```python
# CORRECTION DEBUT tp2 LISTES.py

#TP 2 2023 2024 listes 1ERE PARTIE
##petits exos
## ex 1
def double_present(a,L): #dit si 2*a est ou non dans L
    for x in L:
        if x == 2*a:
            return True # on interrompt dès qu'on trouve 2 *a
    return False


def double_present(a,L): #sans utiliser la propriété du return ( demandé aussi au concours)
    statut = False
    i=0
    while i<len(L) and statut ==False:# ordre des tests important pour ne pas avoir de dépassement d'indice
        if L[i]== a:
            statut = True
        i = i+1
    return statut

def double_present(a,L):# la méthode élégante
    return 2*a in L #on retourne un booléen

## ex 2
def nbre_occurences(a,L): #retourne le nombre d'occurrences de a dans L
    s = 0
    for x in L:
        if x == a:
            s += 1
    return s

##ex 3
def modifie(L):
    for i in range(len(L)): #on parcourt la liste
        if L[i]%2 == 0:
            L[i] = 0
    return None

## ex4 autour du max
#maximum et liste des places
def maxi_et_nb_atteint(L):
    maxi = L[0]
    nbfois = 1
    for i in range(1,len(L)): #sinon, s'il y a un maximum en place numero 0,il sera noté deux fois
        if L[i]>maxi:
            maxi = L[i]
            nbfois = 1
        elif L[i]==maxi:
            nbfois += 1
    return maxi, nbfois



def maxi_et_place_des_max(L):
    maxi = L[0]
    liste_places =[0]
    for i in range(1,len(L)): #sinon, s'il y a un maximum en place numero 0,il sera noté deux fois
        if L[i]>maxi:
            maxi = L[i]
            liste_places=[i]
        elif L[i]==maxi:
            liste_places.append(i)
```

```
062|     return maxi, liste_places
063|
064| ##ex 5
065| # liste à l'envers
066|
067| def copie_miroir(L):
068|     """retourne une copie de L à l'envers"""
069|     return [L[len(L)-1-i] for i in range(len(L))]
070|
071|
072| def modifie_en_miroir(L):
073|     """modifie L en place en son image miroir """
074|     for i in range(len(L)//2): # attention , si on met len(L), on fait 2
échanges et tout revient à sa place
075|         L[len(L)-1-i], L[i] =  L[i], L[len(L)-1-i]
076|     return #
077|
078| ##
079| # exercices plus conséquents
080| ##
081|
082| ##ex 1 les cartes
083| n = 2023
084| paquet = list(range(1,n+1)) # attention au +1
085| while len(paquet) > 1 :
086|     u = paquet.pop(0)
087|     paquet.append(u)
088|     paquet.pop(0)
089| print(paquet)
090|
091| ##ex2
092| def verifie(L,n):
093|     """verfie que la liste L d'entiers ne comporte que des entiers entre 0 et
n"""
094|     for x in L:
095|         if x<0 or x>n:
096|             return False
097|     return True
098|
099| from random import randint
100|
101| def creeliste(longueur,n):
102|     L = [0]*longueur  #création d une liste de bonne longueur
103|     for k in range(longueur):
104|         L[k]=randint(0,n)
105|     return L
106|
107| def listeplaces(L):
108|     n = max(L)
109|     P = [[] for k in range(n+1)]
110|     for k in range(len(L)):
111|         P[L[k]].append(k)
112|     return P
113|
114| def pas_de_doublons(L):
115|     aux = listeplaces(L)
116|     for place in aux:
117|         if len(place) >1:
118|             return False
119|     return True
```