```
## Exercice 2 : distance d'édition
def minimum(L): # fonction utilitaire
    n=len(L)
    mini=L[0]
    for i in range(1,n):
        if L[i]<mini:
            mini=L[i]
    return mini

def d_edition(str1,str2):
    n,p=len(str1),len(str2)
    D=[[0 for j in range(p+1)] for i in
range(n+1)]

    # initialisation de la matrice D
    for i in range(n):
        D[i][p]=n-i
    for j in range(p):
        D[n][j]=p-j

    # remplissage
    for i in range(n-1,-1,-1):
        for j in range(p-1,-1,-1):
            if str1[i]==str2[j]:
                D[i][j]=D[i+1][j+1]
            else:
                D[i][j]=1+minimum( [D[i+1][j],D[i]
[j+1],D[i+1][j+1] ])

    # affichage de la matrice
    for i in range(n+1):
            print(D[i])
    # affichage de la distance d'édition
    print('Distance entre les mots ',str1,' et
',str2,' : ', D[0][0])
```

```
## Récursivité sans mémoïsation
def d_edition2(str1,str2):
    # cas de base
    if str1=='' or str2='':
        return len(str1)+len(str2)
    # recursivite
    if str1[0]==str2[0]:
        return d_edition2(str1[1:],str2[1:])
    else:
        return 1+min( d_edition2(str1,str2[1:]),
d_edition2(str1[1:],str2[1:]),
d_edition2(str1[1:],str2))

## Récursivité avec mémoïsation
def d_edition2_mem(str1,str2):
    dico={}

    def distance(str1,str2):
        # cas de base
        if (str1,str2) in dico:
            return dico[ (str1,str2) ]
        if str1=='':
            dico[ ('',str2) ]=len(str2)
            return len(str2)
        if str2=='':
            dico[ (str1,'')] = len(str1)
            return len(str1)
        # recursivite
        if str1[0]==str2[0]:
            d=distance(str1[1:],str2[1:])
            dico[ (str1,str2) ] = d
            return d
        else:
            d= 1+min( distance(str1,str2[1:]),
distance(str1[1:],str2[1:]),
distance(str1[1:],str2))
            dico[ (str1,str2) ] = d
            return d
    return distance(str1,str2)
```