La Programmation Orientée Objet

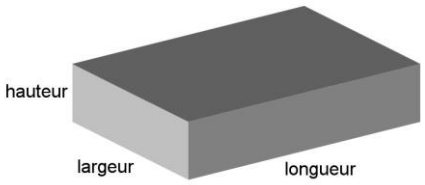
Structures de données

NSI - T^le^ TD

Contenu : Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.
Capacités attendues : Écrire la définition d'une classe.
Accéder aux attributs et méthodes d'une classe.

Exercice 1

On considère des boites assimilées à des pavés droits et définies par leurs dimensions : **longueur ≥ largeur ≥ hauteur.**
(On appelle longueur la plus grande dimension et hauteur la plus petite).



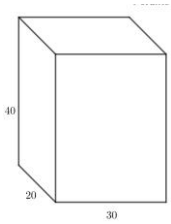
$$Volume = longueur \times largeur \times hauteur$$

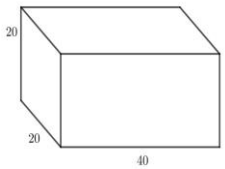
1. Écrire en Python la définition d'une classe **Boite** permettant d'implémenter les données d'une boîte.
   On doit pouvoir écrire les appels :
   petite_boite = Boite(10 , 5 , 4) grosse_boite = Boite(100,100,50).

Le constructeur doit vérifier avec le mot clé **assert** que les paramètres sont dans l'ordre décroissant.

2. Écrire une méthode **volume** qui renvoie le volume de la boite.

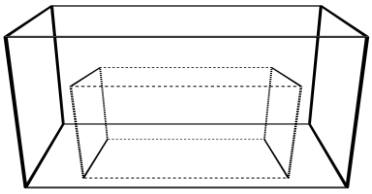
3. On considère les deux boites suivantes :





Écrire un script Python créant les objets associés à ces deux boites et affichant leurs volumes calculés avec la méthode précédente.

4. Un boîte peut être contenue dans une autre si les dimensions de cette dernière sont assez grandes. On néglige l'épaisseur des parois.



Définir une méthode **contenue_dans** prenant en paramètre un objet de la classe **Boite** et qui renvoie **True** si on peut ranger la boite qui appelle la méthode dans la boite argument.
On doit avoir : **petite_boite.contenue_dans(grosse_boite) # renvoie True.**
grosse_boite.contenue_dans(petite_boite) # renvoie False
grosse_boite.contenue_dans(grosse_boite) # renvoie True

5. Écrire une fonction **tri_triplet(a,b,c)** qui renvoie un triplet contenant les valeurs triées par ordre décroissant.
tri_triplet(10,2,5) # renvoie (10,5,2) tri_triplet(5,2,8) # renvoie (8,5,2)

6. Modifier le constructeur de la classe boîte pour qu'il accepte des dimensions dans un ordre quelconque. Les appels
   Boite(5,2,8)
   Boite(8,5,2)
   créeront des objets identiques.

Exercice 2

Définir une classe Intervalle représentant des intervalles de nombres. Cette classe possède deux attributs a et b représentant respectivement l'extrémité inférieure et l'extrémité supérieure de l'intervalle. Les deux extrémités sont considérées comme incluses dans l'intervalle. Tout intervalle avec b < a représente l'intervalle vide.

- Ecrire le constructeur de la classe Intervalle et une méthode est_vide renvoyant True si l'objet représente l'intervalle vide et False sinon.
- Ajouter une méthode __len__ renvoyant la longueur de l'intervalle (l'intervalle vide a une longueur 0).
  Ajouter une méthode __contains__ permettant de tester l'appartenance d'un élément x à un intervalle ivl avec l'instruction x in ivl.
- Ajouter une méthode intersection permettant de renvoyer l'intersection de deux intervalles ivl_1 et ivl_2 avec l'instruction ivl_1.intersection(ivl_2) ou, indifféremment, avec l'instruction ivl_2.intersection(ivl_1).
- (***) Même question avec l'union de deux intervalles.