La Classe Object en Python et ses Méthodes Spéciales

Anis SAIED

November 17, 2023

La Classe Object en Python

La classe `object` est

- La classe de base de toutes les classes en Python.
- Implicitement héritée par toutes les classes définies.
- Au sommet de la hiérarchie de l'héritage des classes.
- Définit des méthodes spéciales et attributs prédéfinis.

Méthodes et attributs prédéfinis de la classe object

Pour connaître les méthodes et attributs prédéfinis de la classe object: `dir(object)`.

```
>>> print(dir(object))
[..., '__class__', '__delattr__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__le__',
'__lt__', '__ne__', '__new__', '__reduce__',
'__reduce_ex__', '__repr__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', ...]
```

Utilisez ces méthodes dans vos classes dérivées pour personnaliser le comportement de vos objets.

Appel des Méthodes Spéciales

Les méthodes spéciales de la classe `object` peuvent être appelées de manière spécifique pour personnaliser le comportement des objets. Voici quelques exemples d'appels :

`__init__(self)` : Appelé lors de la création d'une instance de la classe pour effectuer l'initialisation. Vous pouvez l'appeler en créant une instance de la classe : `obj = MaClasse()`.

`__str__(self)` : Appelé lors de la conversion en chaîne de caractères (par exemple, avec `str()` ou `print()`). Vous pouvez l'appeler en utilisant `str(obj)` ou `print(obj)`.

`__repr__(self)` : Appelé par la fonction `repr()` pour obtenir une représentation "formelle" de l'objet. Vous pouvez l'appeler en utilisant `repr(obj)`.

Rôle et appel des méthodes spéciales

`__class__` : Retourne la classe de l'objet. Vous pouvez l'appeler en utilisant `type(obj)`.

`__dir__` : Retourne la liste des attributs valides de l'objet. Vous pouvez l'appeler en utilisant `dir(obj)`.

`__doc__` : Retourne la documentation de l'objet. Vous pouvez y accéder avec `obj.__doc__`.

`__eq__(self, other)` : Appelé lors de la comparaison d'égalité entre deux objets avec l'opérateur `==`. Vous pouvez l'appeler en utilisant `obj1 == obj2`.

`__ge__(self, other)` : Appelé lors de la comparaison "supérieur ou égal" (`>=`). Vous pouvez l'appeler en utilisant `obj1 >= obj2`.

Méthodes Spéciales pour les Opérations Mathématiques

Voici quelques-unes des méthodes spéciales liées à des opérations mathématiques :

- `__add__(self, other)` : Définie pour l'opérateur d'addition (+).
- `__sub__(self, other)` : Définie pour l'opérateur de soustraction (-).
- `__mul__(self, other)` : Définie pour l'opérateur de multiplication (*).
- `__truediv__(self, other)` : Définie pour l'opérateur de division (/).
- `__floordiv__(self, other)` : Définie pour l'opérateur de division entière (//).
- `__mod__(self, other)` : Définie pour l'opérateur de modulo (%).
- `__pow__(self, other[, modulo])` : Définie pour l'opérateur de puissance (**).

Exemple avec la méthode __add__

```
class Nombre:
    def __init__(self, valeur):
        self.valeur = valeur
    def __add__(self, other):
        if isinstance(other, Nombre):
            return Nombre(self.valeur + other.valeur)
        else:
            raise ValueError("Impossible d'ajouter un objet"
                             "de type différent")
# Utilisation
nombre1 = Nombre(5)
nombre2 = Nombre(10)
# x + y fait appel à type(x).__add__(x, y)
resultat = nombre1 + nombre2
print(resultat.valeur)  # Affiche 15
```

Implication de `__str__` dans `print`

Lorsque vous utilisez `print(x)` et que la classe de l'objet `x` n'implémente pas la méthode spéciale `__str__`, voici ce qui se passe :

```
class Exemple:
    pass

objet = Exemple()
print(objet)
```

Dans cet exemple, la classe `Exemple` n'implémente pas `__str__`, et le message retourné par défaut est de la forme :

```
<__main__.Exemple object at 0x...>
```

Explication

Lorsque vous faites `print(objet)` et que la classe de l'objet ne définit pas `__str__`, Python utilise `__repr__` s'il est défini. Sinon, il utilise une représentation par défaut basée sur l'emplacement mémoire de l'objet.

`<__main__.Exemple object at 0x...>`: Cela indique que l'objet est de la classe `Exemple` définie dans le module principal (`__main__`). La partie at `0x...` indique l'emplacement mémoire de l'objet.

Le message par défaut, tel que `<main.Exemple object at 0x...>`, est généré par la méthode spéciale `__repr__` de la classe parente (`object`) lorsque la classe de l'objet ne définit ni la méthode spéciale `__str__` ni `__repr__`.

Exercice : Implémentation de __str__ et __repr__

Exercice : Implémentez les méthodes spéciales __str__ et __repr__ dans la classe Personne.

```
class Personne:
    def __init__(self, nom, age):
        self.nom = nom
        self.age = age
    def __str__(self):
        pass
    def __repr__(self):
        pass
# Création d'une instance de la classe Personne
personne = Personne("Ali", 30)
print(personne)
p_representation = repr(personne)
print("Représentation formelle :", p_representation)
```

Correction : __str__ et __repr__

```
class Personne:
    def __init__(self, nom, age):
        self.nom = nom
        self.age = age

    def __str__(self):
        return "{}, {} ans".format(self.nom, self.age)

    def __repr__(self):
        return "Personne('{}', {})".format(self.nom, self.age)

# Création d'une instance de la classe Personne
personne = Personne("Ali", 30)

print(personne)# Affichage avec print()
p_representation = repr(personne) #représentation formelle

# Afficher la représentation formelle
```