**Approach : Brute Force**

1) divide n by i starting from i=1 till k-factors found(worst case goes till n)

2) store all factors in an array

3) return k-th factor

Time -> O(n)

Space -> O(n) (to store the factors)

**Code:**

factors = []

count = 0

for i in range(1,n+1):

if n%i==0:

factors.append(i)

count += 1

if k==count:

return i

return -1

**Approach: Efficient/ less than n**

1) find ony factors till root(n)

2) Every factor must have its pair, that is if we have a d that can divide n then n/d can also divide n

3) Except 1 case: n/d = d eg to find factors of 4 till root(4) -> 2 we have [1,2] and 4/2 = 2 d=2 and n/d=2

4) store 1 to root(n) in 1 array

5) store reverse pairs in another(to create a sorted array of factors)

6) return k-th factor

Time -> O(sqrt(n))

Space -> O(sqrt(n)) -> atmost 2\*sqrt(n) factors available

**Code:**

factors = []

f2 = []

count = 0

root = int(sqrt(n))

for i in range(1, root+1):

if n%i==0:

factors.append(i)

f2.append(n//i)

# remove matching or mid factor if exists

if factors[-1] == f2[-1]:

f2.pop()

factors += f2[::-1]

if len(factors) >= k:

return factors[k-1]

return -1