**6 kyu**

**Simple nearest prime**

7286% of 4970 of139[KenKamau](https://www.codewars.com/users/KenKamau)

Python

* [TRAIN AGAIN](https://www.codewars.com/kata/simple-nearest-prime/train/python)
* [NEXT KATA](https://www.codewars.com/trainer/python)

Details

[Solutions](https://www.codewars.com/kata/simple-nearest-prime/solutions/python)

[Forks (5)](https://www.codewars.com/kata/simple-nearest-prime/forks/python)

[Discourse (27)](https://www.codewars.com/kata/simple-nearest-prime/discuss/python)

* Add to Collection
* |
* Share this kata:

In this Kata, you will be given a number and your task will be to return the nearest prime number.

solve(4) = 3. The nearest primes are 3 and 5. If difference is equal, pick the lower one.

solve(125) = 127

We'll be testing for numbers up to 10^10. 500 tests.

More examples in test cases.

Good luck!

If you like Prime Katas, you will enjoy this Kata: [Simple Prime Streaming](https://www.codewars.com/kata/5a908da30025e995880000e3)

<https://www.codewars.com/kata/simple-nearest-prime/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** solve(n):

    der = n

    izq = n

**while**(**not** is\_prime(der)):

        der += 1

**while**(izq >= 2  **and** **not** is\_prime(izq)):

        izq -= 1

**if**(n - izq <= der - n):

**return** izq

**return** der

[lgtech](https://www.codewars.com/users/lgtech)

**def solve(n):**

**print('starting with {0}'.format(n), flush=True)**

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

**if p % 2 == 0 :**

**return False**

**for x in range(3,int(p\*\*.5)):**

**if p % x == 0:**

**return False**

**return True**

**#return not any([p%x==0 for x in range(3,int(p\*\*.5))])**

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

**return n**

**step = (n%2) + 1**

**while 1:**

**if is\_prime(n-step):**

**return n-step**

**elif is\_prime(n+step):**

**return n+step**

**else:**

**step += 2**

**return None**

[Blind4Basics](https://www.codewars.com/users/Blind4Basics)

**from itertools import count**

**from bisect import bisect\_left as bisect**

**n = 100005 # Sieve up to (10\*\*10)\*\*.5 + 5**

**sieve, primes = [0]\*((n>>1)+1), [2] # optimized sieve (store only odd numbers**

**for i in range(3, n+1, 2):**

**if not sieve[i>>1]:**

**primes.append(i)**

**for j in range(i\*\*2>>1, (n+1)>>1, i): sieve[j] = 1**

**def solve(n):**

**if n%2 and n>>1 < len(sieve) and not sieve[n>>1]: return n # easy check: prime if sieve element is set to False**

**idx = bisect(primes,n) # Insertion point of n in the prime list**

**return bigSearch(n, idx) if idx == len(primes) \**

**else min( primes[max(0,idx-1):idx+2], key=lambda x: (abs(x-n), x)) # fast look up in the list of primes (check element before, under and after the insertion point)**

**def isPrime(n, iSq): return all(n%x for x in primes[:iSq])**

**def bigSearch(n, idx):**

**iSq = bisect(primes,n\*\*.5,0,idx) # insertion point of n\*\*.5 in the prime list (for complex primality check)**

**isPair = not n%2**

**for x in count(0):**

**for c in [-1,1]:**

**p = n+c\*(2\*x + isPair)**

**if isPrime(p, iSq): return p**