Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №5**

**Дисциплина: Криптографические протоколы**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Цель работы:** реализовать программный продукт, позволяющий проводить тест Ферма чисел на простоту. Провести от одного до ста тестов для каждого числа. Сформировать список составных чисел до одного миллиарда для которых первое прохождение теста выдает ответ простое, отметить для каждого из этих чисел, какой ответ будет выдавать тест в случае проведения от 2 до 100 тестов.

**Ход работы:**

Программный продукт состоит из следующих функций, которые применяются (вызываются) последовательно:

1) is\_prime(n, k=5): Эта функция выполняет тест Ферма на простоту числа n с k тестами. Тест Ферма основан на малой теореме Ферма, которая гласит, что если p - простое число и a - целое число, не делится на p, то a(p-1) ≡ 1 (mod p).  
Если это условие не выполняется, то число n определенно составное. Если условие выполняется, то число n вероятно простое. Функция выполняет этот тест k раз (по умолчанию выполняется 5 тестов) с различными случайными значениями a. Функция представлена на рисунке 1.

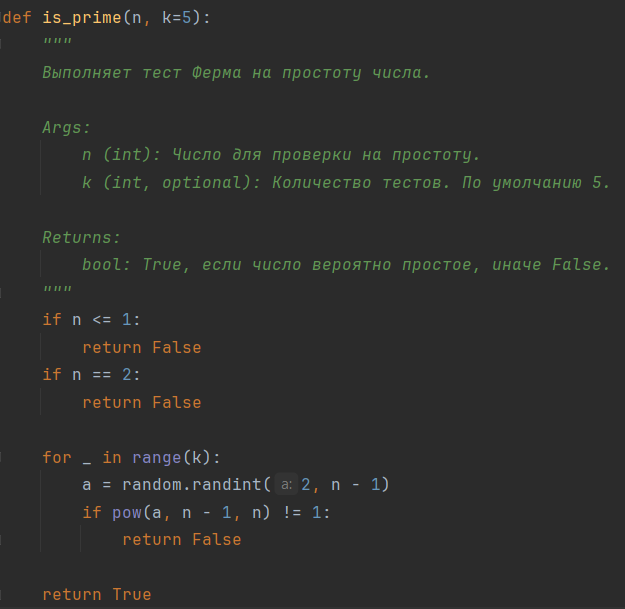


Рисунок 1 – Тест Ферма.

2) find\_composite\_numbers(limit, num\_tests): Эта функция находит составные числа до заданного предела limit, используя написанную выше функцию is\_prime для проведения num\_tests тестов на каждое число. Составное число — это число, которое имеет больше двух делителей. Если функция is\_prime возвращает False для числа, это число добавляется в список составных чисел. Фрагмент кода представлен на рисунке 2.

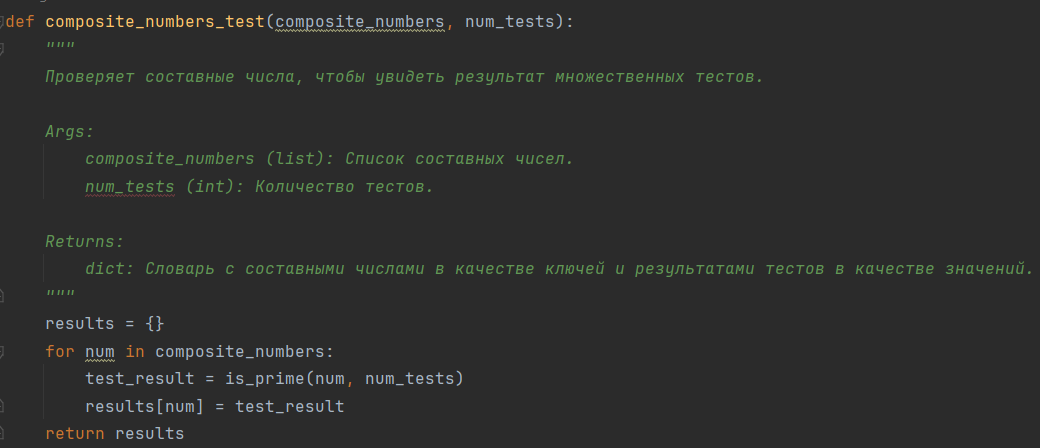


Рисунок 2 – Проверка составных чисел.

3) composite\_numbers\_test(composite\_numbers, num\_tests): Эта функция проверяет составные числа из списка composite\_numbers, чтобы увидеть результаты num\_tests тестов. Это делается для исследования вероятностной природы теста Ферма. Для каждого числа из списка composite\_numbers выполняется num\_tests тестов Ферма, и результаты сохраняются в словаре. Эта функция представлена на рисунке 3.

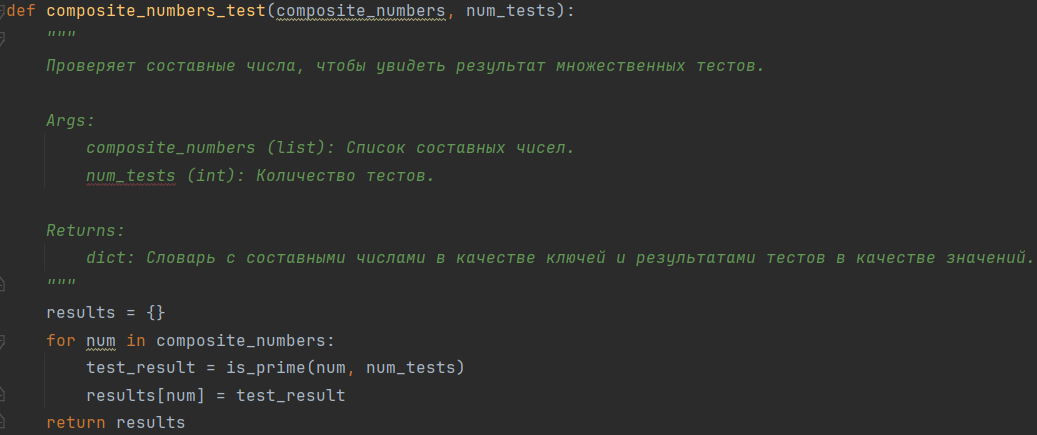


Рисунок 3 – “Прогонка” составного числа на заданном количестве тестов.

В конце программы выполняются следующие действия: находятся составные числа до одного тысячи (если необходимо, то можно задать большее количество чисел), затем проверяются составные числа с 2 до 100 тестов и выводятся соответствующие результаты тестов (рисунок 4).

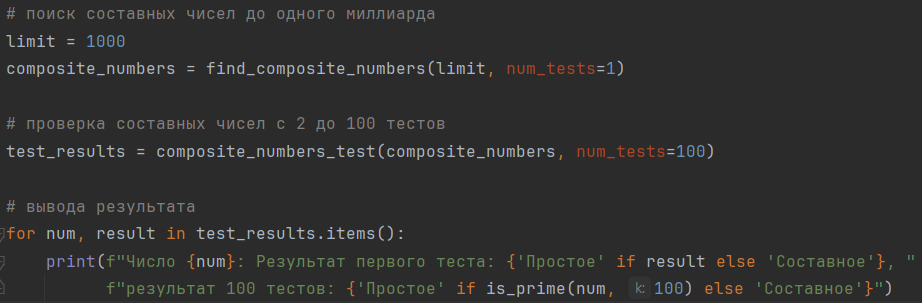


Рисунок 4 – Вызов соответствующих функций.

**Листинг программы**

Файл LW\_5.py

# реализация теста Ферма

import random

def is\_prime(n, k=5):

"""

Выполняет тест Ферма на простоту числа.

Args:

n (int): Число для проверки на простоту.

k (int, optional): Количество тестов. По умолчанию 5.

Returns:

bool: True, если число вероятно простое, иначе False.

"""

if n <= 1:

return False

if n == 2:

return False

for \_ in range(k):

a = random.randint(2, n - 1)

if pow(a, n - 1, n) != 1:

return False

return True

def find\_composite\_numbers(limit, num\_tests):

"""

Находит составные числа до заданного предела.

Args:

limit (int): Верхний предел поиска.

num\_tests (int): Количество тестов для каждого числа.

Returns:

list: Список составных чисел.

"""

composite\_numbers = []

for i in range(2, limit + 1):

if not is\_prime(i, num\_tests):

composite\_numbers.append(i)

return composite\_numbers

def composite\_numbers\_test(composite\_numbers, num\_tests):

"""

Проверяет составные числа, чтобы увидеть результат множественных тестов.

Args:

composite\_numbers (list): Список составных чисел.

num\_tests (int): Количество тестов.

Returns:

dict: Словарь с составными числами в качестве ключей и результатами тестов в качестве значений.

"""

results = {}

for num in composite\_numbers:

test\_result = is\_prime(num, num\_tests)

results[num] = test\_result

return results

# Найдем составные числа до одного миллиарда

limit = 1000

composite\_numbers = find\_composite\_numbers(limit, num\_tests=1)

# Проверим составные числа с 2 до 100 тестов

test\_results = composite\_numbers\_test(composite\_numbers, num\_tests=100)

# Выведем результаты

for num, result in test\_results.items():

print(f"Число {num}: Результат первого теста: {'Простое' if result else 'Составное'}, "

f"результат 100 тестов: {'Простое' if is\_prime(num, 100) else 'Составное'}")