```
import math
from math import sqrt, floor, log
import numpy as np
import time

def prime_sieve(N):
    is_prime = np.full(N, True)
    is_prime[:2] = False
    for p in range(2, math.isqrt(N) + 1):
        if is_prime[p]:
            is_prime[p*p::p] = False
    return np.nonzero(is_prime)[0]

def cptenondivisibles(N, L):
    return sum(all(n % p for p in L) for n in range(1, N))

class Primes():
    def __init__(self, N):
        self.__primes = prime_sieve(N)
    def __str__(self):
        return str(self.__primes)
    def __iter__(self):
        return iter(self.__primes)
    def __len__(self):
        return self.__primes.size
    def __getitem__(self, k):
        return self.__primes[k]
    def __contains__(self, x):
        k = self.index(x)
        return k < len(self) and self.__primes[k] == x
    def index(self, x):
        return np.searchsorted(self.__primes, x, side='left')
    def count(self, x):
        return np.searchsorted(self.__primes, x, side='right')
    def range(self, start, stop, step=1):
        return self.__primes[self.index(start):self.index(stop):step]
    def factors(self, n):
        if n in self:
            return np.array([n])
        else:
            P = self.range(2, n//2 + 1)
            return P[n % P == 0]

tic = time.time()
n = 100000000
P = Primes(int(n/2))
res1 = floor(n**0.5)
nplusmu = P.count(res1)
res2 = floor(n**(1/3))
m = P.count(res2)
print('racine carree : ',res1, ' pi(racinecarree) = ',nplusmu,'le ',nplusmu,'-
ieme nombre premier est ',P[nplusmu],' racine cubique ',res2, 'pi(racinecubique)
= ',m)
mu = nplusmu-m
cumulclnd = 0
for p1 in P[0:m]:
  jusquou = floor(n/p1)
  listepremiersapasser = P[0:P.index(p1)]
  if len(listepremiersapasser) > 0:
    clnd = cptenondivisibles(jusquou,listepremiersapasser)
    cumulclnd = cumulclnd+clnd
premierepartie = floor(n/2)-cumulclnd-1
sommeinterne = 0
for s in range(1,int(mu+1)):
```

```
  sommeinterne = sommeinterne+P.count(floor(n/P[m+s-1]))
print('pi(x) global en utilisant formule de Meissel : ',int(premierepartie-
sommeinterne+int(m*(mu+1))+(mu*(mu-1)/2)-1))
tac = time.time()
print(tac-tic,' s. pour calculer pi(',n,').')
```