Pour accéder ou modifier les attributs d'un objet dans le programme principal, on ne va généralement pas utiliser la méthode précédente ***nom_objet.nom_attribut*** permettant d'accéder aux valeurs des attributs car on ne veut pas forcement que l'utilisateur ait accès à la représentation interne des classes.
On utilisera de préférence des **méthodes spéciales (accesseur et mutateur)** dédiées dont le rôle est de faire l'interface entre l'utilisateur de l'objet et la représentation interne de l'objet (ses attributs).
Les attributs sont alors **encapsulés** dans l'objet, c'est à dire non accessibles directement par le programmeur qui a instancié un objet de cette classe.
Ces techniques ne sont pas obligatoires (et le programme de NSI n'est pas clair là-dessus !!!)

2ème Exemple : Construction ''propre'' d'un objet représentant les **intervalles** bornés :

```
class Intervalle :
    def __init__ ( self , debut , fin ) :
        """ premiere methode : constructeur """
        self.debut = debut # attribut
        self.fin = fin # attribut
    def get_debut(self):
        """ methode de recuperation d un attribut : accesseur """
        return( self.debut )
    def get_fin(self):
        """ methode de recuperation d un attribut : accesseur"""
        return( self.fin )
    def set_debut(self , nouveau_debut):
        """ methode de modification : mutateur """
        self.debut = nouveau_debut
    def set_fin(self , nouveau_fin):
        """ methode de modification : mutateur """
        self.fin = nouveau_fin
```

A retenir : Les méthodes spéciales :

- **Le constructeur** : L'endroit le plus approprié pour déclarer un attribut est à l'intérieur d'une méthode appelée le constructeur qui est alors exécuté lors de la création de chaque instance.
  Le constructeur d'une classe se présente comme une méthode et suit la même syntaxe à ceci près que son nom est imposé : **__init__** (deux underscores de chaque côté...).
  Hormis le premier paramètre, invariablement self, il n'existe pas de contrainte concernant la liste des paramètres excepté que le constructeur ne doit pas retourner de résultat.
- **Les accesseurs ou "getters"**

  Pour **obtenir la valeur** d'un attribut nous utiliserons la méthode des **accesseurs** (ou "getters") dont le nom est généralement : *get_Nom_attribut()* .
- **les mutateurs ou "setters" :**

  Pour **contrôler ou modifier** les valeurs d'un attribut, nous utiliserons les méthodes particulières appelées **mutateurs** (ou "setters") dont le nom est généralement *set_Nom_attribut()* .

remarque : La définition d'une classe est complétée ensuite par d'autres méthodes propre à la classe

Exercice 1 *: Réécrire la classe intervalle définie précédemment et y définir les méthodes suivantes :*

1. *La méthode largeur qui renvoie la largeur d'un intervalle.*
2. *La méthode centre qui renvoie le centre d'un intervalle.*
3. *La méthode translation qui opère une translation de l'intervalle d'un entier donné en argument. (par exemple, la translation de l'intervalle [8,12] par la valeur 3 donne l'intervalle [11,15]).*
4. *La méthode intersection qui prend en argument un 2ème intervalle, et renvoie l'intervalle intersection des deux.*
5. *Créer deux instances (deux intervalles) et tester toutes les méthodes construites ci-dessus*