# Euler 0021

## The Problem:

Let d(n) be defined as the sum of proper divisors of (numbers less than n which divide evenly into n).
If d(a) = b and d(b) = a, where a =/= b, then a and b are an amicable pair and each of a and b are called amicable numbers.

For example, the proper divisors of 220 are 1,2,4,5,10,11,20,22,44,55 and 110; therefore d(220) = 284. The proper divisors of 284 are 1,2,4,71 and 142; so d(285) = 220.

Evaluate the sum of all the amicable numbers under 10000.

## Considerations and Approach:

We can generate divisors up to 10,000 for each number using a modified Euler 3.
We can store a dictionary of sums (this is probably unoptimal) as we go along.
When we store a sum, we can check if the sum is already in the dictionary. If it is and the dictionary value for that sum is the original number, we can add both the number and the sum into the dictionary.

## The Code:

```
import math


#generating the primes beforehand is incredibly helpful, since we really don't need to re-
brute force these
def generate_primes(upper):
    prime_threshold = upper
    number_list = list(range(2, prime_threshold+1))
    prime_list = []

    current = 0
    total_iteration = 0
```

```
    while current < len(number_list):
        if number_list[current] != -1:
            prime = number_list[current]
            prime_list += [prime]

            increment = current + prime
            while increment < len(number_list):
                number_list[increment] = -1
                increment += prime
                total_iteration += 1

        #print(number_list)
        #print(prime_list)
        current += 1

    return prime_list

def divisor_finder(number, primes):
    upper_limit = number
    #upper_limit = 20023110541 #, good to use to double check. Is [2,2,3,167]
    current_upper_limit = upper_limit
    sqrt_limit = math.sqrt(upper_limit)
    prime_factors = []

    #we are going to hardcode 2 and 3 for looping sake
    i = 0
    while primes[i] < sqrt_limit and i < len(primes):
        while current_upper_limit % primes[i] == 0:
            current_upper_limit = current_upper_limit/primes[i]
            prime_factors += [primes[i]]
        i += 1

    if current_upper_limit != 1:
        prime_factors += [current_upper_limit]

    #print(prime_factors)

    prime_counts = [[prime_factors[0], prime_factors.count(prime_factors[0])]]
    #print(prime_counts,prime_counts[-1][0],prime_factors[1])
```

```
    for i in range(1,len(prime_factors)):
        #print(prime_counts[-1][0],prime_factors[i])
        if  prime_counts[-1][0] != prime_factors[i]:
            prime_counts += [[prime_factors[i], prime_factors.count(prime_factors[i])]]

    #print(prime_counts)

    divisors = [1]
    counters = [x[1] for x in prime_counts]
    #print(counters)
    while sum(counters) > 0:
        new_divisor = 1
        for i in range(len(counters)):
            #print(prime_counts[i][0],counters[i], new_divisor)
            new_divisor *= prime_counts[i][0]**counters[i]
        divisors += [new_divisor]

        counters[0] -= 1
        for i in range(len(counters)):
            if counters[i] == -1:
                counters[i] = prime_counts[i][1]
                if i != len(counters) - 1:
                    counters[i+1] -= 1
            else:
                break

    divisors.sort()
    return divisors[:-1]



upper = 10000
prime_list = generate_primes(upper)
#print(prime_list)
all_amicables = []
number_sums = {}
for y in range(2,upper):
    number_sums[y] = sum(divisor_finder(y,prime_list))
    if number_sums[y] in number_sums:
        if number_sums[y] != y:
```

```
            if number_sums[number_sums[y]] == y:
                all_amicables += [number_sums[y], y]



# print(divisor_finder(220), divisor_finder(284))
# print(sum(divisor_finder(220)), sum(divisor_finder(284)))
print(number_sums[220],number_sums[284])
print(all_amicables)
print(sum(all_amicables))
```

---

Revision #2
Created 2025-12-01 20:41:58 UTC by Maxwell
Updated 2025-12-01 20:43:23 UTC by Maxwell