

# FORMATION PYTON

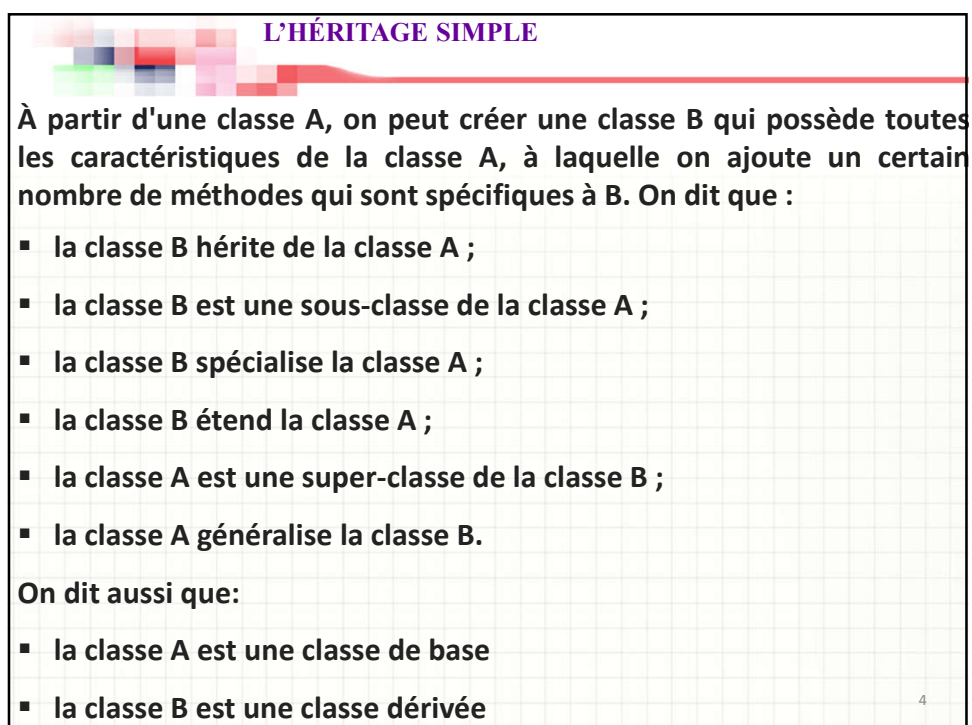
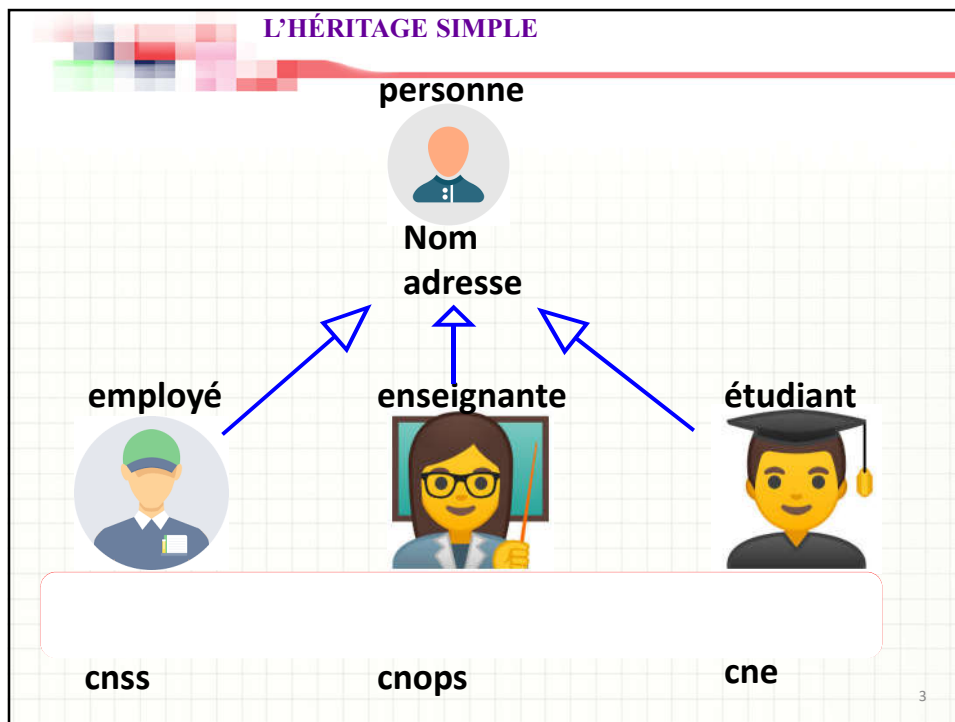
## HÉRITAGE EN PYTHON

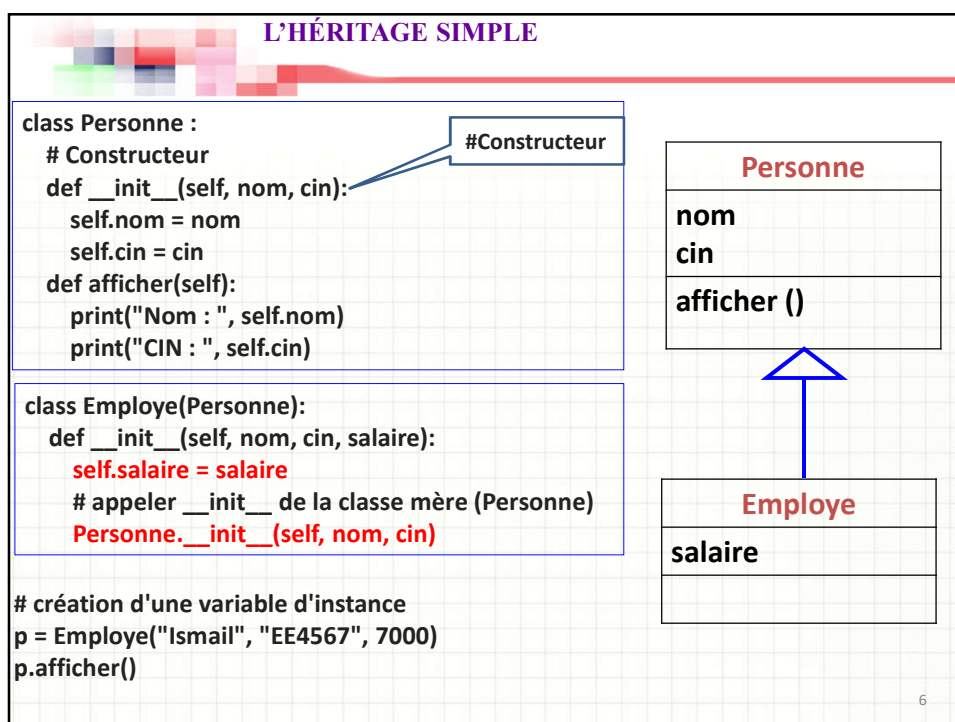
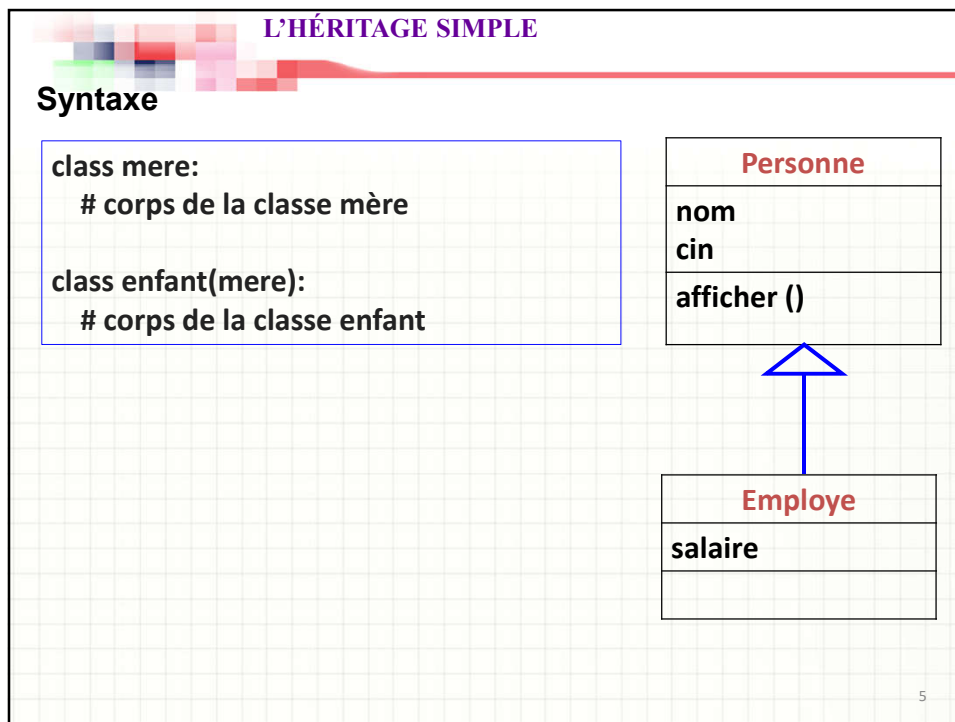
Mustapha HAIN  
ENSAM-Casablanca  
Infohain@gmail.com

### LES RELATIONS- DE UML AU PYTHON OO

**POO offre plusieurs relations, à savoir:**

- Héritage,
- Agrégation,
- Composition
- Association avec cardinalités.





## L'HÉRITAGE SIMPLE

```
class Personne():
    # Constructeur
    def __init__(self, nom, cin):
        self.nom = nom
        self.cin = cin
    def afficher(self):
        print("Nom : ",self.nom)
        print("CIN : ",self.cin)

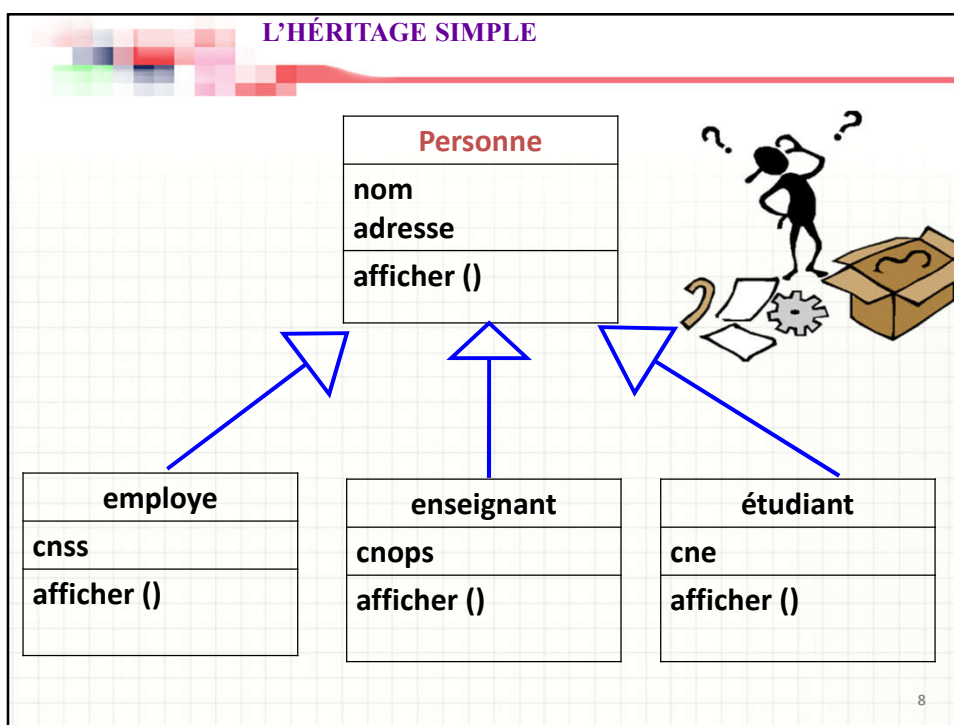
class Employe( Personne ):
    def __init__(self, nom, cin, salaire):
        self.salaire = salaire
    # appeler __init__ de la classe mère (Personne)
    Personne.__init__(self, nom, cin)

p=Employe("Ismail", "EE4567", 7000)
p.afficher()
```

Nom : Ismail  
CIN : EE4567

7

## L'HÉRITAGE SIMPLE

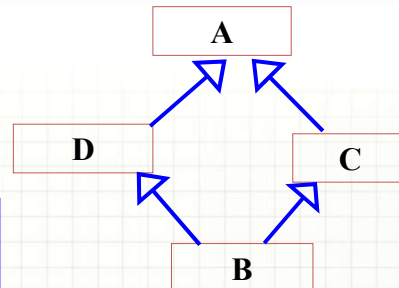


8

## HÉRITAGE MULTIPLE

Python permet de dériver une classe à partir de plusieurs classes à la fois, cela s'appelle HÉRITAGE MULTIPLE.

```
Class mere_1:
    # corps de mere_1
Class mere_2:
    # corps de mere_2
Class mere_3:
    # corps de mere_3
Class enfant(mere_1, mere_2, mere_3):
    # corps de la classe enfant
```



La classe enfant est dérivée de trois classes mere\_1, mere\_2, mere\_3. En conséquence, il héritera des attributs et des méthodes des trois classes.

9

## HÉRITAGE MULTIPLE

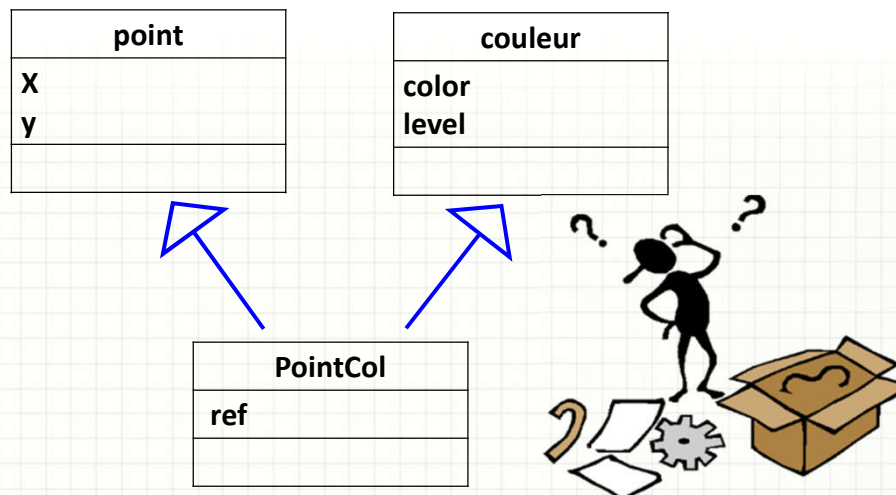
```
class Person(object):
    def __init__(self, name,
idnumber):
        self.name = name
        self.idnumber = idnumber
    def afficher(self):
        print(self.name,self.idnumber)
class Employee(object):
    def __init__(self, salary, post):
        self.salary = salary
        self.post = post
    def afficher(self):
        print(self.salary, self.post)
```

```
class Leader(Person, Employee):
    def __init__(self, name,
idnumber, salary, post, points):
        self.points = points
        Person.__init__(self, name, idnumber)
        Employee.__init__(self, salary, post)
    def afficher(self):
        Person.afficher(self)
        Employee.afficher(self)
        print(self.points)
```

```
p1=Person("Sara",10)
p2=Employee(7000,"Manager")
p3=Leader("Karim",11,6500,"Sales","DG"
)
p1.afficher()
p2.afficher()
p3.afficher()
```

10

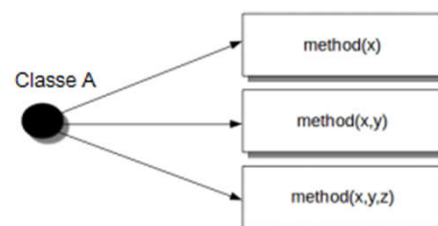
## HÉRITAGE MULTIPLE



11

## SURCHARGE(OVERLOADING)

Le surcharge(Overloading) est de définir plusieurs méthodes de signatures différentes pour un même nom. C'est-à-dire définir, dans la même classe, plusieurs fonctions de même nom mais de différent nombre d'arguments.



En python pas de surcharge, La solution donc est d'utiliser les paramètres avec valeur par défaut.

12

