Adam Rolek

2/19/2020

Exercise 3:

s = [3, 4, 5, 6, 7, 8]  
i = 0  
j = 0  
k = 0  
for i in range(len(s) - 2):  
    for j in range(i + 1, len(s) - 1):  
        for k in range(j + 1, len(s)):  
            print(s[i], s[j], s[k])

Exercise 5:

def gcd(a, b):   
    if (b == 0):  
        return a  
    return gcd(b, a % b)  
  
  
a = 20  
b = 28  
print('GCD of', a, 'and', b, 'is', gcd(a, b))

Exercise 10:

For exercise 3 the basic operation of the algorithm would be the Iteration of the inner loop. The overall time complexity of this algorithm would be an element of O(n3). This is because each for loop would be O(n) and the fact that they are nested means they would be combined to O(n3).

For every step in the greatest common divisor algorithm there are 2 cases:

1. a%b will make b at most half of its previous value
2. a%b will make a at most half of its previous value since b is less than a / 2

So, at every step the algorithm will reduce at least one of the numbers to at least half of its previous value. At most this would be O(lg a) + O(lg b). Which can be reduced down to O(lg n) Where n is the upper limit of a and b.