Ethan Danitz

Prime Number Generator Documentation

6/19/25

In this python script, I have coded a prime number generator which calculates prime numbers less than 10,000,000. The goal of this project is to improve efficiency of the program across iterations and find a much faster way to generate prime numbers. To accomplish this, the time library is used to catch the timestamp before and after the calculation, which is shown in the output, and can be compared across versions. After the first implementation, ~11 seconds was a rough average for the calculation time.

In v1.0.0, I set up an array that stores prime numbers and filled it with 2, 3, and 5. There is a for loop which runs through every odd number from 7 to 10,000,000. In the loop, the iterator is first modded with 5 and if the result is 0, this iteration is skipped. This is because all numbers that end in 5 are divisible by 5, and therefore, running a check on that number would result in false every time. This saves some time. I considered different ways to set up a loop where it would skip 5 after every 4 numbers (7, 9, 1, 3), but no easy way came about without having to deal with the extra overhead of extra variables and calculations. A simple if-statement and modulus calculation in each iteration seemed good enough for v1.0.0. Second in the loop, the number is run through a check_prime method which returns true if it's prime, and false otherwise. If it's prime, the number is added to the array of primes and loop repeats.

In the check_prime method in v1.0.0, a bool flag is set false and keeps track of prime status of candidates. The candidate and a list of primes is passed in. Copying the list of primes into the method for each candidate is a large step which decreases efficiency and is a place for improvement. The square root of the candidate is calculated, which is the largest possible divisor that needs to be checked because all non-prime numbers can be expressed in prime factorization which is the multiplication of primes that produce that number. These prime factorizations must contain a prime number less than or equal to their square root. So, dividing candidates by all prime numbers less than or equal to its square root would show if it's prime or not. The candidate is modded by each prime and checked for an output of 0. If it's 0, it breaks out the loop and returns false. Each prime is checked if it's greater than the square root, and once it is, the flag is switched to true, and it breaks out of the loop and returns true.

To test accuracy of the implementation, the number of elements in the array is checked with the number of primes less than 10,000,000, which is 664579. If these match, accuracy is assumed.