Chap. 3 — Programmation Orienté Objet (pa.dilla.fr/17



1 - Classes et attributs : structurer les données

Une **classe** définit et nomme une structure de données de base du langage qui peut regrouper plusieurs composantes de natures variées

Chacune de ces composantes est appelée un **attribut** et est doté d'un nom.

Exemple

Par exemple, voici une manipulation de trois nombres entiers représentant des durées (heures, minutes, secondes).

On décide d'appeler la classe \mathtt{Chrono} et de la munir de trois attributs :

heures, minutes et secondes.

Voici alors comment on pourrait figurer le temps 21 heures, 34 minutes

: səpuosəs çç tə



1.1 - Description d'une classe

Voici comment définir cette structure sous la forme d'une classe :

Exemple

```
[]: class Chrono:
"""

Une classe pour représenter un temps mesuré

en heures, minutes et secondes."""

def __init__(self, h, m, s):
    self.heures = h

self.minutes = m
    self.secondes = s
```

Pour définir une nouvelle classe, on utilise :

- 1. le mot-clé class
- 2. suivi du nom choisi pour la classe et
- 3. suivi par les deux-points :.

Tout le reste de la définition est alors en retrait (indentation).

Par convention, le nom de la classe doit commencer par une **lettre majus-** cule.

Suivent alors

- une documentation puis
- la définition d'une fonction __init__ possédant
 - comme premier paramètre self puis ensuite
 - les trois paramètres correspondants aux trois composantes d'un objet admettant possédant la structure de chrono.

Les instructions de la forme self.xxx = correspondent aux affectations des valeurs aux trois attributs de la classe.

1.2 — Création d'un objet

Une fois la classe définie, un élément correspondant à la structure Chrono peut être construit avec une expression de la forme Chrono(h, m, s).

septembre 2021

On appelle un tel élément un objet ou une instance de la classe Chrono.

Exemple

X STRUCT.DE DONNÉES

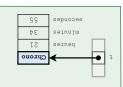
temps "21 heures, 34 mintes et 55 secondes" on écrit : Ainsi, pour définir et affecter à la variable t un objet représentant notre

t = Chrono(21, 34, 55)

REMARQUE

a été alloué à cet objet. à strictement parler l'objet mais un pointeur vers le bloc de mémoire qui On remarque que, comme pour les tableaux, la variable t ne contient pas

La situation correspond donc au schéma suivant :



1.3 — Manipulation des attributs

t. a où a désigne le nom de l'attribut visé. On peut accéder aux attributs d'un objet t de la classe Chrono avec la notation

Tout comme les cases d'un tableau, les attributs d'un objets sont mutables : on

peut les consulter **et** les modifier.





__eq___) et clone en tenant compte de ce changement d'implémenta-Redefinir les méthodes avance, texte (ou __str__), egale (ou

les implémentations de texte et clone. m, s) correspondant. Utiliser cette méthode auxiliaire pour simplifier Ajouter une méthode _conversion qui extrait d'un temps le triplet (h,



classe. auxiliaires et les autres méthodes forment l'interface des objets de cette Les méthodes possédant un nom commençant par _ sont des méthodes

rinətər A

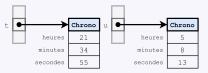
pant dans une même entité des données et le code manipulant ces don-La programmation orientée objet structure les programmes en regrou-

possèdent ses instances. classe, la classe définissant l'ensemble des attributs et méthodes que particulières appelées méthodes. Chaque objet est une instance d'une contenir plusieurs données sous la forme d'attributs à l'aide de fonctions Dans ce paradigme de programmation, on manipule des **objets** pouvant .səən On parle bien d'attribut d'un objet car chaque objet possède pour ses attributs des valeurs qui lui sont propres. On parle alors aussi d'attribut d'instance.

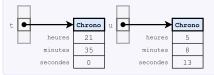
Exemple

Ainsi, chaque objet de la classe Chrono possède trois attributs dont les valeurs sont indépendantes des valeurs des attributs (de même nom) des autres instances.

Les définitions t = Chrono(21, 34, 55) et u = Chrono(5, 8, 13) conduisent donc à la situation suivante :



Une avancée de cinq secondes du chronomètre ${\tt t}$ mènerait ainsi à la situation suivante :



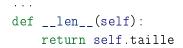
Une classe peut également définir des **attributs de classe**, dont la valeur est attachée à la classe elle même.

Exemple

Ainsi:

class Chrono:

 $heure_max = 24$



REMARQUE

Plusieurs remarques:

- L'appel à la fonction cree() est remplacée par un appel au constructeur Ensemble().
- Les appels de fonctions contient(s,x) et ajoute(s,x) sont transformés en appels de méthodes s.contient(x) et s.ajoute(x).

3 - Retour sur l'encapsulation

Dans la philosophie objet, l'interaction avec les objets se fait essentiellement **avec les méthodes**. Les attributs sont considérés par défaut comme relevant du détail d'implémentation.

Ainsi, pour la classe Chrono, il est essentiel de savoir qu'on peut afficher et faire évoluer les temps, mais l'existance des trois attributs heures, minutes et secondes est anecdotique (et peut être cachée à l'utilisateur).

Exemple

Par exemple, on peut simplifier la définition de la classe en ne définissant qu'un unique attribut $_{ temps}$ mesurant le temps en seconde.

```
class Chrono:
   def __init__(self, h, m, s):
      self._temps = 3600*h + 60*m + s
```

3 - E0.0

septembre 2021

Chrono.heure_max. tance avec theure_max ou depuis la classe elle même avec On peut consulter de tels attributs depuis n'importe quelle ins-

présentes et futures : même pour que la modification soit perseptible par toutes les instances On peut modifier cet attribut de classe en y accédant via la classe elle

Chrono.heure_max = 12

2 – Méthodes : manipuler les données

accéder librement à la totalité de son contenu. à la notion d'encapsulation : un programme manipulant un objet n'est pas censé Dans le paradigme de la programmation objet, la notion de classe est associée

Lutilisateur n'a pas à savoir ou à accéder aux détails d'implémentation.

les **méthodes** de cette classe. de fonctions dédiées qui font partie de la définition de la calsse et sont appelée La manipulation de l'objet passe donc de préférence par une interface constituée

2.1 - Utilisation d'un méthode

appel de méthode s'applique avant tout à un objet de la classe concerné. si les méthodes sont des fonctions qui peuvent recevoir des paramètres, chaque Les méthodes d'une classe servent à manipuler les objets de cette classe. Même

Exemple

une chaîne de caractères décrivant le temps représenté par t est réalisé L'appel d'une méthode texte s'appliquant au chronomètre t et renvoyant

```
at OxiOd8ac198>
adresse mémoire) comme par exemple <__main__.Chrono object
Une instruction Chrono.max(t,u) renvoie donc l'objet (sa classe et son
                         :sətunim.St < sətunim.1t lilə
                                           return t2
                           elif t2.heures > t1.heures:
```

3 - E0.0

2.4 - Une classe pour les ensembles

```
return False
               s.ajoute(x)
           return True
         :(x) tneitnos. z li
                s = Ensemble()
          def contient_doublon(t):
  self.dates[x] = True
      1 =+ ellist.lles
  :(x) insitance. Iles for li
          def ajoute(self, x):
      [x]səfsb. lies mrutər
        def contient(self, x):
88 * [sals] = setsb.llea
           0 = ellist.llea
           def __init__ tell):
                   []: cjssa Ensempje:
                    Exemple
```

Modularité sous la forme d'une classe. Le programme ci-dessus donne une adaptation du programme du chapitre 2-

tion simple d'une méthode __len__ moriser la taille de l'ensemble de dates. Cela permettrait par exemple une défini-Puisqu'une classe peut regrouper plusieurs données, nous en profitons pour mé-

```
cjasa Ensemble:
```

par l'instruction t.texte() et elle pourra renvoyer la chaîne de caractère '21h 34h 55s'.

Cette notation pour l'appel de méthode est la même notation pointée que l'accès aux attributs de t. **Mais** la paire de parenthèse fait bien apparaître une méthode, comme pour une fonction sans paramètre.

Lorsqu'un méthode dépend d'autres paramètres que cet objet principal t, ces autres paramètres apparaissent de la manière habituelle.

Exemple

Par exemple, s'il existe une méthode avance faisant avancer le chronomètre t d'un *certain* nombre de secondes passé en paramètre, on écrira pour avancer le chronomètre de 5 secondes :

t.avance(5)

Ainsi un nouvel appel à t.texte() renverra cette fois-ci '21h 35m 0s'.

REMARQUE

Comme le montre l'exemple, on ne manipule pas directement les attributs d'un objet. On utilise pour cela des méthodes ce qui préserve l'encapsulation du code.

Exemple

On a déjà rencontré des notations de ce type comme par exemple tab.append(42) pour ajouter le nombre 42 au tableau tab.



Par exemple les attributs $\cdot x$ ou $\cdot y$ peuvent être utilisés dans plusieurs classes différentes sans risque de confusion.

Méthodes de classe Il existe des attributs de classe et il existe aussi des **méthodes de classe**, dont la valeur ne dépend pas des instances mais est partagée au niveau de la classe entière.

Ces méthodes de classes sont appelées en programmation objet des **méthodes statiques**.

Exemple

Par exemple:

(1) méthode pertinente pour réaliser des fonctions auxilières ne travaillant pas directement sur les objets de la classe

```
def est_seconde_valide(s):
    return 0 <= s and s < 60</pre>
```

Une instruction Chrono.est_seconde_valide(64) renvoie donc False.

Exemple

(2) opérations d'appliquant à plusieurs instances aux rôles symétriques et dont aucune n'est modifiée (pas d'effets de bords) :

```
def max(t1, t2):
    if t1.heures > t2.heures:
        return t1
```

(nombre, chaîne de caractère, tableau, etc.) que des objets. Les paramètres des méthodes peuvent être aussi bien des valeurs de base

Exemple

X STRUCT.DE DONNÉES

sulvantes:

Par exemple si pour la classe Chrono on admet l'existence des méthodes

- temps représentés - egale s'appliquant à deux chronomètres pour tester l'égalité des
- chronomètre initialisé au même temps que t — clone s'appliquant à un chronomètre t et renvoyant un nouveau

On pourra alors écrire l'instruction suivante

t.egale(u) () enologing = t

qui nous renverra True.

False. Puis après t. avance(3), l'instruction t. egale(u) nous renverra alors

2.2 — Définition d'une méthode

fation self. a

classe. Une méthode ne peut donc pas avoir zéro paramètre. sauf qu'elle doit avoir obligatoirement pour premier paramètre un objet de la une fonction ordinaire, pouvant dépendre d'un nombre de paramètres arbitraire Comme nous venons de le voir, une méthode d'une classe peut être vue comme

Comme ce paramètre est un objet, on pourra accéder à ses attributs avec la nofinition d'un fonction. Le premier paramètre est systématiquement appelé self. La définition d'une méthode de classe se fait avec la même notation que la dé-

> → ab fnamálá [i]t .lfee)__metiteg__ renvoie le i-ième la collection t seulement si x est dans (X __contains__(self, x in t renvoie True si et taille de t entier définissant la (llas)__nal__ renvoie un nombre len(t) appel təffet méthode

Exemple

syntaxe allègèe tement au rôle de la méthode __str__... mais ne bénéficie pas de la Par exemple, la méthode texte de la classe Chrono correspond exac-

même adresse mémoire. ne renvoie True que lorsqu'elle est appliquée deux fois au même objet, ayant la Egalité entre deux objets Par défaut, la comparaison avec == entre deux objets

définir la méthode spéciale __eq__(self, obj). Pour que la comparaison s'effectue en fonction de la valeur des attributs, il faut

REMARQUE

des autres en ce qui concerne le nommage des variables et des autres classe définit un espace de noms, c'est-à-dire une zone mémoire séparée des attributs de même nom. Il n'y a aucun conflit dans ce cas là car une On remarque que deux classes différents peuvent sans problème définir

Exemple

Ainsi, les fonctions texte et avance de la classe Chrono peuvent être implémentée de la façon suivante :

2.3 - Constructeur

La construction d'un nouvel objet avec une expression comme Chrono (21, 34, 55) déclenche deux choses :

- la création de l'objet lui même
- l'appel à une méthode spéciale chargée d'initialiser les valeurs des attributs. Cette méthode, appelée constructeur, est définie par le programmeur. En Python, il s'agit de la méthode __init__ que nous avons pu observer dans le premier exemple.

La définition de la méthode spéciale $__{init}$ _ ne se distingue pas des autres méthodes ordinaires : son premier paramètre est self et représente l'objet auquel elle s'applique. Les autres paramètres sont donnés explicitement lors de la construction.

2.4 – Autre méthodes particulières en Python

Il existe en Python d'autres méthodes particulières :

méthode	appel	effet
str(self)	str(t)	renvoie une chaîne de caractère décrivant t
lt(self, u)	t < u	revoie True si t est strictement plus petit que True donne un code de hachage pour t, par exemple pour l'utiliser comme clé d'un dictionnaire d
hash(self)	hash(t)	