TP 2- Proposition de solutions

Solution 1

```
def nbr(a,b,c):
    '''
    Entrees : a,b,c trois reels avec a non nul
    Sortie : nb de racines reelles du trinome
                    aX^2+bX+c
    '''
        assert a!=0,'ENC'
    d=b**2-4*a*c
    if abs(d)<1e-15: #test de nullite sur les reels
        return 1
    elif d>0:
        return 2
    else:
        return 0
```

Solution 2

```
def factorielle(n):
    '''
    Entree : n un entier naturel
    Sortie : n! (un entier)
    '''
    f=1
    for i in range(1,n+1):
        f=f*i
    return f
```

```
def cb(k,n):
    return factorielle(n)/factorielle(k)/factorielle(n-k)
```

```
def cb2(k,n):
    p=1
    for j in range(1,k+1):
        p=p*(n+1-j)/j
    return p
```

Solution 3

```
import numpy as np

def suite(u,v,n):
    assert u>0 and v>0,'Entree non conforme'
    for k in range(1,n+1):
        u,v=(u+v)/2,np.sqrt(u*v)
    return u,v
```

Solution 4

```
def racine(x,n):
    u=1
    for k in range(1,n+1):
        u=(u+x/u)/2
    return u
```

```
def racine2(x):
    '''
    Calcule une approximation de
    la racine carree de x (un reel positif)
    '''
    u=1
    v=(u+x/u)/2
    n=1
    while abs(u-v)>1e-12:
        u,v,n=v,(v+x/v)/2,n+1
    return v,n
```

Solution 5

```
def somme1(f,g):
    return lambda t:f(t)+g(t)

somme2=lambda f,g:lambda t:f(t)+g(t)

def somme3(f,g):
    def h(t):
        return f(t)+g(t)
    return h
```

Solution 6

```
def balayage(f,a,b,p):
    '''
    Reperage dune racine de f par
    identification d un changement de signe.

    Entrees : f une fonction continue sur [a,b]
              p la precision
    Sortie : la premiere racine detectee
             sinon retourne False
    '''
    if abs(f(a))<1e-15:
        return a
    while a<b and f(a)*f(a+p)>0:
        a=a+p
    if a<b or abs(f(a))<1e-15:
        return a
    else:
        return False
```