**AA\_LAB\_03\_Assignment**

Aim :- Write a Program to Implement Randomized Primality Testing using Fermat's Method.

Code :-

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jul 31 16:14:36 2020

@author: DHRUV

"""

import random as rn

def find\_gcd(num1, num2): # Find GCD of two numbers

if (num1 < num2):

return (find\_gcd(num2, num1))

elif (num1 % num2 == 0):

return num2

else:

return ((find\_gcd(num2, num1 % num2)))

def find\_power(a, num1, num2): # Find power of any number

result = 1

a = a % num2

while(num1 > 0):

if (num1 & 1):

result = (result \* a) % num2

num1 = num1 // 2

a = (a \*\* 2) % num2

return result

def isprime(num): # Check whether the number is prime or not

k = 10 # testing Variable

if (num <= 1 or num == 4):

return False

if (num <= 3):

return True

while (k > 0):

r = rn.randint(2, num-1)

print(r)

if (find\_gcd(num, r) != 1):

return False

if (find\_power(r, num - 1, num) != 1):

return False

k -= 1

return True

if \_\_name\_\_ == "\_\_main\_\_": # Main function

num = int(input("Enter the large number : "))

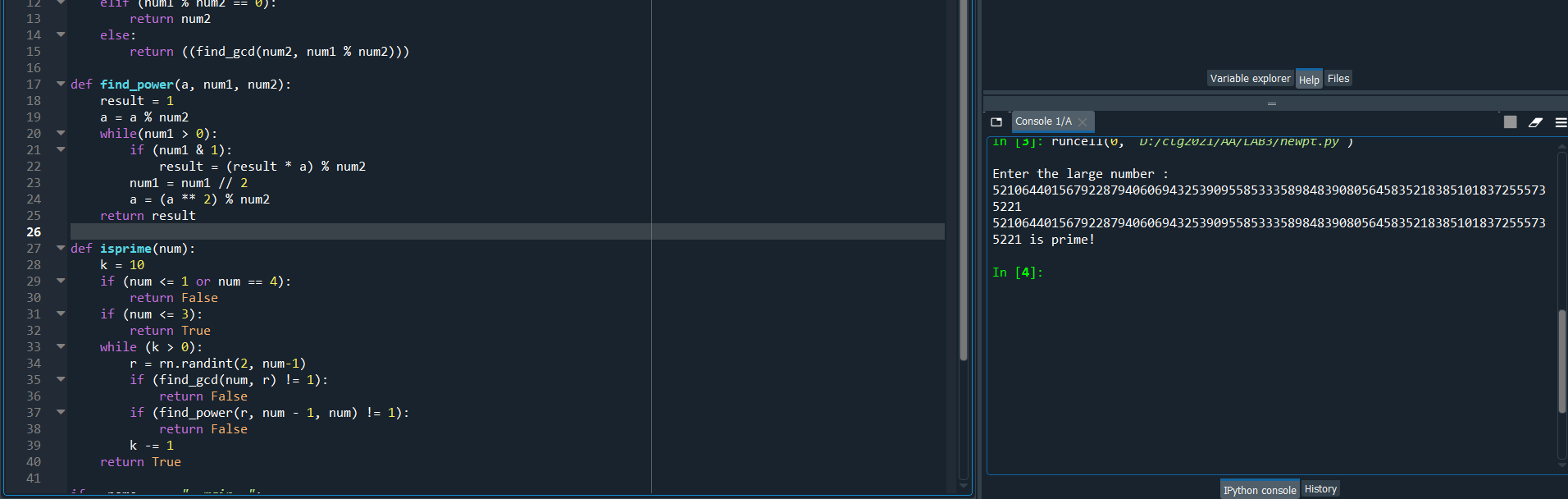
if (isprime(num)):

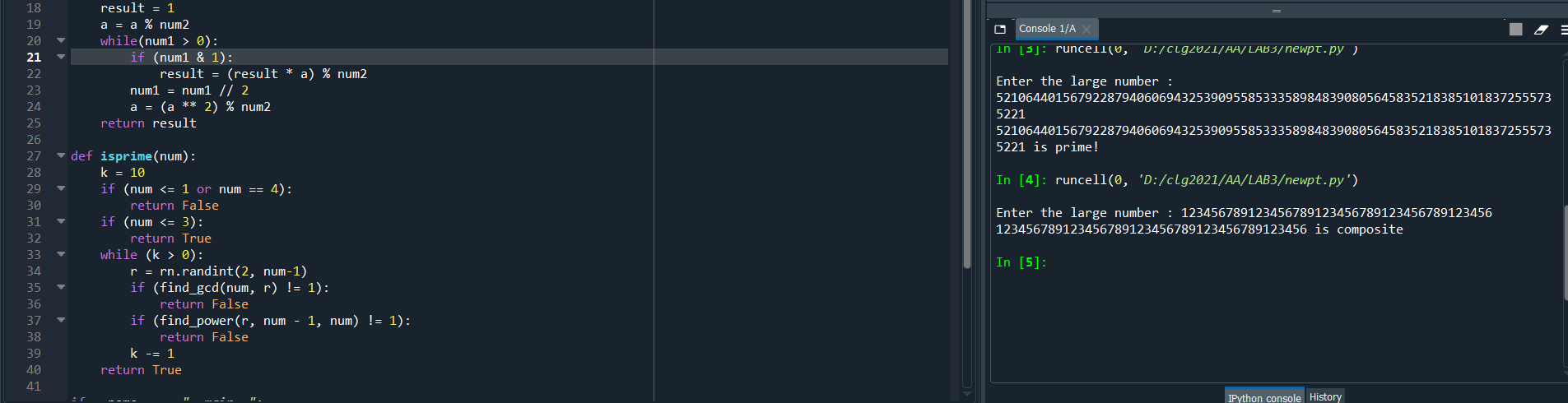
print(f"{num} is prime!")

else:

print(f"{num} is composite")

Output :-





* About Algorithm :-

Time complexity of this solution is O(k Log n). Note that power function takes O(Log n) time.

Note that the above method may fail even if we increase number of iterations (higher k). There exist some composite numbers with the property that for every a < n, gcd(a, n) = 1 and **an-1 ≡ 1 (mod n).** Such numbers are called [Carmichael numbers](https://en.wikipedia.org/wiki/Carmichael_number). Fermat’s primality test is often used if a rapid method is needed for filtering, for example in key generation phase of the RSA public key cryptographic algorithm.