```
## TP 4

## exo 1
# q1
def suite1(n):
    u,v = 3,2
    for k in range(n-1):
        u,v = v, v/(1+u)
    return v

# q2
def suite2(n):
    a,b = 2,1
    for k in range(n-2):
        a,b = b, b+(1/a**2)
    return b

# q3
def suite3(n):
    x, y = 2,2
    for k in range(n-1):
        x,y = y, (k*x+1)/(k**2+y)
    return y

## exo 2
# q1
# B_{n+1} = B_n + B_n*(tau/100) = B_n*(1+tau/100)

# q2
def population(B0,n,tau):
    b = B0
    for k in range(n):
        b = b*(1+tau/100)
    return b

# q3
# population(425, 30, 1.94) vaut environ 756

## exo 3
# q1
# u_{n+1} = u_n + u_n*(1/2**n) = u_n*(1+1/2**n)

# q2
def nenuphar(u0,n):
    u = u0
    for k in range(n):
        u = u * (1+1/2**k)
    return u

# q3
# on teste nenuphar(10,N) avec N = 10 puis 20 puis 100 : on constate qu'on converge vers la
valeur 47,68 environ

## exo 4
# q1
def suite(n):
    u = 4
    for k in range(n):
        u = u/(1+u**2)
    return u

# on teste que suite(0) renvoie 4 et suite(1) renvoie 4/(1+4**2)

# q2
# on teste les valeurs suite(0), suite(1), suite(2), suite(3), etc. Ces valeurs décroissent, il
semble donc que (u_n) soit décroissante.

# q3
def decroissante(N):
    for k in range(N+1):
        if suite(k) < suite(k+1):
            return False
    return True

# q4
# ce n'était pas une bonne idée d'utiliser "suite" au sein de "décroissante" car cela fait
```

```
reprendre le calcul des termes de la suite depuis le départ à chaque fois. Mieux vaut calculer
les termes et les comparer au fur et à mesure.
def decroissante_bis(N):
    u = 4
    for k in range(N+1):
        v = u/(1+u**2) # on calcule le terme suivant
        if u < v:
            return False
        u = v # on update u avec le terme suivant
    return True

## exo 5
# q1
def triangle_rect(n):
    for k in range(1,n+1):
        print(k*"*")

# q2
def triangle_iso(n):
    for k in range(1,n+1):
        espaces = (n-k)*" "
        etoiles = (2*k-1)*"*"
        print(espaces+etoiles)

# q3
def triangle_iso_bis(n): # on fait un triangle iso mais tête en bas
    for k in range(n-1,0,-1): # on prend les mêmes valeurs de k, mais dans l'autre sens ! et
avec une ligne de moins pour ne pas répéter la base du triangle au milieu du losange...
        espaces = (n-k)*" "
        etoiles = (2*k-1)*"*"
        print(espaces+etoiles)

def losange(n): # pour dessiner un losange, on fait un triangle dans un sens, puis un dans
l'autre !
    triangle_iso(n)
    triangle_iso_bis(n) # une ligne de moins pour ne pas répéter la base
```