Partie III

La programmation orientée objets



Informatique 1
Introduction à la programmation

Mission 8 : RESTRUCTURATION

Classes et objets

Kim Mens

Classes et objets

```
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")
```

objet / instance

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

constructeur

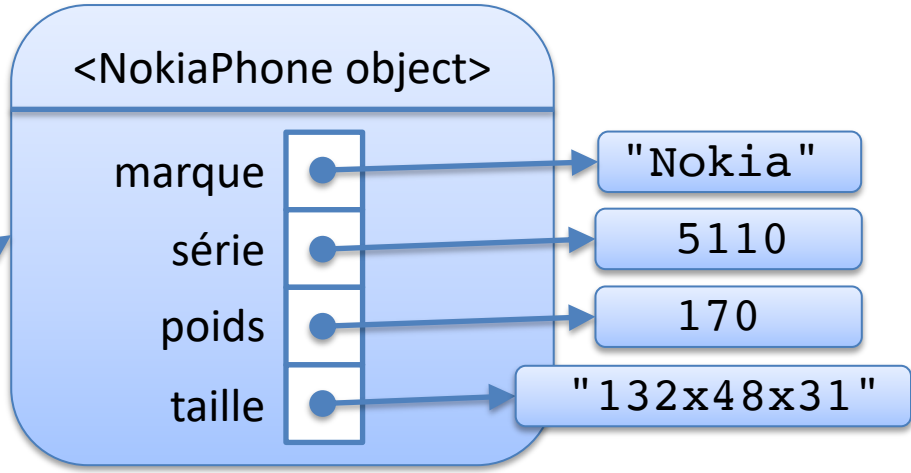
(NokiaPhone)
marque: Nokia
série: 5110
poids: 170g
taille: 132x47x31mm

Manipuler 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) :
      …
```



référence

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

création d'un objet

appel d'une méthode

➔

```
Nokia 5110
Poids:  170 g
Taille: 132x48x31 mm
```

Appeler une méthode

```
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
```

Manipuler 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) :
    …

nokia_kim = NokiaPhone(5110,170,"132x48x31")
nokia_kim.print_specs()
print_specs(nokia_kim)
➔ NameError: name 'print_specs' is not defined
```

Manipuler 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) :
        …
```




```
nokia_kim = NokiaPhone(5110,170,"132x48x31")
nokia_kim.print_specs
```
➔ *< bound method NokiaPhone.print_specs of …>*

```
nokia_kim.print_specs = 2
```

```
nokia_kim.print_specs()
```
➔ `TypeError: 'int' object is not callable`

Manipuler 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,"132x48x31")
```




```
nokia_kim.batterie
➔ AttributeError: 'NokiaPhone' object
   has no attribute 'batterie'
nokia_kim.batterie = 'faible'
nokia_kim.batterie
➔ 'faible'
```

Appeler soi-même

```
class NokiaPhone :

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

  def print_type(self) :
    print(self.marque + " " + str(self.serie))

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

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

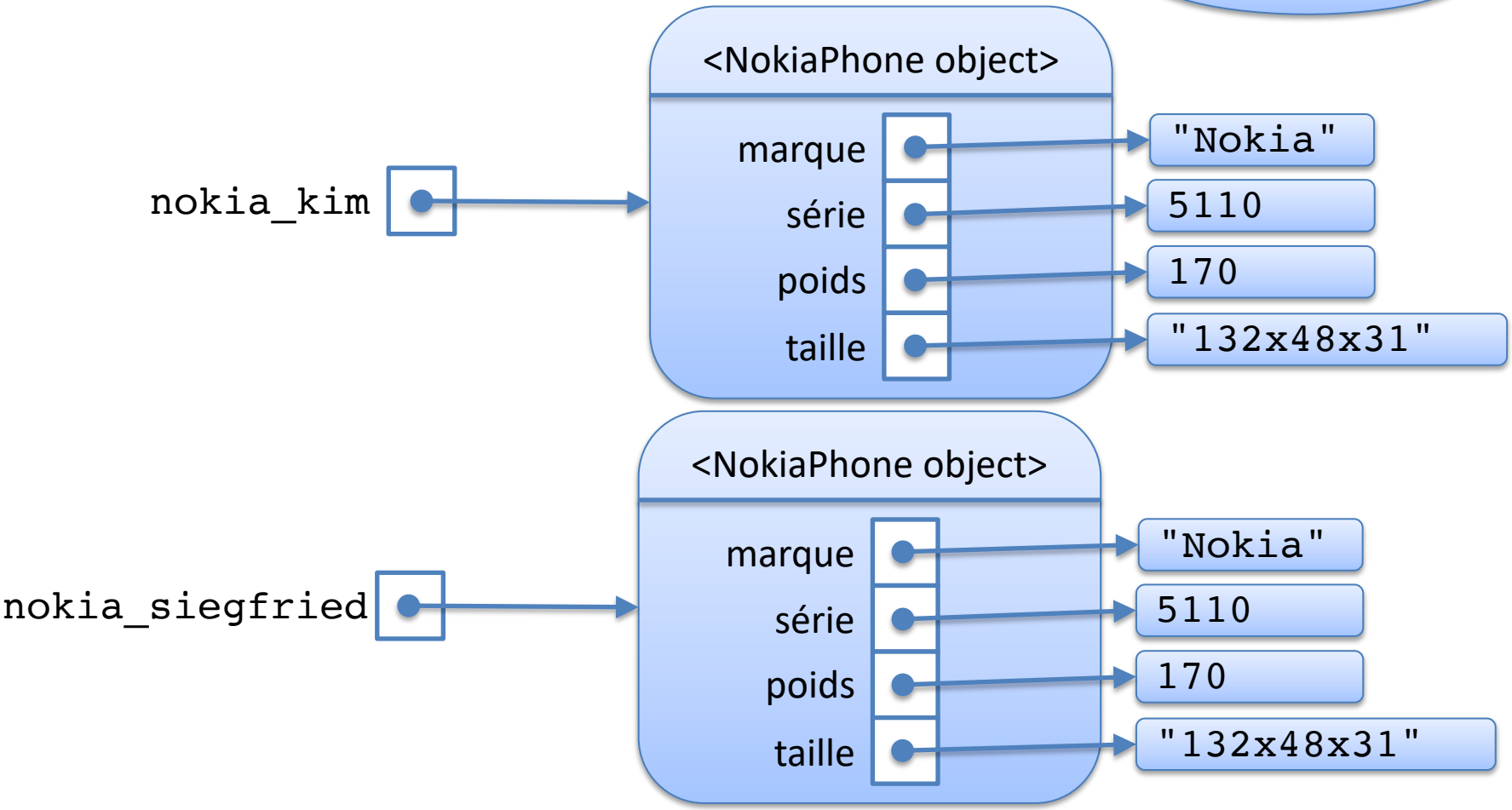
self
le récepteur du message

Nokia 5110
Poids: 170 g
Taille: 132x48x31 mm

Egalité entre objets

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

False



Egalité entre objets

```
class NokiaPhone :
```

(1) **Nommer** la classe

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

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

(6) Créer la méthode **__eq__**

```
def __eq__(self, other) :
  if not isinstance(other,NokiaPhone) :
      return False
  return (self.marque == other.marque)      \
         and (self.serie == other.serie)    \
         and (self.poids == other.poids)    \
         and (self.taille == other.taille)
```

isinstance *vérifie si l'objet donné en premier paramètre est une instance de la classe donnée en second paramètre*

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

True

Méthodes magiques

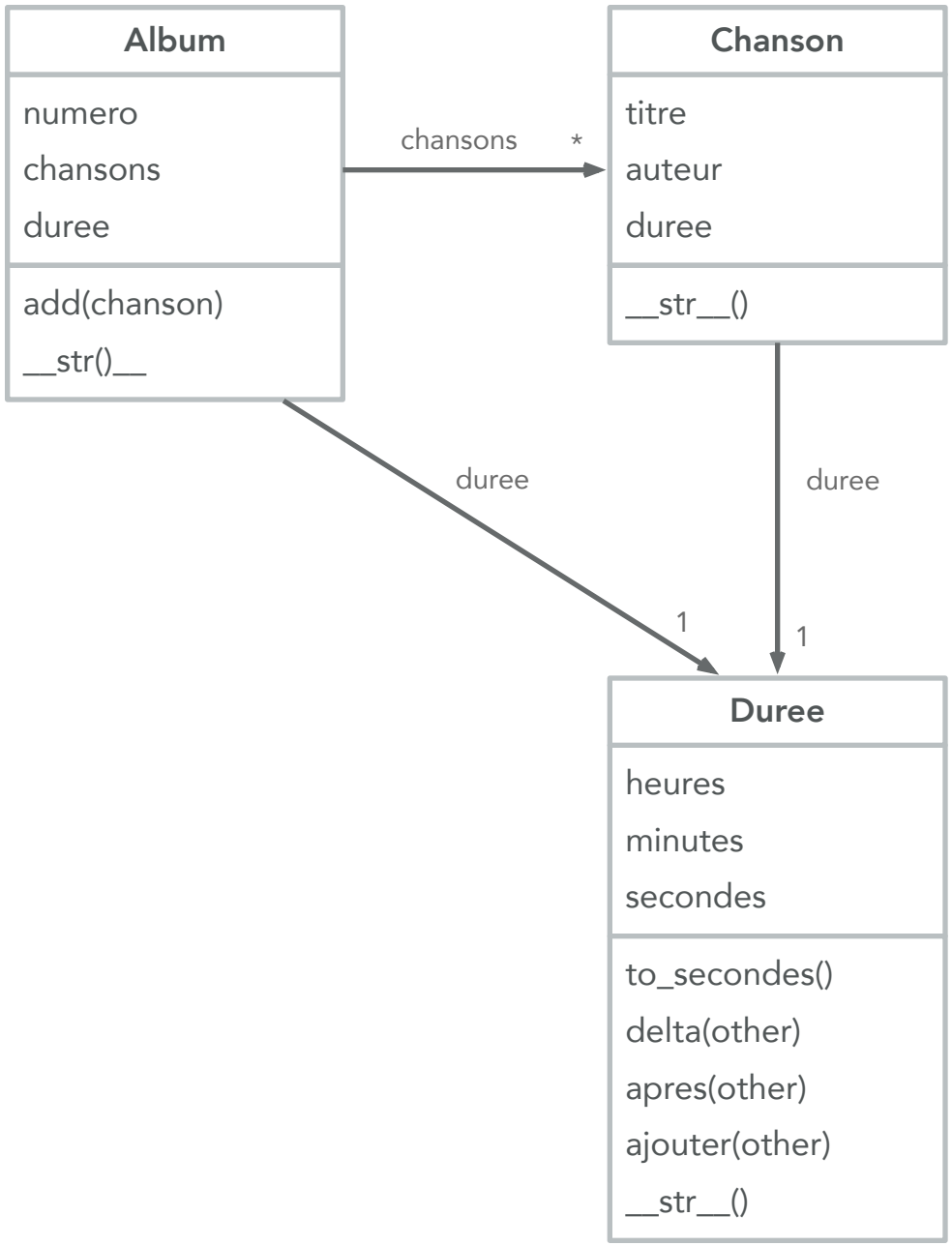
		Appelé “magiquement” par
`__init__`	***méthode d’initialisation***	**le constructeur d’une classe**
`__str__`	***conversion en string***	**str() , print()**
`__eq__`	***égalité***	**==**
`__lt__`	*inférieur à*	<
`__ge__`	*supérieur ou égal à*	>=
`...`	*autres comparateurs*	<= ,!= , >, …
`__add__`	*addition*	+
`__sub__`	*soustraction*	-
…	*autres opérateurs*	* , // , / , % , **, …

Composition de classes

Cf. la mission 8 :

Une classe peut être composé d'une autre.

Un objet peut contenir (une référence vers) un autre.



Composition de classes



NokiaPhone

marque : Nokia
série
poids
taille

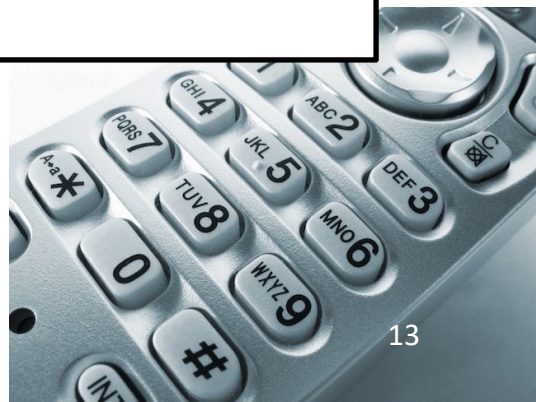
print_specs()
…

numéro

PhoneNumber

country
zone
prefix
postfix

call()
…



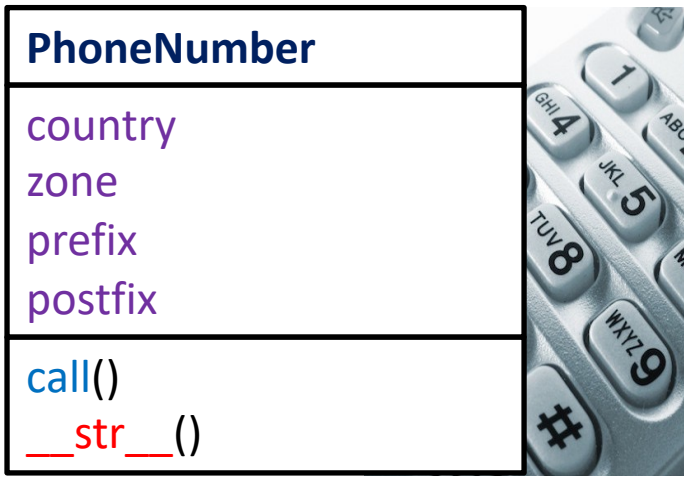
PhoneNumber

```
class PhoneNumber :

  def __init__(self,c,z,pr,po):
    self.country = c
    self.zone = z
    self.prefix = pr
    self.postfix = po

  def call(self):
    # code pour appeler ce numéro de téléphone

  def __str__(self):
      return "+" + self.country + "(0)" + self.zone \
      + "/" + self.prefix + self.postfix
```




```
my_number = PhoneNumber("32","10","4","79111")
print(my_number) ➔ +32(0)10/479111
```

Composition de classes

```
class NokiaPhone :

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

  …

  def print(self) :
    self.print_specs()
    print(self.numero)
```



NokiaPhone
marque : Nokia série poids taille numero
… print ()

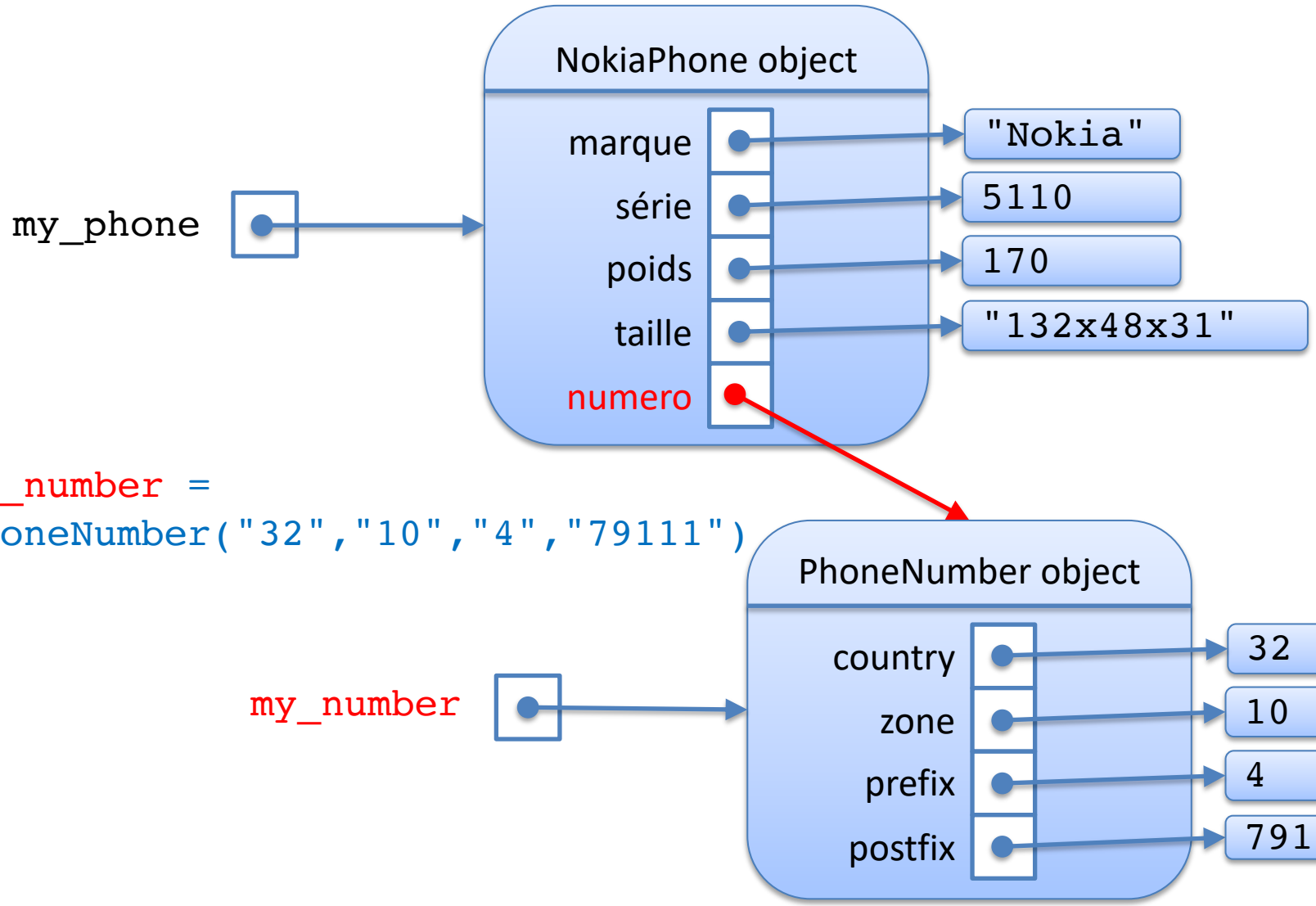
Nouvel attribut d'instance

Nouvelle méthode d'instance

```
my_number = PhoneNumber("32","10","4","79111")
my_phone = NokiaPhone(5110,170,"132x48x31",my_number)
```

Composition de classes

```
my_phone = NokiaPhone(5110,170,"132x48x31",my_number)
```



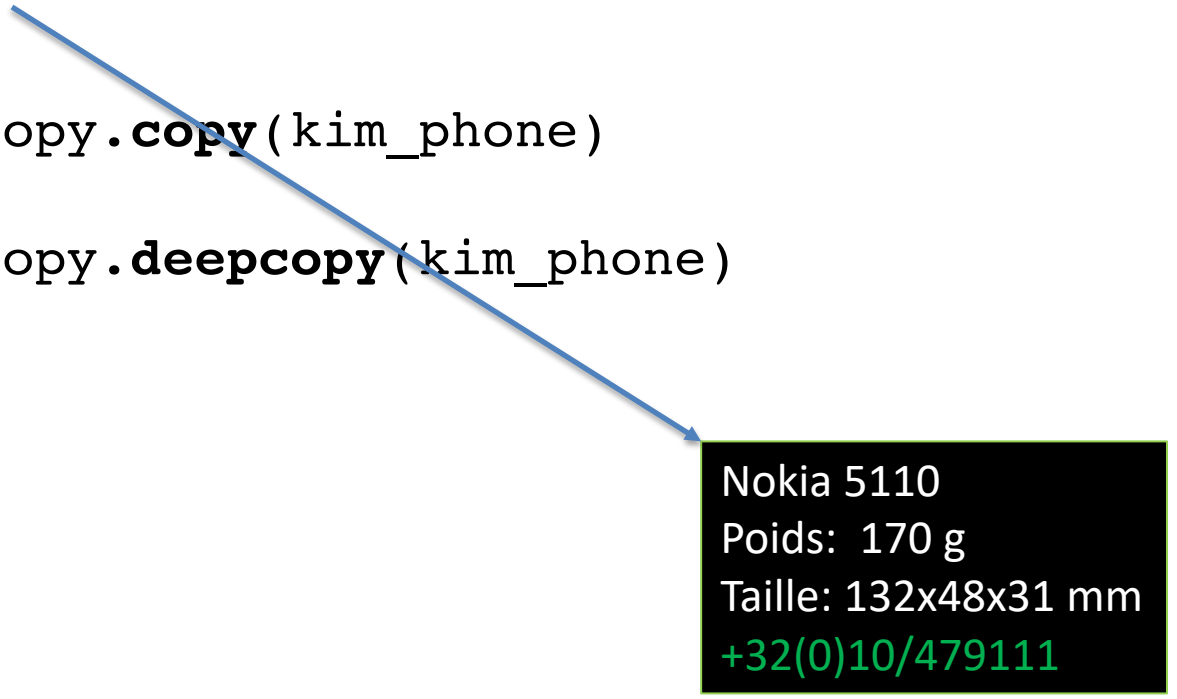
Copie profonde d'un objet

```
import copy

kim_number = PhoneNumber("32","10","4","79111")
kim_phone = NokiaPhone(5110,170,"132x48x31",kim_number)
kim_phone.print()

charles_phone = copy.copy(kim_phone)

cristel_phone = copy.deepcopy(kim_phone)
```



Copie profonde d'un objet

```
import copy

kim_number = PhoneNumber("32","10","4","79111")
kim_phone = NokiaPhone(5110,170,"132x48x31",kim_number)
kim_phone.print()

charles_phone = copy.copy(kim_phone)

cristel_phone = copy.deepcopy(kim_phone)

kim_number.postfix = "79122"

kim_phone.print()
```



Nokia 5110
Poids: 170 g
Taille: 132x48x31 mm
+32(0)10/479122

Copie profonde d'un objet

```
import copy

kim_number = PhoneNumber("32","10","4","79111")
kim_phone = NokiaPhone(5110,170,"132x48x31",kim_number)
kim_phone.print()

charles_phone = copy.copy(kim_phone)

cristel_phone = copy.deepcopy(kim_phone)

kim_number.postfix = "79122"

kim_phone.print()
charles_phone.print()
```



Copie profonde d'un objet

```
import copy

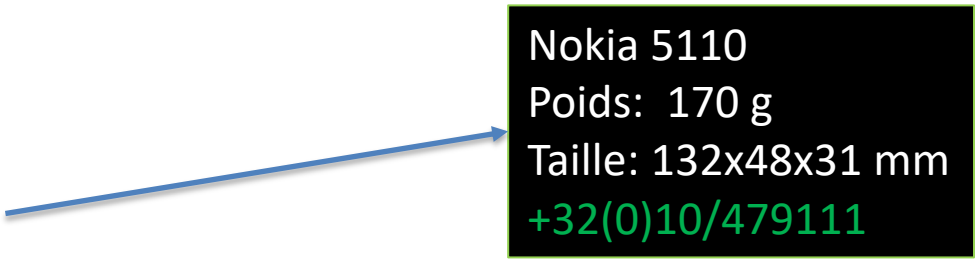
kim_number = PhoneNumber("32","10","4","79111")
kim_phone = NokiaPhone(5110,170,"132x48x31",kim_number)
kim_phone.print()

charles_phone = copy.copy(kim_phone)

cristel_phone = copy.deepcopy(kim_phone)

kim_number.postfix = "79122"

kim_phone.print()
charles_phone.print()
cristel_phone.print()
```

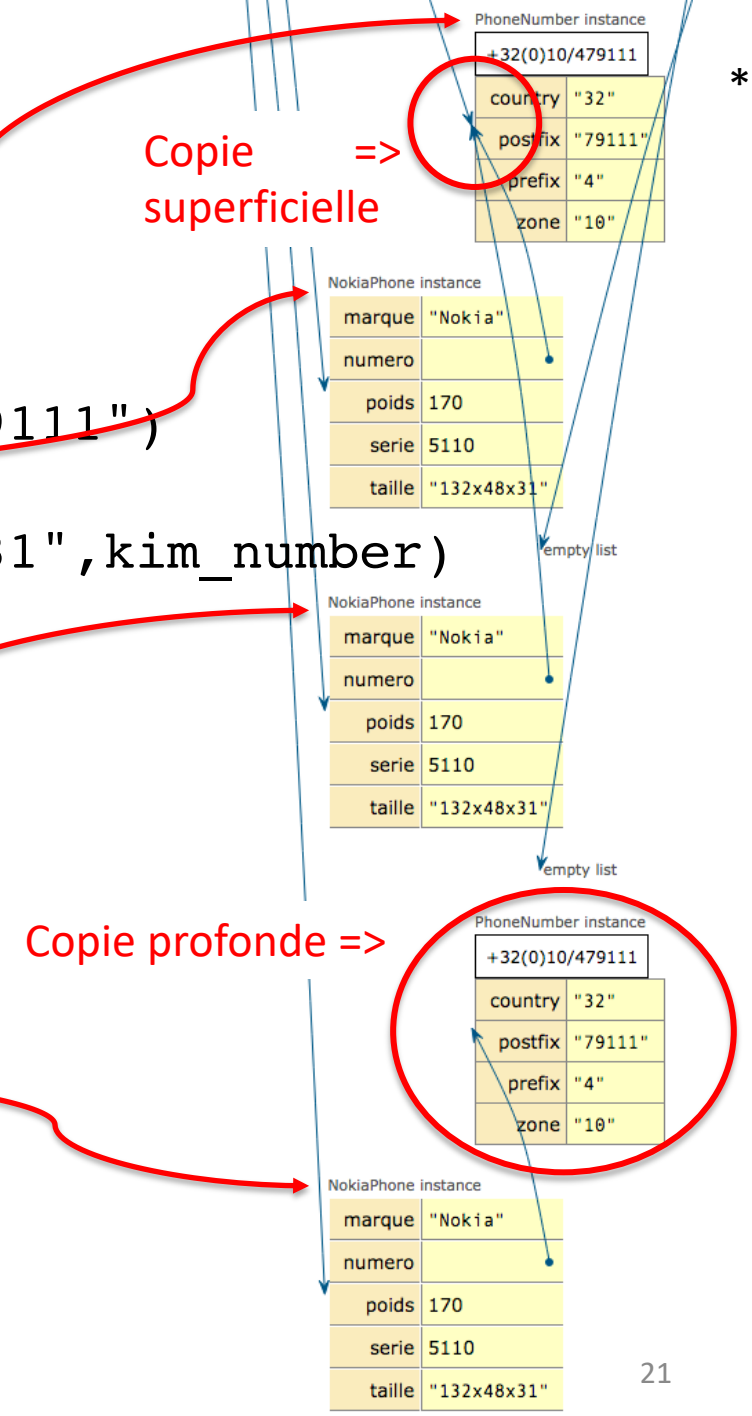


Copie profonde vs. Copie superficielle

```
kim_number =
    PhoneNumber("32","10","4","79111")
kim_phone =
    NokiaPhone(5110,170,"132x48x31",kim_number)
```

```
charles_phone =
    copy.copy(kim_phone)
```

```
cristel_phone =
    copy.deepcopy(kim_phone)
```



* Image produit par PythonTutor

Code

Objects

- 1 - Classes and objects - Basics
- 2 - Classes and objects - Advanced
- 3 - Even more object-oriented programming
- 4 - Collections of objects
- 5 - Inheritance
- 6 - Linked lists
- Appendix - Source code of phone class
- Appendix - Source code of card game
- Appendix - Source code of linked lists

https://syllabus-interactif.info.ucl.ac.be/syllabus/info1-theory/PART_III/appendix_code_phone_class