**6 kyu**

**Backwards Read Primes**

1991688% of 453391of 2,734[g964](https://www.codewars.com/users/g964)

Python

* [TRAIN](https://www.codewars.com/kata/backwards-read-primes/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/backwards-read-primes/solutions/python)

[Forks (4)](https://www.codewars.com/kata/backwards-read-primes/forks/python)

[Discourse (151)](https://www.codewars.com/kata/backwards-read-primes/discuss/python)

* |
* Share this kata:

Backwards Read Primes are primes that when read backwards in base 10 (from right to left) are a different prime. (This rules out primes which are palindromes.)

Examples:

13 17 31 37 71 73 are Backwards Read Primes

13 is such because it's prime and read from right to left writes 31 which is prime too. Same for the others.

**Task**

Find all Backwards Read Primes between two positive given numbers (both inclusive), the second one always being greater than or equal to the first one. The resulting array or the resulting string will be ordered following the natural order of the prime numbers.

**Example**

backwardsPrime(2, 100) => [13, 17, 31, 37, 71, 73, 79, 97] backwardsPrime(9900, 10000) => [9923, 9931, 9941, 9967] backwardsPrime(501, 599) => []

backwardsPrime(2, 100) => [13, 17, 31, 37, 71, 73, 79, 97]

backwardsPrime(9900, 10000) => [9923, 9931, 9941, 9967]

<https://www.codewars.com/kata/backwards-read-primes/python>

**def** \_try\_composite(a, d, n, s):

**if** pow(a, d, n) == 1:

**return** False

**for** i **in** range(s):

**if** pow(a, 2\*\*i \* d, n) == n-1:

**return** False

**return** True *# n  is definitely composite*

**def** is\_prime(n, \_precision\_for\_huge\_n=16):

**if** n **in** \_known\_primes **or** n **in** (0, 1):

**return** True

**if** any((n % p) == 0 **for** p **in** \_known\_primes):

**return** False

    d, s = n - 1, 0

**while** **not** d % 2:

        d, s = d >> 1, s + 1

*# Returns exact according to http://primes.utm.edu/prove/prove2\_3.html*

**if** n < 1373653:

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3))

**if** n < 25326001:

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3, 5))

**if** n < 118670087467:

**if** n == 3215031751:

**return** False

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3, 5, 7))

**if** n < 2152302898747:

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3, 5, 7, 11))

**if** n < 3474749660383:

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3, 5, 7, 11, 13))

**if** n < 341550071728321:

**return** **not** any(\_try\_composite(a, d, n, s) **for** a **in** (2, 3, 5, 7, 11, 13, 17))

*# otherwise*

**return** **not** any(\_try\_composite(a, d, n, s)

**for** a **in** \_known\_primes[:\_precision\_for\_huge\_n])

\_known\_primes = [2, 3]

\_known\_primes += [x **for** x **in** range(5, 1000, 2) **if** is\_prime(x)]

**def** revertir(n):

    rev = 0

**while** (n > 0):

        rev = (rev \* 10) + (n % 10)

        n = n // 10

**return** rev

*#print(revertir(1234))*

**def** backwardsPrime(start, stop):

*# your code*

    ans = []

**for** p **in** range(start, stop + 1):

**if**(is\_prime(p)):

            rev = revertir(p)

**if**(is\_prime(rev) **and** rev != p):

                ans.append(p)

**if**(rev >= start **and** rev <= stop):

                    ans.append(rev)

    d = list(set(ans))

    d.sort()

**return** d

**print**(backwardsPrime(9900, 10000))