```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
# See PyCharm help at https://www.jetbrains.com/help/pycharm/

# Miller-Rabin algorithm. It will work for numbers of arbitrary size.

import random
# need to find the s and t
# t = result from multiple division of 2
# s = power of 2 from the division

def decomposeN(n):
  s = 0
  t = n - 1
  while t % 2 == 0:
    s += 1
    t //= 2
  return s, t

def miller_rabin_test_for_base_a(n, a, s, t):
  if pow(a, t, n) == 1:  # a^t%n==1 => n prime
    return True
  for i in range(s):
    if pow(a, 2**i * t, n) == n - 1:
      return True
  return False   # n composite

def next_prime_value(a):
  a += 1
  while not check_prime(a):
    a += 1
  return a

def check_prime(n):
  if(n<2):
    return False
  if(n>2 and n%2==0):
    return False
  for d in range(3,int(n ** 0.5) + 1,2):  # Check odd divisors only
    if n % d == 0:
      return False
  return True
```

```
def check_if_prime_using_miller_rabin(n, nr_of_iterations):
    # small numbers
    if n <= 1 or n % 2 == 0: #composite
        return False
    if n == 2 or n == 3:   # prime
        return True
    # decompose n to find s and t
    s, t = decomposeN(n)
    a = 2
    # Miller-Rabin test 'nr_of_iterations' times
    for iteration in range(nr_of_iterations):
        print(f"Iteration {iteration + 1}: Testing base a = {a}")
        if not miller_rabin_test_for_base_a(n,a,s,t):
            return False  # composite
        a = next_prime_value(a)
    return True
# 409 => prime
# 413 => not
if __name__ == "__main__":
    n = 409
    nr_of_iterations = 3
    s,t = decomposeN(n)

    is_prime = check_if_prime_using_miller_rabin(n,nr_of_iterations)
    if is_prime:
        print("N is likely prime.")
    else:
        print("N is composite.")
```