Création de classe

Christophe Viroulaud

Terminale - NSI

Lang 05

Activité 1 : Télécharger et décompresser l'annexe.
Ouvrir les deux fichiers Python :

1. **boutique.py** : le programme principal,
2. **classes.py** : le fichier qui contiendra les classes, pour l'instant vide.

Sommaire

Modélisation

À retenir

La modélisation consiste à identifier tous les attributs et méthodes de chaque objet répondant à la problématique.

Utilisateur

- **Attributs :**
  - identifiant : str
  - mot_de_passe : str
  - panier : list
- **Méthodes :**
  - verifier_mdp(mdp_saisi : str) -> bool
  - get_identifiant() -> str
  - set_identifiant(nouveau : str) -> None
  - get_mdp() -> str
  - set_mdp(nouveau : str) -> bool
  - acheter(article : Article) -> None
  - calculer_prix() -> float
  - calculer_remise() -> float

Article

- **Attributs :**
  - nom : str
  - description : str
  - prix : list
- **Méthodes :**
  - get_nom() -> str
  - get_prix() -> str
  - get_description() -> str
  - set_nom(nouveau : str) -> None
  - set_description(nouveau : str) -> None
  - set_prix(nouveau : str) -> None

Sommaire

Créer une classe

À retenir

Une **classe** regroupe les attributs et méthodes d'un objet.

```
1 class Utilisateur
```

Code 1 – Le mot-clé `class`

Remarque

Les bonnes pratiques Python (PEP 8) recommande de mettre une majuscule pour les noms des classes.

Sommaire

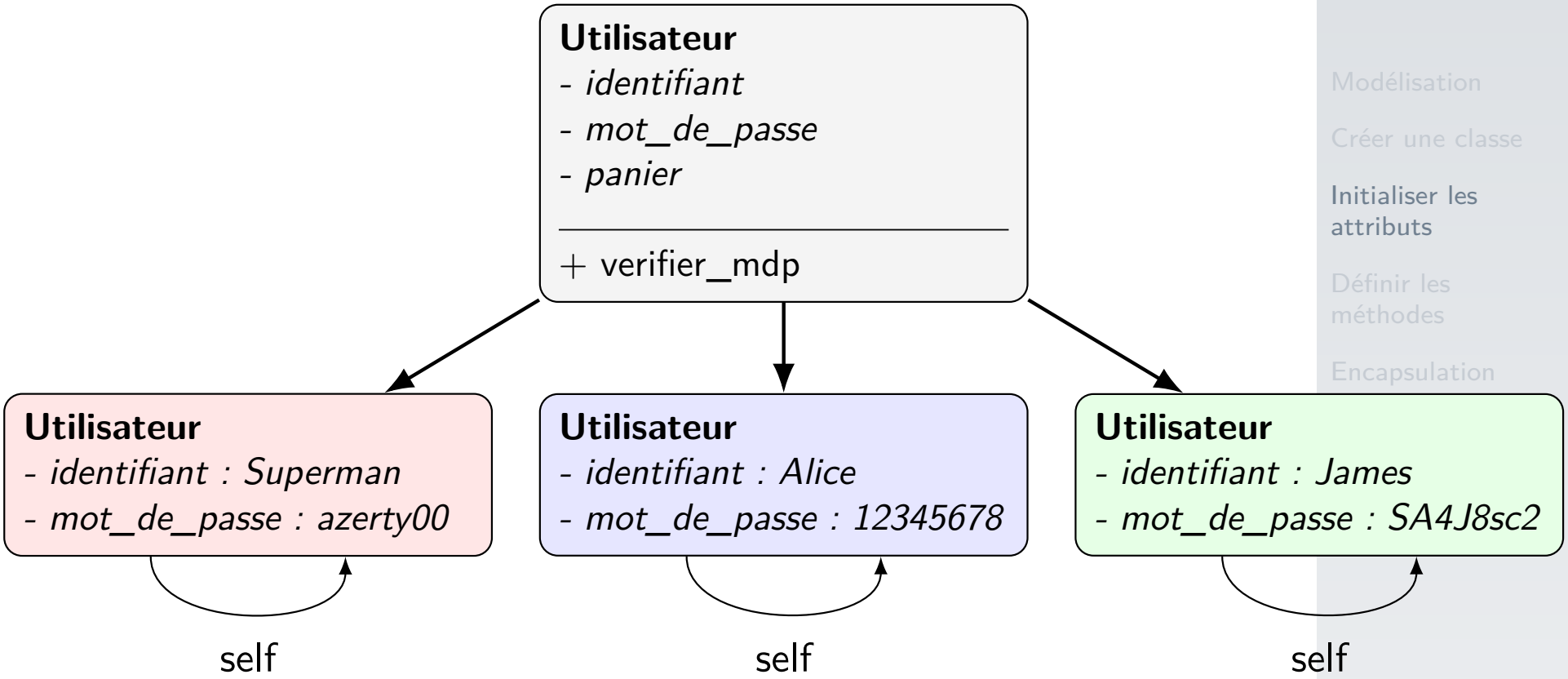
Initialiser les attributs

À retenir

- Les attributs sont initialisés dans la *méthode particulière* **`__init__`**.
- Le mot-clé **`self`** identifie l'instance de l'objet.
- **`self`** est le premier paramètre de chaque méthode.

```
1  class Utilisateur:
2      def __init__(self, ident: str, mdp: str):
3          self.identifiant = ident
4          self.mot_de_passe = mdp
5          self.panier = []
```

Code 2 – Les attributs sont initialisés dans le **constructeur**.



À retenir

`self` est une référence interne à l'instance de l'objet : elle est accessible uniquement à l'intérieur de l'objet.

Sommaire

Définir les méthodes

À retenir

- Une **méthode** est une fonction interne à la classe.
- Rappel : En Python, le premier paramètre d'une méthode est **toujours `self`**.

```
1  class Utilisateur:
2      def __init__(self, ident: str, mdp: str):
3          self.identifiant = ident
4          self.mot_de_passe = mdp
5          self.panier = []
6
7      def verifier_mdp(self, mdp_saisi: str) -> bool:
8          return self.mot_de_passe == mdp_saisi
```

Sommaire

À retenir

Pour manipuler les données dans un objet, il est d'usage de créer certaines méthodes :

- **les accesseurs :** qui renvoient la valeur d'un attribut,
- **les mutateurs :** qui permettent de modifier la valeur d'un attribut.

On n'accède pas aux attributs directement : c'est **l'encapsulation**

```
1 def get_identifiant(self) -> str:
2     return self.identifiant
3
4 def set_identifiant(self, nouveau: str) -> None:
5     self.identifiant = nouveau
```

Activité 2 : Créer l'accesseur et le mutateur de de l'attribut `mot_de_passe`. Cette dernière méthode modifiera le mot de passe seulement si la taille du nouveau est plus grand que 8 et renverra un booléen.

```python
class Utilisateur:
    ...

    def get_identifiant(self) -> str:
        return self.identifiant

    def set_identifiant(self, nouveau: str) -> None:
        self.identifiant = nouveau

    def get_mdp(self) -> str:
        return self.mot_de_passe

    def set_mdp(self, nouveau: str) -> bool:
        if len(nouveau) >= 8:
            self.mot_de_passe = nouveau
            return True
        return False
```

Activité 3 : Créer la classe `Article`, son constructeur, ses accesseurs et mutateurs.
Rappel : La classe possède trois attributs (nom, description, prix).

```python
class Article:
    def __init__(self, nom: str, desc: str, prix: float):
        self.nom = nom
        self.description = desc
        self.prix = prix
```

Code 3 – Constructeur

```python
    def get_nom(self) -> str:
        return self.nom

    def get_description(self) -> str:
        return self.description

    def get_prix(self) -> float:
        return self.prix

    def set_nom(self, nouv: str) -> None:
        self.nom = nouv

    def set_description(self, nouv: str) -> None:
        self.description = nouv

    def set_prix(self, nouv: float) -> None:
        self.prix = nouv
```

Code 4 – Accesseurs et mutateurs

Sommaire

Méthodes supplémentaires

Activité 4 : Écrire les méthodes de la classe `Utilisateur` :

- `acheter(self, article: Article) -> None` : ajouter un article au panier
- `calculer_prix(self) -> float` : calculer le prix total à payer
- `calculer_remise(self) -> float` : calculer la remise de 25% si plus de 3 articles achetés

```
1     def acheter(self, article: Article) -> None:
2         self.panier.append(article)
```

```
1     def calculer_prix(self) -> float:
2         prix = 0
3         for article in self.panier:
4             prix = prix + article.get_prix()
5         return prix
```

```
1     def calculer_remise(self) -> float:
2         prix = self.calculer_prix()
3         remise = 0
4         if len(self.panier) >= 3:
5             remise = prix * 0.25
6         return round(remise, 2)
```