POO

Programmation Orientée Objet

M. Tellene

Programmation Orientée Objet

La POO c'est quoi ?

Programmation Orientée Objet

La POO c'est quoi ?

C'est un paradigme de programmation intégré à Python (et à d'autres langages)

Celui-ci fournit une notion de **classe**, qui permet à la fois de définir (et nommer) des structures de données, et de structurer le code d'un programme

Programmation Orientée Objet

Paradigme de programmation ?

Un paradigme de programmation est une façon d'approcher la programmation informatique et de formuler les solutions aux problèmes et leur formalisation dans un langage de programmation approprié

Programmation Orientée Objet

Paradigme de programmation ?

Un paradigme de programmation est une façon d'approcher la programmation informatique et de formuler les solutions aux problèmes et leur formalisation dans un langage de programmation approprié

Vous en connaissez déjà !

Programmation Orientée Objet

Paradigme de programmation ?

Un paradigme de programmation est une façon d'approcher la programmation informatique et de formuler les solutions aux problèmes et leur formalisation dans un langage de programmation approprié

Vous en connaissez déjà !

$\rightarrow$ la programmation impérative : une suite d'instruction

$\rightarrow$ la programmation événementielle : réagir à des événements

Programmation Orientée Objet

L'élément principal de la programmation orientée objet : **les classes**

Programmation Orientée Objet

L'élément principal de la programmation orientée objet : **les classes**

Une classe définit et nomme une structure de données qui vient s'ajouter aux structures de base du langage

La structure définie par une classe peut regrouper plusieurs composantes de natures variées

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

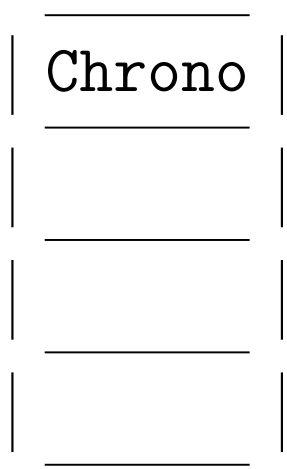
Programmation Orientée Objet

Un exemple : supposons que l'on souhaite manipuler des triplets d'entiers représentant des temps mesurés en heure, minutes et secondes

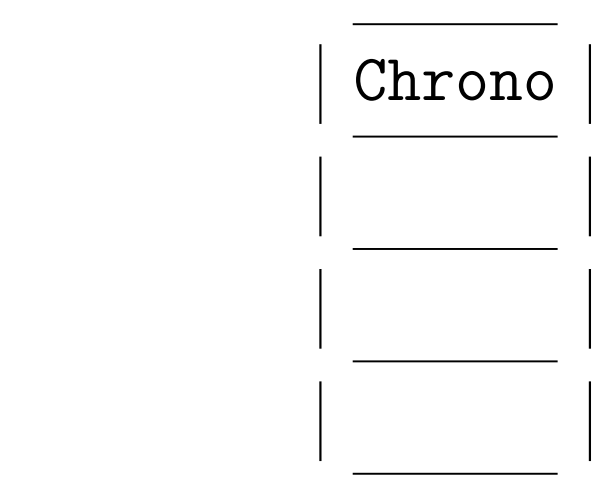
On appellera cette composante `Chrono`

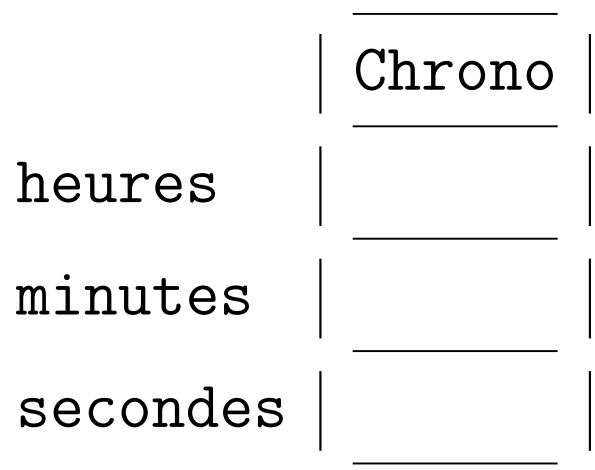
Les trois attributs de `Chrono` seront : `heures`, `minutes` et `secondes`

Programmation Orientée Objet



Programmation Orientée Objet





Programmation Orientée Objet

Qu'est-ce qu'une classe en Python ?

Programmation Orientée Objet

Qu'est-ce qu'une classe en Python ?

Une classe doit avoir un nom : introduit par le mot-clé `class`

Une classe possède un ensemble de fonctions appelées méthodes

La méthode `__init__()` est appelé constructeur

Pour indiquer qu'une méthode appartient à une classe, le premier paramètre de celle-ci doit être `self`

Les attributs d'une classe doivent être précédés du mot-clé `self`

Programmation Orientée Objet

Qu'est-ce qu'une classe en Python ?

Programmation Orientée Objet

Qu'est-ce qu'une classe en Python ?

```
1 class Chrono:
2
3     def __init__(self, h, m, s):
4         self.heures = h
5         self.minutes = m
6         self.secondes = s
```

Programmation Orientée Objet

Qu'est-ce qu'une classe en Python ?

```
1 class Chrono:
2
3     def __init__(self, h, m, s):
4         self.heures = h
5         self.minutes = m
6         self.secondes = s
```

Écrire une classe `Rectangle`, permettant de construire un rectangle doté d'une `longueur` et d'une `largeur`

Programmation Orientée Objet

Comment utiliser les classes ?

$\rightarrow$ en créant des **objets**

Un objet est instance d'une classe et possède :

- un type (la classe dont il est une instance)
- un ensemble de variables appelées attributs d'instance

Programmation Orientée Objet

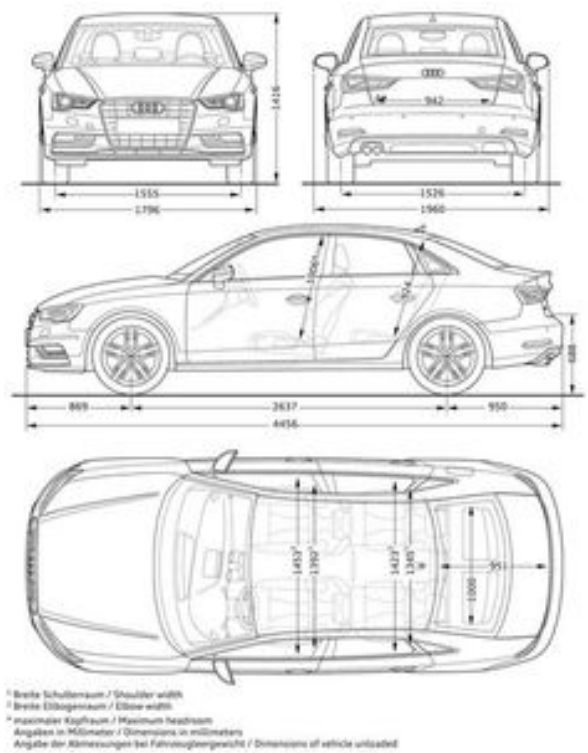


Figure – Classe

Figure – Objet

Programmation Orientée Objet

Comment créer un objet en Python ?

Programmation Orientée Objet

Comment créer un objet en Python ?

```
class Chrono:

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

t = Chrono(21, 24, 55)
```

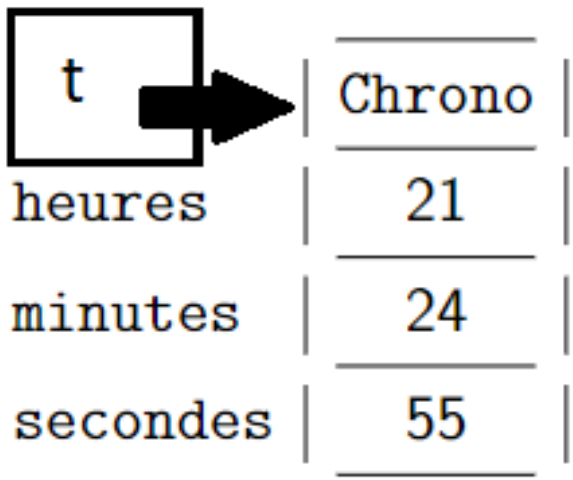
Programmation Orientée Objet

Comment créer un objet en Python ?

```
class Chrono:

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

t = Chrono(21, 24, 55)
```



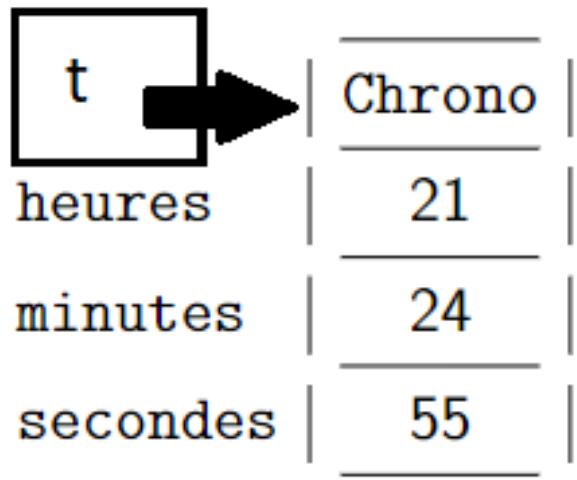
Programmation Orientée Objet

Comment créer un objet en Python ?

```
class Chrono:

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

t = Chrono(21, 24, 55)
```



Créer un objet de la classe `Rectangle` ayant pour `longueur` 12 et pour `largeur` 9

Programmation Orientée Objet

Savoir créer un objet, c'est bien, savoir l'utiliser, c'est mieux !

Programmation Orientée Objet

Savoir créer un objet, c'est bien, savoir l'utiliser, c'est mieux !

L'un des aspects non négligeables de la programmation orientée objet est la manipulation des attributs

En quoi cela consiste ?

```
1 class Chrono:
2
3     def __init__(self, h, m, s):
4         self.heures = h
5         self.minutes = m
6         self.secondes = s
7
8 t = Chrono(21, 24, 55)
```

Programmation Orientée Objet

Savoir créer un objet, c'est bien, savoir l'utiliser, c'est mieux !

L'un des aspects non négligeables de la programmation orientée objet est la manipulation des attributs

En quoi cela consiste ?

```
1 class Chrono:
2
3     def __init__(self, h, m, s):
4         self.heures = h
5         self.minutes = m
6         self.secondes = s
7
8 t = Chrono(21, 24, 55)
```

Comment accéder aux attributs de t ?

Programmation Orientée Objet

Pour accéder à l'attribut `heures` de `t`

Programmation Orientée Objet

Pour accéder à l'attribut `heures` de `t`

$\rightarrow$ `t.heures`

Programmation Orientée Objet

Pour accéder à l'attribut `heures` de `t`

$\rightarrow$ `t.heures`

Pour accéder à l'attribut `minutes` de `t`

$\rightarrow$ `t.minutes`

Pour accéder à l'attribut `secondes` de `t`

$\rightarrow$ `t.secondes`

Programmation Orientée Objet

Est-il possible de faire ça ?

```
class Chrono:

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

t = Chrono(21, 24, 55)
u = Chrono(5, 8, 13)
```

Programmation Orientée Objet

Est-il possible de faire ça ?

```
class Chrono:

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

t = Chrono(21, 24, 55)
u = Chrono(5, 8, 13)
```

Créer un deuxième objet de la classe `Rectangle` ayant pour `longueur` 1 et ayant la même `largeur` que le premier objet créé

Programmation Orientée Objet

Dernier élément des classes : les méthodes

Une méthode est une fonction propre à la classe à laquelle elle appartient

Programmation Orientée Objet

Comment définir une méthode dans une classe ?

Programmation Orientée Objet

Comment définir une méthode dans une classe ?

```
1 class Chrono:
2
3     def __init__(self, h, m, s):
4         self.heures = h
5         self.minutes = m
6         self.secondes = s
7
8     def augmenter_seconde(self, sec):
9         self.secondes = self.secondes + sec
```

Programmation Orientée Objet

Comment définir une méthode dans une classe ?

```
class Chrono:

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

    def augmenter_seconde(self, sec):
        self.secondes = self.secondes + sec
```

Créer une méthode `perimetre` qui renvoie le périmètre du rectangle

Créer une méthode `aire` qui renvoie le périmètre du rectangle

Programmation Orientée Objet

Comment appeler une méthode dans son code ?

Programmation Orientée Objet

Comment appeler une méthode dans son code ?

```
1  class Chrono:
2
3      def __init__(self, h, m, s):
4          self.heures = h
5          self.minutes = m
6          self.secondes = s
7
8      def augmenter_seconde(self, sec):
9          self.secondes = self.secondes + sec
10
11 t = Chrono(21, 24, 55)
12 t.augmenter_seconde(1)
13 print(t.secondes)
14 >>> 56
```

Programmation Orientée Objet

Comment appeler une méthode dans son code ?

```
1  class Chrono:
2
3      def __init__(self, h, m, s):
4          self.heures = h
5          self.minutes = m
6          self.secondes = s
7
8      def augmenter_seconde(self, sec):
9          self.secondes = self.secondes + sec
10
11 t = Chrono(21, 24, 55)
12 t.augmenter_seconde(1)
13 print(t.secondes)
14 >>> 56
```

Vous ne voyez pas une petite erreur arriver ?

Programmation Orientée Objet

Si l'on écrit :

```
1 t = Chrono(21, 24, 55)
2 t.augmenter_seconde(10)
3 print(t.secondes)
4 >>> 65
```

Comment régler ce problème ?

Programmation Orientée Objet

Si l'on écrit :

```
t = Chrono(21, 24, 55)
t.augmenter_seconde(10)
print(t.secondes)
>>> 65
```

Comment régler ce problème ?

```
def augmenter_seconde(self, sec):
        self.secondes = self.secondes + sec

        #dépassement secondes
        self.minutes += self.secondes // 60
        self.secondes = self.secondes % 60

        #dépassement minutes
        self.heures += self.minutes // 60
        self.minutes = self.minutes % 60
```

Programmation Orientée Objet

Dernier point, l'affichage de l'objet :

Programmation Orientée Objet

Dernier point, l'affichage de l'objet :

```
1 t = Chrono(21, 24, 55)
2 print(t)
3 >>> <__main__.Chrono object at 0x7fb8bac711f0>
```

Comment régler ce problème ?

Programmation Orientée Objet

Dernier point, l'affichage de l'objet :

```
t = Chrono(21, 24, 55)
print(t)
>>> <__main__.Chrono object at 0x7fb8bac711f0>
```

Comment régler ce problème ?

En utilisant la méthode particulière `__str__`, cette méthode renvoie une chaîne de caractères décrivant l'objet

Programmation Orientée Objet

Dernier point, l'affichage de l'objet :

```
t = Chrono(21, 24, 55)
print(t)
>>> <__main__.Chrono object at 0x7fb8bac711f0>
```

Comment régler ce problème ?

En utilisant la méthode particulière `__str__`, cette méthode renvoie une chaîne de caractères décrivant l'objet

```
def __str__(self):
    return str(self.heures) + "h " +
           str(self.minutes) + "m " +
           str(self.secondes) + "s"

t = Chrono(21, 24, 55)
print(t)
>>> 21h 24m 55s
```