❑ **Activité 2** : *Correction d'un algorithme*

1. Fonction de recherche du minimum
   On considère la fonction suivante :

```
def minimum(liste):
    mini = liste[0]
    for k in range(1,len(liste)):
        if liste[k]<mini:
            mini=liste[k]
    return mini
```

   a) Que fait cette fonction ?
   b) Ecrire une chaîne de documentation pour cette fonction.
   c) Ajouter les préconditions suivantes sous forme d'instructions `assert` : la liste passée en argument est non vide et ne contient que des valeurs de type entier ou flottant.
   d) Proposer un jeu de tests sous forme d'instruction `assert` pour cette fonction.

2. Problème de la correction
   ⚠ Le terme « correction » ne désigne pas ici l'action de corriger, on dit qu'un algorithme est **correct** lorsqu'il fournit bien la réponse attendue à un problème dans tous les cas.
   a) On note $e_0, e_1, e_2, ...$ les éléments de la liste passée en paramètre. Que contient la variable `mini` avant d'entrer dans la boucle `for` ? Le donner sous la forme du minimum d'une liste d'éléments.
   b) Donner sous la forme du minimum d'une liste d'éléments, le contenu de la variable `mini` après un passage dans la boucle.
   c) On cherche à prouver que notre fonction est *correcte* c'est à dire qu'elle renvoie le minimum de **n'importe** quelle liste passée en paramètre. Les tests effectués suffisent-ils ?
   d) Même question après deux passages, après trois passages.
   e) Proposer une propriété portant sur la variable `mini` et qui reste vraie avant d'entrer dans la boucle et à chaque itération dans la boucle.
   f) Montrer que cette propriété est vraie.
      🛟 On prouvera successivement que :
      — La propriété est vraie avant d'entrer dans la boucle.
      — Si la propriété est vraie lors d'une itération alors elle reste vraie à l'itération suivante.

3. Conclure sur la correction de la fonction.

❑ **Activité 3** : *Terminaison d'un algorithme*

1. Multiplier deux entiers en faisant des additions
   On considère la fonction suivante :

```
def mult(n,p):
    produit = 0
    k = p
    while k>0:
        k=k-1
        produit = produit+n
    return produit
```

   a) Que fait cette fonction ?
   b) Ecrire une chaîne de documentation pour cette fonction.
   c) Ajouter les préconditions suivantes sous forme d'instructions `assert` : les arguments sont des entiers positifs.
   d) Proposer un jeu de tests sous forme d'instruction `assert` pour cette fonction.

2. Terminaison de cet algorithme
   a) Quels sont les valeurs successives prises par la variable `k` lors de chaque passage dans la boucle ?
   b) Conclure.
      🛟 On pourra utiliser la propriété mathématique suivante : « il n'existe pas de suite d'entiers naturels strictement décroissante ».