LISTE D'ALGORITHMES À CONNAÎTRE

1 SUITES RÉCURRENTES

```
# suite récurrente d'ordre 1
# exemple simple u(n+1) = u(n)**2 + 1
def recc(u0,n):
    u = u0
    for i in range(n):
        u = u**2 + 1
    return u

# suite non majorée : premier rang >= M
def rang(u0,M):
    u = u0; n = 0
    while u < M:
        u = u**2 + 1
        n += 1
    return n

# suite non majorée : dernier terme < M
# on suppose M > u0
def terme(u0,M):
    u = u0
    while u < M:
        (u,v) = (u**2 + 1,u)
    return v

# conjecture sur la croissance de la suite
def croissante(u0,n):
    u = u0
    for i in range(n):
        (u,v) = (u**2 + 1, u)
        if u < v:
            return False
    return True


# somme des termes de la suite
def somme(u0,n):
    u = u0; s = u0
    for i in range(n):
        u = u**2 + 1
        s += u
    return s

# suite récurrente d'ordre 2
# exemple simple u(n+2) = 2.u(n+1)-u(n)+3
def recc2(u0,u1,n):
    (u,v) = (u0,u1)
    for i in range(n):
        (u,v) = (v,2*v-u+3)
    return u
```

2 PARCOURS DE LISTE

```
# somme des éléments d'une liste de nombres
def somme(liste):
    s = 0
    for k in liste:
        s += k
    return s

# maximum d'une liste de nombres
def maximum(liste):
    m = liste[0]
    for i in liste[1:]:
        if i > m:
            m = i
```

```
    return m

# (premier) indice du maximum d'une liste de nombres
def maxIndice(liste):
    m = 0
    for i in range(1,len(liste)):
        if liste[i] > liste[m]:
            m = i
    return m

# sous-liste des éléments > M
def listeSup(liste,M):
    res = []
    for i in liste:
        if i > M:
            res.append(i)
    return res

# Par compréhension
def listeSup(liste,M):
    return [i for i in liste if i>M]

# liste de 20 nombres aléatoires entre 1 et 6
[randrange(1,7) for i in range(20)]

# moyenne pondérée arrondie à 0,1
def moyPonderee(notes,coeffs):
    s,c = 0,0
    for i in range(len(notes)):
        s += notes[i]*coeffs[i]
        c += coeffs[i]
    return round(s/c,1)
```

3 TRIS DE LISTE AVEC EFFETS DE BORD

```
def triSelection(liste):
    for i in range(len(liste)):
        mini = i
        for j in range(i+1, len(liste)):
            if liste[j] < liste[mini]:
                mini = j
        liste[i],liste[mini] = liste[mini],liste[i]

def triInsertion(liste):
    for i in range(len(liste)):
        j = i
        while j>0 and liste[j] < liste[j-1]:
            liste[j],liste[j-1] = liste[j-1],liste[j]
            j = j-1
# ou
def triInsertion2(liste):
    for i in range(len(liste)):
        j = i
        carte = liste.pop(i)
        while j>0 and carte < liste[j-1]:
            j = j-1
        liste.insert(j,carte)

def triBulles(liste):
    chaos = True
    while chaos:
        chaos = False
        for i in range(len(liste)-1):
            if liste[i] > liste[i+1]:
                liste[i],liste[i+1]= liste[i+1],liste[i]
                chaos = True
#ou
def triBulles2(liste):
    fin = len(liste)
    while fin > 1:
        last = 0
        for i in range(fin-1):
            if liste[i] > liste[i+1]:
```

```
                liste[i],liste[i+1]= liste[i+1],liste[i]
                last = i+1
        fin = last
```

4 Fonctions

```
# tracé cos
x = linspace(0,2*pi, 100)
y = cos(x)
plot(x,y)

#dichotomie
def dichotomie(f,a,b,eps = 1e-3):
    while abs(b-a) > eps:
        c = (a+b)/2
        if f(c) == 0:
            return c
        if f(a)*f(c) < 0:
            b = c
        else:
            a = c
    return (a+b)/2
```

5 Tirages aléatoires