**Python Example Code**

This page shows a few Python example functions to go with the [CodingBat Python problems](http://codingbat.com/python). Examples:

* [If Boolean](http://codingbat.com/doc/python-example-code.html#ifboolean)
* [Strings](http://codingbat.com/doc/python-example-code.html#strings)
* [Lists](http://codingbat.com/doc/python-example-code.html#lists)
* [None result](http://codingbat.com/doc/python-example-code.html#none)

**If Boolean**

Python boolean operators are spelled out as the words "and" "or" "not", instead of the && syntax in other languages. The following a\_bigger() function should return True if the int parameter a is larger than b by 2 or more.

def a\_bigger(a, b):

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

return True

else:

return False

## Can all be written as just

## return (a > b and (a - b) >= 2)

Notice that the if-test does not need to be in parenthesis, unlike many languages. Forgetting the colon ":" that follows the test is a very common mistake when first learning Python.

Another technique is to return True with one more tests in the body of the function, leaving a return False at the bottom. In some ways, this is simpler than an if/else structure.

def a\_bigger(a, b):

if a > b and a-b >= 2:

return True

# More cases to return True in here

## Catch-all False at the bottom

return False

See the [Python If Boolean](http://codingbat.com/doc/python-if-boolean.html) doc for more information.

**Python Strings**

Make a string out of text by enclosing it in single or double quotes "like this", and use + to combine strings to make bigger strings. The with\_no() example function takes in a string and returns a new string with "No:" added at the front.

def with\_no(str):

return "No:" + str;

The function len(str) returns the length of a string, and str[i] returns the char at index i.

This two\_e() example method returns True if the string contains exactly two 'e' chars. The loopfor ch in str: is a standard loop which iterates over the chars in a string:

def two\_e(str):

count = 0

for ch in str: ## this loops over each char in the string

if ch == 'e':

count = count + 1

if count == 2:

return True

else:

return False

## this last if/else can be written simply as "return (count == 2)"

See the [Python Strings](http://codingbat.com/doc/python-strings.html) doc for more information.

The list example below shows another way to loop over a string or list using index numbers.

**Python Lists**

The same as with strings, the len() function returns the length of a list, and [i] accesses the ith element. The same loop as above, for num in nums:, will loop over all the values in a list. However, here is another way to do it: The function range(n) returns 0, 1, 2, ... n-1. This can be used to write a loop for i in range(len(list)): over the index numbers of a list (or string). This makes it easier to refer to relative (i-1) or (i+1) elements inside the loop. This pair\_13() example function returns True if the list contains a pair of 13's next to each other.

def pair\_13(nums) {

for i in range(len(nums) - 1):

if nums[i]==13 and nums[i+1]==13:

return True

return False ## if we get here, there was not a pair of 13's

## Note: the -1 inside the range() stops the loop one short of the full length,

## so the code in the loop can refer to nums[i+1]

See the [Python Lists](http://codingbat.com/doc/python-lists.html) doc for more information.

**None Result**

If a function does not include a "return", then by default it returns the special value None. So if all of your results are None, you probably just forgot to put in the return. Or perhaps the if/else structure doesn't call return in some cases, so in those cases None is the result.

[CodingBat.com](http://codingbat.com/) code practice.