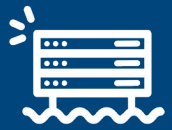


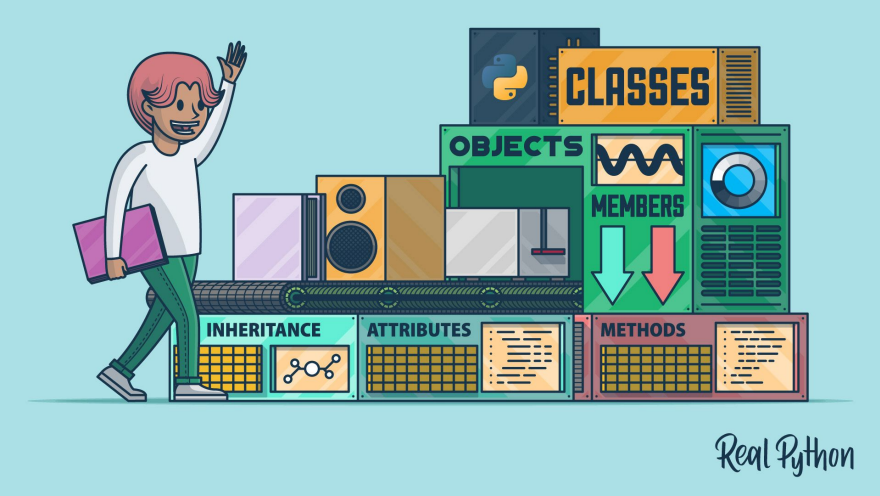
La Plateforme / Python : POO

Sommaire

Qu'est-ce que la POO ?

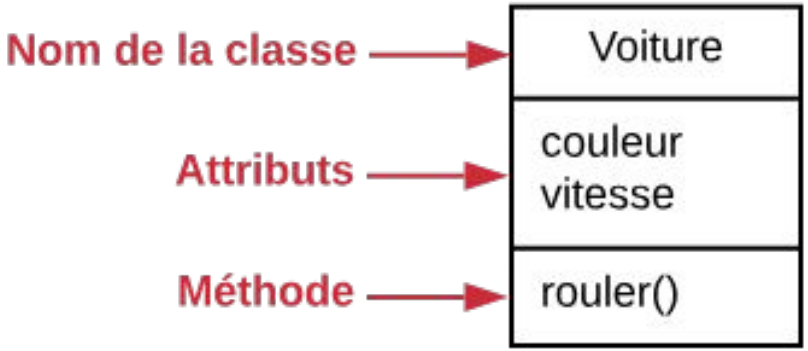
La programmation orientée objet est un modèle de programmation qui repose sur le concept de classes et d'objets.

Quel est le rôle de la POO?



Elle est utilisée pour structurer un programme logiciel en éléments de code simples et réutilisables, appelés classes, qui sont utilisées pour créer des instances d'objets

Qu'est-ce qu'une classe ?



Une classe est un bloc de code qui définit un type d'objet.

Chaque type d'objet a des caractéristiques et des comportements déterminés : des variables et des fonctions.

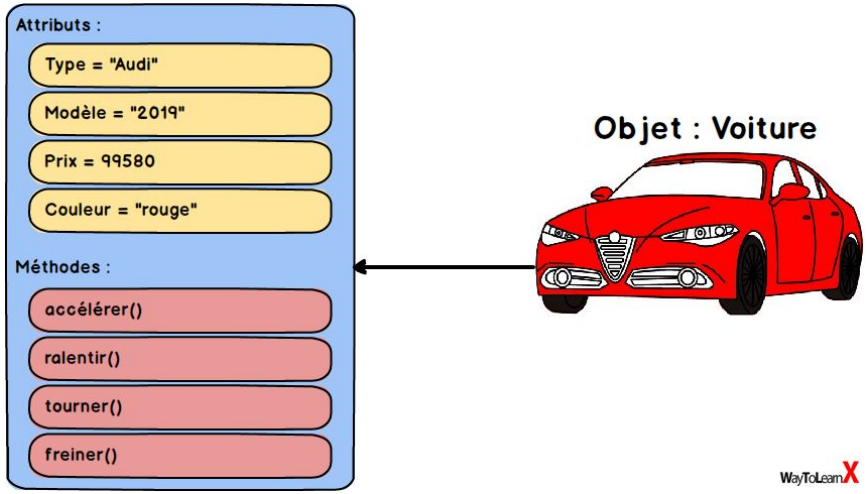
Tous les objets d'une classe disposent de ces variables et de ces fonctions, ce sont ses attributs.

Instance de classe

Un objet avec un comportement et un état, tous deux définis par la classe.

Il s'agit donc d'un objet constituant un exemplaire de la classe.

Objet



on va appeler “objet” un bloc cohérent de code qui possède :

- ses propres variables (qui sont l’ équivalent des caractéristiques des objets de tous les jours)
- fonctions (qui sont nos actions).

Comme les objets de la vie courante, les objets informatiques peuvent être très simples ou très complexes.

Les attributs

Les attributs correspondent aux caractéristiques de l'objet (donc les variables décrivant l'état de l'objet)

Les méthodes

Les méthodes correspondent aux actions que peut faire cet objet (les fonctions applicables à l'objet).

Les niveaux de visibilité

Niveau public	Niveau privé	Niveau protégé
Accessible depuis n'importe quelle instance (ou objet) de la classe ou d'une classe fille.	Accessible depuis l'intérieur de la classe ;	Accessible depuis l'intérieur de la classe ou depuis une classe fille ;

La Plateforme

Exemple

```python
class Utilisateur:



    def __init__(self, nom, age):
        self.user_name = nom
        self.user_age = age

    def getNom(self):
        print("Salut, je suis ", self.user_name,
              "et j'ai ", self.user_age, " ans")


pierre = Utilisateur("Pierre", 29)
pierre.getNom()

. . .



mathilde = Utilisateur("Mathilde", 28)
mathilde.getNom()
```

- mot-clé class

```python
class Utilisateur:

    def __init__(self, nom, age):
        self.user_name = nom
        self.user_age = age
```

- Constructeur
- méthode appelée lorsque l'objet est créé (instanciation)
- initialise les attributs

```python
class Utilisateur:

    def __init__(self, nom, age):
        self.user_name = nom
        self.user_age = age
```

- désigne l'instance de classe (objet) en elle-même

```python
class Utilisateur:

    def __init__(self, nom, age):
        self.user_name = nom
        self.user_age = age
```

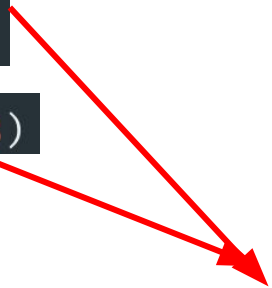
- attribut
- caractéristique de l'objet
- self.<quelquechose>, on définit un attribut des objets de la classe
- tous les objets de cette classe auront ces caractéristiques

```
    def getNom(self):
        print("Salut, je suis ", self.user_name,
              "et j'ai ", self.user_age, " ans")
```

- méthode
- indentée dans la classe
- tous les objets qui sont des instances de cette classe Utilisateur auront la possibilité d'appeler cette méthode

```
pierre = Utilisateur("Pierre", 29)
```

```
mathilde = Utilisateur("Mathilde", 28)
```



- Création d'objets avec les attributs prénom et âge

```python
class Utilisateur:
    niveau = "Prépa"

pierre = Utilisateur()
pierre.niveau = "Accompagnateur Prépa"
print(pierre.niveau)

mathilde = Utilisateur()
print(mathilde.niveau)
```

- Création d'objets

```
pierre.getNom()
```

- Pour appeler une méthode d'instance, on utilise un point et puisqu'il s'agit d'une sorte de fonction (méthode), on utilise des parenthèses.

Questions ?