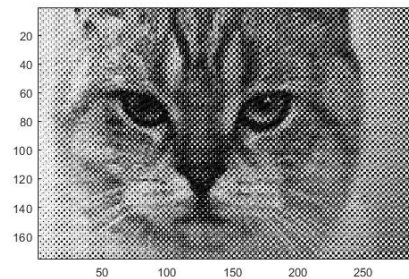
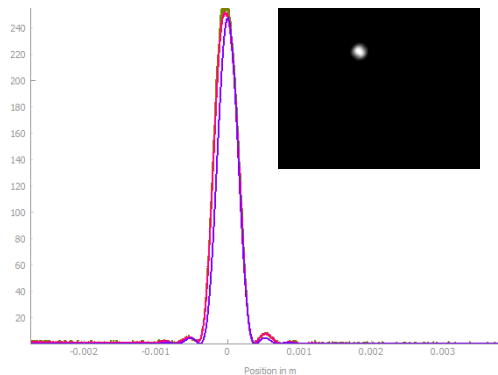
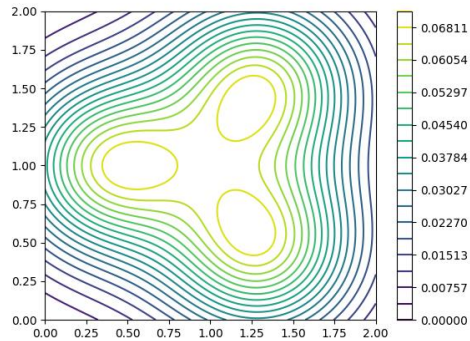


ONIP-2 / FISA

Programmation Orientée Objet

Outils Numériques / Semestre 6
/ Institut d'Optique / ONIP-2

ONIP-2 / Déroulement



TP1 - Diffraction

TP2/3 - Filtrage Détramage

3 séquences

Programmation Objet

Filtrage

Diffraction

ENTREPRISE

X

TP1a

ENTREPRISE

TP1b

TP2a

TP2b

ENTREPRISE

TP3a

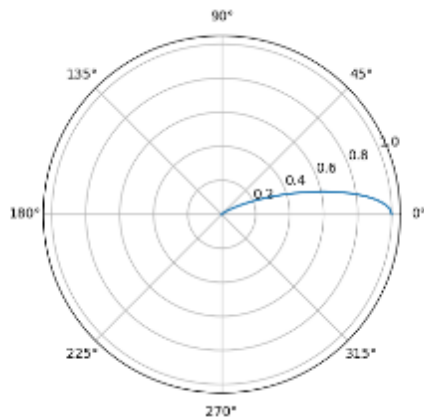
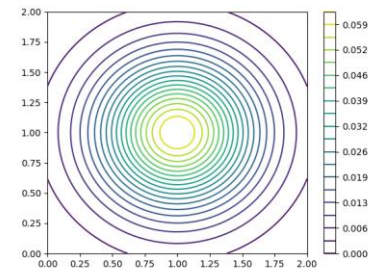
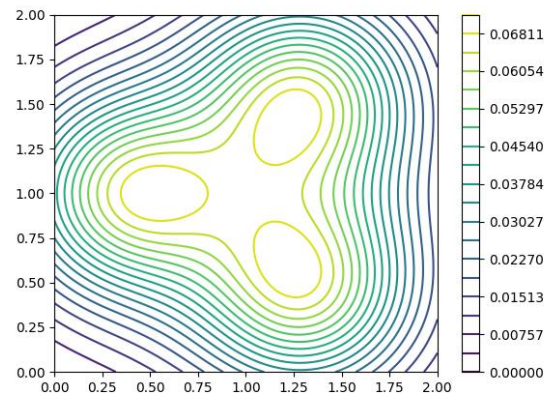
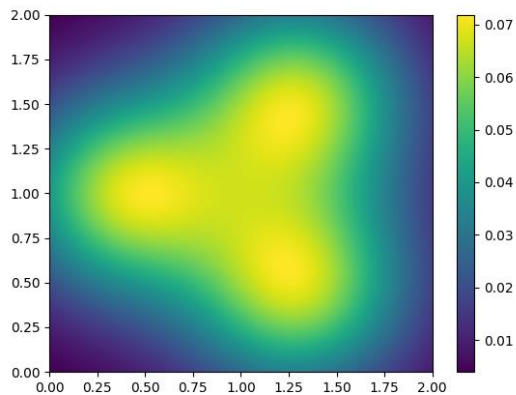
TP3b

ONIP-2 / Mini-projet - Programmation Objet

Programmation Objet

4 séances

Carte d'éclairement de sources incohérentes



Source caractérisée par leur indicatrice de rayonnement

$$I(\alpha) = I_0 \cdot \exp(-4 \cdot \ln(2)) \cdot (\alpha/\Delta)^2$$

Eclairement d'une source ponctuelle donnée par la formule de Bouguer

$$E = \frac{I \cdot \cos(\psi)}{d^2}$$

ENTREPRISE

TP1a

X

X

ENTREPRISE

TP1b

X

TP2a

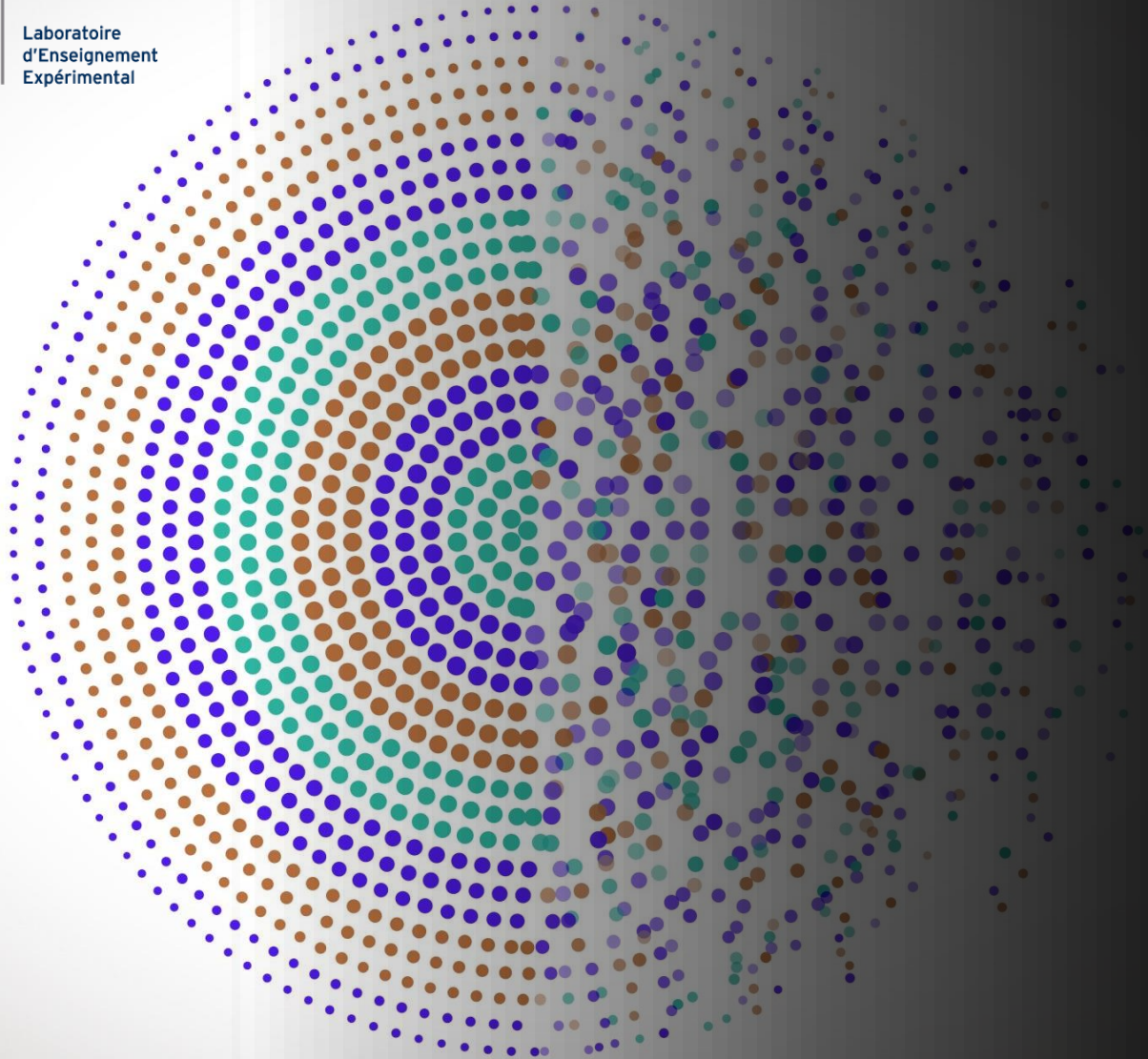
TP2b

ENTREPRISE

TP3a

TP3b

Code commenté
Validation des simulations
Figures pertinentes



Un monde d'objets

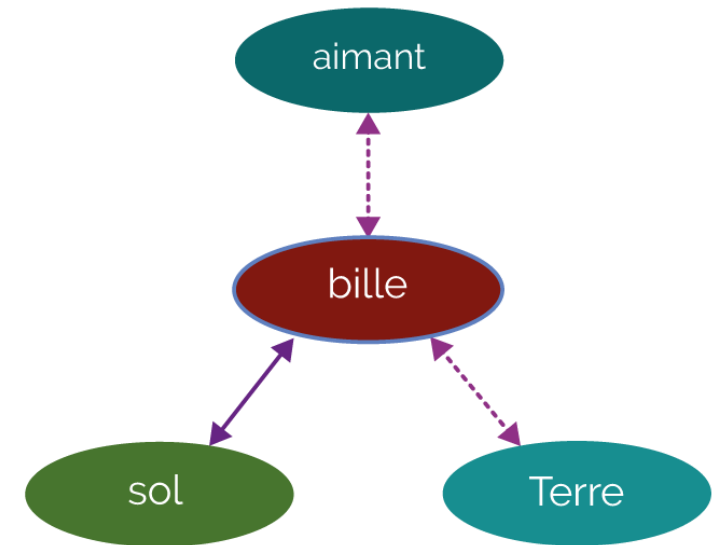
Outils Numériques / Semestre 6
/ Institut d'Optique / ONIP-2

Des objets qui interagissent

https://masevaux.fr/objets_trouves/



<https://www.lepoint.fr/dossiers/societe/velo-libre-service-velib/>



<https://www.maxicours.com/se/cours/les-diagrammes-objet-interaction/>

Des objets qui interagissent

Un objet est caractérisé par :

ETAT

COMPORTEMENT



Des objets qui interagissent

Un **objet** est caractérisé par :

ETAT

COMPORTEMENT



<https://www.lepoint.fr/dossiers/societe/velo-libre-service-velib/>

CHIEN

nom, couleur, race, poids...

manger, courir, aboyer...

TRAIN

marque, type, vitesse max...

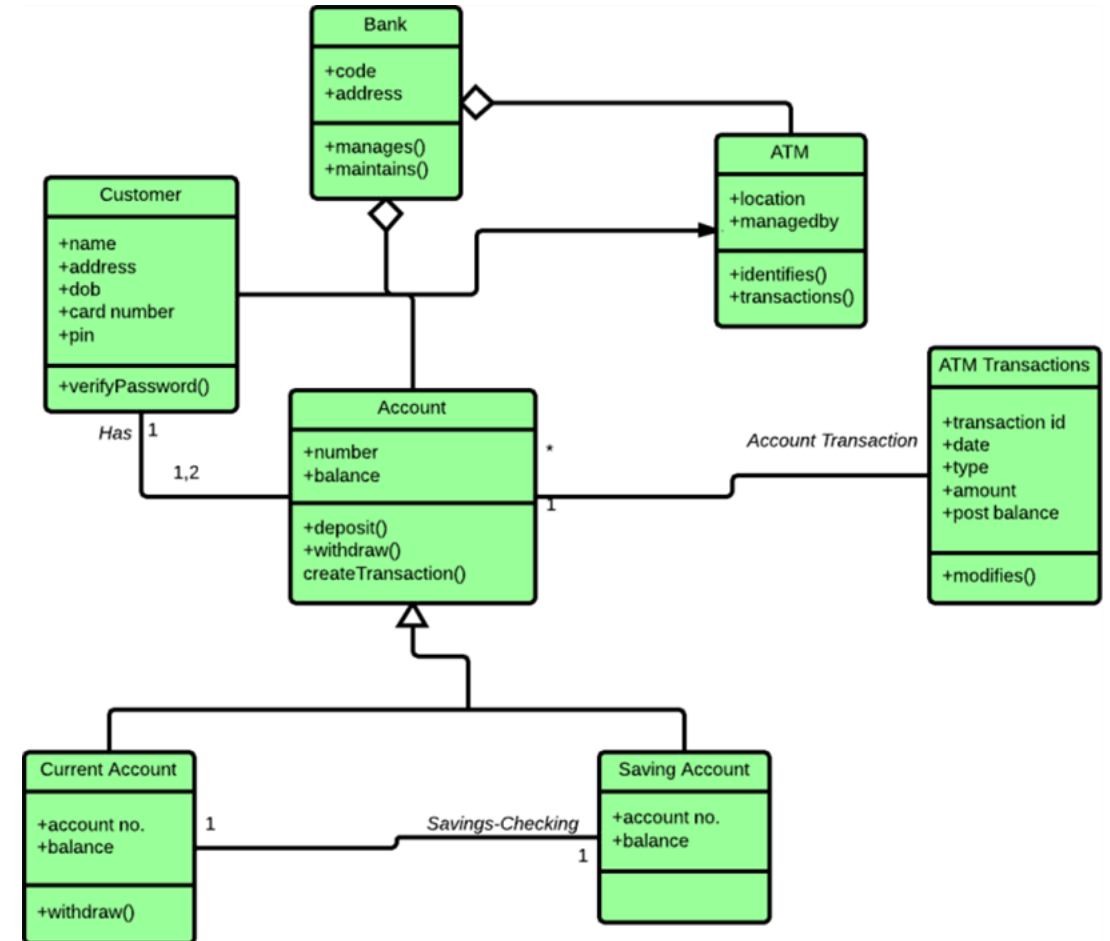
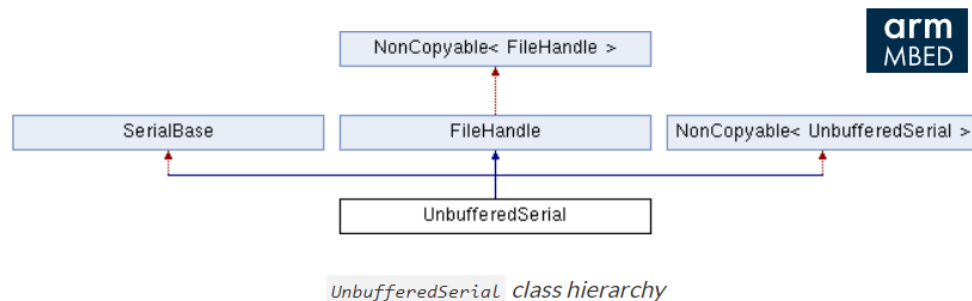
rouler, freiner, klaxonner...

Des objets en informatique

Un **objet** est une **instance** de **classe**, possédant son propre état et son propre comportement

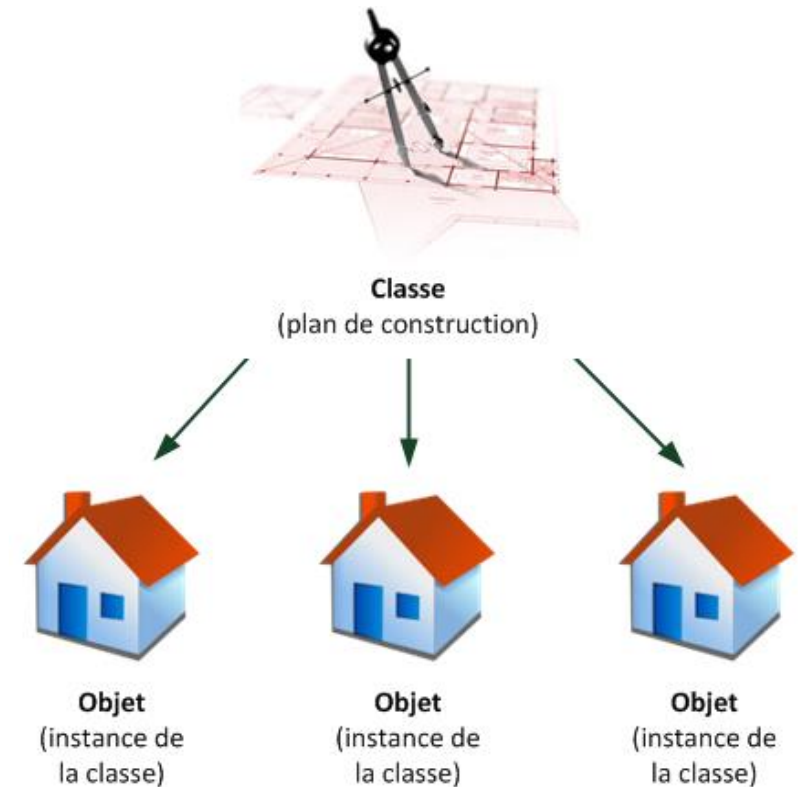
Docs › API references and tutorials › Drivers › Serial (UART) APIs › UnbufferedSerial

UnbufferedSerial



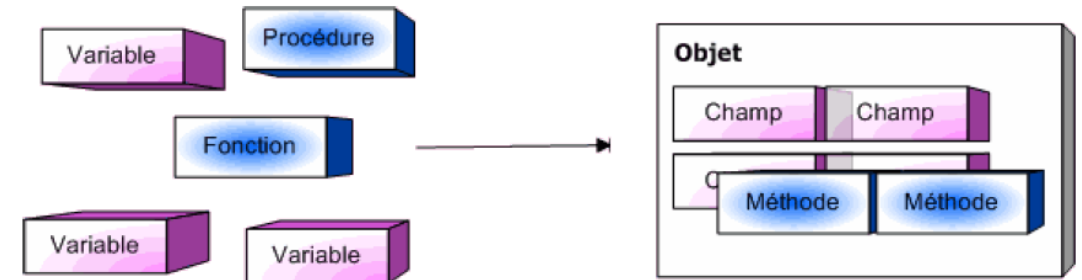
Éléments de base

- **Classe** : rassemblement de différents **attributs** (état d'un objet) et **méthodes** (actions possibles d'un objet)
- **Objet** : instance d'une classe



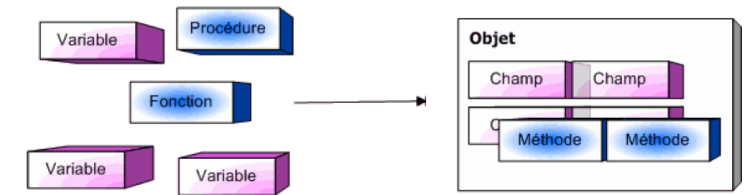
Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
(notion non abordée dans ce module)



Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
(notion non abordée dans ce module)



classe ***numpy.ndarray***

Attributs

- *shape* (Tuple d'entiers)
- *data* (buffer)

Méthodes

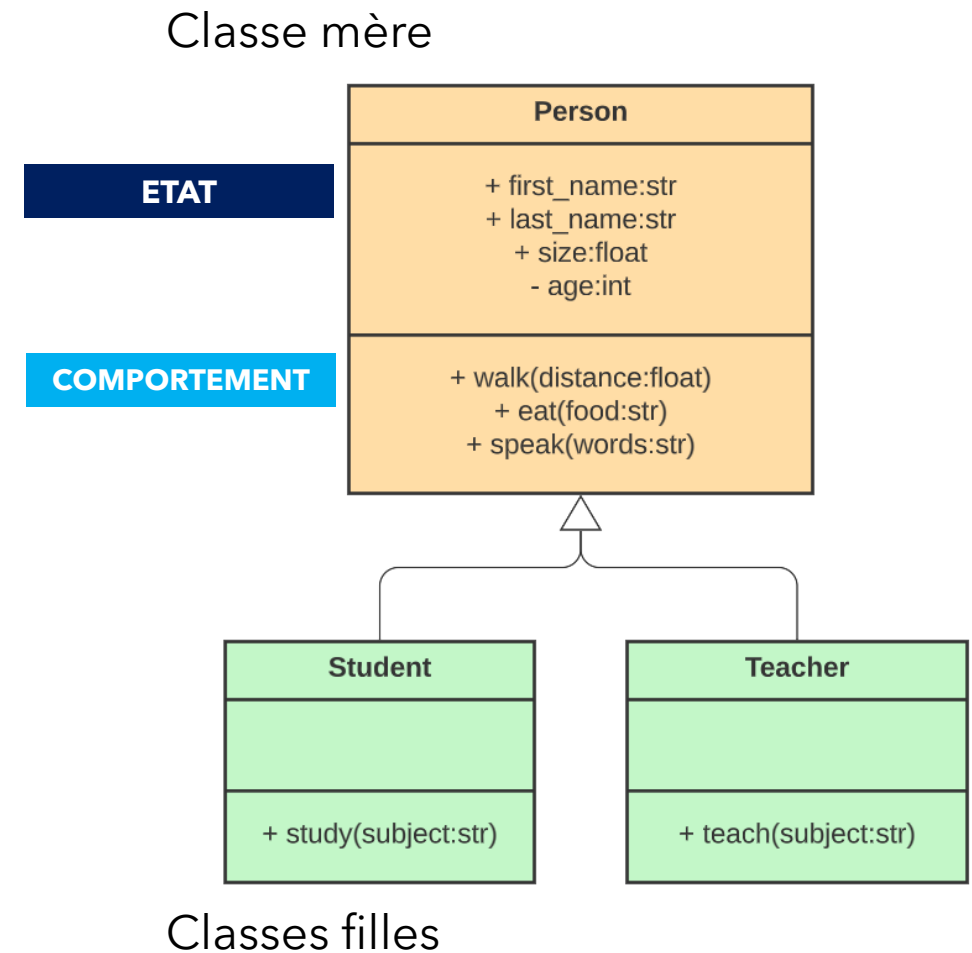
- *max* ([*axis...*])
- *resize* (*new_shape...*)

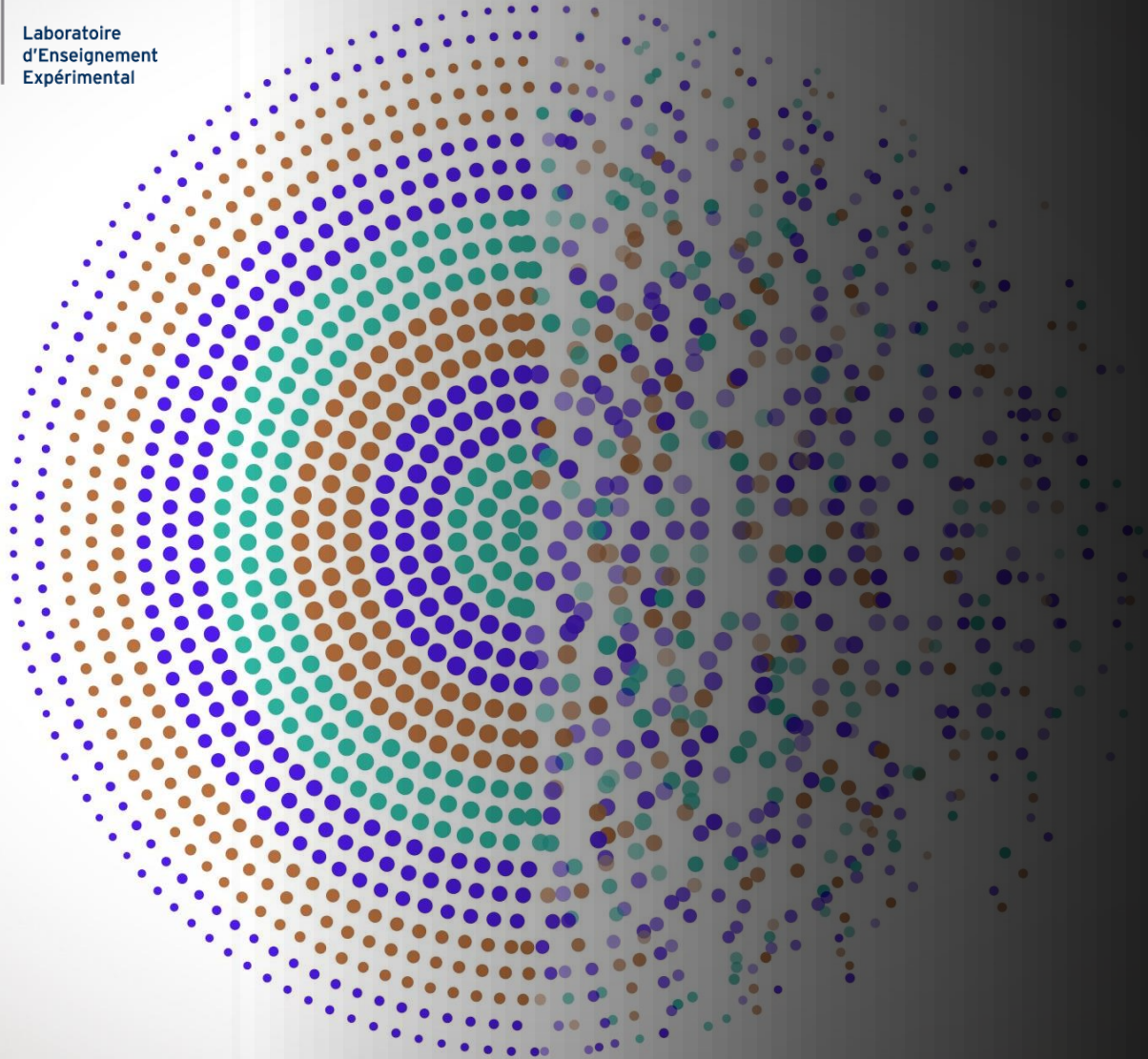
ETAT

COMPORTEMENT

Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
(notion non abordée dans ce module)





POO en Python

Outils Numériques / Semestre 6
/ Institut d'Optique / ONIP-2

Exemple d'une classe

Encapsulation : regroupement de différentes données et fonctions sous une même entité

```
import datetime

class Animal:
    """ object class Animal """
    def __init__(self, name:str, birthyear:int):
        """ Animal class constructor
        :param name: name of the animal
        :param birthyear: year of birth of the animal
        """
        self.name = name
        self.birthyear = birthyear

    def move(self):
        print(f"\t[ {self.name} ] is moving")

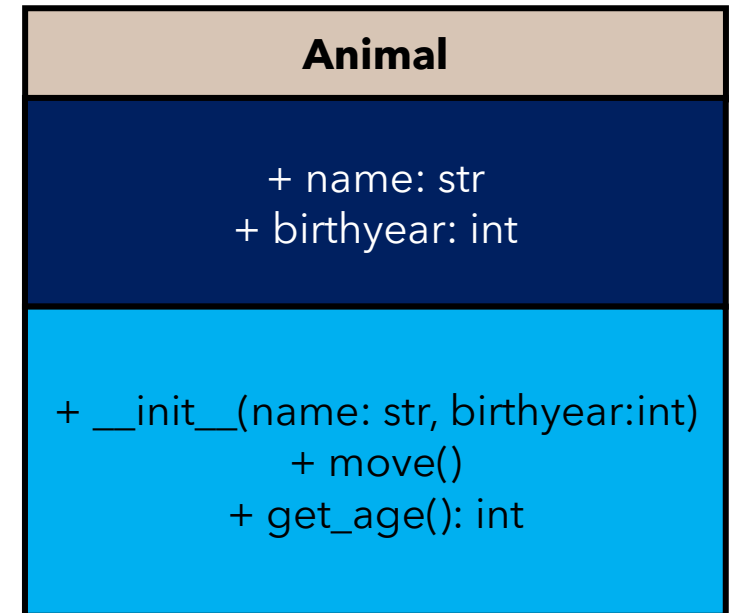
    def get_age(self) -> int:
        return datetime.date.today().year - self.birthyear
```

COMPORTEMENT

ETAT

ETAT

COMPORTEMENT



Exemple d'une classe

Encapsulation : regroupement de différentes données et fonctions sous une même entité

ETAT

variables, propres à un objet (instance d'une classe), nommées **attributs**

COMPOTEMENT

Fonctions associées à un objet (instance d'une classe), nommées **méthodes**

`__init__(self,...)` est le **constructeur** : méthode appelée à l'instanciation d'un objet - **OBLIGATOIRE !**

`move()` et **`get_age()`** sont des fonctions propres à cette classe

ETAT

Animal

+ name: str
+ birthyear: int

COMPOTEMENT

+ **`__init__(name: str, birthyear:int)`**
+ **`move()`**
+ **`get_age(): int`**

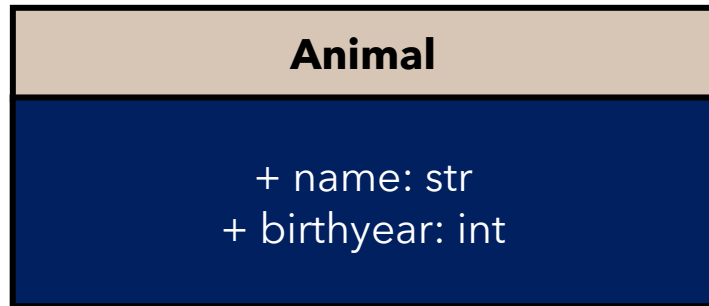
`self` est le mot clé utilisé pour **accéder aux méthodes et attributs d'instance**

Utilisation d'une classe

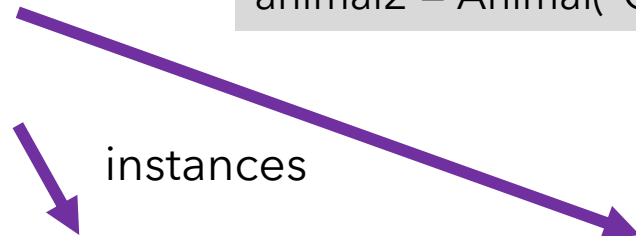
Instanciation d'un objet

```
def __init__(self, name:str, birthyear:int):
    self.name = name
    self.birthyear = birthyear
```

ETAT



```
animal1 = Animal("Felix", 2021)
animal2 = Animal("Garfield", 2015)
```

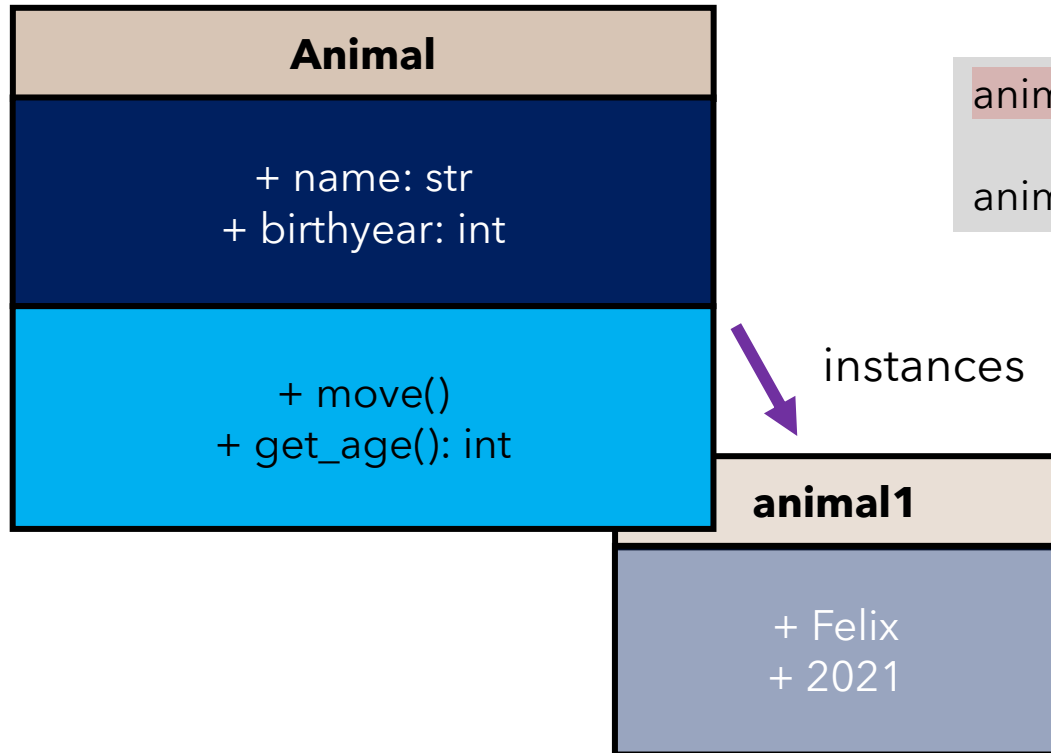


Utilisation d'une classe

Accès aux attributs d'un objet

```
def __init__(self, name:str, birthyear:int):
    self.name = name
    self.birthyear = birthyear
```

ETAT
COMPORTEMENT



```
animal1 = Animal("Felix", 2021)
animal2 = Animal("Garfield", 2015)
```

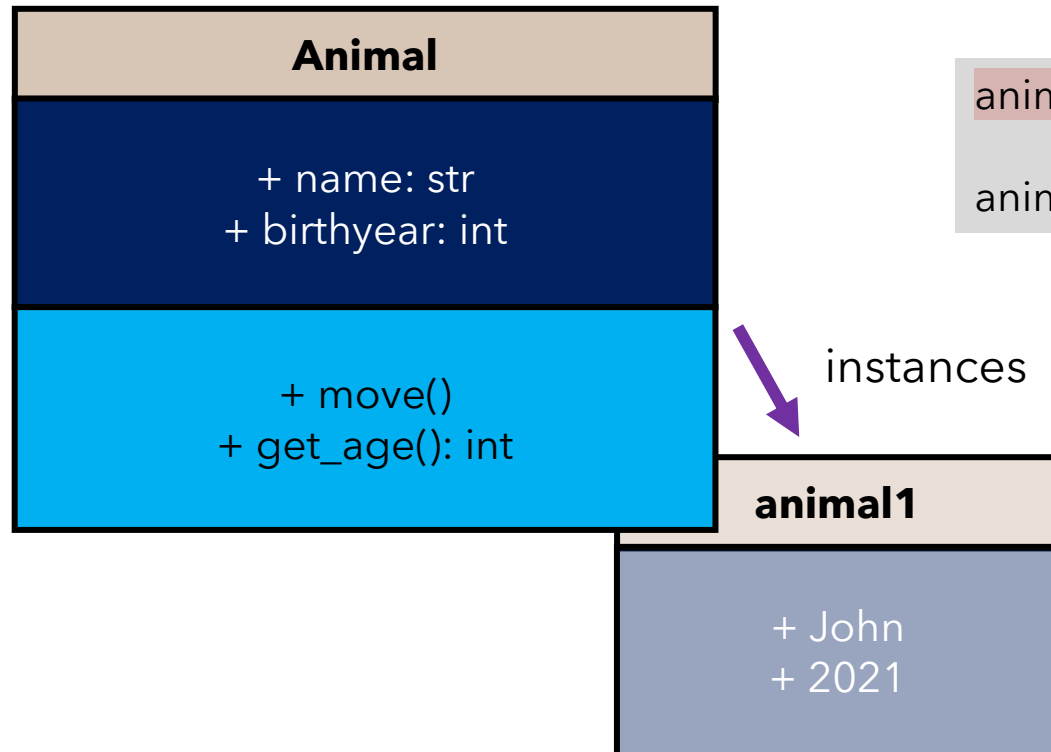
```
print("Animal 1 Name = ", animal1.name)
```

```
>>> Animal 1 Name = Felix
```

Utilisation d'une classe

Accès aux attributs d'un objet

```
def __init__(self, name:str, birthyear:int):
    self.name = name
    self.birthyear = birthyear
```



```
animal1 = Animal("Felix", 2021)
```

```
animal2 = Animal("Garfield", 2015)
```

```
print("Animal 1 Name = ", animal1.name)
```

```
>>> Animal 1 Name = Felix
```

```
animal1.name = "John"
print("Animal 1 Name = ", animal1.name)
```

```
>>> Animal 1 Name = John
```

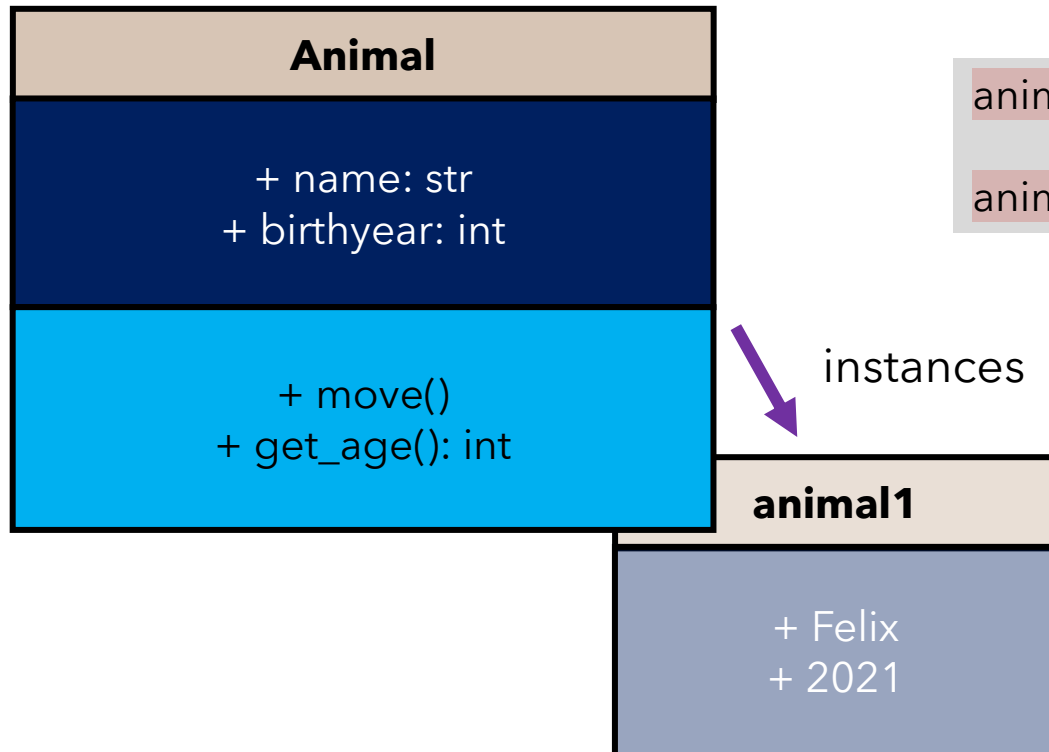
Utilisation d'une classe

Appel à une méthode à l'extérieur d'une classe

```
def move(self):
    print(f"\t[ {self.name} ] is moving")
```

ETAT

COMPORTEMENT



```
animal1 = Animal("Felix", 2021)
```

```
animal2 = Animal("Garfield", 2015)
```

```
animal1.move()
```

```
>>> [ Felix ] is moving
```

```
animal2.move()
```

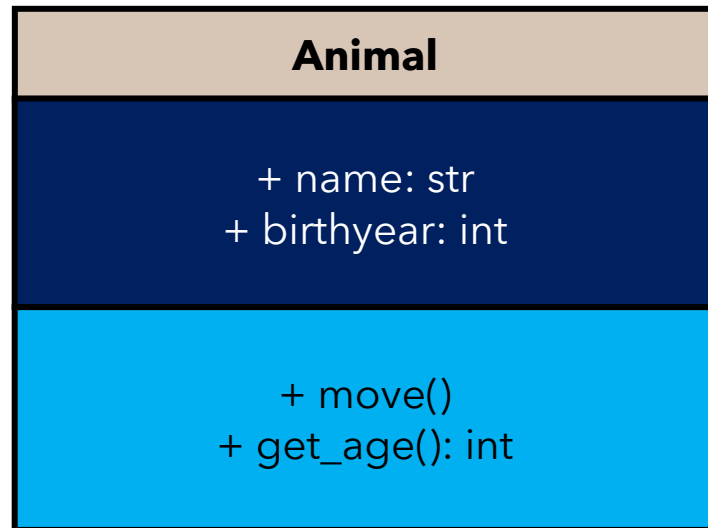
```
>>> [ Garfield ] is moving
```

Utilisation d'une classe

Liste d'objets

ETAT

COMPORTEMENT



```
animal1 = Animal("Felix", 2021)
```

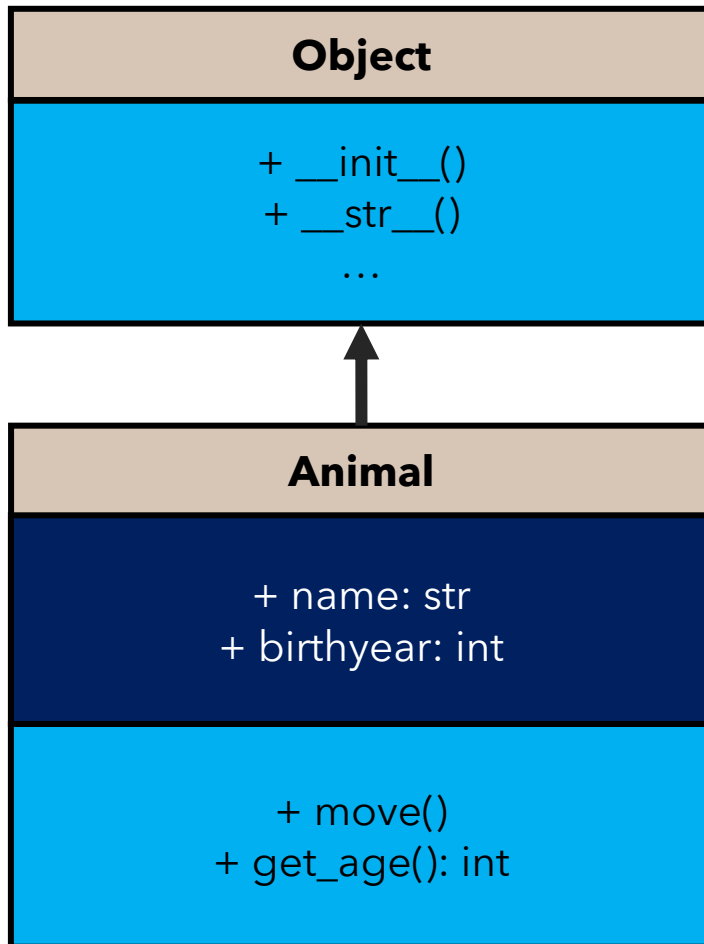
```
animal2 = Animal("Garfield", 2015)
```

```
animaux = []
animaux.append(animal1)
animaux.append(animal2)
animaux[0].move()
```

```
>>> [ Felix ] is moving
```

Objets en Python

Gestion des objets



```
animal1 = Animal("Felix", 2021)
```

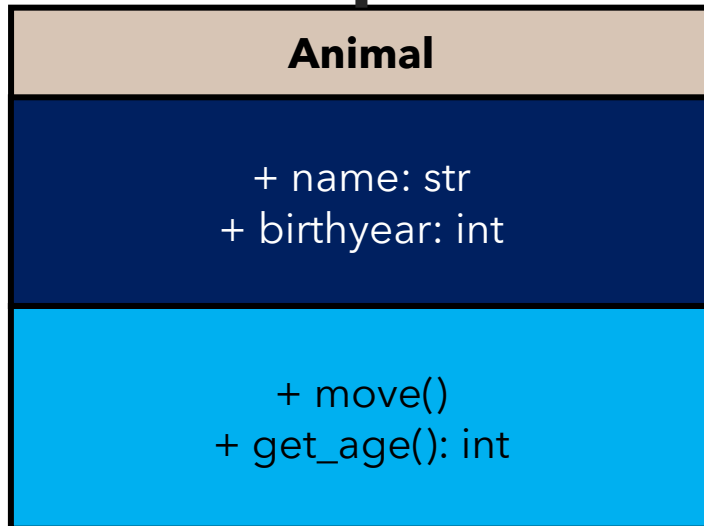
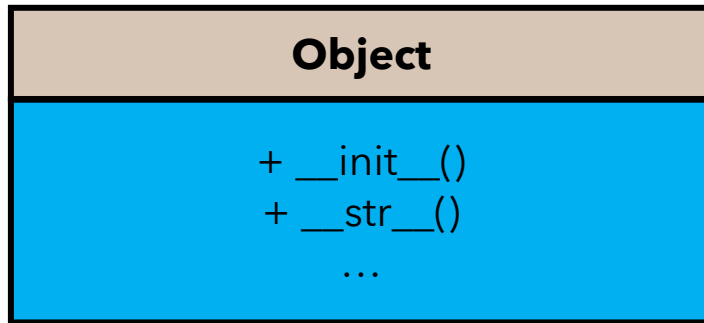
```
print(animal1)
```

```
>>> <__main__.Animal object at 0x000001E4FA066750>
```

Objets en Python

Gestion des objets / Redéfinition de fonctions

Super classe !



ETAT

COMPORTEMENT

```
def __str__(self):
    str = f"Animal [ {self.name} ] born in {self.birthyear}"
    return str
```

```
animal1 = Animal("Felix", 2021)
```

```
print(animal1)
```

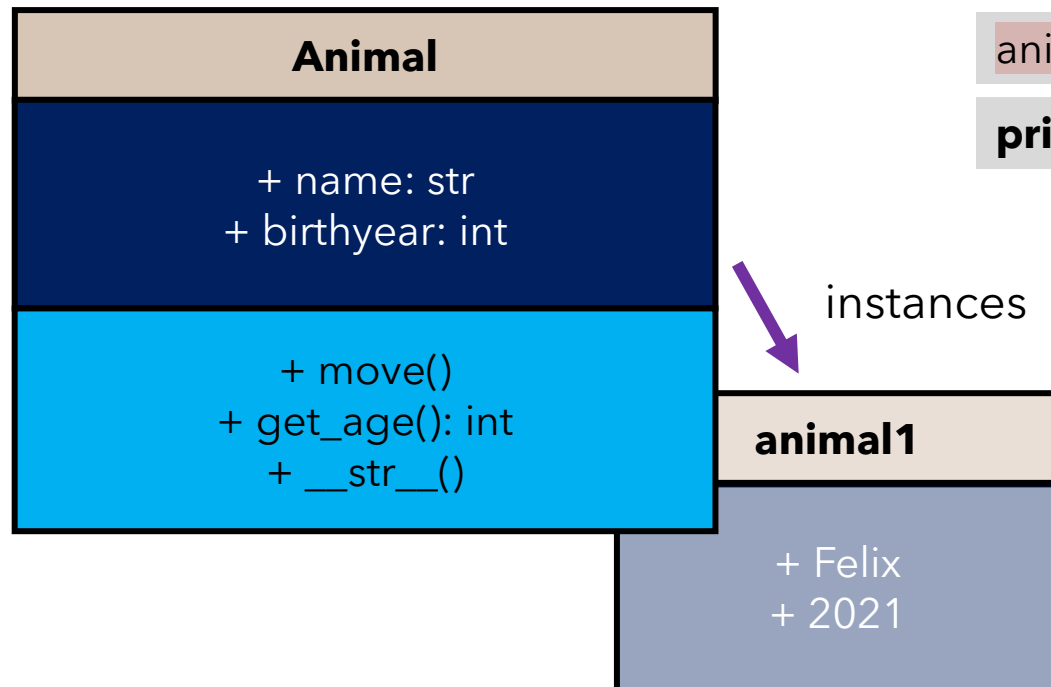
```
>>> <__main__.Animal object at 0x000001E4FA066750>
```

```
>>> Animal [ Felix ] born in 2021
```

Utilisation d'une classe

Appel à une méthode à l'intérieur d'une classe

```
def __str__(self):
    str = f"Animal [ {self.name} ] born in {self.birthyear}"
    str += f" ( {self.get_age()} yo)"
    return str
```



```
animal1 = Animal("Felix", 2021)
```

```
print(animal1)
```

```
>>> Animal [ Felix ] born in 2021 (4 yo)
```

Programmation orientée objet

Quelques règles

- Une classe possède **obligatoirement** un **constructeur** `__init__`
- Le **nom des méthodes ne doit pas commencer** par `__` (double underscore)
(signification très particulière en Python - utilisation réservée à certaines méthodes ou attributs)

The [Google Python Style Guide](#) has the following convention:

ClassName

method_name

function_name

GLOBAL_CONSTANT_NAME

global_var_name

instance_var_name

function_parameter_name

local_var_name