

Partie III

La programmation orientée objets



Informatique 1

Introduction à la programmation

Mission 8 : RESTRUCTURATION

Classes et objets

Kim Mens – Siegfried Nijssen – Charles Pecheur

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

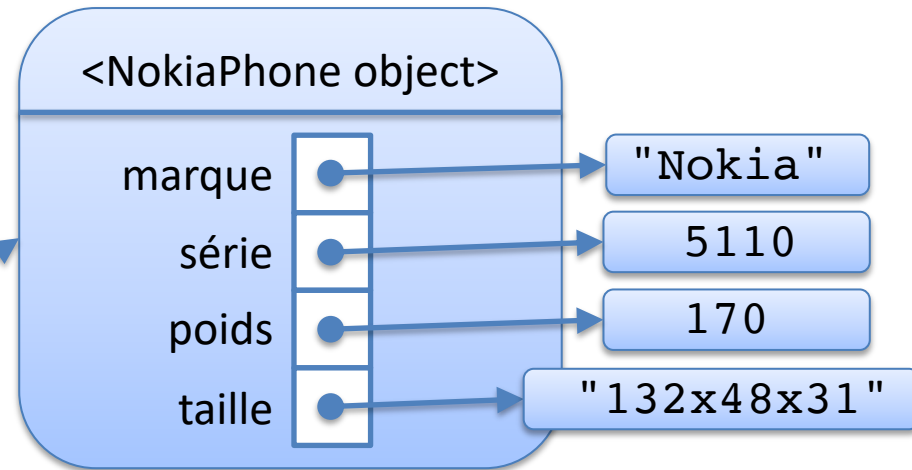
série: 5110

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) :  
        ...
```



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

référence

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")
```

self
le récepteur du message

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

```
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)
```

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()
```

```
nokia_kim.print_specs(nokia_kim)
```

➔ TypeError: print_specs() takes 1 positional argument but 2 were given

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) :  
        print_type(self)  
        print("Poids: " + str(self.poids) + " g")  
        print("Taille: " + self.taille + " mm")  
  
nokia_kim = NokiaPhone(5110,170,"132x48x31")  
nokia_kim.print_specs()  
→ NameError: name 'print_type' is not defined
```

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

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

```
print(nokia_kim is nokia_tom)
```

False

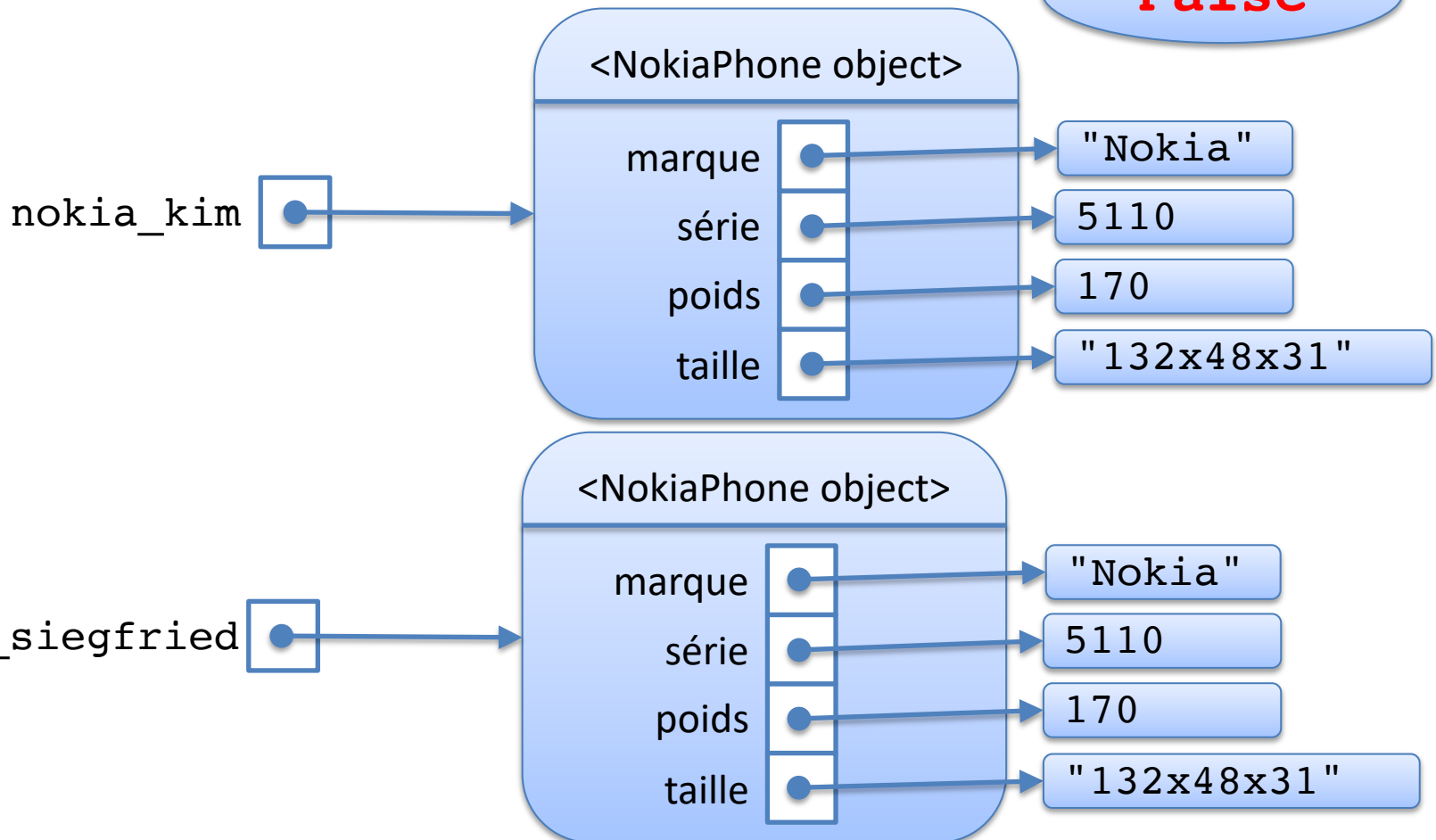
```
print(nokia_kim == nokia_tom)
```

False

Egalité entre objets

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

```
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)
```

(6) Créer la méthode `__eq__`

*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

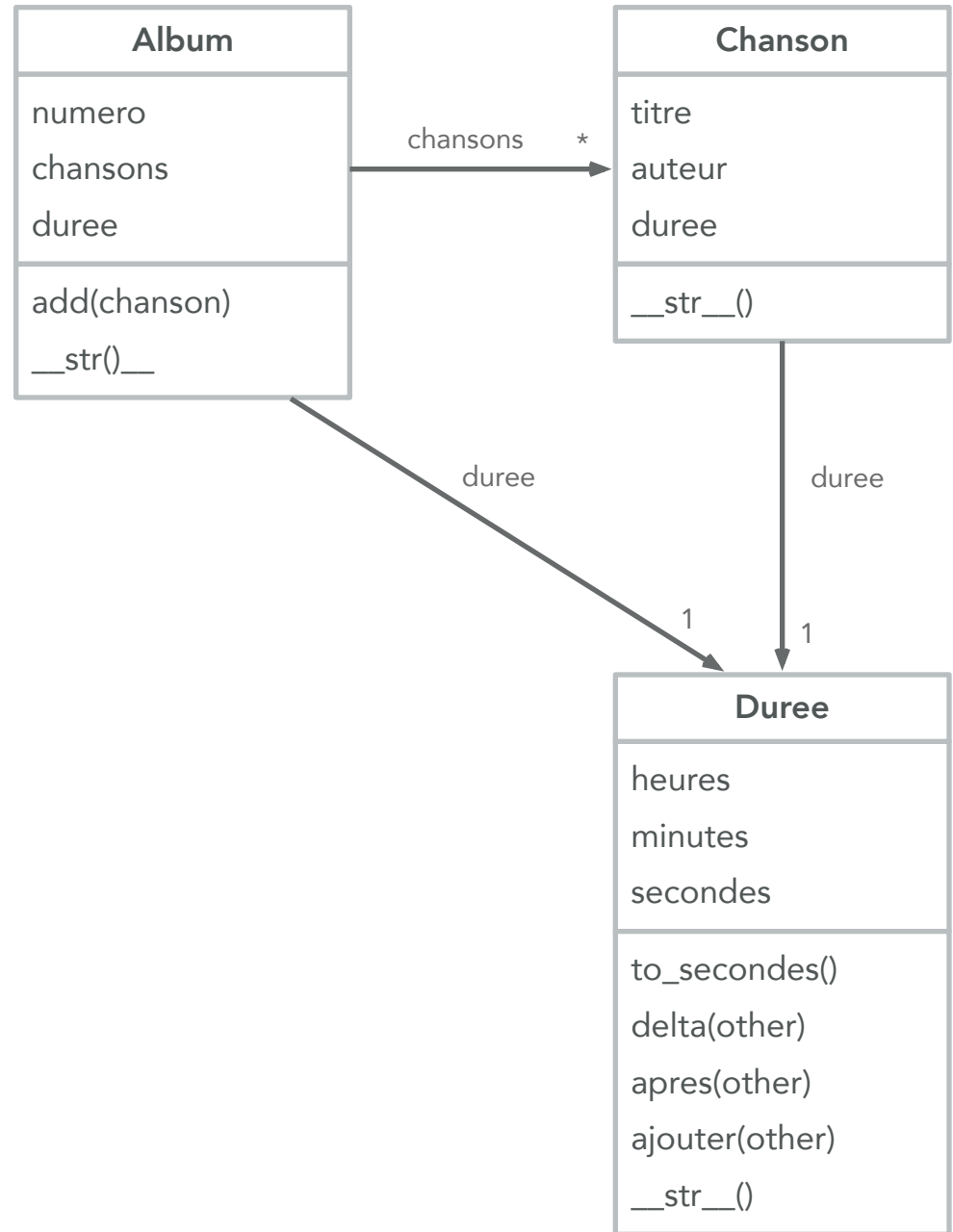
		Appelé “magiquement” par
<code>__init__</code>	<i>méthode d'initialisation</i>	le constructeur d'une classe
<code>__str__</code>	<i>conversion en string</i>	<code>str()</code> , <code>print()</code>
<code>__eq__</code>	<i>égalité</i>	<code>==</code>
<code>__lt__</code>	<i>inférieur à</i>	<code><</code>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

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

Cf. la mission 8 :

Un **Album** est composé d'une **Duree** et d'une liste de **Chanson**

Une **Chanson** contient aussi une **Duree** (et d'autres attributs primitifs)



Album instance	
Album 1 (3 chansons, 00:13:47)	
01: Let's_Dance - David_Bowie - 00:04:05	
02: Relax - Frankie_Goes_To_Hollywood - 00:03:54	
03: Purple_Rain - Prince - 00:05:48	

chansons	
duree	
number	1

list		
0	1	2

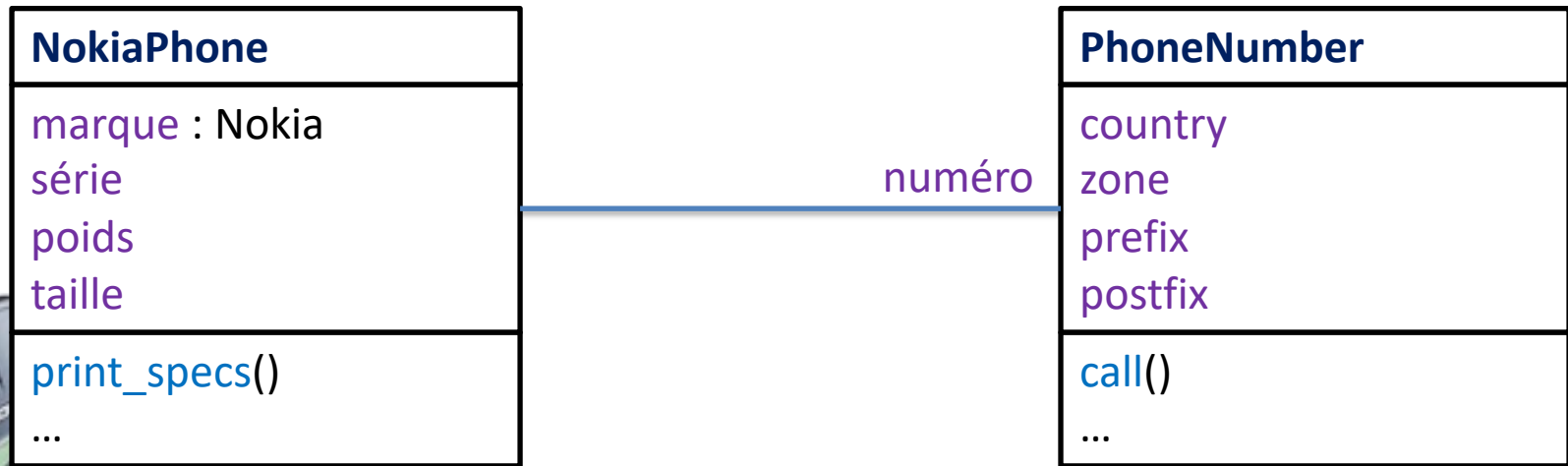
Duree instance	
00:13:47	
hours	0
minutes	13
seconds	47

Duree instance	
00:04:05	
hours	0
minutes	4
seconds	5

Chanson instance	
Let's_Dance - David_Bowie - 00:04:05	
author	"David_Bowie"
duree	
title	"Let's_Dance"

* Image produit par PythonTutor

Composition de classes



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):
```

```
        ...
```

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

PhoneNumber
country zone prefix postfix
call() __str__()



PhoneNumber

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

...

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

```
my_number = PhoneNumber("32", "10", "4", "79111")  
print(my_number) → TypeError: __str__ returned  
non-string (type NoneType)
```

PhoneNumber
country zone prefix postfix
call() __str__()



PhoneNumber

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

...

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

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

```
TypeError: __str__ returned non-string (type NoneType)
```

PhoneNumber
country zone prefix postfix
call() __str__()



PhoneNumber

```
class PhoneNumber :
```

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

...

```
    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
```

PhoneNumber
country zone prefix postfix
call() __str__()



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 ()

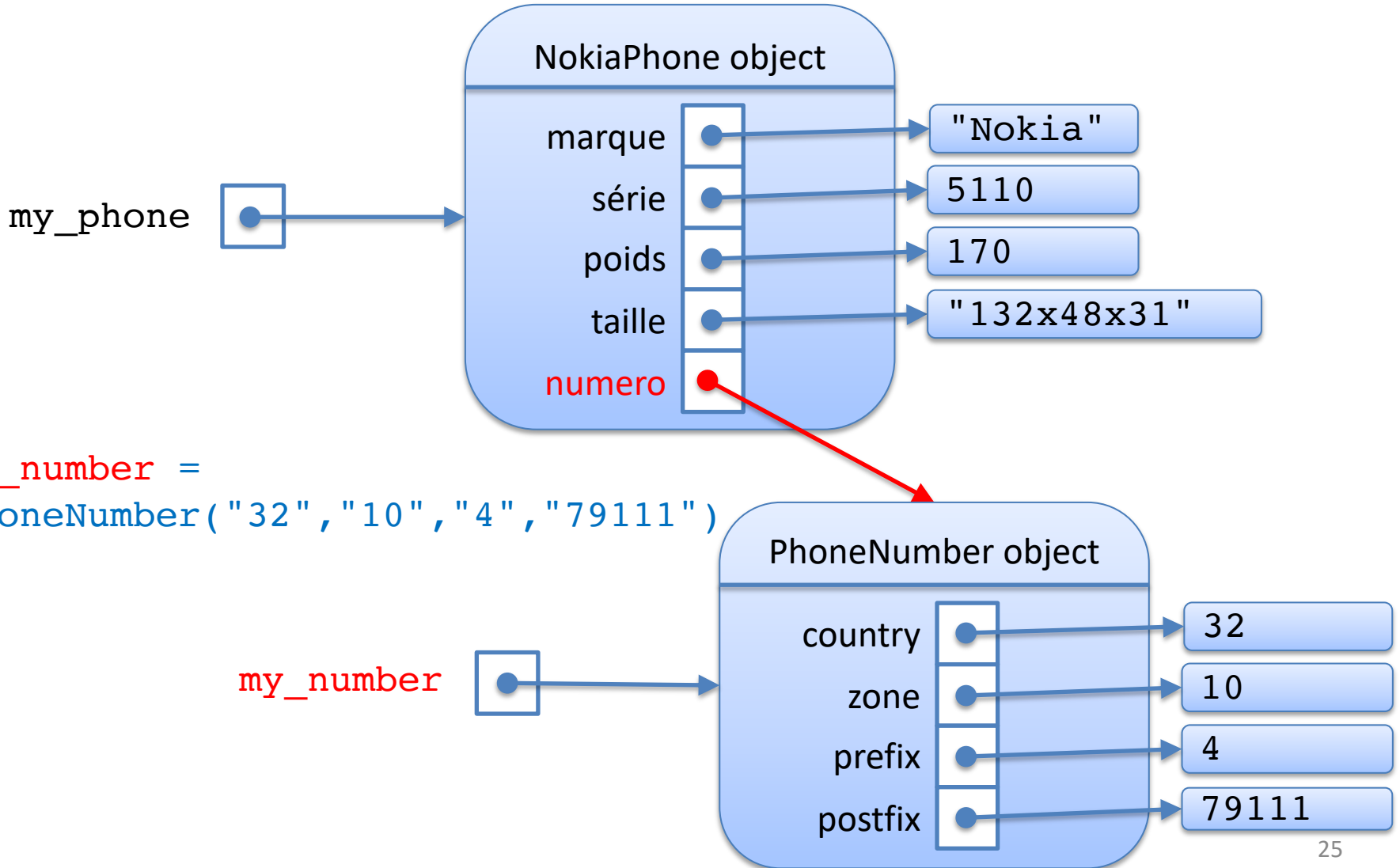
Nouvel attribut
d'instance

Nouvelle méthode
d'instance

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

Composition de classes

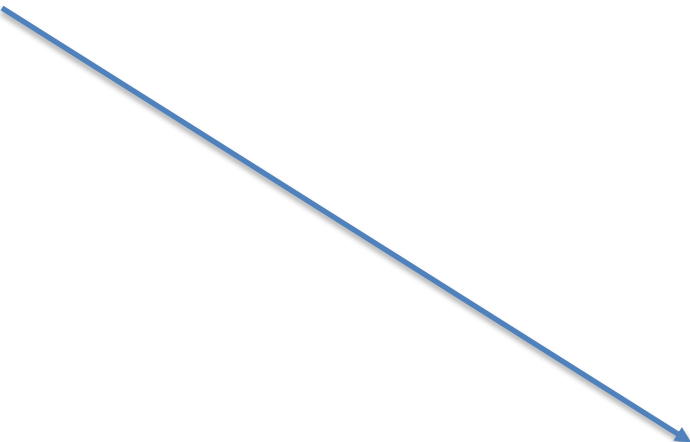
```
my_phone = NokiaPhone(5110,170,"132x48x31",my_number)
```



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()
```



```
Nokia 5110  
Poids: 170 g  
Taille: 132x48x31 mm  
+32(0)10/479111
```

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)
```

```
siegfried_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)
```

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

```
kim_number.postfix = "79122"
```

```
kim_phone.print()  
charles_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)
```

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

```
kim_number.postfix = "79122"
```

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



```
Nokia 5110  
Poids: 170 g  
Taille: 132x48x31 mm  
+32(0)10/479111
```

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)
```

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

Copie superficielle =>

PhoneNumber instance	
+32(0)10/479111	
country	"32"
postfix	"79111"
prefix	"4"
zone	"10"

NokiaPhone instance	
marque	"Nokia"
numero	
poids	170
serie	5110
taille	"132x48x31"

NokiaPhone instance	
marque	"Nokia"
numero	
poids	170
serie	5110
taille	"132x48x31"

Copie profonde =>

PhoneNumber instance	
+32(0)10/479111	
country	"32"
postfix	"79111"
prefix	"4"
zone	"10"

NokiaPhone instance	
marque	"Nokia"
numero	
poids	170
serie	5110
taille	"132x48x31"

* 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