Exercise 1

Here is the [lecture from 6.00.1x on generators](https://www.youtube.com/watch?v=BLWn90kEYMk). Additionally, you can also take a look at Chapter 8.3 in the textbook.

For the following problem, consider the following way to write a power set generator. The number of possible combinations to put n items into one bag is 2n. Here, items is a Python list. If need be, also check out the [docs on bitwise operators](https://wiki.python.org/moin/BitwiseOperators) (<<, >>, &, |, ~, ^).

# generate all combinations of N items

def powerSet(items):

N = len(items)

# enumerate the 2\*\*N possible combinations

for i in range(2\*\*N):

combo = []

for j in range(N):

# test bit jth of integer i

if (i >> j) % 2 == 1:

combo.append(items[j])

yield combo

As above, suppose we have a generator that returns every combination of objects in one bag. We can represent this as a list of 1s and 0s denoting whether each item is in the bag or not.

Write a generator that returns every arrangement of items such that each is in one or none of two different bags. Each combination should be given as a tuple of two lists, the first being the items in bag1, and the second being the items in bag2.

def yieldAllCombos(items):

"""

Generates all combinations of N items into two bags, whereby each

item is in one or zero bags.

Yields a tuple, (bag1, bag2), where each bag is represented as

a list of which item(s) are in each bag.

"""

Note this generator should be pretty similar to the powerSet generator above.

We mentioned that the number of possible combinations for N items into one bag is 2n. How many possible combinations exist when there are two bags? Think about this for a few minutes, then click the following hint to confirm if your guess is correct. Remember that a given item can only be in bag1, bag2, or neither bag -- it is not possible for an item to be present in both bags!

[How many possible combinations exist for N items into two bags?](https://courses.edx.org/xblock/block-v1:MITx+6.00.2x+1T2021+type@vertical+block@4f367aea066648389a812de4b5102d15?show_title=0&show_bookmark_button=0&recheck_access=1&view=student_view&format=Lecture%20Sequence)

* With two bags, there are 3n possible combinations available.
* With one bag we determined there were 2n possible combinations available by representing the bag as a list of binary bits, 0 or 1. Since there are N bits, and they can be one of two possibilities, there must be 2n possibilities.
* With two bags there thus must be 3n possible combinations. You can imagine this by representing the two bags as a list of "trinary" bits, 0, 1, or 2 (a 0 if an item is in neither bag; 1 if it is in bag1; 2 if it is in bag2). With the "trinary" bits, there are N bits that can each be one of three possibilities - thus there must be 3n possible combinations.

1

# Answer:

def yieldAllCombos(items):

"""

Generates all combinations of N items into two bags, whereby each item is in one or zero bags.

Yields a tuple, (bag1, bag2), where each bag is represented as a list of which item(s) are in each bag.

"""

N = len(items)

# Enumerate the 3\*\*N possible combinations

for i in range(3\*\*N):

bag1 = []

bag2 = []

for j in range(N):

if (i // (3 \*\* j)) % 3 == 1:

bag1.append(items[j])

elif (i // (3 \*\* j)) % 3 == 2:

bag2.append(items[j])

yield (bag1, bag2)