# Chapitre 16 :
# Correction d'un programme

## I Tests

Les tests permettent de vérifier le comportement d'un programme sur des exemples. Ils ne permettent pas de prouver que le programme fonctionne mais peuvent montrer qu'il ne fonctionne pas.

Pour tester un programme, il faut construire un jeu de test, c'est-à-dire choisir des valeurs intéressantes pour lesquelles le programme va être exécuté. On peut choisir :

- des cas simples,
- des valeurs extrêmes, voire interdites,
- un grand nombre de valeurs (dans ce cas, on écrira une fonction afin de réaliser les tests), ces valeurs pouvant être choisies aléatoirement,
- des valeurs représentant des catégories différentes (par exemple nombres pairs/impairs),
- ...

➪ **Exemple 1 :** On veut tester une fonction DivisionEuclidienne qui prend comme arguments deux entiers $a$ et $b$ et qui renvoie le couple $(q,r)$ tel que $q$ est le quotient et $r$ le reste de la division euclidienne de $a$ par $b$.

```
def test1() :
    for a in range(10) :
        for b in range(1,10) :
                (q,r)=DivisionEuclidienne(a,b)
                assert a==b*q+r and 0<=r<b

import random

def test2() :
    for i in range(100) :
            a=random.randint(0,1000)
            b=random.randint(1,1000)
            (q,r)=DivisionEuclidienne(a,b)
            assert a==b*q+r and 0<=r<b
```

La fonction test1 réalise des tests sur les entiers strictement inférieurs à 10 et la fonction test2 réalise des tests sur 100 entiers aléatoires inférieurs ou égaux à 1000.

## II Terminaison

Les boucles for se terminent forcément, elles ne sont donc pas concernées par cette étude. Le seul souci pourrait venir d'une boucle while et des fonctions récursives. En effet, dans le cas d'une boucle while, le critère d'arrêt dépend d'une condition et pour les fonctions récursives, le cas de base doit être atteint. Il faut donc s'assurer que la condition de sortie de boucle est vérifiée en un nombre fini d'étapes. Pour cela, on utilise une expression qui prend un nombre fini de valeurs, cette expression est appelée un **variant de boucle**.

L'argument le plus courant pour justifier la terminaison d'un algorithme est d'exhiber un variant de boucle à valeurs **entières** qui croit **strictement** à chaque itération et qui est **majorée** (ou qui décroit strictement à chaque itération et est minorée). En effet, la condition de sortie de boucle sera alors nécessairement satisfaite en un nombre fini d'itérations.

➪ **Exemple 2 :**

```
def recherche(l,x) :
    """Recherche si un entier x et dans une liste d'entiers x """
    i=0
    while i<len(l) and l[i] !=x :
        i+=1
    if i==len(l) :
        return False
    else :
        return True
```

La variable $i$ est un variant de boucle qui décrit une suite d'entiers strictement croissante et majorée par len(l), il s'agit donc d'une suite finie. Ainsi la terminaison est prouvée.

➪ **Exemple 3 :**

```
def dichotomie(f,a,b,epsilon) :
    """f est une fonction continue, a et b sont des flottants tels que f(a) et f(b) soient
    de signe opposé et epsilon un flottant strictement positif.
    dichotomie(f,a,b,epsilon) renvoie une liste [x,y] telle que |y-x|≤ε et telle que f
     admette au moins un zéro dans l'intervalle [x,y]."""
    assert f(a)*f(b)<=0 :
    x=a
    y=b
    while abs(y-x)>epsilon :
        z=(x+y)/2
        if f(x)*f(z)<=0 :
            y=z
        else :
            x=z
    return [x,y]
```

L'expression $y-x$ est un variant de boucle. Elle décrit une suite géométrique de raison $\frac{1}{2}$, elle tend donc vers 0. Ainsi, en un nombre fini d'itérations : $|y-x| \leq \varepsilon$.Ainsi la terminaison est prouvée.

➪ **Exemple 4 :**

```
def puissance(x,n) :
    """Calcule x puissance n, avec x un flottant et n un entier positif """
    if n==0
        return 1
    else :
        return x*puissance(x,n-1)
```

Il s'agit d'une fonction récursive.

La variable $n$ est un variant de boucle qui décrit une suite d'entiers positifs strictement décroissante , le cas de base $n=0$ sera donc atteint. Ainsi la terminaison est prouvée.

# III Correction

On appelle **invariant de boucle** une propriété portant sur les variables d'une boucle qui est vraie au cours de la boucle et qui permet de prouver qu'on obtient le résultat voulu en sortie de boucle.

Les invariants de boucles peuvent porter sur les valeurs en entrée ou en sortie de boucle.

Les preuves d'invariants de boucles sont des preuves par récurrence.

On dit que :

- la **correction est partielle** quand le résultat est correct (prouvé par un invariant de boucle) lorsque l'algorithme s'arrête,
- la **correction est totale** quand le résultat est correct (prouvé par un invariant de boucle) et si l'algorithme se termine (prouvé par un variant de boucle).

➪ **Exemple 5 :**

```
def somme(n) :
    s=0
    for i in range(n+1) :
        s+=i
    return s
```

La fonction somme(n) renvoie la valeur : $\sum_{k=1}^{n} k$.

On considère, pour $i \in \mathbb{N}$, la propriété suivante :

$$\mathcal{P}(i) : \text{"à l'entrée du tour de boucle } i\text{, on a : } s = \sum_{k=1}^{i-1} k\text{"}$$

Montrons que cette propriété est un invariant de boucle.

- Pour $i = 0$ :
  On se place à l'entrée du tour de boucle 0, donc $s = 0$, ainsi $s = \sum_{k=1}^{i-1} k$.
- Soit $i \in \mathbb{N}$, supposons $\mathcal{P}(i)$ vraie.
  On se place à l'entrée du tour de boucle $i+1$, comme le tour de boucle $i$ a été effectué, on a alors :
  $$s = \sum_{k=1}^{i-1} k + i = \sum_{k=1}^{i} k.$$
  Donc $\mathcal{P}(i+1)$ est vraie.
- Donc, pour tout $i \in \mathbb{N}$, $\mathcal{P}(i)$ est vraie.

Le résultat renvoyé correspond à la sortie du tour de boucle $n$, c'est-à-dire à l'entrée du tour de boucle $n+1$, donc, d'après $\mathcal{P}(n+1)$, le résultat renvoyé est :
$$s = \sum_{k=1}^{n} k$$

➪ **Exemple 6 :**

```
def max(l) :
    m=l[0]
    for i in range(1,len(l)) :
        if m<l[i] :
            m=l[i]
    return m
```

La fonction max(l) renvoie la valeur : $\max(l[k], 0 \leq k \leq len(l)-1)$.
On considère, pour $i \in \mathbb{N}^*$, la propriété suivante :

$$\mathcal{P}(i) : \text{"à l'entrée du tour de boucle } i\text{, on a : } m = \max(l[k], 0 \leq k \leq i-1)\text{"}$$

Montrons que cette propriété est un invariant de boucle.

- Pour $i = 1$ :
  On se place à l'entrée du tour de boucle 1, donc $m = l[0]$, ainsi $m = \max(l[k], 0 \leq k \leq i-1)$.
- Soit $i \in \mathbb{N}^*$, supposons $\mathcal{P}(i)$ vraie.
  On commence par se placer à l'entrée du tour de boucle $i$, on a alors, par hypothèse : $m = \max(l[k], 0 \leq k \leq i-1)$. De plus :
  - Si $m < l[i]$, alors $\max(l[k], 0 \leq k \leq i-1) < l[i]$ donc : $\max(l[k], 0 \leq k \leq i) = l[i]$. De plus, dans ce cas, à l'entrée du tour de boucle $i+1$ : $m = l[i]$. Donc : $m = \max(l[k], 0 \leq k \leq i)$.
  - Sinon, alors $m \geq l[i]$ donc $\max(l[k], 0 \leq k \leq i-1) \geq l[i]$ d'où : $\max(l[k], 0 \leq k \leq i) = \max(l[k], 0 \leq k \leq i-1)$. De plus, dans ce cas, à l'entrée du tour de boucle $i+1$ : $m = \max(l[k], 0 \leq k \leq i-1)$. Donc : $m = \max(l[k], 0 \leq k \leq i)$.
  -

  Donc $\mathcal{P}(i+1)$ est vraie.
- Donc, pour tout $i \in \mathbb{N}^*$, $\mathcal{P}(i)$ est vraie.

Le résultat renvoyé correspond à la sortie du tour de boucle $len(l)-1$, c'est-à-dire à l'entrée du tour de boucle $len(l)$, donc, d'après $\mathcal{P}(len(l))$, le résultat renvoyé est :
$$m = \max(l[k], 0 \leq k \leq len(l)-1).$$

➪ **Exemple 7 :**

```
def recherche(l,x) :
    i=0
    while i<len(l) and l[i] !=x :
        i+=1
    if i==len(l) :
        return False
    else :
        return True
```

On considère, pour $i \in \mathbb{N}$, la propriété suivante :

$$\mathcal{P}(i) : \text{"à l'entrée du tour de boucle } i\text{, on a, pour tout } k \in [\![0, i]\!],\ x \neq l[k]\text{"}$$

Montrons que cette propriété est un invariant de boucle.

- Pour $i = 0$ :
  On se place à l'entrée du tour de boucle 0, comme la condition correspondant au tour de boucle 0 est vérifiée, on a $x \neq l[0]$ .
- Soit $i \in \mathbb{N}$, supposons $\mathcal{P}(i)$ vraie.
  On commence par se placer à l'entrée du tour de boucle $i+1$. Comme le tour de boucle $i$ a été effectué, on a, d'après $\mathcal{P}(i)$ : $\forall k \in [\![0, i]\!]$, $x \neq l[k]$.
  De plus, on se place à l'entrée du tour de boucle $i+1$, donc la condition correspondant au tour de boucle $i+1$ est vérifiée, c'est-à-dire : $x \neq l[i+1]$.
  Ainsi : $\forall k \in [\![0, i+1]\!]$, $x \neq l[k]$. Donc $\mathcal{P}(i+1)$ est vraie.
- Donc, pour tout $i \in \mathbb{N}$, $\mathcal{P}(i)$ est vraie.

Donc la propriété $\mathcal{P}$ est un invariant de boucle.

La boucle se termine lorsque :

- $i = len(l)$, dans ce cas le dernier tour de boucle effectué est $i = len(l) - 1$ donc, d'après $\mathcal{P}(len(l) - 1)$ :

$$\forall k \in [\![0, len(l) - 1]\!], x \neq l[k].$$

  Donc $x$ n'est pas dans $l$, ce qui correspond bien à la valeur False renvoyée.
- $i < len(l)$ et $l[i] = x$, dans ce cas $x$ est dans $l$, ce qui correspond bien à la valeur True renvoyée.

➪ **Exemple 8 :**

```
def puissance(x,n) :
    """Calcule x puissance n, avec x un flottant et n un entier positif """
    if n==0
        return 1
    else :
        return x*puissance(x,n-1)
```

On considère, pour $n \in \mathbb{N}$, la propriété suivante :

$$\mathcal{P}(n) : \text{" puissance(x,n) renvoie } x^n \text{"}.$$

Montrons que cette propriété est un invariant de boucle.

- Pour $n = 0$ :
  Il s'agit du cas de base donc puissance(x,n) renvoie $1 = x^n$.
- Soit $n \in \mathbb{N}$, supposons $\mathcal{P}(n)$ vraie.
  puissance(x,n+1) renvoie x*puissance(x,n)$= x.x^n = x^{n+1}$.
- Donc, pour tout $n \in \mathbb{N}$, $\mathcal{P}(n)$ est vraie.

Donc la propriété $\mathcal{P}$ est un invariant de boucle et le résultat renvoyé est prouvé.

# IV Etude complète d'un exemple : le tri par sélection

L'algorithme du tri par sélection est :

```
def TriSelection(L) :
    l=copy.copy(L)
    n=len(l)
    for i in range(n) :
        imin=i #recherche du minimum à partir du rang i
        m=l[i]
        for j in range(i+1,n) :
            if l[j]<m :
                imin=j
                m=l[j]
        l[imin],l[i]=l[i],l[imin] # on place le minimum en position i
    return l
```

Ici, on a réalisé une copie de la liste afin d'éviter un effet de bord et de pouvoir comparer la liste triée avec la liste initiale.

On rappelle que sa complexité est quadratique.

**Tests :**

On écrit une fonction permettant de tester si la liste est bien triée par ordre croissant :

```
def EstCroissante(l) :
    n=len(l)
    for i in range(n-1) :
        if l[i]>l[i+1] :
            return False
    return True
```

On écrit une fonction permettant de tester si deux listes s'obtiennent par permutation :

```
def EstPermutation(l1,l2) :
    d1={}
    d2={}
    for x in l1 : #dictionnaire dont les clés sont les éléments de l1
            et les valeurs leur nombre d'apparitions
        if x in d1 :
            d1[x]+=1
        else :
            d1[x]=1
    for x in l2 : #de même avec l2
        if x in d2 :
            d2[x]+=1
        else :
            d2[x]=1
    return d1==d2
```

On teste la fonction sur :

- une liste de taille 1,
- une liste de petite taille triée par ordre croissant,
- une liste de petite taille triée par ordre décroissant,
- une liste aléatoire de grande taille.

```
>>> l1=[2]
>>> l2=[i for i in range(20)]
>>> l3=[20-i for i in range(20)]
>>> import random
>>> l4=[random.randint(1,10000) for i in range(1000)]
>>> EstCroissante(TriSelection(l1)) and EstPermutation(TriSelection(l1),l1)
True
>>> EstCroissante(TriSelection(l2)) and EstPermutation(TriSelection(l2),l2)
True
>>> EstCroissante(TriSelection(l3)) and EstPermutation(TriSelection(l3),l3)
True
>>> EstCroissante(TriSelection(l4)) and EstPermutation(TriSelection(l4),l4)
True
```

**Terminaison :**

L'algorithme est itératif et composé de boucles for donc se termine.

**Correction :**

Comme dans l'exemple 6, on peut montrer que le résultat obtenu en sortie de la boucle for portant sur la variable $j$ est :

$$\text{l[imin]}=\min_{i\leq j<n} \text{l[j]} \text{ et } i \leq \text{imin} < n.$$

On considère, pour $i \in \mathbb{N}$, la propriété suivante :

$$\mathcal{P}(i) : \text{"à la sortie du tour de boucle } i,\ l[0] \leq l[1] \leq \cdots \leq l[i] \text{ et pour tout } j > i,\ l[j] \geq l[i]\text{"}$$

Montrons que cette propriété est un invariant de boucle.

- Pour $i = 0$ :
  Avant l'échange, on a : l[imin]=min $l$ donc, après l'échange, $l[0] = \min l$.
  Ainsi, à la sortie du tour de boucle 0, pour tout $j > 0$, $l[j] \geq l[0]$ donc $\mathcal{P}(0)$ est vraie.
- Soit $i \in \mathbb{N}$, supposons $\mathcal{P}(i)$ vraie.
  On commence par se placer à l'entrée du tour de boucle $i+1$, on a alors, par hypothèse : l[imin]= $\min_{i \leq j < n}$ l[j] et $i \leq$ imin $< n$.
  Puis, on effectue la boucle for sur $j$, on a alors : l[imin]= $\min_{i+1 \leq j < n}$ l[j] et $i + 1 \leq$ imin $< n$.
  Ensuite, on effectue l'échange, on a donc : $l[i+1] = \min_{i+1 \leq j < n} l[j]$.
  Ainsi :
  - comme pour tout $j > i$, $l[j] \geq l[i]$, on a : $l[i+1] \geq l[i]$ donc : $l[0] \leq l[1] \leq \cdots \leq l[i+1]$,
  - comme $l[i+1] = \min_{i+1 \leq j < n} l[j]$, on a : pour tout $j > i+1$, $l[j] \geq l[i+1]$.

  Donc $\mathcal{P}(i+1)$ est vraie.
- Donc, pour tout $i \in \mathbb{N}$, $\mathcal{P}(i)$ est vraie.

Le résultat renvoyé correspond à la sortie du tour de boucle $n-1$, donc, d'après $\mathcal{P}(n-1)$, $l[0] \leq l[1] \leq \cdots \leq l[n-1]$ ainsi $l$ est triée par ordre croissant.