1. Trier par sélection ou par insertion… à la main !
   Appliquer « à la main » les deux algorithmes du cours sur les différentes listes ci-dessous :
   a = [3, 9, 8, 4, 10, 12, 7]
   b = [1, 2, 3, 9, 8, 7, 1, 2, 3]
   c = [″Paul″, ″Rémi″, ″Edgar″, ″Nicolas″, ″Etienne″, ″Noémie″]

2. Cas extrêmes
   a) Combien de comparaisons et d'échanges d'éléments seront effectuées sur la liste : a = [1, 2, 3, 4, 5] avec chacun des deux algorithmes de tri du cours ?
   b) Même question avec la liste : b = [5, 4, 3, 2, 1].
   c) Reprendre les deux questions précédentes avec des listes a et b de 10 éléments
   d) Peut-on généraliser à des listes de *n* éléments ?

3. Fonctions de tri du cours en Python
   a) Ecrire une fonction tri_select(liste, mode="") qui applique l'algorithme de tri par sélection sur une liste.
      - mode = ″n″ : permet d'afficher le nombre de comparaisons et d'échanges effectués à la fin.
      - mode = ″d″ : permet d'afficher le détail du debug avec l'évolution du contenu de la liste étape par étape.
   b) Même question avec une fonction tri_insert(liste, mode="") qui applique l'algorithme de tri par insertion.

4. Comparaison avec d'autres algorithmes de tri
   a) Observer l'efficacité des différents algorithmes de tri à cette adresse : http://lwh.free.fr/pages/algo/tri/tri.htm
   Remarque : Un tri est dit « stable » lorsque deux éléments égaux pour la relation d'ordre utilisée se retrouvent toujours dans le même ordre après le tri.
   b) Tester les deux algorithmes du cours avec a = [*2*, **2**, 1]

5. Fonctions de tri intégrée en Python
   Soit : a = [(1, 2), (2, 3), (3, 2), (1, 1), (1, 3), (2, 2), (2, 1), (3, 1)]
   a) Observer le résultat des commandes a.*sort* et *sorted*(a).
   b) Tester avec les paramètres suivants :
      a.sort(key = lambda x : x[0]).
      Puis : a.*sort*(*key* = *lambda* x : x[1])

6. Autre algorithme : Le tri à bulles
a) Observer la méthode utilisée pour trier la liste [4, 10, 8, 3, 9, 7]

[4, 10, 8, 3, 9, 7]	[4, 8, 3, 9, 7, 10]	[4, 3, 8, 7, 9, 10]
[4, **8, 10**, 3, 9, 7]	[4, **3, 8**, 9, 7, 10]	[**3, 4**, 8, 7, 9, 10]
[4, 8, **3, 10**, 9, 7]	[4, 3, 8, **7, 9**, 10]	[3, 4, **7, 8**, 9, 10]
[4, 8, 3, **9, 10**, 7]		
[4, 8, 3, 9, **7, 10**]		

   Penser à des bulles qui remonteraient à la surface…
   Appliquer la même méthode pour trier à la main les listes :
   a = [3, 9, 8, 4, 10, 12, 7]
   b = [1, 2, 3, 9, 8, 7, 1, 2, 3]
   c = [″Paul″, ″Rémi″, ″Edgar″, ″Nicolas″, ″Etienne″, ″Noémie″]
a) Ecrire une fonction tri_bulles(liste, mode="") qui applique l'algorithme de tri à bulles sur une liste.
   - mode = ″n″ : permet d'afficher le nombre de comparaisons et d'échanges effectués à la fin.
   - mode = ″d″ : permet d'afficher le détail du debug avec l'évolution du contenu de la liste étape par étape.

➢ **Formule mathématique à connaitre :**

Soit *n* un entier naturel non nul et S la somme des entiers de 1 à *n*, alors :

$$\mathbf{S = 1 + 2 + 3 + \ldots + n = \frac{n(n+1)}{2}}$$

Démonstration :
S = 1 + 2 + … + (*n* – 1) + *n* dans l'ordre croissant
S = *n* + (*n* – 1) + … + 2 + 1 dans l'ordre décroissant
Additionnons membre à membre ces deux égalités, on obtient :
2 S = (*n* + 1) + (*n* + 1) + … + (*n* + 1) + (*n* + 1) *n* fois la même valeur
2 S = *n*(*n* + 1) Donc : $S = \frac{n(n+1)}{2}$