#!/usr/bin/python -tt

# Copyright 2010 Google Inc.

# Licensed under the Apache License, Version 2.0

# http://www.apache.org/licenses/LICENSE-2.0

# Google's Python Class

# http://code.google.com/edu/languages/google-python-class/

# Basic string exercises

# Fill in the code for the functions below. main() is already set up

# to call the functions with a few different inputs,

# printing 'OK' when each function is correct.

# The starter code for each function includes a 'return'

# which is just a placeholder for your code.

# It's ok if you do not complete all the functions, and there

# are some additional functions to try in string2.py.

# A. donuts

# Given an int count of a number of donuts, return a string

# of the form 'Number of donuts: <count>', where <count> is the number

# passed in. However, if the count is 10 or more, then use the word 'many'

# instead of the actual count.

# So donuts(5) returns 'Number of donuts: 5'

# and donuts(23) returns 'Number of donuts: many'

def donuts(count):

if 0 <= count < 10: x = 'Number of donuts: %d' %count

elif count >=10: x = 'Number of donuts: many'

return x

# B. both\_ends

# Given a string s, return a string made of the first 2

# and the last 2 chars of the original string,

# so 'spring' yields 'spng'. However, if the string length

# is less than 2, return instead the empty string.

def both\_ends(s):

x = s[:2]

y = s[-2:]

z = x + y

if len(s) <= 2: return ''

else: return z

# C. fix\_start

# Given a string s, return a string

# where all occurences of its first char have

# been changed to '\*', except do not change

# the first char itself.

# e.g. 'babble' yields 'ba\*\*le'

# Assume that the string is length 1 or more.

# Hint: s.replace(stra, strb) returns a version of string s

# where all instances of stra have been replaced by strb.

def fix\_start(s):

if len(s) != 0: return s[0] + s[1:].replace(s[0], '\*')

else: return 'String is null'

# D. MixUp

# Given strings a and b, return a single string with a and b separated

# by a space '<a> <b>', except swap the first 2 chars of each string.

# e.g.

# 'mix', pod' -> 'pox mid'

# 'dog', 'dinner' -> 'dig donner'

# Assume a and b are length 2 or more.

def mix\_up(a, b):

if len(a) and len(b) >=2:

x = a.replace(a[:2], b[:2])

y = b.replace(b[:2], a[:2])

return '%s %s' %(x, y)

else: return 'null'

# Provided simple test() function used in main() to print

# what each function returns vs. what it's supposed to return.

def test(got, expected):

if got == expected:

prefix = ' OK '

else:

prefix = ' X '

print '%s got: %s expected: %s' % (prefix, repr(got), repr(expected))

# Provided main() calls the above functions with interesting inputs,

# using test() to check if each result is correct or not.

def main():

print 'donuts'

# Each line calls donuts, compares its result to the expected for that call.

test(donuts(4), 'Number of donuts: 4')

test(donuts(9), 'Number of donuts: 9')

test(donuts(10), 'Number of donuts: many')

test(donuts(99), 'Number of donuts: many')

print

print 'both\_ends'

test(both\_ends('spring'), 'spng')

test(both\_ends('Hello'), 'Helo')

test(both\_ends('a'), '')

test(both\_ends('xyz'), 'xyyz')

print

print 'fix\_start'

test(fix\_start('babble'), 'ba\*\*le')

test(fix\_start('aardvark'), 'a\*rdv\*rk')

test(fix\_start('google'), 'goo\*le')

test(fix\_start('donut'), 'donut')

test(fix\_start(''),'String is null')

print

print 'mix\_up'

test(mix\_up('mix', 'pod'), 'pox mid')

test(mix\_up('dog', 'dinner'), 'dig donner')

test(mix\_up('gnash', 'sport'), 'spash gnort')

test(mix\_up('pezzy', 'firm'), 'fizzy perm')

# Standard boilerplate to call the main() function.

if \_\_name\_\_ == '\_\_main\_\_':

main()