def fib(x):
    """assumes x an int >= 0
    Returns Fibonacci of x"
    assert type(x) == int and x >= 0
    if x == 0 or x == 1:
        return 1
    else:
        return fib(x-1) + fib(x-2)

def testFib(n):
    fibs = []
    for i in range(n+1):
        nextFib = fib(i)
        fibs.append(nextFib)
    return fibs

-----------------

def removeElems(L1, L2):
    """Assumes that L1 and L2 are lists
    removes from L1 those elements that are also in L2
    does not mutate L2
    E.g., if L1 = [1,2,1] and L2 = [1,4], L1 will have the
    value [2] when removeElems returns"
    for i in range(len(L1)):
        if L1[i] not in L2:
            continue
        else:
            L1.pop(i)
    return

def getPerms(L):
    """Assumes that L is either a list or a str
    returns a list containing all permutations of the elements of L"
    if len(L) <= 1:
        return [L]
    permList = []
    for perm in getPerms(L[1:]):
        for i in range(len(L)):
            permList.append(perm[:i] + L[0:1] + perm[i:])
    print permList

def printPermsOfDiff(L1, L2):
    """Assumes L1 and L2 are lists
    removes all elements of L1 that are also in L2
    prints all the permutations of L1
    leaves L2 unchanged"
    L1 = [1,2,3,4,1,3]
    L2 = [1, 4]
    printPermsOfDiff(L1, L2)
def removeElems(L1, L2):
    """Assumes that L1 and L2 are lists
    removes from L1 those elements that are also in L2
    does not mutate L2
    E.g., if L1 = [1,2,1] and L2 = [1,4], L1 will have the
    value [2] when removeElems returns"
    badIndices = []
    for i in range(len(L1)):
        if L1[i] not in L2:
            continue
        else:
            badIndices.append(i)
    i = len(badIndices) - 1
    while i >= 0:
        L1.pop(badIndices[i])
        i -= 1

def getPerms(L):
    """Assumes that L is either a list or a str
    returns a list containing all permutations of the elements of L"
    if len(L) <= 1:
        return [L]
    permList = []
    for perm in getPerms(L[1:]):
        for i in range(len(L)):
            permList.append(perm[:i] + L[0:1] + perm[i:])
    return permList

def printPermsOfDiff(L1, L2):
    """Assumes L1 and L2 are lists
    removes all elements of L1 that are also in L2
    prints all the permutations of L1
    leaves L2 unchanged"
    removeElems(L1, L2)
    perms = getPerms(L1)
    print 'Permutations are', perms