Euler 0035

The Problem:

Circular primes are primes that can be caesar'd and still be the prime all the way through.

Find how many circular primes there are below 1 million.

Considerations and Approach:

We are going to generate all the primes under 1000000, then iterate through all of them.
For each prime we will rotate it by it's length - 1 and if every rotation is a prime, then it is a circular prime

The Code:

```
import math

prime_threshold = 1000000
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

print(len(prime_list))

circ_primes = [2]

prime_list = [prime for prime in prime_list if '0' not in str(prime) and '2' not in
str(prime) and '4' not in str(prime) and '6' not in str(prime) and '8' not in str(prime)]

print(len(prime_list))

iterations = 0
for prime in prime_list:
    iterations += 1
    circular = True
    current = prime
    for i in range(1,int(math.log(prime, 10))+1):
        current = int(str(current)[-1] + str(current)[0:-1])
        #print(prime, current)
        #print(prime, current)
        if current not in prime_list:
            circular = False
            break

    if circular:
        circ_primes += [prime]

    if iterations % 10000 == 0:
        print(iterations/len(prime_list))

#print(circ_primes)
print(len(circ_primes))
```

Revision #2
Created 2026-04-20 19:57:26 UTC by Maxwell

Updated 2026-04-20 19:58:36 UTC by Maxwell