```
#Correction_info3

#EX1
from numpy import *

def puiss(n:int,A:array)->array:
    """renvoie A**n"""
    p,q=shape(A)
    assert p==q #on teste quand même si la matrice est carrée.
    P=eye(p)
    for k in range(1,n+1):
        P=dot(P,A)
    return P

def puissrec(n:int,A:array)->array:
    """renvoie A**n en récursif"""
    p,q=shape(A)
    assert p==q
    if n==0:
        return eye(p)
    else:
        return dot(A,puissrec(n-1,A))

#from numpy.linalg import *
#On importe le sous-module linalg et on appelle eigvals(A)... on aura
alors 3 valeurs propres distinctes, donc 3 vecteurs propres
indépendants qui définiront une base de diagonalisation : A est donc
diagonalisable.

def expo(n:int,A:array)->array:
    """calcule la somme partielle associée à exp(A)"""
    p,q=shape(A)
    assert p==q
    S=zeros((p,q)) #on fera bien attention à l'initialisation !
    for k in range(0,n+1):
        S=S+(1/factorial(k))*puiss(k,A)
    return S

#EX2
from numpy import *

def cherchermatrice(A:array)->tuple:
    """renvoie le max et sa place dans le tableau"""
    p,q=shape(A)
    #on va stocker la place et la valeur max au fur et à mesure qu'on
traverse la matrice.
    ind,max=(0,0),A[0,0]
    for i in range(0,p):
        for j in range(0,q):
            if A[i,j]>max:
                ind,max=(i,j),A[i,j]
            else:
                pass
    return ind,max
```



```
#EX3

#A=array([[0,1],[1,1]])

def fibo(n:int)->array:
    """renvoie le vecteur Xn=[[u_n],[u_n+1]]"""
    A=array([[0,1],[1,1]])
    if n==0:
        return array([[1],[1]])
    else:
        return dot(A,fibo(n-1))

#EX4

from random import *

def mataleatoire()->array:
    """renvoie une matrice aléatoire inversible"""
    M=zeros((2,2))
    while det(M)==0:
        for i in range(0,2):
            for j in range(0,2):
                M[i,j]=random()
    return M

from numpy.linalg import *

def mataleatoire2()->array:
    """renvoie une matrice aléatoire inversible et à coefficients
dans N, et son inverse"""
    M=zeros((2,2))
    while det(M)==0:
        for i in range(0,2):
            for j in range(0,2):
                M[i,j]=randint(1,1000)
    return M,inv(M)

def ordre(M:array,N:int)->int:
    """renvoie l'ordre éventuel d'une matrice donnée"""
    p,q=shape(M)
    assert p==q
    for k in range(1,N+1):
        for i in range(0,p):
            for j in range(0,q):
                if puiss(k,M)[i,j]==eye(p)[i,j]: #on teste si les
coefficients coïncident
                    return k
    return 0

#EX5

def listetomatrice(L:list)->array:
    """renvoie la matrice d'adjacence d'un graphe défini par la liste
des voisins"""
    n=len(L) #c'est le nombre de sommets
```

```
    M=zeros((n,n))
    for s in range(0,n): #on va parcourir les listes de voisins dans
L
        for v in L[s]:
            M[s,v]=1 #on met lorsque v fait partie des voisins de s
    return M
```