

Jour 1 : Les Classes

La programmation c'est class

Simplifiez votre code, organisez vos idées, structurez vos programmes

Jusqu'à aujourd'hui, vous avez développé vos programmes au moyen de la programmation procédurale. La résolution de vos problèmes se faisait par un ensemble d'instructions qui se découpait en plusieurs actions afin de résoudre votre problème. Cette méthode est intuitive lorsque l'on apprend à développer, mais vous avez dû vous rendre compte que cette méthode pose un certain nombre d'inconvénients, votre code est long, fastidieux et difficile à maintenir.

La POO (programmation orientée objet) ne permet pas de faire plus de choses, mais elle permet toutefois d'organiser son code de façon claire et structurée, elle facilite le travail coopératif et rend son code maintenable et réutilisable.



Job 1

Créer une classe **Operation** avec un constructeur (méthode qui sera appelée lors de l'instance de la classe). Ajouter les attributs "**nombre1**" et "**nombre2**" initialisés avec des valeurs par défaut. Instancier votre première classe et imprimer l'objet en console.

Résultat attendu :

```
<__main__.Operation object at 0x0000002541F869CF0>
```

Job 2

À l'aide de la classe **Operation** créée au-dessus, imprimer en console la valeur des attributs "**nombre1**" et "**nombre2**".

Résultat attendu :

```
Le nombre1 est 12  
Le nombre2 est 3
```

Job 3

Modifier votre classe **Operation** afin d'y ajouter la méthode **addition()**. Cette méthode additionne "**nombre1**" et "**nombre2**" et affiche en console le résultat.

Job 4

Créez une classe **Personne** avec les attributs **nom** et **prenom**. Ajoutez une méthode **SePresenter** qui retourne le nom et le prénom de la personne. Ajoutez aussi un constructeur prenant en paramètres de quoi donner des valeurs initiales aux attributs **nom** et **prenom**. Instanciez plusieurs personnes avec les valeurs de construction de votre choix et faites appel à la méthode **SePresenter** afin de vérifier que tout fonctionne correctement.

Résultat attendu :

```
Je suis John Doe  
Je suis Jean Dupond
```

Job 5

Créer une classe nommée **Point** avec les attributs **x** et **y** correspondant aux coordonnées horizontales et verticales du point. Les deux attributs doivent être initialisés dans le constructeur.

La classe **Point** doit posséder les méthodes suivantes :

- **afficherLesPoints** qui affiche les coordonnées des points.
- **afficherX** et **afficherY** qui affiche respectivement **x** et **y**.
- **changerX** et **changerY** qui change la valeur des attributs **x** et **y**.

Job 6

Créez une classe **Animal** avec un attribut `age` initialisé à zéro et `prenom` initialisé vide dans son constructeur.

Instancier un objet `Animal` et écrire en console l'attribut `âge`. Créer une méthode **vieillir** qui ajoute 1 à l'âge de l'animal. Faire vieillir votre animal et afficher son âge en console.

Résultat attendu :

```
L'age de l'animal 0 ans
# Age de l'animal apres appel de la methode vieillir
L'age de l'animal 1 ans
```

Créer une méthode **nommer** qui prend en paramètre le nom de l'animal. Nommer votre animal et afficher en console le nom de l'animal.

Résultat attendu :

```
L'animal se nomme Luna
```

Job 7

Créer une classe **Personnage** représentant un personnage de jeu. Le plateau de jeu est représenté par une matrice. Le joueur est défini par ses attributs `x` et `y`, représentant les index du tableau.

Écrire le constructeur de cette classe en initialisant la position (`x` et `y`).

Écrire les méthodes : **gauche**, **droite**, **bas** et **haut** permettant respectivement de faire avancer à gauche et à droite, en haut ou en bas.

Implémenter une méthode "**position**" renvoyant les coordonnées sous forme d'un tuple.

Job 8

Créer la classe "**Cercle**" recevant en argument son rayon initialisé dans le constructeur.

Ajouter les méthodes suivantes :

- **changerRayon** qui permet de modifier le rayon.
- **afficherInfos** qui permet d'afficher toutes les informations du cercle.
- **circonférence** qui permet de retourner la circonférence.
- **aire** qui permet de retourner l'aire du cercle.
- **diametre** qui permet de retourner le diamètre du cercle.

Créer deux cercles avec comme rayon 4 et 7. Pour chaque cercle, afficher ses informations, afficher sa circonférence, son diamètre et son aire.

Job 9

Créer la classe **Produit** avec des attributs **nom**, **prixHT**, **TVA**. Implémenter la méthode **CalculerPrixTTC** qui retourne le prix produit avec la TVA. Ajouter la méthode **afficher** qui liste l'ensemble des informations du produit.

Créer plusieurs produits et calculer leurs TVA.

Ajouter des méthodes permettant de modifier le nom du produit et son prix. Ainsi que des méthodes permettant de retourner chaque information du produit.

Modifier le nom et le prix de chaque produit et afficher en console le nouveau prix des produits.

La fonction print() ne doit pas être utilisée dans la classe Produit.

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
-

Base de connaissances

- [Les classes - Documentation officielle](#)
- [Apprenez la programmation orientée objet avec Python](#)
- [Tutoriel Class python](#)
- [Class python](#)
- [Passage de référence](#)
- [L'héritage](#)