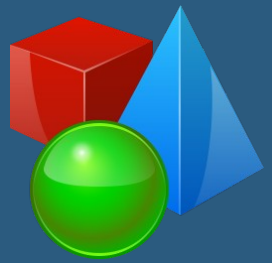
Lycée Philippe Lamour - 2021 / 2022



Spécialité NSI - Terminale

Programmation orientée objet

M. Pagès

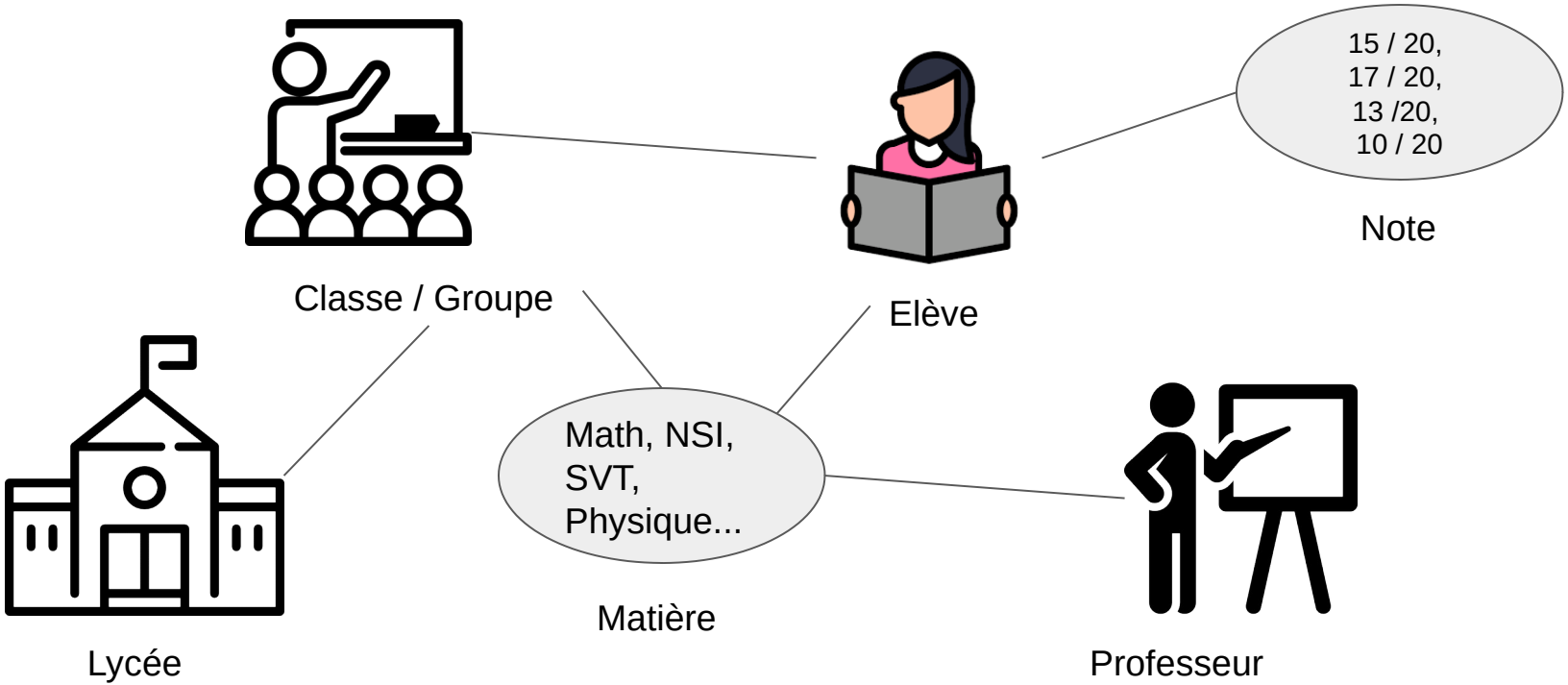
Programmation Objet

Représenter des concepts, des entités complexes du monde réel de manière informatique par une structure de données particulière.

Un des premiers avantages de ce concept est de pouvoir rapidement et facilement modéliser certains “gros” projets.

Exemple : Organiser la gestion des notes des élèves d’un lycée.

Programmation Objet



Attributs et méthodes

Exemple : Objet “Pokemon”

Attributs

Numéro
Nom
Points de vie
Points d'expérience
Force
Défense
Poids
Type

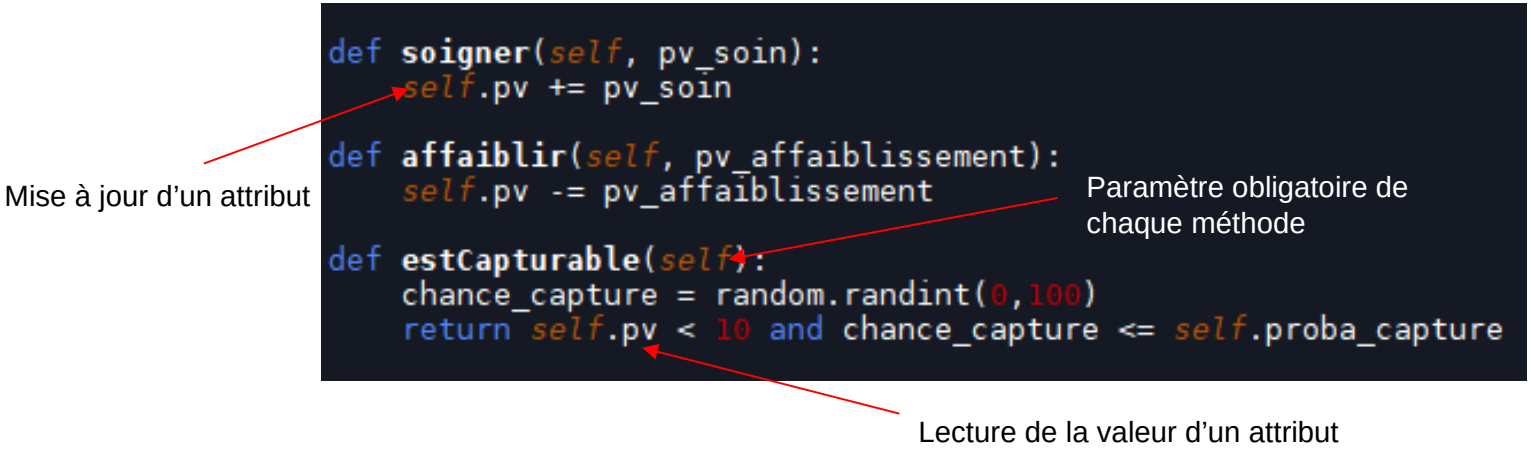
Méthodes

soigner(pv)
affaiblir(pv)
gagnerCombat()
estCapturable()
faireEvoluer()
exposerAUnePierre(pierre)

En python - Les classes

```python
import random

class Pokemon:
    def __init__(self, numero, nom, typePokemon, pv, force, poids, proba_capture):
        self.numero = numero
        self.nom = nom
        self.typePokemon = typePokemon
        self.pv = pv
        self.force = force
        self.poids = poids
        self.proba_capture = proba_capture
        self.xp = 0

    def soigner(self, pv_soin):
        self.pv += pv_soin

    def affaiblir(self, pv_affaiblissement):
        self.pv -= pv_affaiblissement

    def estCapturable(self):
        chance_capture = random.randint(0,100)
        return self.pv < 10 and chance_capture <= self.proba_capture
```

En python - Les classes - Mot clé “self”



- Désigne l’objet courant, sur lequel la méthode est appelée
- Permet d'accéder aux **attributs** et aux **méthodes** de l’objet à **l’intérieur de la classe**
- On ne l’utilise pas à l’extérieur de la classe.

En python - Les classes - Constructeur

```python
class Pokemon:
    def __init__(self, numero, nom, typePokemon, pv, force, poids, proba_capture):
        self.numero = numero
        self.nom = nom
        self.typePokemon = typePokemon
        self.pv = pv
        self.force = force
        self.poids = poids
        self.proba_capture = proba_capture
        self.xp = 0
```

```python
bublizare = Pokemon(1,"Bulbizare", "Plante",100, 10,20,30)
```

- Fonction qui va permettre de créer une **instance** de l'objet en initialisant ses **attributs**
- Certaines valeurs ne sont pas obligatoirement passées en paramètre (exemple : xp)
- Si on n'a pas besoin d'initialiser des valeurs, le constructeur n'est pas obligatoire

En python - Les classes - Méthodes

```python
def soigner(self, pv_soin):
    self.pv += pv_soin

def affaiblir(self, pv_affaiblissement):
    self.pv -= pv_affaiblissement

def estCapturable(self):
    chance_capture = random.randint(0,100)
    return self.pv < 10 and chance_capture <= self.proba_capture
```

```
In [3]: bulbizare.pv
Out[3]: 100

In [4]: bulbizare.affaiblir(50)

In [5]: bulbizare.pv
   ...:
Out[5]: 50
```

- Fonctions qui permettent d'exécuter une action ou de récupérer des données sur un objet
- L'objet est donc représenté par "**self**"
- On n'utilise pas "self" lors de **l'appel** de la méthode, car l'objet est déjà connu, c'est donc implicite (on appelle la méthode sur l'objet)

En python - Les classes - En bref

La **classe** : **MonsieurPatate**

Attributs : Bras, oreilles, chapeau, yeux, nez, chaussures, etc…

Instanciation (utilisation du **constructeur**) : Placer les accessoires sur un monsieur patate vierge.

5 **instances** de **MonsieurPatate**

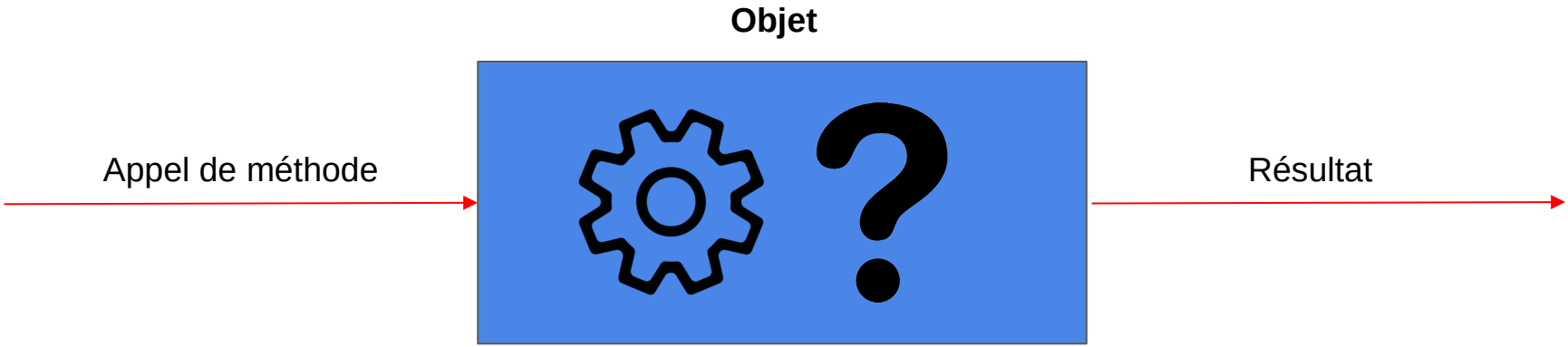
Des valeurs pour les attributs

Programmation Objet - Utilisation

```
In [2]: bulbizare = Pokemon(1, "Bulbizare","Plante",100,10,20,30)

In [3]: bulbizare.pv
Out[3]: 100

In [4]: bulbizare.affaiblir(50)

In [5]: bulbizare.pv
   ...:
Out[5]: 50

In [6]: bulbizare.affaiblir(45)

In [7]: bulbizare.pv
Out[7]: 5

In [8]: bulbizare.soigner(4)

In [9]: bulbizare.pv
Out[9]: 9
```

```
In [10]: bulbizare.estCapturable()
Out[10]: False

In [11]: bulbizare.estCapturable()
Out[11]: True

In [12]: bulbizare.soigner(20)

In [13]: bulbizare.estCapturable()
Out[13]: False

In [14]: pikachu = Pokemon(25, "Pikachu", "Electrique", 150,5,20,5)

In [15]: pikachu.typePokemon
Out[15]: 'Electrique'
```

Programmation Objet - Principe d’encapsulation



Agit comme une boîte noire

Programmation Objet - Principe d’encapsulation

Un conducteur sait comment manier sa voiture : La faire avancer, reculer, passer les vitesses, mais sait-il réellement ce qui se passe dans la voiture quand il agit sur les commandes ?

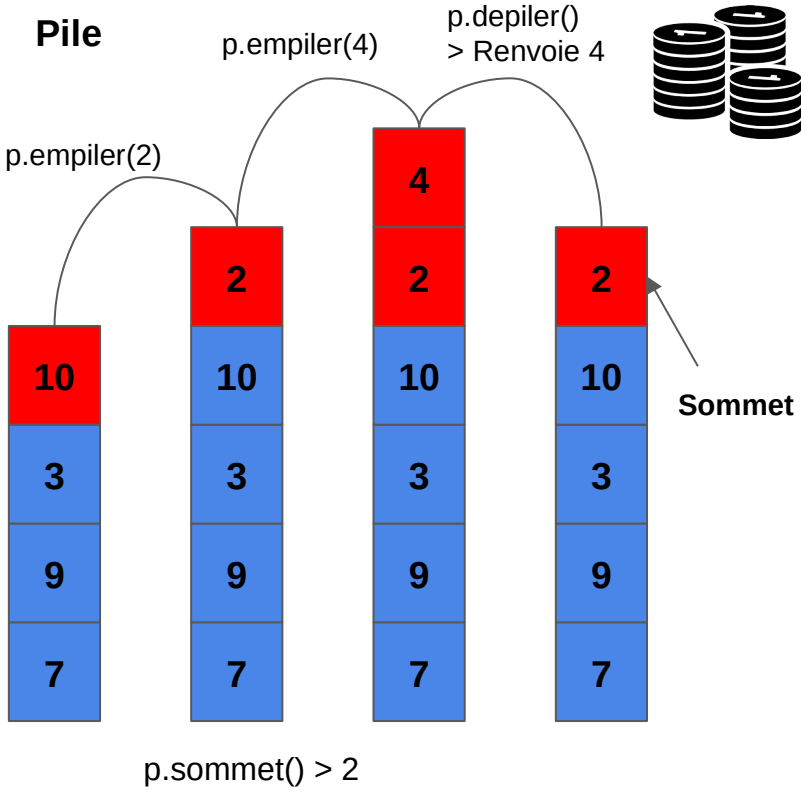
Non, ou du moins, il ne connaît pas tous les détails d'ingénierie derrière chacune de ses actions, **et il n’en a pas besoin!**

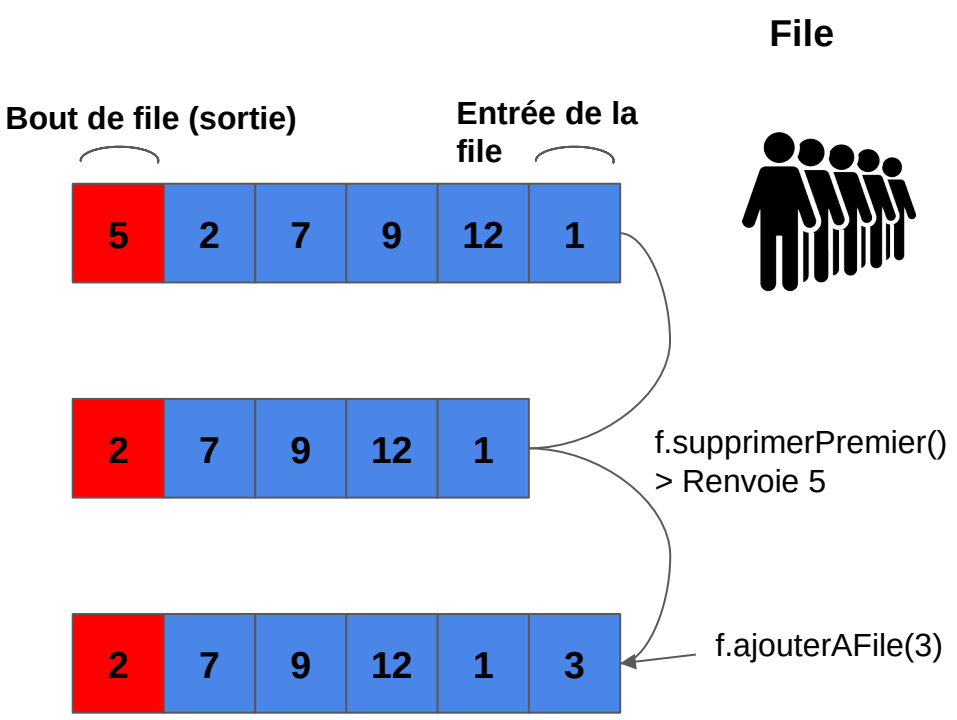
C’est le même principe pour les objets, celui qui l’utilise n’a pas besoin de connaître son fonctionnement interne, il interagit avec lui au travers d’une **interface**.

Programmation Objet - Vous en avez déjà utilisé !

```
In [19]: ma_liste = list() #Instanciation

In [20]: ma_liste.append(5) #Appel de la méthode append sur l'objet

In [21]: ma_liste.pop() #Appel de la méthode pop sur l'objet
Out[21]: 5
```

Programmation Objet - Pile et File, des objets!





TP - Programmation objet

- Découverte et création de votre première classe
- Exercices sur les objets
- Création de “Pile” et “File” en objet
- Gestion des élèves et des notes d’un lycée en objet
- 4 niveaux de difficultés :
  - * : Facile
  - ** : Intermédiaire
  - *** : Avancé
  - **** : Très avancé