Jour 4 : L'héritage

La programmation, c'est Class

C'est juste une histoire de famille...

L'héritage permet de construire une classe B à partir d'une classe A. Il permet de définir un lien de parenté entre deux classes. Ces deux classes partagent donc de nombreuses choses, leurs attributs, méthodes, etc.

Job 1

Créer une classe **Personne** qui aura comme attribut age prenant un entier et initialisé à 14 par défaut. Ajouter une méthode **afficherAge** qui affichera en console l'âge de la personne et une méthode **bonjour** qui écrit en console '**Hello**'. Créer une méthode **modifierAge** prenant en paramètre un entier et permettant de modifier l'âge de la personne.

Créer une classe **Eleve** qui hérite de la classe **Personne** qui n'a pas d'attribut et une méthode publique **allerEnCours** qui affiche : "**Je vais en cours**" ainsi qu'une méthode **afficherAge** qui écrit en console : "**J'ai XX ans**".

Créer une classe **Professeur** avec un attribut privé **matiereEnseignee**, qui indiquera la matière du professeur, et une méthode publique **enseigner** qui affiche : "**Le cours va commencer**".

Instancier une classe **Personne** et une classe **Eleve**. Afficher l'âge par défaut de l'élève en console.

Job 2

À l'aide de la classe **Personne**, **Eleve** et **Professeur** créent au-dessus, faites dire bonjour à votre élève grâce à la méthode **bonjour** ainsi que "**Je vais en cours**" grâce à la méthode **allerEnCours**. Redéfinir l'âge de l'élève à **15 ans**.

Créez un objet **Professeur**, 40 ans, faites dire bonjour à votre professeur et commencez le cours grâce à la méthode enseigner.

Job 3

Créer une classe **Rectangle** avec comme attribut **privé** longueur et largeur. Créer la méthode **perimètre** permettant de calculer le périmètre du rectangle ainsi que la méthode **surface** permettant de calculer la surface du rectangle.

Créer les **assesseurs** et **mutateurs** permettant de manipuler les attributs de la classe.

Créer une classe **Parallelepipede** héritant de la classe **Rectangle** avec en plus un attribut hauteur et une autre méthode **volume**, permettant de calculer le volume du parallélépipède.

Instancier la classe Rectangle et assurez-vous que toutes les méthodes fonctionnent.

Job 4

Créer une classe nommée **Forme** possédant une méthode nommée **aire** qui renvoie 0.

Créer une classe **Rectangle** qui hérite de la classe **Forme** et qui possède deux attributs largeur et hauteur. Surcharger la méthode aire dans la classe Rectangle afin qu'elle ne renvoie plus 0, mais l'aire du rectangle.

Écrire en console le résultat de la méthode aire de la classe Rectangle.

Job 5

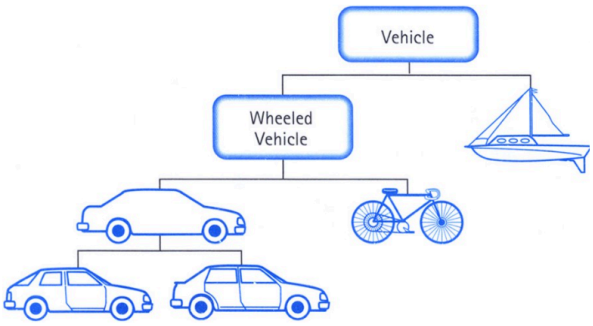
Récupérer votre classe **Forme** crée juste au-dessus.

Créer une classe fille nommée **Cercle** qui hérite de la classe **Forme** et possédant un attribut radius.

Surcharger la méthode **aire** dans la classe Cercle pour qu'elle renvoie l'aire du cercle.

Créez une instance de chaque classe Rectangle et Cercle et utilisez-les pour tester les différentes surcharges de la méthode aire en affichant les résultats en console.

Job 6



Créer une classe **Vehicule** avec comme attribut une marque, le modèle, une année et un prix. Créer la méthode **informationsVehicule** qui écrit en console la marque, le modèle, l'année et le prix du véhicule.

Créer la classe **Voiture** qui hérite de la classe **Vehicule**. Dans le constructeur de la classe Voiture, ajouter un attribut **portes** qui contient le nombre entier **4**. Créer dans cette classe, une méthode **informationsVehicule** qui affiche, en console, les informations générales du véhicule et le nombre de portes de la voiture.

Instancier un objet Voiture, passer les informations dont la classe a besoin et faites appel à la méthode **informationsVehicule**.

Résultat attendu :

```
Marque = Mercedes
Model = Classe A
Année = 2020
Prix = 18500
Nombre de porte = 4
```

Créer une classe **Moto** qui hérite de la classe **Vehicule** et ajouter l'attribut **roue** qui contient par défaut l'entier **2**. Créer à nouveau une méthode **informationsVehicule** dans la classe Moto qui affiche l'intégralité des informations de la moto.

Instancier un objet Moto et faites appel à la méthode **informationsVehicule**.

Résultat attendu :

```
Marque = Yamaha
Model = 1200 Vmax
Année = 1987
Prix = 4500
Nombre de roue = 2
```

Créer la méthode **demarrer** dans la classe **Véhicule** qui écrit en console "**Attention, je roule**". Surcharger la méthode **demarrer** dans la classe Moto et Voiture afin d'afficher un message personnalisé. Faites démarrer votre voiture et votre moto.

Job 7

Développer votre version du célèbre jeu Blackjack. Le but est de faire le plus de points sans dépasser 21. Chaque carte représente une valeur :

- de 2 à 10 : ces cartes ont pour valeur sa valeur nominale
- une figure a pour valeur 10 points
- un as 1 ou 11 points au choix

Le jeu commence avec les joueurs qui reçoivent chacun 2 cartes. Ensuite, le joueur peut choisir de "prendre" (recevoir) une ou plusieurs cartes supplémentaires pour tenter d'améliorer sa main, ou de "passer" et laisser le tour au croupier. Le croupier prend des cartes jusqu'à ce qu'il ait au moins 17 points, puis s'arrête. Si la main du joueur dépasse 21, il perd immédiatement. Si le total de la main du joueur est supérieur à celui du croupier, le joueur gagne. Sinon, c'est le croupier qui gagne.

Créer au minimum deux classes **Carte** et **Jeu**.

La classe **Carte** aura au minimum un attribut valeur et couleur. La classe **Jeu** quant à elle devra gérer l'ensemble des cartes. Les cartes du jeu seront stockées dans un attribut paquet représenté par une liste et contenant 52 cartes.

Créer toutes les méthodes nécessaires pour jouer une partie.

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 l’héritage

Base de connaissances

- Les classes - Documentation officielle
- Apprenez la programmation orientée objet avec Python
- Tutoriel Class python
- Class python
- L'héritage