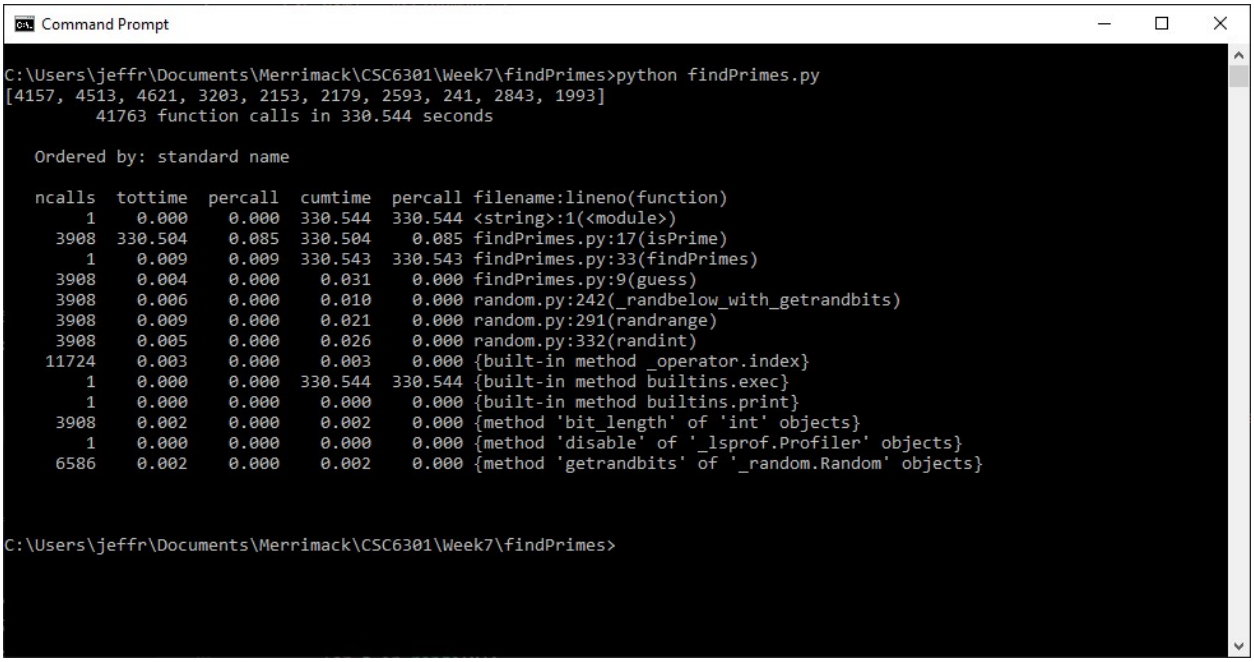
CSC-6301 – Assignment 7

Jeffrey Frost

The 1.0 original version took approximately 120 seconds to execute. Based off of the cProfile (Reference Figure 1 - Original Implementation (v1.0)), the line of code that took the longest to execute was the call to the isPrime function 330.504s out of the total 330.543s that the calling function findPrimes took. This means that isPrime took up 99.99% of the execution time of findPrime.



```
Command Prompt

C:\Users\jeffr\Documents\Merrimack\CSC6301\Week7\findPrimes>python findPrimes.py
[4157, 4513, 4621, 3203, 2153, 2179, 2593, 241, 2843, 1993]
         41763 function calls in 330.544 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000  330.544  330.544 <string>:1(<module>)
     3908  330.504    0.085  330.504    0.085 findPrimes.py:17(isPrime)
        1    0.009    0.009  330.543  330.543 findPrimes.py:33(findPrimes)
     3908    0.004    0.000    0.031    0.000 findPrimes.py:9(guess)
     3908    0.006    0.000    0.010    0.000 random.py:242(_randbelow_with_getrandbits)
     3908    0.009    0.000    0.021    0.000 random.py:291(randrange)
     3908    0.005    0.000    0.026    0.000 random.py:332(randint)
    11724    0.003    0.000    0.003    0.000 {built-in method _operator.index}
        1    0.000    0.000  330.544  330.544 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
     3908    0.002    0.000    0.002    0.000 {method 'bit_length' of 'int' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
     6586    0.002    0.000    0.002    0.000 {method 'getrandbits' of '_random.Random' objects}


C:\Users\jeffr\Documents\Merrimack\CSC6301\Week7\findPrimes>
```

Figure 1 - Original Implementation (v1.0)

For version 1.1, the isPrime function was refactored using a prime number finding implementation from CSC-6303 week 2. The isPrime function now took only 88ms to execute and the overall findPrimes function took 137ms. Reference the cProfile output in Figure 2 - New Implementation (v1.1). This time the total execution time of the isPrime function was brought down to just under 2 minutes, which is a huge time savings.

```
Command Prompt

C:\Users\jeffr\Documents\Merrimack\CSC6301\Week7\findPrimes>python findPrimes.py
[1151, 1907, 479, 1559, 197, 1907, 3719, 4283, 1787, 4931]
         76275 function calls in 0.140 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.137    0.137 <string>:1(<module>)
     7162    0.088    0.000    0.088    0.000 findPrimes.py:17(isPrime)
        1    0.006    0.006    0.137    0.137 findPrimes.py:38(findPrimes)
     7162    0.005    0.000    0.043    0.000 findPrimes.py:9(guess)
     7162    0.010    0.000    0.016    0.000 random.py:242(_randbelow_with_getrandbits)
     7162    0.012    0.000    0.033    0.000 random.py:291(randrange)
     7162    0.005    0.000    0.038    0.000 random.py:332(randint)
    21486    0.005    0.000    0.005    0.000 {built-in method _operator.index}
        1    0.003    0.003    0.140    0.140 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.print}
     7162    0.002    0.000    0.002    0.000 {method 'bit_length' of 'int' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
    11812    0.003    0.000    0.003    0.000 {method 'getrandbits' of '_random.Random' objects}


C:\Users\jeffr\Documents\Merrimack\CSC6301\Week7\findPrimes>
```

Figure 2 - New Implementation (v1.1)