**6 kyu**

**Group prime numbers**

11100% of1529[NikosAthens](https://www.codewars.com/users/NikosAthens)

* Python
* 3.6.0

Instructions

Output

* Write a function generator that will generate the first n primes grouped in tuples of size m. If there are not enough primes for the last tuple it will have the remaining values as None.

Examples

For n = 11 and m = 2:

(2, 3), (5, 7), (11, 13), (17, 19), (23, 29), (31, None)

For n = 11 and m = 3:

(2, 3, 5), (7, 11, 13), (17, 19, 23), (29, 31, None)

For n = 11 and m = 5:

(2, 3, 5, 7, 11), (13, 17, 19, 23, 29), (31, None, None, None, None)]

For n = 3 and m = 1:

(2,), (3,), (5,)

Note: large numbers of n will be tested, up to 50000

[**https://www.codewars.com/kata/group-prime-numbers/solutions?show-solutions=1**](https://www.codewars.com/kata/group-prime-numbers/solutions?show-solutions=1)

**--------------MI SOLUCION NO ACEPTADA----------------**

**def esPrimo(n):**

**if(n < 2): return False**

**if(n == 2): return True**

**if(n % 2 == 0): return False**

**i = 3**

**while (i \* i <= n):**

**if(n % i == 0):**

**return False**

**i += 2**

**return True**

**def get\_primes(how\_many, group\_size=2):**

**# yield**

**arr = []**

**cont = 0**

**i = 2**

**while(cont < how\_many):**

**if(esPrimo(i)):**

**arr.append(i)**

**cont += 1**

**i += 1**

**'''**

**for j in range(0, len(arr)):**

**print( j+1, " ", arr[j])**

**'''**

**i = 0**

**cont = 0**

**grupo = []**

**r = []**

**while i < len(arr):**

**grupo.append(arr[i])**

**cont += 1**

**if(cont >= group\_size):**

**r.append(tuple(grupo))**

**cont = 0**

**grupo = []**

**i += 1**

**#print(cont )**

**#print(i)**

**if(i <= how\_many and cont > 0):**

**while(cont < group\_size):**

**grupo.append(None)**

**cont += 1**

**r.append(tuple(grupo))**

**return r**

**#res = get\_primes(11, 5)**

**res = get\_primes(33,34)**

**print(res)**

**---------------------**

* **# generate primes up to limit**
* **LIMIT = 10\*\*6**
* **sieve = [0]\*2 + list(range(2, LIMIT))**
* **for n in sieve:**
* **if n:**
* **for i in range(n\*n, LIMIT, n):**
* **sieve[i] = 0**
* **PRIMES = list(n for n in sieve if n)**
* **def get\_primes(n, m=2):**
* **primes\_ = PRIMES[:n] + [None] \* m**
* **return ( tuple(primes\_[i:i+m]) for i in range(0, n, m) )**
  + - Best Practices2
    - Clever0
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=598cd87e4f4214876d000060&review_id=593e95e33bb47ac8bd0002b5)
  + [Link](https://www.codewars.com/kata/reviews/593e95e33bb47ac8bd0002b5/groups/598cd87e4f4214876d000060)
* [ChristianECooper](https://www.codewars.com/users/ChristianECooper)
* **# Known primes**
* **primes = [2, 3, 5, 7, 11]**
* **# Generate prime numbers in sequence**
* **def gen\_primes():**
* **yield from primes**
* **candidate = primes[-1]**
* **candidate\_root = int(candidate \*\* 0.5) + 1**
* **while True:**
* **candidate += 2**
* **if candidate\_root \* candidate\_root < candidate:**
* **candidate\_root += 1**
* **for p in primes:**
* **if candidate % p == 0:**
* **break**
* **if p > candidate\_root:**
* **primes.append(candidate)**
* **yield candidate**
* **break**
* **def get\_primes(how\_many, group\_size=2):**
* **result = []**
* **for i, p in enumerate(gen\_primes(), start=1):**
* **result.append(p)**
* **if len(result) == group\_size:**
* **yield tuple(result)**
* **result = []**
* **if i == how\_many:**
* **break**
* **if len(result) > 0:**
* **while len(result) < group\_size:**
* **result.append(None)**
* **yield tuple(result)**
  + - Best Practices1
    - Clever1
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=593fe86c56e7c76419000969&review_id=593e95e33bb47ac8bd0002b5)
  + [Link](https://www.codewars.com/kata/reviews/593e95e33bb47ac8bd0002b5/groups/593fe86c56e7c76419000969)
* [OlegRadchenko](https://www.codewars.com/users/OlegRadchenko)
* **def get\_primes(how\_many, group\_size):**
* **primes = [False] \* 2 + list(range(2, 13\*how\_many))**
* **for x in primes:**
* **if x:**
* **for i in range(x\*x, 13\*how\_many, x):**
* **primes[i] = False**
* **seq = sorted(set(primes))[1:how\_many+1]**
* **while len(seq) % group\_size != 0:**
* **seq += [None]**

**return (tuple(seq[x:x+group\_size]) for x in range(0, len(seq), group\_size))**

* + - Best Practices1
    - Clever0
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=5a1f57cbfebd6d27a10000b8&review_id=593e95e33bb47ac8bd0002b5)
  + [Link](https://www.codewars.com/kata/reviews/593e95e33bb47ac8bd0002b5/groups/5a1f57cbfebd6d27a10000b8)
* [Blind4Basics](https://www.codewars.com/users/Blind4Basics)
* **def makePrimes(n):**
* **sieve, primes = [0]\*(n+1), []**
* **for i in range(2, n+1):**
* **if not sieve[i]:**
* **primes.append(i)**
* **for j in range(i\*\*2, n+1, i): sieve[j] = 1**
* **return primes**
* **PRIMES = makePrimes(650000) # => 52831 prime numbers**
* **def get\_primes(how\_many, group\_size=2):**
* **lst = PRIMES[:how\_many] + [None] \* (group\_size - how\_many%group\_size)**
* **for n in range(how\_many//group\_size + bool(how\_many%group\_size)):**

**yield tuple(lst[n\*group\_size : (n+1)\*group\_size])**

* + - Best Practices1
    - Clever0
  + 0
  + [Fork](https://www.codewars.com/kumite/new?group_id=5974d8aa1b1cea96ce000b08&review_id=593e95e33bb47ac8bd0002b5)
  + [Link](https://www.codewars.com/kata/reviews/593e95e33bb47ac8bd0002b5/groups/5974d8aa1b1cea96ce000b08)
* [Voile](https://www.codewars.com/users/Voile)
* **from itertools import islice, chain, repeat**
* **def gen\_primes():**
* **D = {}; q = 2**
* **while True:**
* **if q not in D: yield q; D[q \* q] = [q]**
* **else:**
* **for p in D[q]: D.setdefault(p + q, []).append(p)**
* **del D[q]**
* **q += 1**
* **def get\_primes(how\_many, group\_size=2):**

**return (t for t in zip(\*[chain(islice(gen\_primes(), how\_many), repeat(None, group\_size-1))]\*group\_size))**