```
from math import floor
from random import gauss

def gauss_int(a, b):
    n = b + 1
    while n > b or n < a:
        n = floor(gauss(b, (b-a)))
    return n

if __name__ == '__main__':
    count = [0] * 39
    for i in range(1000000):
        count[gauss_int(0, 38)] += 1
    print(count)


------------------------------------------------------------------------------

"""This is the standard Python 3.6 implementation of choices  """

from random import random
import itertools as _itertools
import bisect as _bisect

def choices(population, weights=None, *, cum_weights=None, k=1):
    """Return a k sized list of population elements chosen with replacement.

    If the relative weights or cumulative weights are not specified,
    the selections are made with equal probability.

    """
    if cum_weights is None:
        if weights is None:
            _int = int
            total  = len(population )
            return [population [_int(random() * total)] for i in range(k)]
        cum_weights  = list(_itertools .accumulate (weights ))
    elif weights is not None:
        raise TypeError('Cannot specify both weights and cumulative weights  ')
    if len(cum_weights ) != len(population ):
        raise ValueError('The number of weights does not match the population  ')
    bisect = _bisect.bisect
    total = cum_weights [-1]
    return [population [bisect(cum_weights , random() * total)] for i in range(k)]
```