La Plateforme

Jour 3 : Le passage par référence

La programmation c'est class

Job 1

Créer une classe **Ville** avec comme attributs **privés** un nom et un nombre d'habitants.
Créer une classe **Personne** avec les attributs privés suivants : nom, âge et un objet de la classe ville.
Ajouter la méthode **ajouterPopulation** dans la classe **Personne** qui permet d'augmenter de 1 le nombre d'habitants de la ville.

Créer un objet **Ville** avec comme arguments "Paris" et 1000000.
Afficher en console le nombre d'habitants de la ville de Paris.

Créer un autre objet **Ville** avec comme arguments "Marseille" et 861635.
Afficher en console le nombre d'habitants de la ville de Marseille.

Créer les objets suivants :

- John, 45 ans, habitant à Paris
- Myrtille, 4 ans, habitant à Paris.
- Chloé, 18 ans, habitant à Marseille.

Afficher le nombre d'habitants de Paris et de Marseille après l'arrivée de ces nouvelles personnes.

Résultat attendu :

```
Population de la ville de Paris: 1000000 habitants
Population de la ville de Marseille: 861635 habitants
Mise a jour de la population de la ville de Paris 1000002 habitants
Mise a jour de la population de la ville de Marseille 861636 habitants
```

Job 2

Créer une classe **CompteBancaire** avec les attributs **privés**, numéro de compte, nom, prénom et solde. Cette classe doit posséder les méthodes suivantes :

- ➔ **afficher :** qui affiche le détail sur le compte.
- ➔ **afficherSolde :** cette méthode affiche dans le terminal le solde du client.
- ➔ **versement :** cette méthode prend un paramètre le montant du versement et ajoute celui-ci au solde du client.
- ➔ **retrait** : cette méthode prend un entier en argument (le montant à retirer) ,enlève ce montant au solde du compte et affiche le nouveau solde.
  Veillez à ce que le compte possède bien le montant disponible sinon un message d'erreur est affiché.

Créez un compte avec les valeurs de construction de votre choix et faites appel aux différentes méthodes afin de vérifier que tout fonctionne correctement.

Ajoutez l'attribut decouvert à votre classe **CompteBancaire**, cet attribut aura pour valeur un booléen. Si le client a le droit à un découvert, la valeur de cet attribut sera True et des opérations pourront être effectuées même si le solde est de zéro (méthode retrait).

Ajouter les méthodes suivantes :

- ➔ **agios :** cette méthode permet d'appliquer des agios au solde du compte si celui-ci est négatif.
- ➔ **virement** : cette méthode prend en paramètre une référence, un compte bancaire (celui qui reçoit l'argent) et un montant. Un message de confirmation ou d'erreur doit être affiché.

Créez une deuxième instance de la classe **CompteBancaire**. Ce deuxième compte doit être à découvert (solde négatif). Faire un versement du premier compte vers celui à découvert afin de le remettre à zéro.

Job 3

Dans cet exercice, vous allez créer votre to do list.

Créer une classe **Tache** qui représente une tâche à faire. Cette classe a comme attribut un titre, une description et un statut (à faire ou terminer) initialisés dans le constructeur.

Créer une classe **ListeDeTaches** qui représente la liste des tâches à faire ainsi que toutes les méthodes nécessaires à la gestion de celles-ci avec comme attribut **taches**(liste).

Ajouter les méthodes suivantes :

- ➔ **ajouterTache** : qui permet d'ajouter une tâche.
- ➔ **supprimerTache** : qui permet de supprimer une tâche.

- ➔ **marquerCommeFinie** : qui permet de signaler que la tâche est faite.
- ➔ **afficherListe** : qui permet de retourner une liste de toutes les tâches.
- ➔ **filterListe** : qui permet de filtrer les tâches par rapport à un statut et retourne cette liste.

Tester votre code en créant plusieurs instances de **Tache**, les ajouter à la classe **listeDeTache**, supprimer une tache, changer le statut d'une tâche, afficher toutes les tâches et afficher les tâches à faire.

Job 4

Créer une classe pour représenter un joueur ainsi qu'une classe pour représenter une équipe de foot.

La classe **Joueur** doit avoir les attributs suivants : nom, numéro, position, nombre de buts marqués, passes décisives effectuées, cartons jaunes reçus et cartons rouges reçus. Tous ces attributs doivent être initialisés lors de la création de l'objet **Joueur**.

Cette classe doit posséder les méthodes suivantes :

- ➔ **marquerUnBut**,
- ➔ **effectuerUnePasseDecisive**,
- ➔ **recevoirUnCartonJaune**,
- ➔ **recevoirUnCartonRouge**,
- ➔ **afficherStatistiques**.

Ces méthodes permettent de mettre à jour les statistiques du joueur.

La classe **Equipe** doit avoir les attributs nom et liste de joueurs. Le nom de l'équipe et la liste de joueurs (liste vide par défaut) doivent être initialisés dans le constructeur.

Ajouter les méthodes suivantes dans la classe **Equipe**:

- **ajouterJoueur** : cette méthode ajoute un joueur à l'équipe.
- **AfficherStatistiquesJoueurs** : cette méthode permet d'afficher toutes les statistiques de l'ensemble des joueurs.
- **mettreAJourStatistiquesJoueur** : cette méthode permet de mettre à jour les statistiques d'un joueur (buts, cartons ...).

Créez plusieurs joueurs avec les paramètres de votre choix et ajoutez-les aux équipes.

Présenter l'ensemble des joueurs de chaque équipe. Utiliser les différentes méthodes afin de simuler un match, marquer un but, avoir un carton rouge… Et afficher à nouveau les statistiques des joueurs.

Job 5

Créez un jeu de combat en utilisant la POO.

À tour de rôle, votre personnage et l'ennemi attaquent. Le but étant de vaincre l'ennemi (vie à zéro).

Votre programme doit contenir au minimum deux classes, **Personnage** et **Jeu**.

Commencer par créer une classe nommée **Personnage** prenant des paramètres de construction : nom (string) et vie(int).

Créez au minimum une méthode **attaquer** qui enlève des points à son adversaire.

Ensuite créer la classe **Jeu** ne prenant pas de paramètre. Créer une méthode **choisirNiveau** qui permet de demander au joueur le niveau de difficulté. Celui-ci sera stocké dans l'attribut niveau.

En fonction du niveau choisi, le nombre de points de vie du joueur ainsi que de l'ennemi seront différents.
Créer **lancerJeu**, méthode qui utilise l'attribut niveau. Cette méthode aura pour but d'instancier deux objets **Personnage**, un qui représente le joueur et l'autre l'ennemi avec un nombre de points défini en fonction du niveau.

Implémenter le déroulement d'une partie en demandant au joueur le niveau de difficulté et pensez à ajouter une méthode qui vérifie la santé de vos personnages ainsi qu'une méthode permettant de vérifier qui a gagné.

Sur vos scripts doit apparaître l'ensemble des méthodes appelées tout au long des exercices.

Rendu

Le projet est à rendre sur **https://github.com/prenom-nom/runtrack-python-poo**. Pour chaque jour, créer un dossier nommé "**jourXX**" et pour chaque job, créer un fichier "**jobXX**" ou **XX** est le numéro du job.
N'oubliez pas d'envoyer vos modifications dès qu'une étape est avancée ou terminée et utilisez des commentaires explicites.

Compétences visées

- Maîtriser l'architecture POO en Python
- Maîtriser le passage en référence

Base de connaissances

- Les classes - Documentation officielle
- Apprenez la programmation orientée objet avec Python
- Tutoriel Class python
- Class python
- Passage de référence