

# Converting Iterative Code to Recursive Code

In this lesson, we will learn how to convert an iterative code into recursive code.

## We'll cover the following

- Steps for Converting Iterative Code to Recursive
  - Reverse a String
  - Explanation

The key to converting iterative code to recursive code is to find the specific lines of code that get transformed between the two implementations. Let's take a look at an example:

## Steps for Converting Iterative Code to Recursive #

1. Identify the **main** loop
  - This loop should modify one or more variables
  - It should return a result based on its final values.
2. Use the loop condition as the **base case** and the body of the loop as the **recursive case**.
3. The local variables in the iterative version turn into the parameters of the recursive version.
4. Compile and rerun tests.
5. Refactor the new function: You may be able to remove some temps and find a better structure for the conditional in the recursive function.

Let's take a look at an example:

## Reverse a String #

Reversing means that we take the string of code and flip it. The letters at the front go to the back, and vice versa.

H e l l o   W o r l d



d l r o W   o l l e H

String reversal

The codes below show the two implementations of the solution for reversing a string: iterative and recursive.



Notice the differences in the code before looking at the explanation:

Iterative

Recursive

```
1 def reverseString(string):
2     # Base Case
3     if len(string) == 0:
4         return string
5
6     # Recursive Case
7     else:
8         return reverseString(string[1:]) + string[0]
9
10 # Driver Code
11 targetVariable = "Educative"
12 print(reverseString(targetVariable))
```

Recursive method for reversing a string

## Explanation #

- **Step 1:** Identify the **loop** in the iterative code (**line number 5-7**). This loop has *two* features:
  - It will **modify** a variable
  - At the end, it will give an output

Analyze the iterative code. In each iteration, the last letter of the input string is added to another string, `reverse`. The iterator `length` for the input `string` is then reduced.

- **Step 2(a):** Convert the **loop condition** `length >= 0` to **base case** for recursive code.
  - This can be done by analyzing the situation for the last iteration, where any further iterations will stop.
  - In our case, the last condition will be `length == 0`. This will be our **base case**.
- **Step 2(b):** Convert the **body** of the loop to **recursive case**.
  - In the iterative version, we append the **last** letter of the input `string` to the beginning of a new string `reverse`. We then decrease the length of input `string`.
  - We now perform this recursively: `reverseString(string[1:]) + string[0]`.
  - We are pass the decreased `string` to the function `reverseString` but append the **first** letter to the **end** of the `string`.

- **Step 3:** We can skip step 3 since there are no major local variables. The variable `reverse` in our code, stores the result. We do not need to pass this as an input parameter.
- **Step 4 and 5:** Perform any required modifications to get the desired output.



In the next lessons, we will look at both the iterative and recursive methods of solving a problem. This will help you get into the groove of solving problems with recursion for the later chapters.

[← Back](#)[Next →](#)

Differences Between Iterative and Rec...

Count Vowels in a String

☒ **Mark as Completed**



Report an  
Issue



Ask a Question

([https://discuss.educative.io/tag/converting-iterative-code-to-recursive-code\\_\\_iteration-vs-recursion\\_\\_recursion-for-coding-interviews-in-python](https://discuss.educative.io/tag/converting-iterative-code-to-recursive-code__iteration-vs-recursion__recursion-for-coding-interviews-in-python))