**6 kyu**

**The Sum of The First and The Last Prime Factor Make Chains of Numbers**

14198% of 2946 of77[raulbc777](https://www.codewars.com/users/raulbc777)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/the-sum-of-the-first-and-the-last-prime-factor-make-chains-of-numbers/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/the-sum-of-the-first-and-the-last-prime-factor-make-chains-of-numbers/solutions/python)

[Discourse (3)](https://www.codewars.com/kata/the-sum-of-the-first-and-the-last-prime-factor-make-chains-of-numbers/discuss/python)

* Add to Collection
* |
* Share this kata:

Every positive integer number, that is not prime, may be decomposed in prime factors. For example the prime factors of 20, are:

2, 2, and 5, because: 20 = 2 . 2 . 5

The first prime factor (the smallest one) of 20 is 2 and the last one (the largest one) is 5. The sum of the first and the last prime factors, sflpf of 20 is: sflpf = 2 + 5 = 7

The number 998is the only integer in the range [4, 1000] that has a value of 501 , so its sflpf equals to 501, but in the range [4, 5000] we will have more integers with sflpf = 501 and are: 998, 1996, 2994, 3992, 4990.

We need a function sflpf\_data() (javascript: sflpfData()that receives two arguments, val as the value of sflpf and nMax as a limit, and the function will output a sorted list of the numbers between 4 to nMax(included) that have the same value of sflpf equals to val.

Let's see some cases:

sflpf\_data(10, 100) == [21, 25, 63]

/// the prime factorization of these numbers are:

Number Prime Factorization Sum First and Last Prime Factor

21 = 3 . 7 ----> 3 + 7 = 10

25 = 5 . 5 ----> 5 + 5 = 10

63 = 3 . 3 . 7 ----> 3 + 7 = 10

sflpf\_data(10, 200) == [21, 25, 63, 105, 125, 147, 189]

sflpf\_data(15, 150) == [26, 52, 78, 104, 130]

(Advice:Try to discard primes in a fast way to have a more agile code)

Enjoy it!

<https://www.codewars.com/kata/the-sum-of-the-first-and-the-last-prime-factor-make-chains-of-numbers/python>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

// MAXN 100001

// stores smallest prime factor for every number

// int spf[MAXN];

// Calculating SPF (Smallest Prime Factor) for every

// number till MAXN.

// Time Complexity : O(nloglogn)

static List<int> sieve(int MAXN )

{

int[] spf = new int[MAXN];

spf[1] = 1;

for (int i = 2; i < MAXN; i++)

// marking smallest prime factor for every

// number to be itself.

spf[i] = i;

// separately marking spf for every even

// number as 2

for (int i = 4; i < MAXN; i += 2)

spf[i] = 2;

for (int i = 3; i \* i < MAXN; i++)

{

// checking if i is prime

if (spf[i] == i)

{

// marking SPF for all numbers divisible by i

for (int j = i \* i; j < MAXN; j += i)

// marking spf[j] if it is not

// previously marked

if (spf[j] == j)

spf[j] = i;

}

}

return spf.ToList();

}

// A O(log n) function returning primefactorization

// by dividing by smallest prime factor at every step

static List<int> getFactorization(int x, int[] spf)

{

List<int> ret = new List<int> ();

while (x != 1)

{

ret.Add(spf[x]);

x = x / spf[x];

}

return ret;

}

static int[] sflpf\_data(int val, int nMax)

{

//# your code here

//return []

List<int> ans = new List<int>();

int[] si = sieve(nMax + 1).ToArray();

for (int i = 4; i <= nMax; i++)

{

List<int> pf = getFactorization(i, si);

if (pf.Count > 1)

{

if(pf[0] + pf[pf.Count-1] == val)

{

ans.Add(i);

}

}

}

return ans.ToArray();

}

static void Main(string[] args)

{

//int[] res = sflpf\_data(15, 150);

//foreach(int item in res)

//{

// Console.Write(item + " ");

//}

List<int> si = sieve(100);

for(int i =0; i<si.Count; i++)

{

Console.Write(si[i] + " ");

}

Console.ReadLine();

}

}

}

**import** math

**def** sieve( MAXN ):

    spf = [0] \* MAXN;

    spf[1] = 1;

*#for (int i = 2; i < MAXN; i++)*

**for** i **in** range(2, MAXN):

*# marking smallest prime factor for every*

*# number to be itself.*

        spf[i] = i;

*# separately marking spf for every even*

*# number as 2*

*#for (int i = 4; i < MAXN; i += 2)*

**for** i **in** range(4, MAXN, 2):

        spf[i] = 2

*#for (int i = 3; i \* i < MAXN; i++)*

*#for i in range(3, MAXN, i \* i):*

    i = 3

**while**(i\*i < MAXN):

*#checking if i is prime*

**if** (spf[i] == i):

*# marking SPF for all numbers divisible by i*

*#for (int j = i \* i; j < MAXN; j += i)*

**for** j **in** range(i \* i, MAXN, i):

*#marking spf[j] if it is not*

*#previously marked*

**if** (spf[j] == j):

                    spf[j] = i

        i+=1

**return** spf

**def** getFactorization( x,  spf):

    ret = []

**while** (x != 1):

        ret.append(spf[x])

        x = x // spf[x]

**return** ret

**def** sflpf\_data(val, nMax):

    ans = []

    si = sieve(nMax + 1)

*#for (int i = 4; i <= nMax; i++)*

**for** i **in** range(4, nMax+1):

        pf = getFactorization(i, si)

**if**(len(pf) > 1):

**if**(pf[0] + pf[len(pf) - 1] == val): ans.append(i)

**return** ans

**print**(sflpf\_data(15, 150))

'''

s = ""

for i in range(0, len(lista)):

   s += str(lista[i] ) + " "

print(s)

'''