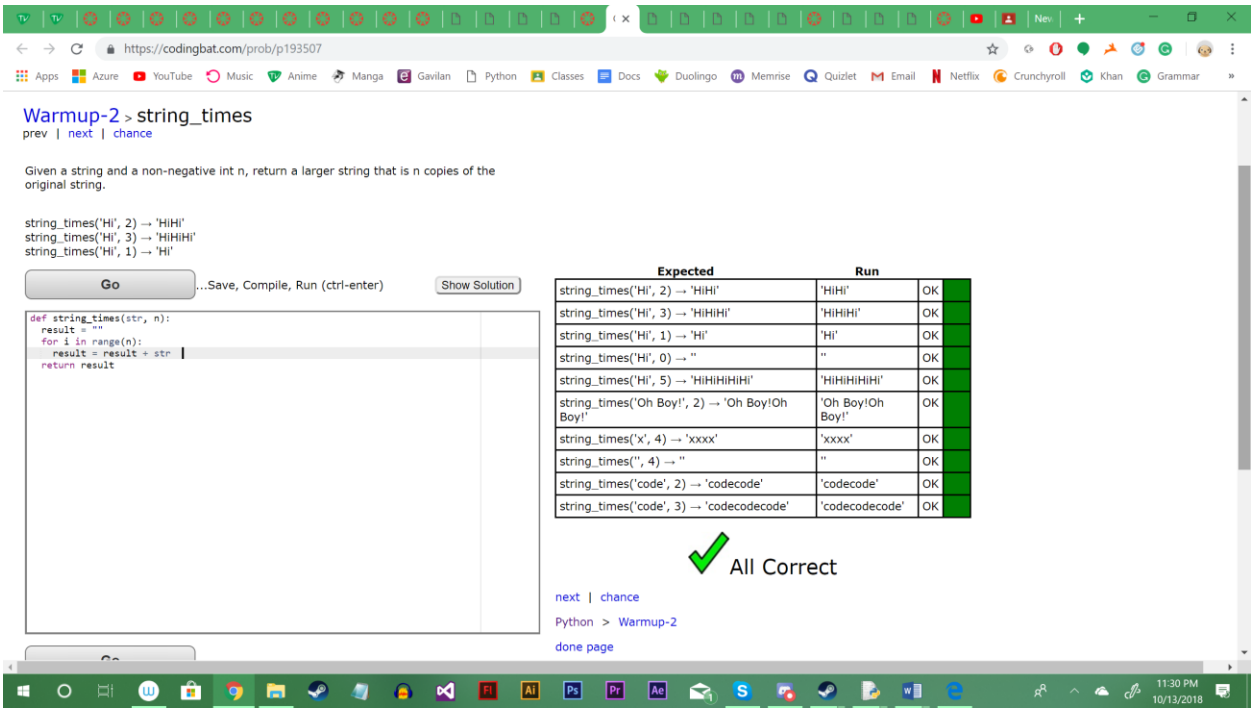
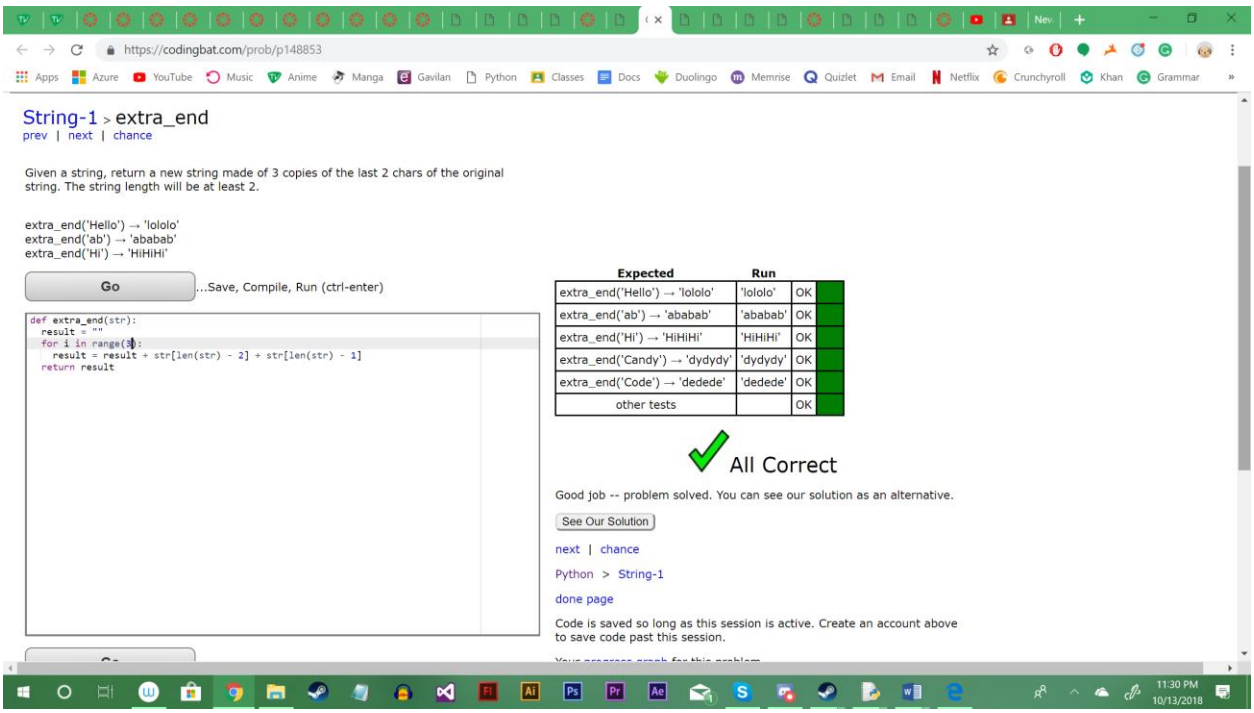
P36

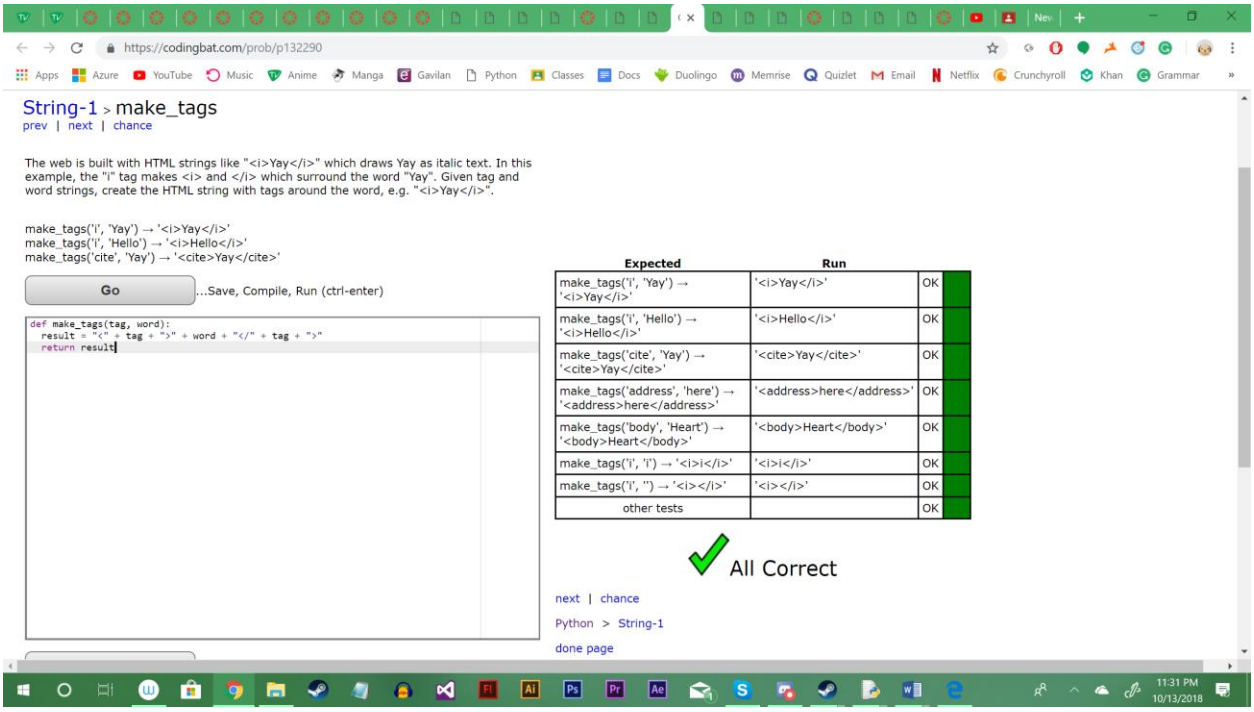
1.


https://codingbat.com/prob/p193507
Warmup-2 > string_times
prev | next | chance
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'
Go
...Save, Compile, Run (ctrl-enter)
Show Solution
def string_times(str, n):
result = ""
for i in range(n):
result = result + str
return result
Expected
Run
string_times('Hi', 2) → 'HiHi' 'HiHi' OK
string_times('Hi', 3) → 'HiHiHi' 'HiHiHi' OK
string_times('Hi', 1) → 'Hi' 'Hi' OK
string_times('Hi', 0) → '' '' OK
string_times('Hi', 5) → 'HiHiHiHiHi' 'HiHiHiHiHi' OK
string_times('Oh Boy!', 2) → 'Oh Boy!Oh Boy!' 'Oh Boy!Oh Boy!' OK
string_times('x', 4) → 'xxxx' 'xxxx' OK
string_times('', 4) → '' '' OK
string_times('code', 2) → 'codecode' 'codecode' OK
string_times('code', 3) → 'codecodecode' 'codecodecode' OK
All Correct
next | chance
Python > Warmup-2
done page

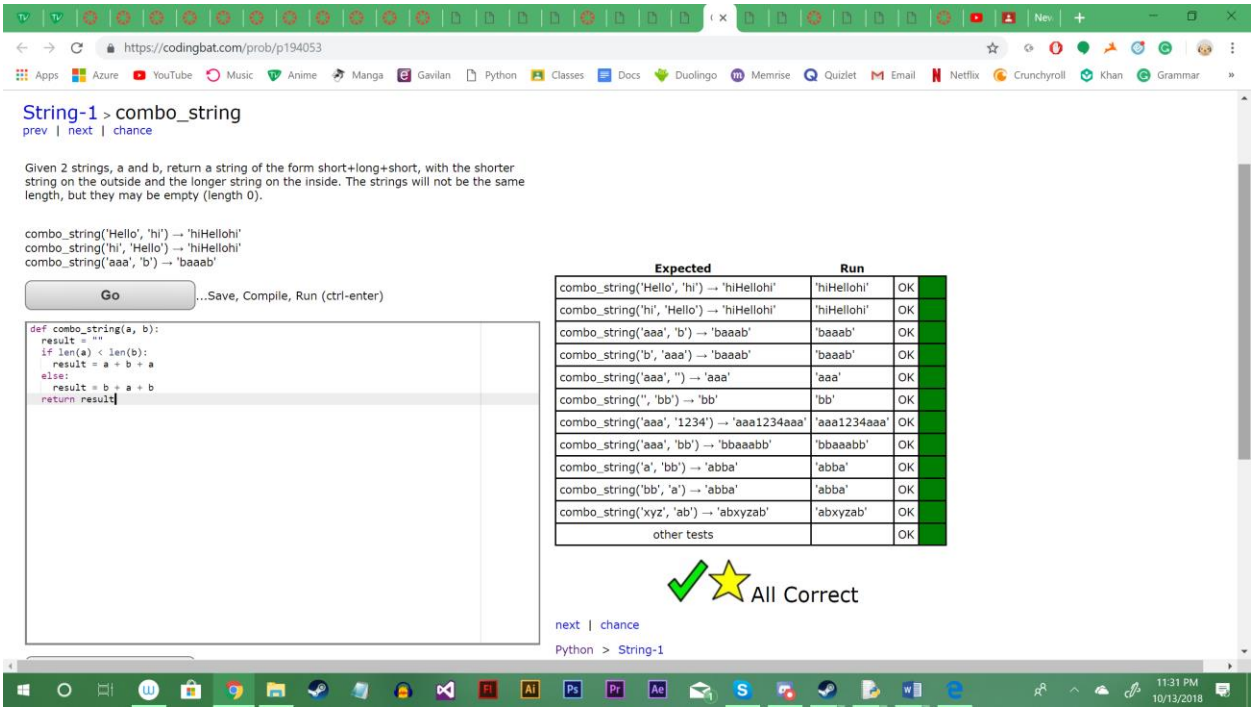
2.


https://codingbat.com/prob/p148853
String-1 > extra_end
prev | next | chance
Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.
extra_end('Hello') → 'lololo'
extra_end('ab') → 'ababab'
extra_end('Hi') → 'HiHiHi'
Go
...Save, Compile, Run (ctrl-enter)
def extra_end(str):
result = ""
for i in range(3):
result = result + str[len(str) - 2] + str[len(str) - 1]
return result
Expected
Run
extra_end('Hello') → 'lololo' 'lololo' OK
extra_end('ab') → 'ababab' 'ababab' OK
extra_end('Hi') → 'HiHiHi' 'HiHiHi' OK
extra_end('Candy') → 'dydydy' 'dydydy' OK
extra_end('Code') → 'dedede' 'dedede' OK
other tests OK
All Correct
Good job -- problem solved. You can see our solution as an alternative.
See Our Solution
next | chance
Python > String-1
done page
Code is saved so long as this session is active. Create an account above to save code past this session.

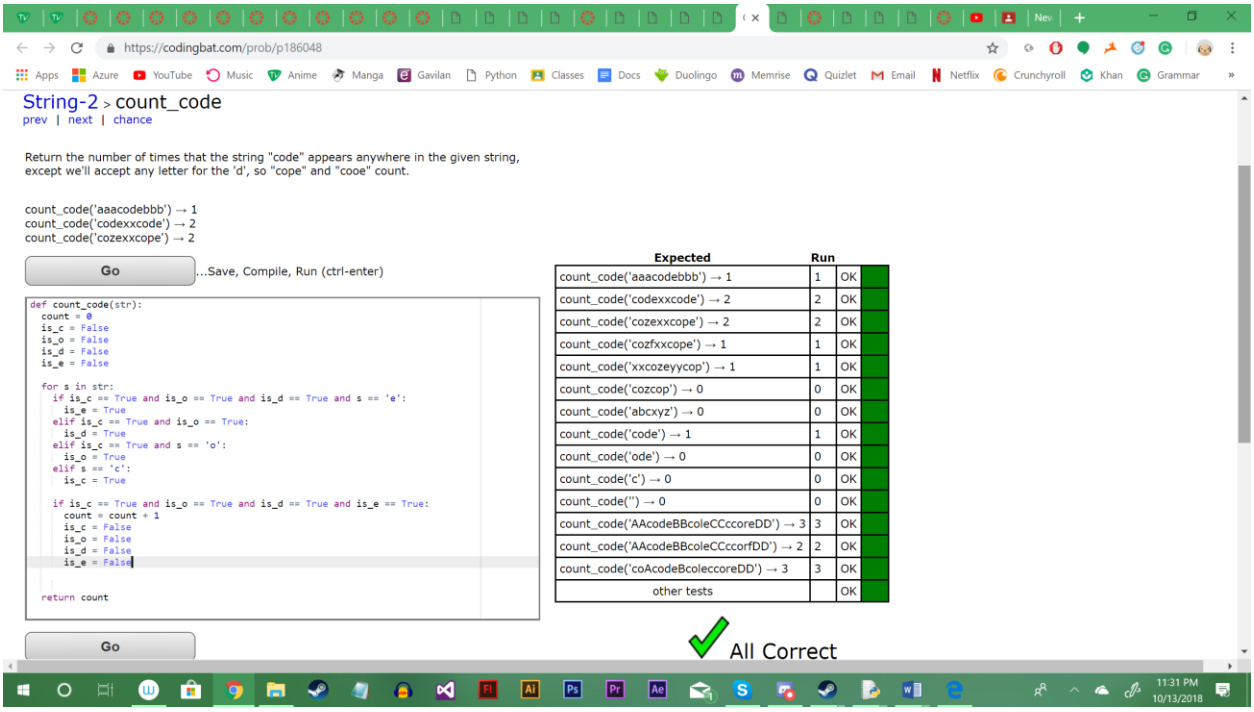
3.


https://codingbat.com/prob/p132290
String-1 > make_tags
prev | next | chance
The web is built with HTML strings like "Yay" which draws Yay as italic text. In this example, the "i" tag makes and which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "Yay".
make_tags('i', 'Yay') → 'Yay'
make_tags('i', 'Hello') → 'Hello'
make_tags('cite', 'Yay') → '<cite>Yay</cite>'
Go
...Save, Compile, Run (ctrl-enter)
def make_tags(tag, word):
result = "<" + tag + ">" + word + "</" + tag + ">"
return result
Expected
Run
make_tags('i', 'Yay') → 'Yay' 'Yay' OK
make_tags('i', 'Hello') → 'Hello' 'Hello' OK
make_tags('cite', 'Yay') → '<cite>Yay</cite>' '<cite>Yay</cite>' OK
make_tags('address', 'here') → '<address>here</address>' '<address>here</address>' OK
make_tags('body', 'Heart') → '<body>Heart</body>' '<body>Heart</body>' OK
make_tags('i', 'i') → 'i' 'i' OK
make_tags('i', '') → '' '' OK
other tests OK
All Correct
next | chance
Python > String-1
done page

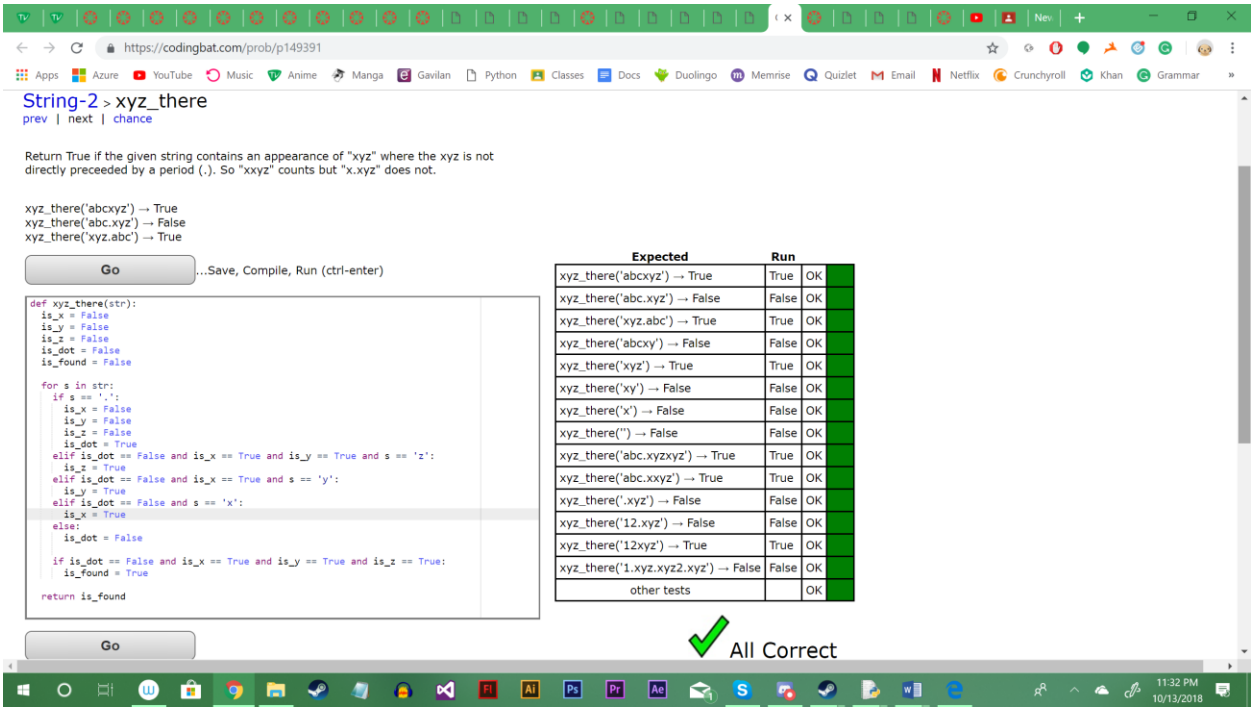
4.


https://codingbat.com/prob/p194053
String-1 > combo_string
prev | next | chance
Given 2 strings, a and b, return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).
combo_string('Hello', 'hi') → 'hiHellohi'
combo_string('hi', 'Hello') → 'hiHellohi'
combo_string('aaa', 'b') → 'baaab'
Go
...Save, Compile, Run (ctrl-enter)
def combo_string(a, b):
result = ""
if len(a) < len(b):
result = a + b + a
else:
result = b + a + b
return result
Expected
Run
combo_string('Hello', 'hi') → 'hiHellohi' 'hiHellohi' OK
combo_string('hi', 'Hello') → 'hiHellohi' 'hiHellohi' OK
combo_string('aaa', 'b') → 'baaab' 'baaab' OK
combo_string('b', 'aaa') → 'baaab' 'baaab' OK
combo_string('aaa', '') → 'aaa' 'aaa' OK
combo_string('', 'bb') → 'bb' 'bb' OK
combo_string('aaa', '1234') → 'aaa1234aaa' 'aaa1234aaa' OK
combo_string('aaa', 'bb') → 'bbaaabb' 'bbaaabb' OK
combo_string('a', 'bb') → 'abba' 'abba' OK
combo_string('bb', 'a') → 'abba' 'abba' OK
combo_string('xyz', 'ab') → 'abxyzab' 'abxyzab' OK
other tests OK
All Correct
next | chance
Python > String-1

5.


https://codingbat.com/prob/p186048
String-2 > count_code
prev | next | chance
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
Go
...Save, Compile, Run (ctrl-enter)
def count_code(str):
count = 0
is_c = False
is_o = False
is_d = False
is_e = False
for s in str:
if is_c == True and is_o == True and is_d == True and s == 'e':
is_e = True
elif is_c == True and is_o == True:
is_d = True
elif is_c == True and s == 'o':
is_o = True
elif s == 'c':
is_c = True
if is_c == True and is_o == True and is_d == True and is_e == True:
count = count + 1
is_c = False
is_o = False
is_d = False
is_e = False
return count
Go
Expected
Run
count_code('aaacodebbb') → 1 1 OK
count_code('codexxcode') → 2 2 OK
count_code('cozexxcope') → 2 2 OK
count_code('cozfxxcope') → 1 1 OK
count_code('xxcozeyycop') → 1 1 OK
count_code('cozcop') → 0 0 OK
count_code('abcxyz') → 0 0 OK
count_code('code') → 1 1 OK
count_code('ode') → 0 0 OK
count_code('c') → 0 0 OK
count_code('') → 0 0 OK
count_code('AAcodeBBcoleCCccoreDD') → 3 3 OK
count_code('AAcodeBBcoleCCccorfDD') → 2 2 OK
count_code('coAcodeBcoleccoreDD') → 3 3 OK
other tests OK
All Correct

6.


https://codingbat.com/prob/p149391
String-2 > xyz_there
prev | next | chance
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
Go
...Save, Compile, Run (ctrl-enter)
def xyz_there(str):
is_x = False
is_y = False
is_z = False
is_dot = False
is_found = False
for s in str:
if s == '.':
is_x = False
is_y = False
is_z = False
is_dot = True
elif is_dot == False and is_x == True and is_y == True and s == 'z':
is_z = True
elif is_dot == False and is_x == True and s == 'y':
is_y = True
elif is_dot == False and s == 'x':
is_x = True
else:
is_dot = False
if is_dot == False and is_x == True and is_y == True and is_z == True:
is_found = True
return is_found
Go
Expected
Run
xyz_there('abcxyz') → True True OK
xyz_there('abc.xyz') → False False OK
xyz_there('xyz.abc') → True True OK
xyz_there('abcxy') → False False OK
xyz_there('xyz') → True True OK
xyz_there('xy') → False False OK
xyz_there('x') → False False OK
xyz_there('') → False False OK
xyz_there('abc.xyzxyz') → True True OK
xyz_there('abc.xxyz') → True True OK
xyz_there('.xyz') → False False OK
xyz_there('12.xyz') → False False OK
xyz_there('12xyz') → True True OK
xyz_there('1.xyz.xyz2.xyz') → False False OK
other tests OK
All Correct