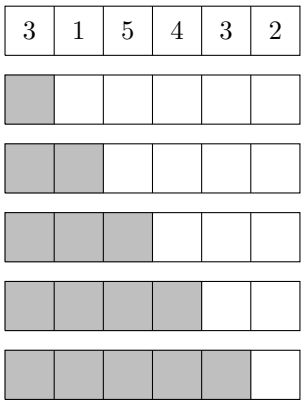
L'objectif de ce TP est d'écrire des algorithmes permettant de trier par ordre croissant les coefficients d'une liste de nombres. Ces nombres seront généralement de type *flottant*.
Pour écrire ces programmes, on utilise des boucles plutôt que des fonctions spécifiques à `Python`.

I Le tri sélection.

1 Principe



Pour trier une liste, cet algorithme repose sur le principe suivant :

- rechercher le plus petit coefficient de la liste.
- échanger ce coefficient avec le premier coefficient de la liste : ce coefficient est bien placé.
- répéter ces opérations sur les coefficients de la queue de liste à trier.

Ainsi, pour qu'une liste de longueur n soit triée, on va effectuer $n-1$ fois ces opérations.

Appliquer à la main cet algorithme sur la liste `[3,1,5,4,3,2]`.

2 Algorithme

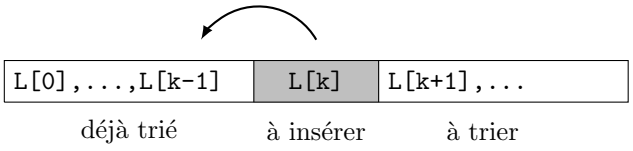
1. Écrire une fonction `indiceMin(L,k)` qui prend en argument une liste `L` et un indice `k` et qui renvoie l'indice `kmin` du plus petit coefficient de la liste extraite de `L` à partir de l'indice `k`.
2. Écrire une fonction `echange(L,i,j)` qui prend en argument une liste `L` et deux indices `i` et `j`, et qui renvoie la liste obtenue en échangeant les éléments d'indices `i` et `j`.
3. En combinant les deux fonctions précédentes, écrire une fonction `triSelection(L)` qui prend en argument une liste `L` et qui renvoie la liste triée correspondante.
4. Tester le programme sur plusieurs listes.

II Le tri insertion.

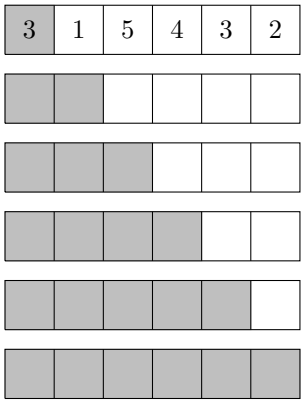
1 Principe

Le tri insertion est le tri le plus naturel. Il consiste à insérer successivement les éléments dans la liste des éléments déjà triés. C'est la méthode que l'on utilise lorsqu'on trie un jeu de cartes.

- le premier élément de la liste forme à lui seul une liste triée.
- on parcourt la liste à partir de l'élément d'indice 1 et à chaque étape `k` :
  - `L[:k]` est triée,
  - on insère l'élément d'indice `k` en le permutant avec les éléments précédents qui lui sont supérieurs.



Appliquer cet algorithme à la liste `[3,1,5,4,3,2]`.



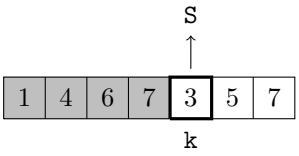
2 Déplacement dans une liste

On souhaite écrire une fonction `insere(L,k)` qui prend en argument une liste `L` et un indice `k` tels que les `k` premiers éléments sont triés, c'est-à-dire les éléments dont les indices sont compris entre `0` et `k-1`, et qui insère au bon endroit l'élément d'indice `k` de `L`. La liste obtenue est telle que les `k+1` premiers éléments sont triés, c'est-à-dire les éléments dont les indices sont compris entre `0` et `k`.

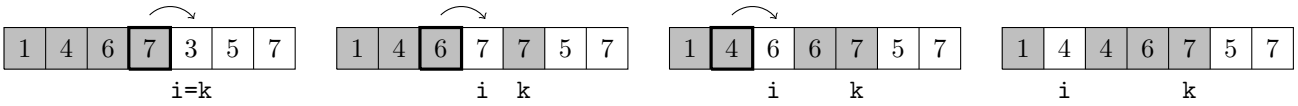
Par exemple, on considère `L=[1,4,6,7,3,5,7]` et `k=4`.

Méthode :

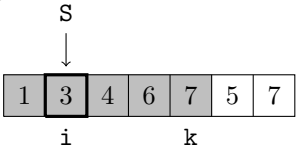
1. On stocke dans une variable `S` le terme en position `k`, que l'on souhaite insérer à la bonne place.



2. On parcourt de droite à gauche les termes déjà triés : tant que le terme rencontré est supérieur à `S` on le décale vers la droite jusqu'à trouver la position `i` où placer l'élément stocké dans `S`.



3. On place le terme stocké dans `S` en position `i`.



3 Algorithme

1. Écrire la fonction `insere(L,k)`.
2. En utilisant la fonction précédente, écrire une fonction `triInsertion(L)` qui prend en argument une liste `L` et qui renvoie la liste triée correspondante.
3. Tester le programme sur plusieurs listes.

III Tri par comptage

Dans le cas où les données à trier ne prennent qu'un petit nombre de valeurs différentes dans un intervalle connu, une autre méthode de tri consiste à :
• Compter pour chaque valeur de l'intervalle le nombre de fois où cette valeur apparaît dans la liste,
• Créer une liste triée en lisant une *table de comptage* qui indique combien de fois chaque valeur apparaît dans la liste.

Exemple : Soit la liste $L = [1, 5, 2, 2, 3, 5, 1, 6, 4, 4, 2, 1, 1, 1, 5, 5, 1]$.
Cette liste ne contient que des valeurs entières de l'intervalle $[\![1, 6]\!]$.
Pour la trier, on compte combien de fois apparaît la valeur 1 (ici 6 fois), combien de fois apparaît la valeur 2 (ici 3 fois) etc
On crée ainsi une liste d'effectifs : $E = [6, 3, 1, 2, 4, 1]$, qui sera la table de comptage, et qui sert à produire ensuite la liste triée : $T = [1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 4, 4, 5, 5, 5, 5, 6]$.

1. Écrire une fonction `bornes(L)` prenant pour argument une liste L d'entiers et renvoyant le *tuple* `(a,b)` formé par la plus petite valeur a et la plus grande valeur b de la liste L.
2. En déduire une fonction `tableComptage(L)` d'argument une liste L d'entiers, renvoyant la liste d'effectifs E associée à la liste L.
3. Écrire une fonction `triComptage(L)` renvoyant la liste triée associée à L.

IV Application aux statistiques

1. Écrire une fonction `mediane(L)` renvoyant la *médiane* d'une série statistique dont les valeurs sont enregistrées dans une liste L.
2. Écrire une fonction `quantile(L,n)` renvoyant les quantiles de la liste L ($n = 2$: médiane, $n = 4$: quartiles, $n = 10$: déciles, etc).