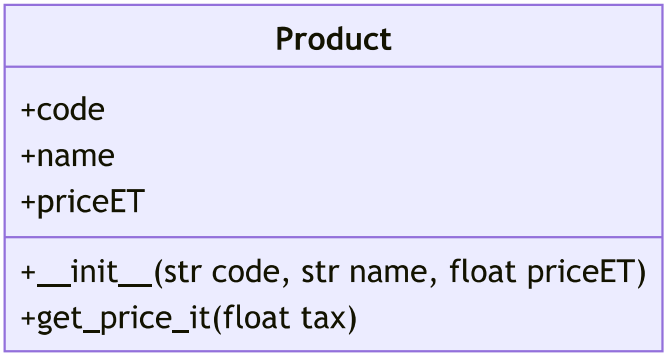
TP 4 - Introduction à la POO

Petit rappel théorique ou apprentissage complet, regardez ce cours : OpenClassrooms

Les concepts étant parfoit compliquer à comprendre même si le code fonctionne, ne pas hésiter à soliciter l'enseignant pour des précisisons.

Exo 01

Dans un fichier Product.py, créez une classe Product selon ce diagramme de classe.



La fonction get_price_it doit retourner le prix incluant la taxe.

Une fois cela réalisez, créez un programme qui créé plusieurs products et affiche leur prix TTC sous cette forme

```
CodeProduit - NomProduit - PrixTTC
```

Modifiez ensuite votre programme pour que la taxe ne soit plus un paramètre de get_price_it mais un attribut de classe initialisé à 0.2.

Exo 02

Via ce lien, vous pourrez retrouver toute les méthodes à surcharger : Surcharge d'opérateur

Créez un fichier nommé Fraction.py puis créez une classe nommé Fraction. Celle-ci présentera 2 attributs dans son constructeur. Un numérateur et un dénominateur.

Comme le type Fraction n'est pas nativement géré par les opération standards, c'est à nous de redéfinir comment les gérer.

Dans votre classe redéfinissez les opérateurs suivant :

- la fonction **str**
- +,-,*,/
- >,<,<=, >=, ==, !=

Puis faites des tests dans un programme principal.

Exo 03

Imaginons un jeu de carte traditionnel. Une carte est représenté par une valeur (5,10,Q,K) et une couleur (♠,♣,♦,♥). Chaque information peut être considéré comme une information complexe et donx un objet.

Première étape

Créer les classes CardValue et CardColor.

La classe CardValue aura 2 attributs :

- value_txt : La valeur affiché (5,10,K)
- value_pts : La valeur numérique (5,11,13,...)

La CardColor aura ces attributs :

- shade : La forme de la couleur (♠,♣,♦,♥)
- shade_name : Le nom de la forme
- foreground_color : La couleur du texte
- background_color : La couleur du fond pour l'afficher

Nous pouvons maintenant créer une classe Card. L'idée est que l'on puisse aussi comparer les cartes donc pensez à la surcharge d'opérateur.

Vu que l'égalité ne se compare que sur la valeur (car un 4 de trèfle n'est pas un 4 de carreau mais leurs valeurs sont égales), je vous invite à créer une methode is_equal_value(card: Card). Elle renvera vrai si les 2 cartes ont la même valeur.

N'oubliez de redéfinir **str** et **repr** pour pouvoir afficher une carte dans le terminal. Voir ici pour gérer les couleurs

Deuxième étape

Maintenant que nous avons modélisé une carte, il ne manque plus qu'a modéliser le deck de carte.

Un deck sera représenté par 2 listes, le deck en lui même et la défausse

Vous aurez les méthodes suivantes :

- init52_cards() : Remplie le deck avec les 52 cartes usuelles
- Shuffle() : Melange le deck
- Draw() : Retire la première carte du deck et la retourne
- Discard(card) : Met la carte dans la défausse

Vous pourrez ajouter des méthodes si nécessaire. (Comme **str**)

Troisème étape

Recoder le jeu de duel présenté dans le DLC du TP1 à l'aide des classes.