```
#ShowMergeSort.py
""" Contains two implementations of Merge and a recursive implementation
of MergeSort. An application script uses random examples to check that
the list returned from MergeSort is the same a the list produced via
the built-in sort method.
"""

from random import randint as randi

def Merge1(x,y):
    """ Returns a list of ints that is the merge of sorted lists x and y.

    Uses the pop method on copies of x and y

    PreC: x and y are lists of ints that are sorted from small to big.
    """
    # Make copies of x and y so as not to modify these lists in the caller
    u = list(x)
    v = list(y)
    # z will be built up through repeated appending
    z = []
    while len(u)>0 and len(v)>0 :
        if u[0]<= v[0]:
            g = u.pop(0)
        else:
            g = v.pop(0)
        z.append(g)
    # Either u or v is the empty list.
    # Append u onto z (no harm ih u is empty)
    z.extend(u)
    # Append v onto z (no harm if v is empty)
    z.extend(v)
    return z

def Merge2(x,y):
    """ Returns a list of ints that is the merge of sorted lists x and y.

    Uses the pop method on copies of x and y

    PreC: x and y are lists of ints that are sorted from small to big.
    """
    n = len(x)
    m = len(y)
    # ix and iy always indicate  the "next" element that is to be picked
    # from x and y respectively.
    ix = 0
    iy = 0
    # The merged list z will be constructed vis repeated appending.
    z = []
    for iz in range(n+m):
       # Append the next value to z
       if   ix>=n:
          # all elements of x have been merged
          z.append(y[iy]);  iy+=1
       elif iy>=m:
          # All elements of y have been merged
          z.append(x[ix]);  ix+=1
       elif x[ix] <= y[iy]:
            z.append(x[ix]);  ix+=1
         elif x[ix] > y[iy]:
            z.append(y[iy]);  iy+=1
    return z


def MergeSort(a):
    """ Returns a list of ints that is the sorted version of a.

    Uses the method of merge sort.

    PreC: a is a list of sorted ints
    """
    n = len(a)
    if n<=1:
       # Nothing to do if a is empty or if it consists of a single int
       return a
    else:
       # Split a into a pair of half-sized lists.
        m  = n/2
        u0 = a[:m]
        u1 = a[m:]
       # Sorth them both and merge the results.
        y0 = MergeSort(a[:m])
        y1 = MergeSort(a[m:])
        z = Merge1(y0,y1)        # Can also use z = Merge2(y0,y1)
        return z


if __name__ == '__main__':
    """ Chack Mergesort with the built in sort method."""
    n = 1000
    a = []
    for k in range(n):
       a.append(randi(0,2*n))
    z = MergeSort(a)
    a.sort()
    print '\nMergeSort(a) is the same as a.sort().'
    print z==a
```