The problem in normal order evaluation is that the function arguments used are evaluated when they’re being used which can lead to the functions being performed multiple times which can cause the program to crash or run a lot slower than when using eager evaluation. The example in this python code that can cause the functions to repeat is:

Def factorial(n):  
 if (n==0):  
 return 1  
 else:

Return n \* factorial(n-1)

This causes repetition by having the factorial function calculate each time before the multiplication takes place in the else statement.

A way to solve this problem is by rewriting the code where the factorial is calculated by multiplying the numbers 1 to 3 in a loop which is more efficient than the recursive call which causes repetition and slowness. The way to rewrite it would be:

Def factorial(n):  
 result = 1  
 for I in range(1, n+1):  
 result \*= i  
 return result