```
## TP2 : if

## exo 1

# q1
x = 1 # ou x=-1
if x>0:
    x = x+2
# avec x=1 au départ, x vaut 3 à la fin
# avec x=-1 au départ, x vaut -1 à la fin

# q2
y = 1 # ou y=6
if y<2:
    y = y + 2
y = y - 2
# quand y=1 au départ, y vaut 1 à la fin
# quand y=3 au départ, y vaut 4 à la fin
# la ligne 2 est prise en compte uniquement si y<2
# la ligne 3 est toujours prise en compte

# q3
# fun(3) vaut 1, fun(-2) vaut -1


## exo 2
# q1
# 4 et 1

# q2
# 1 et 0
# (pour le deuxième : attention, à la fin du premier if, a vaut 0.5, donc on
rentre dans le second if)

# q3
# -1
# (on ne rentre pas dans le premier if, donc pas dans le second qui est à
l'intérieur du premier)

## exo 3
# q1
def valabs(x):
    if x>0:
        return x
    else :
        return -x

# q2
def un_ou_deux(x):
    if x==1 or x==2 :
        return "oui"
    else :
        return "non"

## exo 4
def maxi(a,b):
    if a>b:
        return a
    else :
        return b
```

```python
def maxi3(a,b,c):
    if a>=b and a>=c:
        return a
    if b>=a and b>=c:
        return b
    else :
        return c

def maxi3_bis(a,b,c):
    max_ab = maxi(a,b)
    max_abc = maxi(max_ab,c)
    return max_abc


## exo 5

# question 1

def etat1(T):
    if T<0 :
        return "solide"
    if T>=0 and T<100 :
        return "liquide"
    else :
        return "gazeux"
# on peut aussi être plus précis et dire que pour T=0, on a les 2 phases
solide et liquide, et que pour T=100 on a les 2 phases liquide et gazeux

# question 2

import numpy as np

def etat2(T,P):
    log10P = np.log(P)/np.log(10)
    if log10P <= T/50-2 :
        return "gazeux"
    else :
        if T<0 :
            return "solide"
        else :
            return "liquide"

# idem on peut raffiner avec les cas d'égalité où il y a 2 phases en
simultané, voire 3 pour le point triple...

## exo 6

def racines(a,b,c):
    delta = b**2 - 4*a*c
    if delta<0:
        return "pas de solution réelle"
    if delta ==0 :
        return -b/(2*a)
    if delta>0:
        alpha = (-b + delta**(1/2) ) / (2*a)
        beta = (-b - delta**(1/2) ) / (2*a)
        return alpha, beta

# racines(1,-4,3) renvoie (3.0, 1.0)
# racines(1,4,4) renvoie -2
# racines(1,0,1) renvoie 'pas de solution réelle'
```

```python
def racines2(a,b,c):
    if a==0:
        if b==0:
            if c==0:
                return 'tous les réels sont solutions'
            else :
                return 'pas de solution'
        else :
            return -c/b
    else :
        return racines(a,b,c)

# racines2(0,2,1) renvoie -.5
# racines2(0,0,1) renvoie 'pas de solution'
# racines2(0,0,0) renvoie 'tous les réels sont solutions'
```