Practice – On MITx

```
import numpy as np
def find_combination(choices, total):
    """
    choices: a non-empty numpy.array of ints
    total: a positive int

    Returns result, a numpy.array of length len(choices)
    such that
        * each element of result is 0 or 1
        * sum(result*choices) == total
        * sum(result) is as small as possible
    In case of ties, returns any result that works.
    If there is no result that gives the exact total,
    pick the one that gives sum(result*choices) closest
    to total without going over.
    """
    counter = 1
    result = np.array([0 for i in range(len(choices))])
    while len(bin(counter)[2:]) <= len(choices):
        a = np.array(list(map(int, bin(counter)[2:].zfill(len(choices)))))

        if sum(a * choices) <= total:
            if total - sum(a * choices) < total - sum(result * choices):
                result = a
            elif total - sum(a * choices) == total - sum(result * choices):
                if sum(a) < sum(result):
                    result = a
        counter += 1
    return result

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

import random

def oneTrial():
    '''
    Simulates one trial of drawing 3 balls out of a bucket containing
    3 red and 3 green balls. Balls are not replaced once
    drawn. Returns True if all three balls are the same color,
    False otherwise.
    '''
    balls = ['r', 'r', 'r', 'r', 'g', 'g', 'g', 'g']
    chosenBalls = set([])
    for t in range(3):
        # For three trials, pick a ball
        ball = random.choice(balls)
        chosenBalls.add(ball)
        # Remove the chosen ball from the set of balls
        balls.remove(ball)
        # and add it to a set of balls we picked
        chosenBalls.add(ball)
    # If there's only one ball in the set then we picked the same ball each
time.
    return len(chosenBalls) == 1
```

```
def drawing_without_replacement_sim(numTrials):
    '''
    Runs numTrials trials of a Monte Carlo simulation
    of drawing 3 balls out of a bucket containing
    3 red and 3 green balls. Balls are not replaced once
    drawn. Returns the a decimal - the fraction of times 3
    balls of the same color were drawn.
    '''
    numTrue = 0
    for trial in range(numTrials):
        if oneTrial():
            numTrue += 1

    return float(numTrue)/float(numTrials)

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

def greedySum(L, s):
    current_sum = 0.0
    multiples = []
    for val in L:
        # find the largest multiple
        multiple = int( (s - current_sum) / val)
        current_sum += (val * multiple)
        multiples.append(multiple)

    if current_sum != s:
        return "no solution"
    else:
        return sum(multiples)
```