Python Exercises

In this lab, you will write many Python functions to practice using loops. You do not have to complete every function but should complete shuffledNumbers(n) and attempt to get as many done as possible in your lab time.

• **shuffledNumbers(n)**

Write a function implementing the following pseudocode:

1. Check that n is non-negative, and return an empty list otherwise
2. Initialize a length-n list “out” so that, for all i, entry i, in “out” has value i.
3. For i from n-1 down to 1
4. Pick an int j at random between 0 and i inclusive
5. If j is less than i, swap out[i] and out[j]
6. Return out

Tip: For step 4, call random.randint(0, i) and make sure to import random.

The following problems are sample problems courtesy of codingbat.com

• **string_times(str, n)**

Given a string and a non-negative int n, return a larger string that is n copies of the original string.
string_times('Hi', 2) → 'HiHi'
string_times('Hi', 3) → 'HiHiHi'
string_times('Hi', 1) → 'Hi'

• **front_times(str, n)**

Given a string and a non-negative int n, we'll say that the front of the string is the first 3 chars, or whatever is there if the string is less than length 3. Return n copies of the front;
front_times('Chocolate', 2) → 'ChoCho'
front_times('Chocolate', 3) → 'ChoChoCho'
front_times('Abc', 3) → 'AbcAbcAbc'

• **string_splosion(str)**

Given a non-empty string like "Code" return a string like "CCoCodCode".
string_splosion('Code') → 'CCoCodCode'
string_splosion('abc') → 'aababc'
string_splosion('ab') → 'aab'

• **last2(str)**

Given a string, return the count of the number of times that a substring length 2 appears in the string and also as the last 2 chars of the string, so "hixxxhi" yields 1 (we won't count the end substring).
last2('hixxhi') → 1
last2('xaxxaxaxx') → 1
last2('axxxaaxx') → 2

• **array_count9(nums)**

Given an array of ints, return the number of 9's in the array.
array_count9([1, 2, 9]) → 1
array_count9([1, 9, 9]) → 2
array_count9([1, 9, 9, 3, 9]) → 3

• **array_front9(nums)**

Given an array of ints, return True if one of the first 4 elements in the array is a 9. The array length may be less than 4.
array_front9([1, 2, 9, 3, 4]) → True
array_front9([1, 2, 3, 4, 9]) → False
array_front9([1, 2, 3, 4, 5]) → False

• **array123(nums)**

Given an array of ints, return True if the sequence of numbers 1, 2, 3 appears in the array somewhere.
array123([1, 1, 2, 3, 1]) → True
array123([1, 1, 2, 4, 1]) → False
array123([1, 1, 2, 1, 2, 3]) → True

• **string_match(a, b)**

Given 2 strings, a and b, return the number of the positions where they contain the same length 2 substring. So "xxcaazz" and "xxbaaz" yields 3, since the "xx", "aa", and "az" substrings appear in the same place in both strings.
string_match('xxcaazz', 'xxbaaz') → 3
string_match('abc', 'abc') → 2
string_match('abc', 'axc') → 0

• **count_evens(nums)**

Return the number of even ints in the given array. Note: the % "mod" operator computes the remainder, e.g. 5 % 2 is 1.
count_evens([2, 1, 2, 3, 4]) → 3
count_evens([2, 2, 0]) → 3
count_evens([1, 3, 5]) → 0

• **big_diff(nums)**

Given an array length 1 or more of ints, return the difference between the largest and smallest values in the array. Note: the built-in min(v1, v2) and max(v1, v2) functions return the smaller or larger of two values.
big_diff([10, 3, 5, 6]) → 7
big_diff([7, 2, 10, 9]) → 8
big_diff([2, 10, 7, 2]) → 8

• **centered_average(nums)**

Return the "centered" average of an array of ints, which we'll say is the mean average of the values, except ignoring the largest and smallest values in the array. If there are multiple copies of the smallest value, ignore just one copy, and likewise for the largest value. Use int division to produce the final average. You may assume that the array is length 3 or more.
centered_average([1, 2, 3, 4, 100]) → 3
centered_average([1, 1, 5, 5, 10, 8, 7]) → 5
centered_average([-10, -4, -2, -4, -2, 0]) → -3

• **sum13(nums)**

Return the sum of the numbers in the array, returning 0 for an empty array. Except the number 13 is very unlucky, so it does not count and numbers that come immediately after a 13 also do not count.
sum13([1, 2, 2, 1]) → 6
sum13([1, 1]) → 2
sum13([1, 2, 2, 1, 13]) → 6

• **sum67(nums)**

Return the sum of the numbers in the array, except ignore sections of numbers starting with a 6 and extending to the next 7 (every 6 will be followed by at least one 7). Return 0 for no numbers.
sum67([1, 2, 2]) → 5
sum67([1, 2, 2, 6, 99, 99, 7]) → 5
sum67([1, 1, 6, 7, 2]) → 4

• **has22(nums)**

Given an array of ints, return True if the array contains a 2 next to a 2 somewhere.
has22([1, 2, 2]) → True
has22([1, 2, 1, 2]) → False
has22([2, 1, 2]) → False

• **double_char(str)**

Given a string, return a string where for every char in the original, there are two chars.
double_char('The') → 'TThhee'
double_char('AAbb') → 'AAAAbbbb'
double_char('Hi-There') → 'HHii--TThheerree'

• **count_hi(str)**

Return the number of times that the string "hi" appears anywhere in the given string.
count_hi('abc hi ho') → 1
count_hi('ABChi hi') → 2
count_hi('hihi') → 2

• **cat_dog(str)**

Return True if the string "cat" and "dog" appear the same number of times in the given string.
cat_dog('catdog') → True
cat_dog('catcat') → False
cat_dog('1cat1cadodog') → True

• **count_code(str)**

Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" count.
count_code('aaacodebbb') → 1
count_code('codexxcode') → 2
count_code('cozexxcope') → 2

• **end_other(a, b)**

Given two strings, return True if either of the strings appears at the very end of the other string, ignoring upper/lower case differences (in other words, the computation should not be "case sensitive"). Note: s.lower() returns the lowercase version of a string.
end_other('Hiabc', 'abc') → True
end_other('AbC', 'HiaBc') → True
end_other('abc', 'abXabc') → True•

• **xyz_there(str)**

Return True if the given string contains an appearance of "xyz" where the xyz is not directly preceeded by a period (.). So "xxyz" counts but "x.xyz" does not.
xyz_there('abcxyz') → True
xyz_there('abc.xyz') → False
xyz_there('xyz.abc') → True