```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Oct 14 14:29:54 2021

@author: stephane
"""

#from math import factorial
#from random import random, randint
#from time import clock


# Exo 2

#def penaux():
#    ...

#nb_penaux = 10**5
#proba_o = sum(penaux() for _ in range(nb_penaux)) / nb_penaux

"""
>>> proba_o
0.86265
"""

# Exo 3


#def penaux_subite():
#    ...

#proba_o_sub = sum(penaux_subite() for _ in range(nb_penaux)) / nb_penaux

"""
>>> proba_o_sub
0.55542


Que dit la théorie ?

La probabilité qu'Olivier gagne (dans lepremier cas) vérifie :
p = 5/6 + 1/6 * 1/5 * p

>>> 5/6 / (1-1/30)
0.8620689655172414

Pas mal ! Et pour la mort subite :
p = 5/6 * 1/5 + (5/6*4/5 + 1/6*1/5) * p

>>> 1/6 / (1-(5/6*4/5+1/6*1/5))
0.5555555555555557

Trop fastoche...
"""

# Exo 4


#def penaux_5():
#    ...
```



```
#res = [0, 0, 0]
#for _ in range(nb_penaux):
#    res[penaux_5()] += 1

#freq = [r/nb_penaux for r in res]

#p_ol_5 = factorial(5)**2*sum(1/factorial(ol)/factorial(5-ol)*
#         (5/6)**ol*(1/6)**(5-ol)*sum(1/factorial(fr)/factorial(5-fr)*
#         (4/5)**fr*(1/5)**(5-fr) for fr in range(ol)) for ol in range(1, 6))

#p_fr_5 = factorial(5)**2*sum(1/factorial(fr)/factorial(5-fr)*
#         (4/5)**fr*(1/5)**(5-fr)*sum(1/factorial(fr)/factorial(5-fr)*
#         (5/6)**ol*(1/6)**(5-ol) for ol in range(fr)) for fr in range(1, 6))

"""
>>> freq
[0.28237, 0.3863, 0.33133]
>>> p_ol_5, p_fr_5
(0.38529938271604950, 0.08256148148148151)

"""

# Cadeaux pour les tris


#def tableau_aleatoire(n):
#    return [randint(1, 10*n) for _ in range(n)]


#def test_tri(tri, taille):
#    foo = tableau_aleatoire(taille)
#    t0 = clock()
#    tri(foo)  # si quelque chose (<> None) est renvoyé, poubelle !
#    t1 = clock()
#    return t1-t0

# Exo 5


#def tri_bulles(t):
#   ...

#foo = tableau_aleatoire(10)
#print(foo)
# [71, 87, 32, 98, 66, 61, 99, 41, 68, 14]

#tri_bulles(foo)
#print(foo)
# [14, 32, 41, 61, 66, 68, 71, 87, 98, 99]


"""
>>> test_tri(tri_bulles, 100)
0.0005693990005966043
>>> test_tri(tri_bulles, 1000)
0.04688860200076306
>>> test_tri(tri_bulles, 10000)
5.333151953000197
"""

# Exo 6
```

```
#def maxi(t):
#    ...


#def pos_maxi(t):
#    ...


#def pos_maxi_parmi(t, k):
#    """ position du maximum parmi les k premiers éléments (de 0 à k-1, donc)"""
    ...

# Exo 7


#def tri_selection(t):
#    ...

#foo = tableau_aleatoire(10)
#print(foo)
# [15, 45, 66, 55, 78, 31, 81, 38, 58, 41]

#tri_selection(foo)
#print(foo)
# [15, 31, 38, 41, 45, 55, 58, 66, 78, 81]


"""
>>> test_tri(tri_selection, 100)
0.0002116310006385902
>>> test_tri(tri_selection, 1000)
0.02172732899998664
>>> test_tri(tri_selection, 10000)
2.1681362630006333

Quadratique mais meilleur que le bulle. Avec un facteur entre 2 et 3.
"""

# Exos 8, 9 et 10


#def position(t, i):
#    ...


#def tri_insertion(t):
#    ...

#foo = tableau_aleatoire(10)
#print(foo)
#tri_insertion(foo)
#print(foo)

# [41, 66, 43, 57, 37, 22, 98, 38, 4, 20]
# [4, 20, 22, 37, 38, 41, 43, 57, 66, 98]

"""
>>> test_tri(tri_insertion, 100)
0.0003228599998692516
>>> test_tri(tri_insertion, 1000)
0.020980328998120967
```



```
>>> test_tri(tri_insertion, 10000)
2.1806308980012545

Quadratique; très semblable au tri sélection.
"""


#def position_dicho(t, i):
#    """ En quelle position doit-on insérer t[i] ?"""
#    ...


#def tri_insertion_bis(t):
#    ...

"""
>>> test_tri(tri_insertion_bis, 100)
0.00017717599985189736
>>> test_tri(tri_insertion_bis, 1000)
0.0047557660000165924
>>> test_tri(tri_insertion_bis, 10000)
0.17951199999879464
>>> test_tri(tri_insertion_bis, 100000)
27.3524443339993

C'est TRÈS nettement meilleur ! Quadratique chaotique...

Avant : C1 n^2 (insertion) + C2 n^2 (recherche)
Après : C1 n^2 (insertion) + C3 n.ln(n) (recherche)
Conclusion : on peut imaginer que C2 est de l'ordre de 10 C1.
"""
```