Correction

```
# 1.
def decimal(L):
    L2 = [L[-k-1]*2**k for k in range(len(L))]
    return sum(L2)

# 4.
def binaire(n):
    if n<0:
        return
    L=[n%2]
    n=n//2
    while n>0:
        L.append(n%2)
        n=n//2
    return L

# 5.
# a)
def dicho(f,a,b,epsilon):
    assert f(a)*f(b)<=0 and epsilon>0
    c=(b-a)/2
    while abs(c)>epsilon:
        if f(a+c)*f(a)>0:
            a=a+c
            c=c/2
        else:
            b=a+c
            c=c/2
    return a+c

# b)
r2=dicho(lambda x:x**2-2,1,2,1e-7)
# c)
print(r2,abs(r2-2**0.5))
# d)
r2=dicho(lambda x:x**2-2,1,2,1e-15)
print(r2,abs(r2-2**0.5))
r3=dicho(lambda x:x**2-3,1,2,1e-15)
print(r3,abs(r3-3**0.5))
r5=dicho(lambda x:x**2-5,2,3,1e-15)
print(r5,abs(r5-5**0.5))
r1000=dicho(lambda x:x**2-1000,30,40,1e-15)
print(r1000,abs(r1000-1000**0.5))
r10001=dicho(lambda x:x**2-10001,100,101,1e-15)
print(r10001,abs(r10001-10001**0.5))
```

Remarque à propos de la question 4) : deux problèmes peuvent se produire lors de l'utilisation

de l'algorithme de dichotomie liés à la représentation approximative des nombres réels en informatique

- la boucle `while` ne se termine pas : c'est le cas si on fait porter le test de terminaison directement sur les variables représentant a_n et b_n. En effet, ***il est possible qu'à cause de l'approximation due à l'utilisation de flottants au lieu de réels***, les variables `a` et `b` représentant ces suites soient ***constantes bien distinctes à partir d'un certain rang***. Si cela se produit, il est possible que `abs(b-a)` ne soit jamais inférieure à la précision exigée, d'où une boucle infinie ;
- la boucle se termine bien mais l'approximation obtenue n'est pas de la qualité désirée. C'est ce qui se produit dans la correction donnée plus haut pour `r1000` et `r10001`. Une étude attentive montre en effet (pour le dernier cas) qu'à partir d'un certain rang
  `a` vaut 100.00499987500623
  `b` vaut 100.00499987500625
  et `(a+b)/2` vaut 100.00499987500623 (problème de représentation des flottants).
  Une manière d'éviter ce problème est de tester laquelle des valeurs `a` ou `b` est la meilleure approximation et de renvoyer cette dernière. Ainsi, la fonction suivante est meilleure que celle donnée plus haut :

```
# Version amelioree
def dicho(f,a,b,epsilon):
    assert f(a)*f(b)<=0 and epsilon>0
    c=(b-a)/2
    while abs(c)>epsilon:
        if f(a+c)*f(a)>0:
            a=a+c
            c=c/2
        else:
            b=a+c
            c=c/2
    if abs(f(a))<abs(f(b)):
        return a
    else:
        return b
```