L'encapsulation

Définition :

Technique pour restreindre l'accès direct aux attributs d'une classe.

Avantages :

• Contrôle de la lecture/écriture.

• Protection contre des modifications non autorisées.

• Respect du principe de modularité et sécurité.

Exemple avec _ (convention protégée)

```python
class Ordinateur:
    def __init__(self, cpu):
        self._cpu = cpu

ordi = Ordinateur("Intel i5")
print(ordi._cpu)  # Possible mais déconseillé
ordi._cpu = "AMD Ryzen 7"  # Modifiable
print(ordi._cpu)
```

Accès direct possible mais **Usage réservé** à la classe ou ses sous-classes.

Exemple avec __ (encapsulation forte)

```python
class Ordinateur:
    def __init__(self, cpu):
        self.__cpu = cpu

ordi = Ordinateur("Intel Xeon")
print(ordi.__cpu)  # Provoque une erreur !
```

Name mangling :

_**Ordinateur__cpu** est le vrai nom de l'attribut.

L'encapsulation classique avec property()

```python
class Ordinateur:
    def __init__(self, cpu):
        self._cpu = cpu

    def _get_cpu(self):     # Accesseurs
        print("Accesseur appelé")
        return self._cpu

    def _set_cpu(self, new_value): # mutateurs
        print("Mutateur appelé")
        self._cpu = new_value

    # Définir une propriété
    cpu = property(_get_cpu, _set_cpu)
```

```python
ordi = Ordinateur("Intel i7")
print(ordi.cpu)  # Appel de l'accesseur
ordi.cpu = "AMD Ryzen 5"  # Appel du mutateur
print(ordi.cpu)  # Appel de l'accesseur
```

_get_cpu pour **la lecture**
_set_cpu pour **l'écriture**.

Encapsulation avec la technique moderne

Depuis les versions modernes de Python, le décorateur **@property** simplifie l'écriture des accesseurs et mutateurs.

```python
class Ordinateur:
    def __init__(self, cpu, ram):
        self._cpu = cpu  # Privé

    @property
    def cpu(self):
        return self._cpu

pc = Ordinateur("Intel", "4 To")
print(pc.cpu)
```

Obligation de déclarer l'accesseurs avant le mutateur

Encapsulation avec la technique moderne

@cpu.setter : Ajoute une méthode pour modifier la valeur de la propriété cpu.

```python
class Ordinateur:
    def __init__(self, cpu, ram):
        self._cpu = cpu

    @property
    def cpu(self):
        return self._cpu

    @cpu.setter
    def cpu(self, newCpu):
        self._cpu = newCpu

pc = Ordinateur("Intel", "4 To")
pc.cpu = "AMD"
print(pc.cpu)
```

Obligation de déclarer
l'accesseurs avant le mutateur