

Euler 0010

The Problem:

We are finding primes again, this time all the primes up to 2000000!

We are going to modify the code that was created in Euler Problem 7 to generate up to a number instead of up to a prime count.

Then it is a simple matter of summing up all of the primes.

The issue... the last prime generator took 2.4 seconds to generate up to 100,000. So.. It's probably not going to make the Euler 1 minute guarantee if we are going to 20 times the number.

Considerations:

Yeah, I tried the more brutish generator and it didn't work... I'm going to try a different and possibly even stranger approach.

I am going to represent the threshold that we want to get to as an array of numbers. This does mean I am trading computation for space... but space is cheap.

The Approach:

We will start with an array of 2000000 and then cull out 0 and 1.

Then we start at 2, ignore it, and mark everything that is a multiple of 2.

Then we go to 3, ignore it, and mark everything that is a multiple of 3.

We will keep going until there are no more numbers to check.

Basically, it's the sieve of Eratosthenes.

The Code:

```
prime_threshold = 2000000
number_list = list(range(2, prime_threshold+1)) #initiating the sieve. Lot's of space here
```

```
on startup being used
prime_list = []

current = 0
total_iteration = 0
while current < len(number_list): #while we haven't iterated through the whole sieve yet
    if number_list[current] != -1: #find the next number that isn't marked
        prime = number_list[current]
        prime_list += [prime] #add it to our prime list, because it is prime.

        increment = current + prime
        while increment < len(number_list): #and go through the rest of the sieve marking
            numbers with it
                number_list[increment] = -1
                increment += prime
                total_iteration += 1
        current += 1

print(len(prime_list)) #look at the length of the prime list, this is the number of primes
under prime_threshold
print(total_iteration) #the amount of iterations that we made, much better than
recalculating primes for each number
print(sum(prime_list)) #the answer
```

Revision #2

Created 2025-08-11 14:56:11 UTC by Maxwell

Updated 2025-08-11 15:02:13 UTC by Maxwell