```
## TP3
## exo 1

## q1
for k in range(6,12):
    print(k)

## q2
for k in range(101,122,2):
    print(k)

## q3
for k in range(20,-21,-5):
    print(k)


## exo 2
# q1
def f(n):
    S = 0
    for k in range(3,n+2):
        S = S + (2*k+1)**(1/2)
    return S

# on teste dans la console que f(4) renvoie 7**(1/2) + 9**(1/2) + 11**(1/2)

# q2
def fact(n):
    P = 1
    for k in range(1,n+1):
        P = P*k
    return P

# on teste que fact(4) renvoie 24

# q3
def g(x,n):
    S = 0
    for k in range(n+1):
        S = S + x**k/fact(k)
    return S

# on teste par exemple que g(2,3) renvoie 2^0/0! + 2^1/1! + 2^2/2! + 2^3/3! = 1 + 2 + 2 + 8/6 =
6,3333...

# q4
def suite1(n):
    u = 2
    for k in range(n):
        u = 2*u + 1/u
    return u

# on teste suite1(10) environ égal à 2379

# q5
def suite2(n):
    v = 1.4
    for k in range(1,n):
        v = v*(3-v)
    return v

# on teste suite2(10) environ égal à 2,16

## exo 3
# q1
def suite3(n):
    u = 3
    for k in range(0,n):
        u = (u+3*k)/(2*k+1)
    return u

def suite4(n):
    v = 1
    for k in range(2,n+1):
        v = (v + k)**2
    return v
```

```python
def somme(n):
    S = 0
    for k in range(n+1):
        S = S + k/(2*k+n+1)
    return S

## exo 4
# q1
def suite(n):
    u = 1
    for k in range(n):
        u = 3*u**(1/2) + 1
    return u

# suite(0) renvoie 1, suite(1) renvoie 4, suite(2) renvoie 7

# q2
def somme(n):
    s = 0
    for k in range(n+1):
        s = s + suite(k)
    return s

# q3
# Pour évaluer somme(5) Python appelle séparément suite(k) pour k compris entre 0 et 5. Or
chaque appel de suite(k) pour k >= 1 demande de calculer u_1. Dans la boucle for de la fonction
somme, lorsqu'on demande d'ajouter suite(3) immédiatement après avoir calculé suite(2), Python
reprend le calcul de u_3 depuis le début, sans utiliser la valeur de u_2 qu'il vient de
calculer ! En tout, on calcule donc 5 fois la valeur de u_1 pour calculer somme(5) alors qu'une
seule fois aurait suffi si on a avait gardé en mémoire la valeur de u !

# q4
def somme_bis(n):
    u = 1
    S = u # on initialise S à u_0 plutôt qu'à 0 car...
    for k in range(n):
        u = 3*u**(1/2) + 1
        S = S + u #... lors du premier tour de la boucle for, on ajoute ici u_1 à S
    return S

# q5
import time

# à l'exécution du code ci-dessous :

##
start_time = time.time()
print(somme(1000))
print("--- %s seconds ---" % (time.time() - start_time))
##

# on obtient un temps d'exécution d'environ 0.1 seconde
# tandis que pour le code

##
start_time = time.time()
print(somme_bis(1000))
print("--- %s seconds ---" % (time.time() - start_time))
##

# on obtient un temps d'exécution d'environ 0.001 seconde.
# Attention ces temps d'exécution dépendent de la machine sur laquelle on travaille !


## exo 4 : voir TP 4

## exo 5 :
# il faut utiliser des fonctions sinus et cosinus, en important une bibliothèque...
```