```
# ========================================================== #
# TRI A BULLE                                                #
# Evaluation expérimentale de la complexité moyenne.         #
# (nombre de comparaisons et nombre d'inversions effectuées). #
# Version du 21/11/2015                                      #
# ========================================================== #

# Importations
# ============
from random import randint
from time import perf_counter

# La fonction de tri
# ==================
def tri_bulle(L):
    global c, inv
    k = len(L)
    # Liste vide ou ne contenant qu'un élément
    if k<= 1:
        return L
    else:
        Done = False
        while not Done and k >= 2:
            c += (k-1)
            Done = True
            for i in range(k-1):
                if L[i] > L[i+1]:
                    inv += 1
                    L[i], L[i+1] = L[i+1], L[i]
                    Done = False
            k -= 1

# Saisie des paramètres et initialisations
# ========================================
c, inv = 0, 0
ntris = int(input('Combien de tris souhaitez-vous ? '))
n = int(input('\nLongueur des listes à trier ? '))
valmax = int(input('\nValeur maxi des nombres générés ? '))
complex = 0

for i in range(ntris):
    LNA = [randint(0,valmax) for i in range(n)]
    t_start = perf_counter()
    tri_bulle(LNA)
    t_end = perf_counter()
    complex += (t_end - t_start)

# Affichage de :
# (1) la valeur moyenne de "temps de tri/n²" qui doit être à peu près constant
# (2) la valeur moyenne de "nombre de comaparisons/n²" qui doit être proche de 1/2.
# (3) la valeur moyenne de "nombre d'inversions/n²" qui doit être proche de 1/4.
# ================================================================================
print('\nDurée du tri : ',complex/(ntris*n**2))
print('\nComparaisons : ',c/(ntris*n**2))
print('\nInversions : ',inv/(ntris*n**2))

# ===============
# FIN DU PROGRAMME
```