# TP 23- Proposition de solutions

**Solution 1** Permutations d'une liste

```
def permutations(L):
    if len(L)==0: return [[]]
    else:
        T,intermediaire=[],permutations(L[1:])
        for p in intermediaire:
            for i in range(len(L)):
                T=T+[p[:i]+[L[0]]+p[i:]]
    return T
```

**Solution 2** Sous-listes d'une liste

Méthode par disjonction suivant le premier élément des sous-listes : éléments parmi les éléments de la liste.

```
def sous_listes(L):
    if len(L)==0: return [[]]
    else:
        T=[[]]
        for i in range(len(L)):
            intermediaire=sous_listes(L[i+1:])
            for l in intermediaire:
                T.append([L[i]]+l)
    return T
```

Méthode par disjonction suivant le premier de la liste : soit il est pris, soit non.

```
def sous_listes(L):
    if len(L)<1: return [[]]
    else:
        T,S=sous_listes(L[1:]),[]
        for l in T:
            S=S+[l,[L[0]]+l]
        return S
```

**Solution 3** Les tours de Hanoï

```
def hanoi(n,A,B,C):
    '''deplacer les n galets de A sur B
       les galets sont numerotes de 0 a n'''
    if n==0:
        print('Deplacer',0,'de',A,'a',B)
        return
    hanoi(n-1,A,C,B)
    print('Deplacer',n,'de',A,'a',B)
    hanoi(n-1,C,B,A)
```

Le nombre $a_n$ d'étapes nécessaires pour déplacer $n$ galets vérifie : $a_1 = 1$ et pour $n \geq 1$, $a_{n+1} = 2a_n + 1$

La suite est arithmético-géométrique et on trouve :

$$\forall n \in \mathbb{N}, \qquad a_n = 2^n - 1$$

➜ Animation des tours d'Hanoï : (nouvelle version)

On introduit une liste de 3 listes pour suivre les galets sur les trois tours.

```
def hanoi_G(n,A,B,C):
    global L # liste de 3 listes pour les 3 tours
    if n==0:
        L.append([0,A,B])
        return
    hanoi_G(n-1,A,C,B)
    L.append([n,A,B])
    hanoi_G(n-1,C,B,A)

n=int(input('Donner n : '))
# Les rayons
R=[0.5+ k*0.5 for k in range(n)]
# Epaisseurs et hauteurs
e=0.5
H=[0.2+k*(e+0.2) for k in range(n)]
# Support
S=[(n+1)//2,(n+1)//2+n+1,(n+1)//2+2*n+2]
plt.fill([0,0,(n+1)//2*2+2*n+2,(n+1)//2*2+2*n+2,0],
                    [0,-0.5,-0.5,0,0],'b',ec='b')
for i in S:
    plt.fill([i-0.1,i-0.1,i+0.1,i+0.1,i-0.1],
             [0,H[-1]+2*e,H[-1]+2*e,0,0],'b',ec='b')
plt.axis('scaled')
plt.show()
# Initialisation des galets
def galet(i,r,h,c='r'):
    plt.fill([S[i]-R[r],S[i]-R[r],S[i]+R[r],
              S[i]+R[r],S[i]-R[r]],[H[h],H[h]+e,
              H[h]+e,H[h],H[h]],c,ec='none')
    if c=='w':
        plt.fill([S[i]-0.1,S[i]-0.1,S[i]+0.1,
              S[i]+0.1,S[i]-0.1],[H[h],H[h]+e,
              H[h]+e,H[h],H[h]],'b',ec='b')

T=[list(range(n-1,-1,-1)),[],[]]
for h,r in enumerate(T[0]):
    galet(0,r,h)
plt.draw()
# Deroulement du jeux
L=[]
hanoi_G(n-1,'0','2','1')
# Affichage
for l in L:
    plt.pause(0.5)
    r,d,a=l[0],int(l[1]),int(l[2])
    galet(d,r,len(T[d])-1,'w')
    galet(a,r,len(T[a]))
    T[a].append(T[d].pop())
    print('Deplacer',r,'de',d,'a',a)
    plt.draw()
```