

# Programmation objet

J. Boucher

Lycée Paul Riquet, 1<sup>er</sup> NSI

15 septembre 2024

# Plan

1 I. Introduction

2 II. Classes et attributs

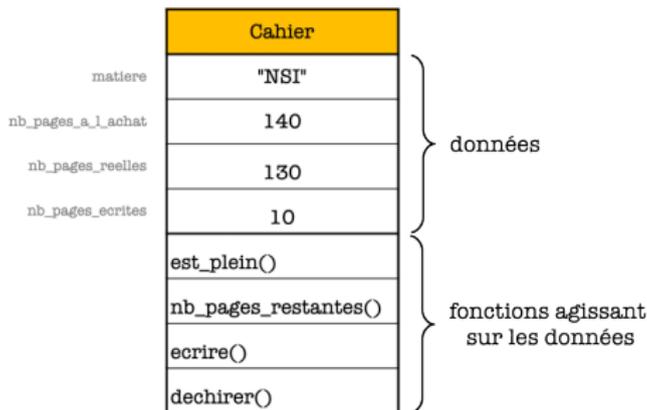
3 III. Méthodes

4 IV. Encapsulation

# I. Introduction

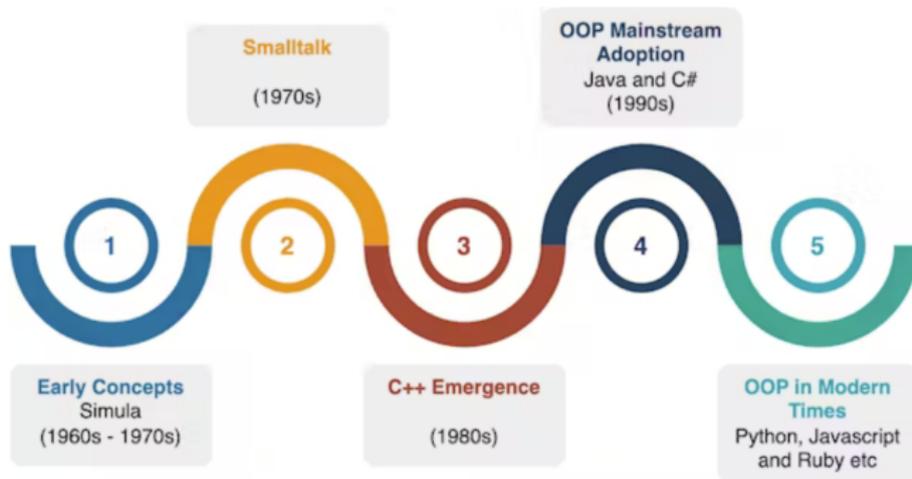
## Le paradigme de la programmation objet

La **programmation orientée objet** ou POO, est un des grands paradigmes<sup>1</sup> de programmation. Elle permet de réunir « données » et « fonctions qui agissent sur ces données », au sein d'une même structure appelée **objet**.



Représentation d'un objet de type Cahier

1. « manière de voir les choses »



2

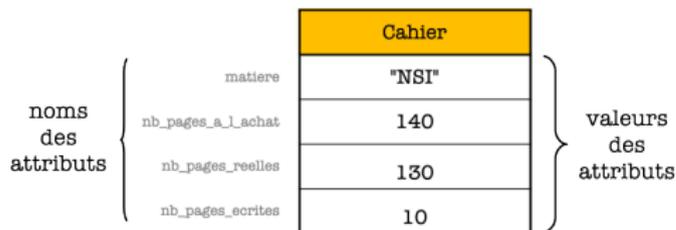
Évolution de la POO depuis les années 60 jusqu'à nos jours.

2. source : <https://aigents.co/data-science-blog/publication/object-oriented-programming-a-primer>

## II. Classes et attributs

### Classes et attributs

- Une **classe** définit et nomme une structure de données, qui vient s'ajouter aux structures de base du langage.
- Elle permet de regrouper plusieurs composantes de natures variées, appelées **attributs** (ou bien *champs* ou *propriétés*) et dotées d'un nom.
- Les valeurs des attributs représentent l'*état* d'un objet *instancié* à partir d'une classe.



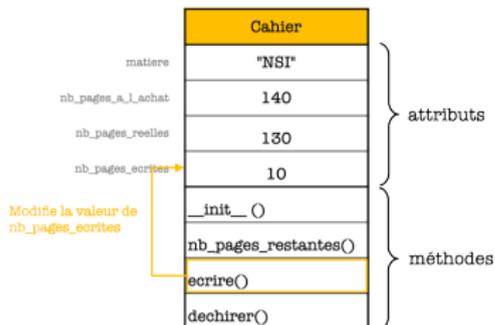
```
class Cahier:
    def __init__(self, matiere, nb_pages):
        self.matiere = matiere
        self.nb_pages_a_l_achat = nb_pages
        self.nb_pages_reelles = nb_pages
        self.nb_pages_ecrites = 0
```

La classe Cahier regroupe quatre attributs de type string et int ; c'est une structure de données dite *composites*.

## III. Méthodes

### Méthodes

- La manipulation des objets s'effectue préférentiellement par un ensemble de fonctions dédiées, faisant partie de la définition de la classe, qui sont appelées **méthodes** de cette classe.
- Pour construire un nouvel objet, on appelle une méthode spéciale chargée d'initialiser la valeurs des attributs, appelée **constructeur**.



```
class Cahier:
    def __init__(self, matiere, nb_pages):
        self.matiere = matiere
        self.nb_pages_a_l_achat = nb_pages
        self.nb_pages_reelles = nb_pages
        self.nb_pages_ecrites = 0

    def ecrire(self, nb_pages_a_ecrire):
        self.matiere = matiere
```

```
mon_cahier_de_nsi = Cahier("NSI", 140)
mon_cahier_de_nsi.ecrire(10)
```

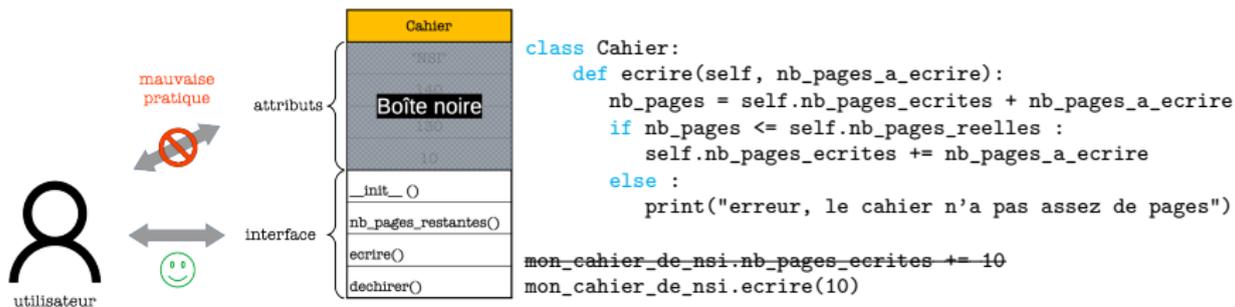
Appel du constructeur de la classe Cahier puis de la méthode ecrire pour modifier la valeur de l'attribut nb\_pages\_ecrites.



## IV. Encapsulation

### Principe d'encapsulation

- L'interaction avec un objet se fait avec un ensemble de méthodes, appelé **interface**.
- Les attributs doivent rester « masqués » à l'utilisateur de l'objet, et relèvent du détail d'implémentation à la charge du concepteur de la classe.



→ L'encapsulation permet de modifier les structures de données internes sans modifier l'interface, mais aussi de rajouter des règles de validation, par exemple pour éviter d'écrire plus de pages que n'en possède le cahier.