D. D'autres méthodes spéciales :

1. Méthode d'affichage

Une méthode spécifique **__str__** est utilisée pour permettre à l'objet d'être affiché dans le format que l'on veut en utilisant la fonction **print** :

Exemple pour l'affichage d'un intervalle sous la forme [a,b]

```
class Intervalle :
        def __init__ ( self , debut , fin ) :
        """ premiere methode : constructeur """
             self.debut = debut # attribut
             self.fin = fin # attribut
        def __str__(self) :
             return " [" +str(self.debut ) + ","+str(self.fin)+ "]"
```

Ce qui donne :

```
>>> int1 = Intervalle(8,12)
>>> print(int1)
[ 8 , 12 ]
```

Pour certains types d'affichage (avec des tableaux) on utilise **__repr__** au lieu de **__str__**

2. Surcharge d'opérateur

La surcharge permet à un **opérateur** de posséder un **sens différent** suivant le type de ses opérandes. Les méthodes spéciales __add__ , __sub__ ou __eq__ permettent aux instances une utilisation spécifique des opérateurs + , - ou =

Exemple : dans l'expression [2,3] + 4 = [6,7] : le + correspond à une translation de 4 des bornes

```
class Intervalle :
        def __init__ ( self , debut , fin ) :
         """ premiere methode : constructeur """
              self.debut = debut # attribut
              self.fin = fin # attribut

        def __add__(self, valeur ) :
          """ permet l utilisation du + """
             self.debut = self.debut + valeur
             self.fin = self.fin + valeur

         def __sub__(self, valeur ) :
          """ permet l utilisation du - """
            self.debut = self.debut - valeur
            self.fin = self.fin – valeur

       def __eq__(self,autre ) :
          """ permet l utilisation du - """
            return self.debut= = autre.debut and self.fin == autre.fin
```

Ce qui donne :

```
>>> int1 = Intervalle(8,12)
>>> print(int1)
[ 8 , 12 ]
>>> int1+3
>>> print(int1)
[11,15]
>>> int1-5
>>> print(int1)
[6,10]
>>> int2=Intervalle(8,11)
>>> int1==int2
False
>>> int1==int2
True
```

Remarque : on peut accéder à toutes les méthodes d'une classe par *dir(nom_de_la_classe)*