def primetest(n):
    if n==1:
        return False
    i=2
    while i<=n^(.5):
        if n%i==0:
            return False
        i=i+1
    return True

def primelist(limit):
    output=[]
    n=2
    while n<=limit:
        if primetest(n):
            output.append(n)
        n=n+1
    return output

def smallest_prime_divisor(n):
    i=2
    while i<=n:
        if n%i==0 and primetest(i):
            return i
        i=i+1
    return

def experiment(): #experiment for rolling a die, \n    success is if a prime number comes up
    roll=randint(1,6)
    return primetest(roll)
experiment()  #just test to make sure there are \no errors
True
def trials(n):
    #this function repeats the 
experiment n times, counts the number of successes, and returns 
the ratio (number of successes)/n
    i=1
    success=0
    while i<=n:
        if experiment():
            #run the experiment. If true, 
increase the success count.
            success=success+1
        i=i+1
    return (success+0.0)/n  #add the 0.0 to turn the result \into a decimal, instead of a fraction
trials(10)  #do 10 trials
0.800000000000000
trials(100)  #do 100 trials
0.520000000000000
trials(1000)  #do 1000 trials
0.492000000000000
trials(10000)  
0.489800000000000

testbag=['R', 'R', 'R', 'G', 'G', 'Y', 'Y', 'Y', 'Y']  #this is the list \representing the bag of balls: 3 red, 2 green, 4 yellow
def experiment():
    #this experiment \picks a ball, takes it out of the bag, then picks a second ball. \Success if the two balls have the same color
    a=randint(0,8)  #choose first ball
    ball1=testbag[a]
    new_bag=testbag[:a]+testbag[a+1:]  #remove the chosen ball from the bag
    b=randint(0,7)
    ball2=new_bag[b]  #choose second ball
    if ball1==ball2:  #see if they are equal
        return True
    return False
experiment()  #again, just do one trial run to make sure there \are no obvious coding errors
False
trials(100000)  #do 100,000 trials.
0.277260000000000

5.0/18
0.27777777777778

def experiment():  #this experiment picks a random \n    number from 1 to 100,000. success if the number happens to be \n    prime.
    a=randint(1,10^5)
    return is_prime(a)  #you could use primetest(a) or \n    is_prime(a) here. The latter is the built-in Sage function, and \n    is faster than ours.

trials(1000000)
0.095794000000000

# This tells us that about 1/10 of the numbers from 1 to 10^5 are \nprime. That’s a lot of primes!