Partie III

La programmation orientée objets



Informatique 1
Introduction à la programmation *orientée objets*

Mission 8 : (une courte) INTRODUCTION

Classes et objets

Kim Mens

Mission 8 : Matière à étudier !!

Objects

1 - Classes and objects - Basics
2 - Classes and objects - Advanced
3 - Even more object-oriented programming
4 - Overloading and polymorphism
5 - Collections of objects
6 - Inheritance
7 - Linked lists

1 - Classes and objects - Basics

- Object-oriented programming
- User-defined compound data types
- Attributes
- Improving the initialiser
- Adding other methods to the class
- Instances as arguments and parameters
- Converting an instance to a string
- Instances as return values
- A change of perspective
- Objects can have state

2 - Classes and objects - Advanced

- Rectangles
- Objects are mutable
- Sameness
- Copying

Qu'est-ce qu'une **classe** ?

- On aimerait manipuler différents objets similaires d'un même type
- Une classe décrit les caractéristiques communes (le type) de tous ces objets
  - **Attributs** (état)
    et valeurs initiales
  - **Méthodes** (fonctionnalité)
- La classe est comme une « usine » pour créer des **objets** de cette classe
  - **Instances** (objets)



Chaque objet/instance à sa propre identité et état

Créer une classe

```
class NokiaPhone :
```

(1) **Nommer** la classe

(2) Déterminer **les attributs** et **méthodes**

(3) Créer la méthode d'initialisation **__init__**

```
def __init__(self,s,p,t) :
  self.marque = "Nokia"
  self.serie = s
  self.poids = p
  self.taille = t
```

NokiaPhone
marque : Nokia série poids taille
print_specs()

(4) Créer les autres **méthodes** d'instances

```
def print_specs(self) :
  print(self.marque + " " + str(self.serie))
  print("Poids:  " + str(self.poids) + " g")
  print("Taille: " + self.taille + " mm")
```

```
nokia_kim = NokiaPhone(5110,170,"132x47.5x31")
```

(NokiaPhone)
marque: Nokia
série: 5110
poids: 170g
taille: 132x47x31mm

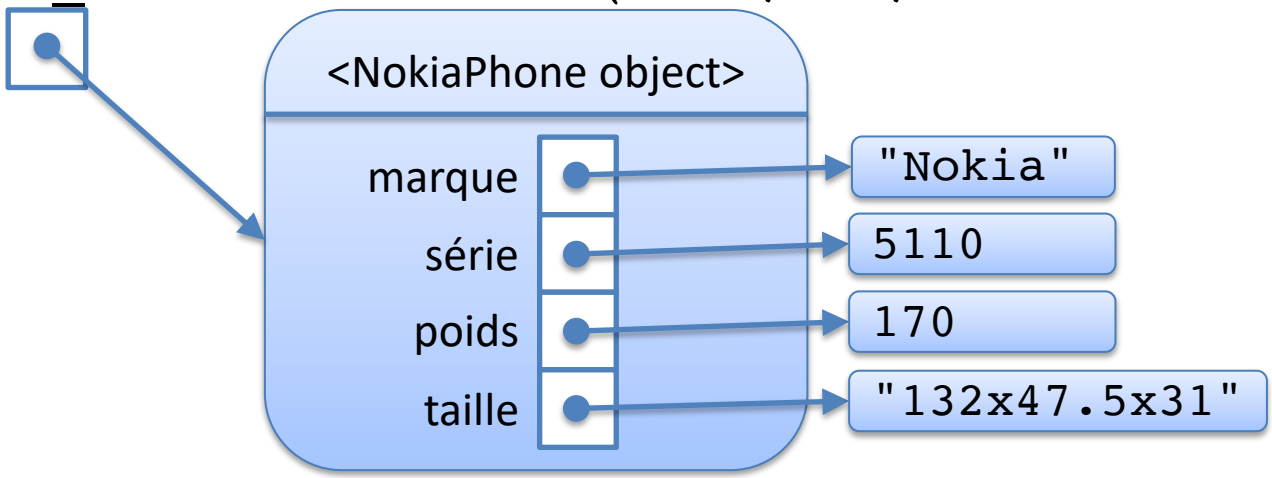
Créer un objet

```
class NokiaPhone :

  def __init__(self,s,p,t) :
    self.marque = "Nokia"
    self.serie  = s
    self.poids  = p
    self.taille = t
  …
nokia_kim = NokiaPhone(5110,170,"132x47.5x31")
```

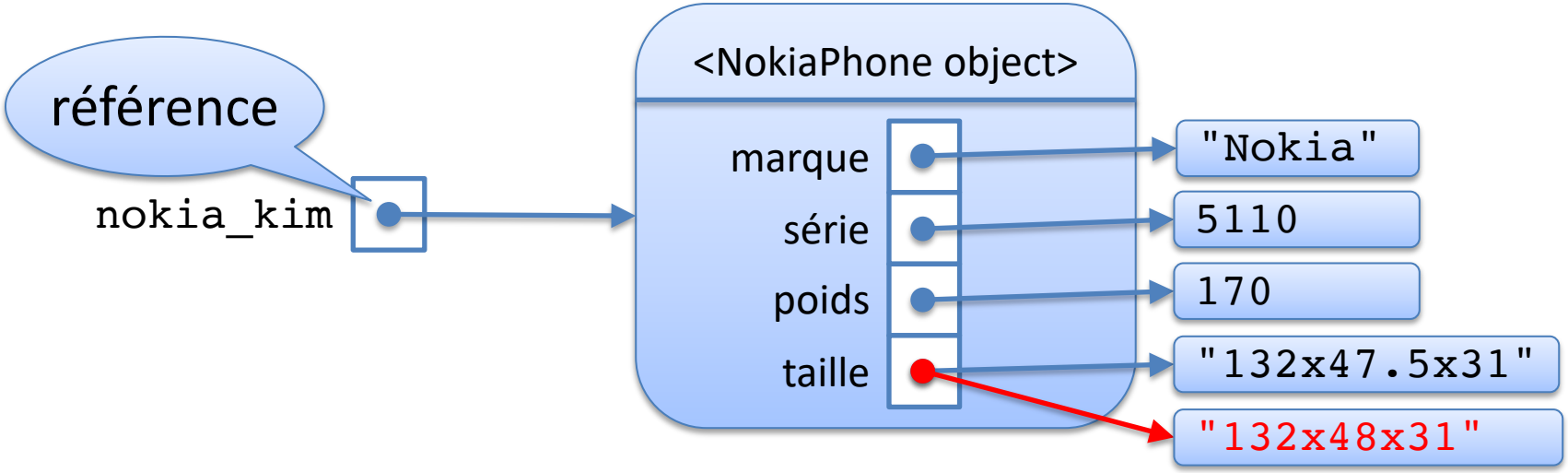
méthode d'initialisation __init__
(initialise les variables d'instance)

self
l'objet nouvellement créé



Accéder aux attributs d'un objet

```
>>> nokia_kim = NokiaPhone(5110,170,"132x47.5x31")
>>> nokia_kim
<__main__.NokiaPhone object at 0x1042686a0>
```




```
>>> nokia_kim.marque
'Nokia'
>>> nokia_kim.poids
170
>>> nokia_kim.taille = "132x48x31"
```

Appeler une méthode sur un objet

```
class NokiaPhone :

  def __init__(self,s,p,t) :
    self.marque = "Nokia"
    self.serie = s
    self.poids = p
    self.taille = t

  def print_specs(self) :
    print(self.marque + " " + str(self.serie))
    print("Poids:  " + str(self.poids) + " g")
    print("Taille: " + self.taille + " mm")

nokia_kim = NokiaPhone(5110,170,"132x48x31")
nokia_kim.print_specs()
```

self
le récepteur du message

Nokia 5110
Poids: 170 g
Taille: 132x48x31 mm

Références

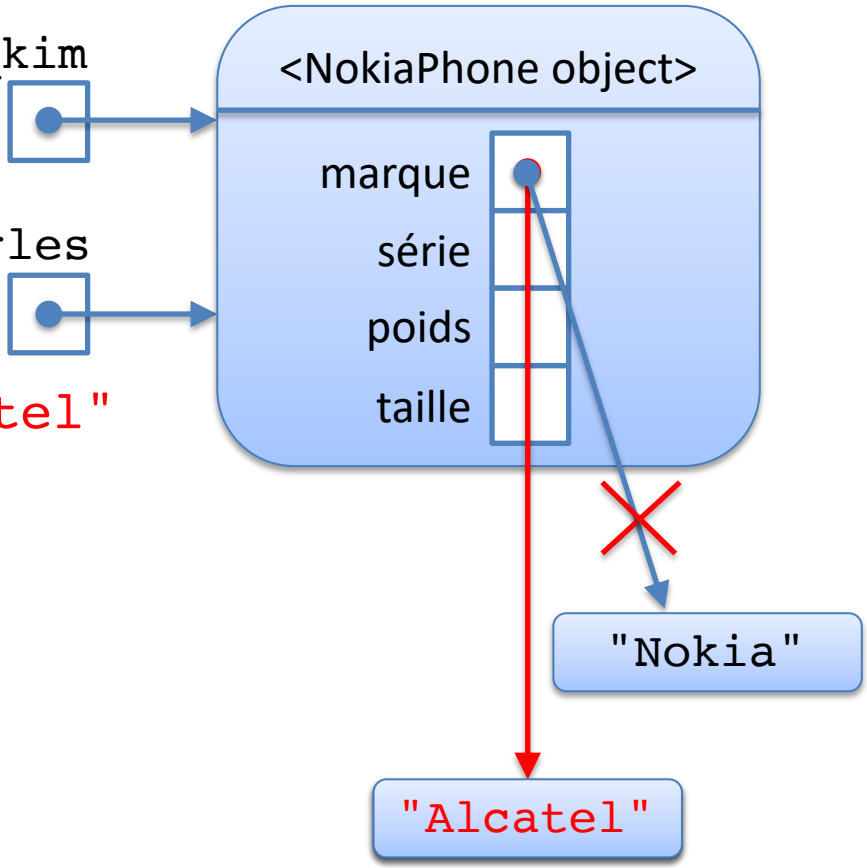
```
nokia_kim = NokiaPhone(5110,170,"132x48x31")

phone_charles = nokia_kim

phone_charles.marque = "Alcatel"

phone_charles = None

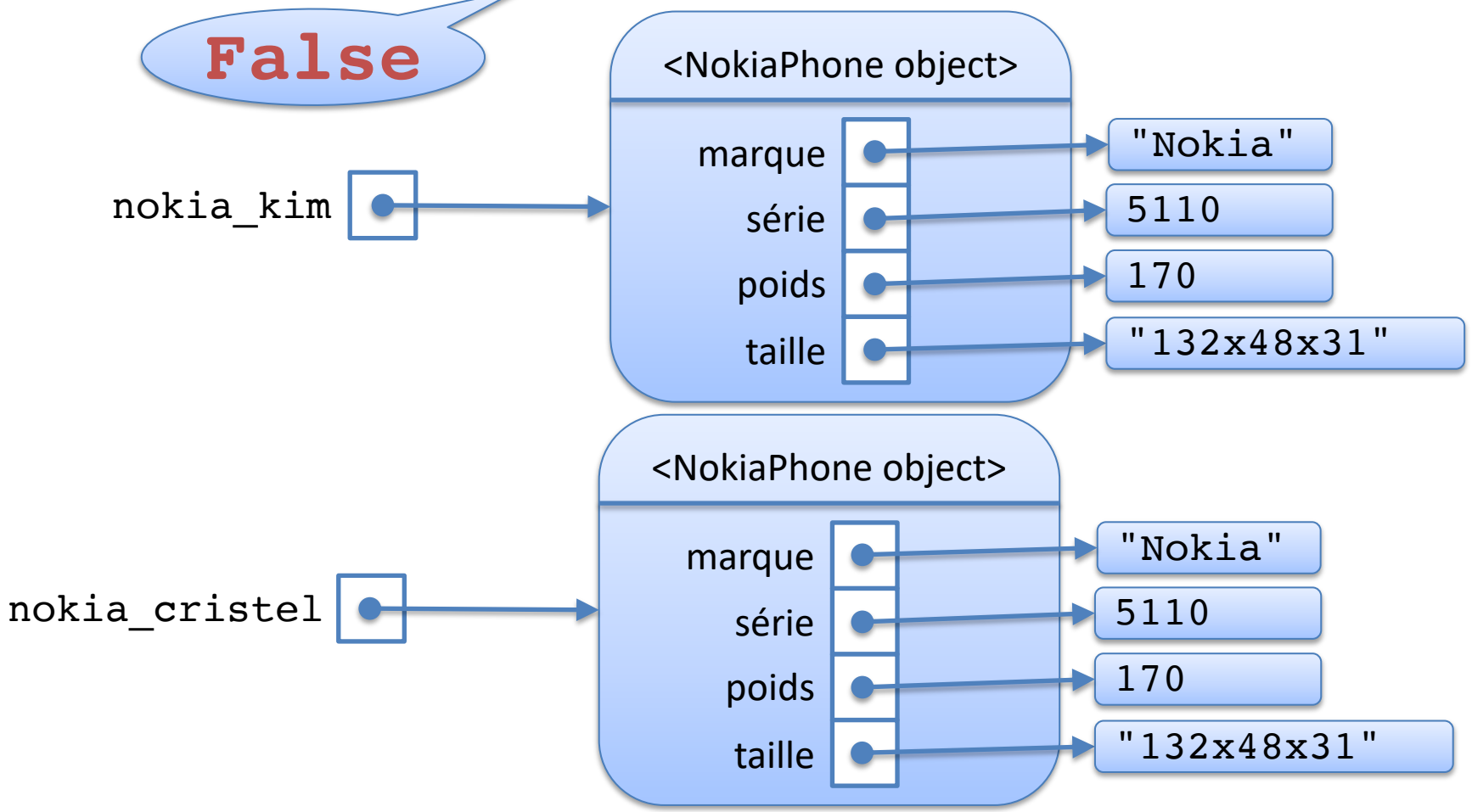
nokia_kim.print_specs()
```



Alcatel 5110
Poids: 170 g
Taille: 132x48x31 mm

Egalité entre objets

```
nokia_kim =      NokiaPhone(5110,170,"132x48x31")
nokia_cristel = NokiaPhone(5110,170,"132x48x31")
print(nokia_kim is nokia_cristel)
```



La méthode magique __str__

NokiaPhone

marque : Nokia
série
poids
taille

```
class NokiaPhone :
```

(1) **Nommer** la classe
(2) Déterminer **les attributs** et **méthodes**
(3) Créer la méthode d'initialisation **__init__**

```
def __init__(self,s,p,t) :
    self.marque = "Nokia"
    self.serie = s
    self.poids = p
    self.taille = t
```

```
def print_specs(self) :
    print(self.marque + " " + str(se
    print("Poids:  " + str(self.poid
    print("Taille: " + self.taille +
```

(5) Créer la méthode **__str__**

```
def __str__(self) :
    return self.marque + " " + str(self.serie) + "\n" \
           + "Poids:  " + str(self.poids) + " g" + "\n" \
           + "Taille: " + self.taille + " mm" + "\n"
```

```
nokia_kim = NokiaPhone(5110,170,"132x48x31")
print(nokia_kim)
nokia_kim.__str__()
```

Nokia 5110
Poids: 170 g
Taille: 132x48x31 mm

MISSION 8

Objectif

Apprendre à programmer avec des objets

Problème

- Albums de chansons
- Définir des objets pour représenter un album, une durée, une chanson

Album 1 (17 chansons, 01:10:55)
1: Let's_Dance - David_Bowie - 00:04:05
2: Relax - Frankie_Goes_To_Hollywood - 00:03:54
3: Purple_Rain - Prince - 00:05:48
4: Enjoy_The_Silence - Depeche_Mode - 00:04:13
5: Chacun_Fait_C'Qui_Lui_Plait - Chagrin_D'amour - 00:04:08
6: Love_Missile_F1-11 - Sigue_Sigue_Sputnik - 00:04:06
7: Spaceman - Babylon_Zoo - 00:03:59
8: Hijo_De_La_Luna - Mecano - 00:04:19
9: 7_Seconds - Youssou_N'Dour_&_Neneh_Cherry - 00:03:48
10: Osez_Joséphine - Alain_Bashung - 00:02:57
11: Déjeuner_En_Paix - Stephan_Eicher - 00:03:27
12: New_Gold_Dream - Simple_Minds - 00:05:31
13: Missing - Everything_But_The_Girl - 00:04:44
14: Nineteen - Paul_Hardcastle - 00:03:38
15: Killer - Adamski - 00:04:13
16: Unbelievable - EMF - 00:03:29
17: Overload - Sugababes - 00:04:36

Album 2 (17 chansons, 01:11:24)
1: Ice_Ice_Baby - Vanilla_Ice - 00:04:29
2: Do_You_Really_Want_To_Hurt_Me - Culture_Club - 00:04:23
3: Under_The_Milky_Way - The_Church - 00:04:57
4: Shout - Tears_For_Fears - 00:06:29
5: Pure_Morning - Placebo - 00:04:15
6: Porcelain - Moby - 00:04:00
7: Toi_Mon_Toit - Elli_Medeiros - 00:03:37
8: Just_A_Friend_Of_Mine - Vaya_Con_Dios - 00:03:22
9: Sleeping_Satellite - Tasmin_Archer - 00:04:36
10: I_Won't_Let_You_Down - DPH - 00:04:05
11: A_Girl_Like_You - The_Smithereens - 00:04:36