page avec un système de session pour conserver les informations de connexion d'un utilisateur (nom et email).
    - Ajouter un bouton "Déconnexion" pour détruire la session.

Niveau 1 - Moyen

1. **Approfondissement des tableaux** :
    - Créer un tableau associatif de produits (nom, prix, stock) et calculer la valeur totale des stocks.
    - Manipuler des chaînes : écrire une fonction qui vérifie si un mot est un palindrome.
2. **Introduction à la POO** :
    - Créer une classe `Utilisateur` avec des propriétés (nom, email) et une méthode pour les afficher.
    - Mettre en place un système de connexion simplifié utilisant des sessions.
3. **Gestion avancée des fichiers** :
    - Implémenter un script pour organiser des données dans un fichier CSV et les visualiser sous forme de tableau dynamique avec des options de tri.
4. **Exercice supplémentaire - Gestion des erreurs** :
    - Ajouter des blocs `try-catch` pour gérer les erreurs lors de la manipulation de fichiers ou de l'accès à des données non définies.

Niveau 2 - Bon

1. **POO avancée** :
    - Créer une classe `Produit` avec des propriétés et méthodes pour gérer le stock (ajouter/retirer des quantités).
    - Implémenter une classe `Commande` qui utilise `Produit` pour calculer un total avec TVA incluse.
2. **Introduction au Design Pattern Singleton** :
    - Implémenter un design pattern "Singleton" pour créer une classe `Configuration` permettant de charger les paramètres d'une application (par exemple, nom de l'application, version, URL de base).
    - Ajouter une méthode pour lire les paramètres depuis un fichier JSON et les rendre accessibles dans toute l'application via une instance unique.
3. **Exercice supplémentaire - Classe Router (Design pattern)** :
    - Implémenter une classe `Router` pour gérer les routes de l'application.
    - La classe doit permettre d'enregistrer des routes avec des callbacks associées, et exécuter les callbacks en fonction des URL demandées.
    - **Étendre l'exercice** : Ajouter une méthode qui vérifie si une route existe avant de l'exécuter, et définir une route par défaut (exemple : afficher une erreur 404 si la route demandée n'est pas trouvée).

Niveau 3 - Très Bon

1. **Architecture MVC avancée** :
    - Créer une application simple en MVC pour gérer des utilisateurs avec un modèle MySQL.
    - Ajouter des contrôleurs pour afficher, ajouter et supprimer des utilisateurs.
2. **Design patterns** :
    - Implémenter un système de routage simple (design pattern "Router") pour gérer les différentes pages de l'application.
    - Ajouter un design pattern "Singleton" pour gérer une connexion unique à la base de données.