```python

##Exercice 3

def valeur_min(L):
    mini=L[0]  #on initialise provisoirement la valeur minimum
    for elt in L:
        if elt<mini:    #on change cette valeur uniquement si on en rencontre
            mini=elt       #une strictemetn plus petite
    return mini

def indice_min(L):
    mini=L[0]
    ind=0
    for k in range(len(L)):
        if L[k]<mini:
            mini=L[k]
            ind=k
    return ind

def couple_min(L):
    mini=L[0]
    ind=0
    for k in range(len(L)):
        if L[k]<mini:
            mini=L[k]
            ind=k
    return [ind,mini]

##Exercice 4 - Liste étiquetée

def somme(L,E):
    '''L est une liste de nombres, E est une liste de booléens(les étiquettes des nombres)'''
    n=len(L)
    res=0 #pour stocker la somme
    test=0 #pour savoir si toutes les étiquettes sont à False
    for i in range(len(L)):
        if E[i] :    #on ajoute le nombre L[i] uniquement si son étiquette est True
            res+=L[i]
        else:
            test+=1
    if test==n:  #cas où toutes les étiquettes sont False
        return False
    else:
        return res

def valeur_min2(L,E):
    mini=float('inf')  #on initialise provisoirement la valeur minimum

    for i in range(len(L)):
        if  E[i] and L[i]<mini:
            mini=L[i]

    return mini

##Exercice 5 - Tri par sélection

def indice_min_tri(L,g,d):
    mini=L[g]
    ind=g
    for k in range(g,d+1):
        if L[k]<mini:
            mini=L[k]
            ind=k
    return ind

def tri_selection(L):
    for k in range(len(L)-1):
        ind=indice_min_tri(L,k,len(L)-1)
        L[ind],L[k]=L[k],L[ind]
    return L

```

```
72  ##Exercice 5 - Tri rapide
73
74
75  def partition(a):
76      '''partitionne le tableau a en deux,le premier tableau renvoyé contient les
        éléments inférieurs
77              ou égaux au pivot (sans le pivot), le deuxième tableau contient les
                éléments strictement supérieurs'''
78
79      pivot=a[0]  #on choisit toujours le premier élément de la liste comme pivot
80      n=len(a)
81      b=[]   #b contiendra les éléments plus petits que le pivot
82      c=[]   #c contiendra les éléments plus grands que le pivot
83      for x in a[1:n]:
84          if x<=pivot:
85              b.append(x)
86          else:
87              c.append(x)
88
89      return [pivot,b,c]
90
91
92
93
94  def tri_rapide(t):
95
96      if len(t)==0 or len(t)==1:  # si la liste est vide ou contient un seul élément,
        elle est déjà triée
97          return t
98
99      pivot,b, c=partition(t)             # on partionne la liste autour du pivot
100     t=tri_rapide(b)+[pivot]+tri_rapide(c) # on fait un appel récursif
101                                          #en appliquant la fonction aux deux
                                             sous-tableaux
102     return t
103
104
105
106
107
```