# Математические основы защиты информации и информационной безопасности. Отчет по лабораторной работе №5

## Вероятностные алгоритмы проверки чисел на простоту

Серенко Данил Сергеевич 1132236895

## Содержание



## Цель работы

Освоить на практике алгоритмы проверки чисел на простоту.

## Выполнение лабораторной работы

Требуется реализовать:

1. Алгоритм, реализующий тест Ферма
2. Алгоритм вычисления символа Якоби
3. Алгоритм, реализующий тест Соловэя-Штрассена
4. Алгоритм, реализующий тест Миллера-Рабина.

### Алгоритм, реализующий тест Ферма

Алгоритм основан на малой теореме Ферма, которая утверждает, что если n - простое число, то для любого целого числа a, не являющегося кратным n, выполняется a^(n-1) ≡ 1 (mod n). Алгоритм выбирает случайные значения a и проверяет условие. Если оно не выполняется для какого-либо a, то n считается составным. Если оно выполняется для всех выбранных a, то n вероятно является простым.

Реализация на Python предствлена на рисунке 1.

```python
def is_prime_fermat(n, k=5):
    if n <= 1:
        return False
    if n <= 3:
        return True
    for _ in range(k):
        a = random.randint(2, n - 2)
        if pow(a, n - 1, n) != 1:
            return False
    return True
```

*fermat*

## Символ Якоби

Символ Якоби обобщает символ Лежандра и используется для определения вычетов в кольце вычетов по модулю n. Для нечетного простого числа n и целого числа a, символ Якоби Jacobi(a, n) равен 1, если a является квадратичным вычетом по модулю n, -1, если a является квадратичным невычетом, и 0, если a кратно n. Символ Якоби используется в различных алгоритмах для проверки простоты и для решения квадратичных уравнений по модулю.

Реализация на Python предствлена на рисунке 2.

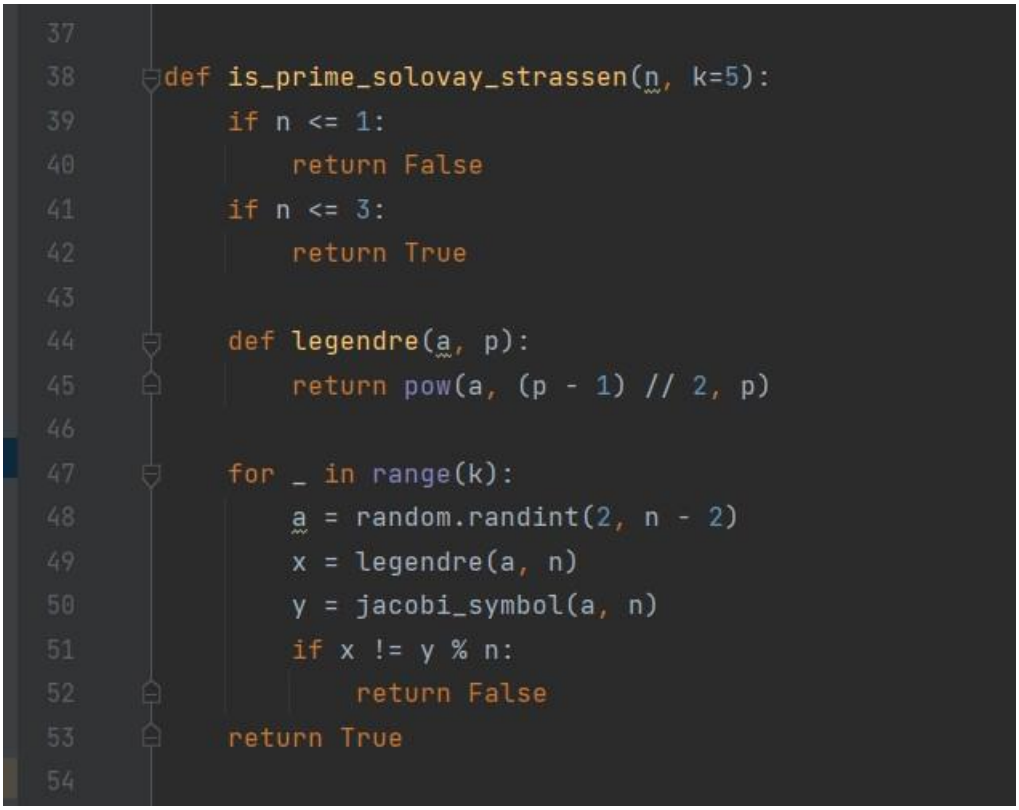
```python

def jacobi_symbol(a, n):
    if n % 2 == 0 or n <= 0:
        raise ValueError("n должно быть нечетным и положительным")
    a = a % n
    t = 1
    while a != 0:
        while a % 2 == 0:
            a /= 2
            r = n % 8
            if r == 3 or r == 5:
                t = -t

        a, n = n, a
        if a % 4 == 3 and n % 4 == 3:
            t = -t
        a = a % n
    if n == 1:
        return t
    else:
        return 0

```

*jacobi*

## Тест Соловея-Штрассена

Этот алгоритм использует символ Якоби и проверяет, является ли число простым. Алгоритм выбирает случайное целое число a и проверяет два условия: 1) a не делится на n, и 2) символ Якоби Jacobi(a, n) равен результату вычисления с использованием символа Лежандра. Если оба условия выполняются для всех выбранных a, то n вероятно является простым числом.

Реализация на Python предствлена на рисунке 3.



```
def is_prime_solovay_strassen(n, k=5):
    if n <= 1:
        return False
    if n <= 3:
        return True

    def legendre(a, p):
        return pow(a, (p - 1) // 2, p)

    for _ in range(k):
        a = random.randint(2, n - 2)
        x = legendre(a, n)
        y = jacobi_symbol(a, n)
        if x != y % n:
            return False
    return True
```

*solovay_strassen*

## Тест Миллера-Рабина

Этот алгоритм также использует вероятностный метод для проверки простоты числа. Алгоритм выбирает случайное целое число a и разлагает n - 1 на 2^s * d, где s - четное, и d нечетное. Затем алгоритм проверяет условия Миллера-Рабина: 1) a^d ≡ 1 (mod n), и 2) для всех i от 0 до s-1, $a^{(2i} * d) \equiv -1 \pmod n$ или $a^{(2i} * d) \equiv 1 \pmod n$. Если оба условия выполняются для всех выбранных a, то n вероятно является простым числом.

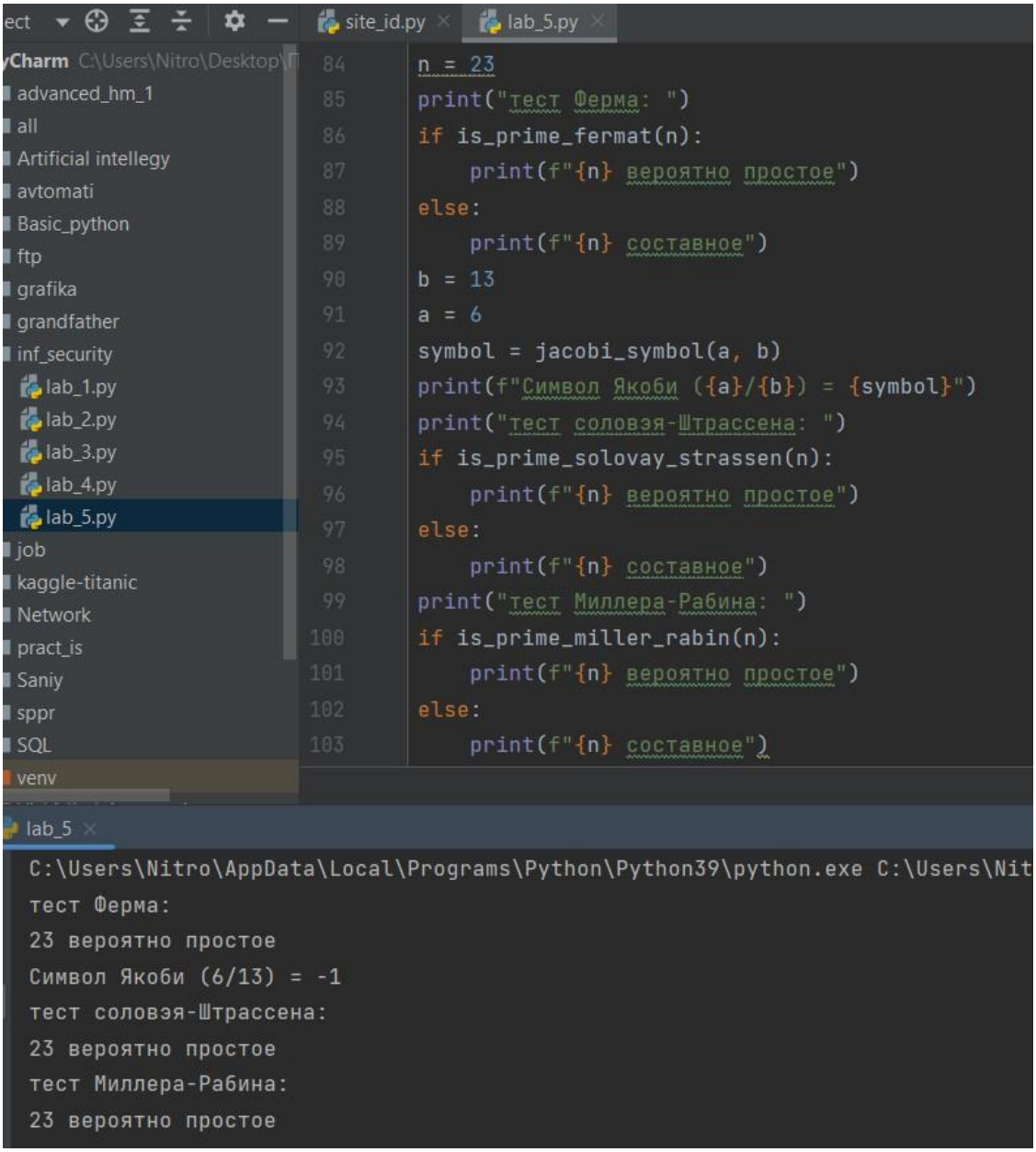
Реализация на Python предствлена на рисунке 4.

```python
def is_prime_miller_rabin(n, k=5):
    if n <= 1:
        return False
    if n <= 3:
        return True

    def miller_rabin_test(a, s, d, n):
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            return True

        for _ in range(s - 1):
            x = (x * x) % n
            if x == n - 1:
                return True
        return False
    s, d = 0, n - 1
    while d % 2 == 0:
        s += 1
        d //= 2

    for _ in range(k):
        a = random.randint(2, n - 2)
        if not miller_rabin_test(a, s, d, n):
            return False

    return True
```

*miller_rabin*

## Результат работы программы

функция запуска и выходные значения программы.




*main*

Выходные значения программы.

## Выводы

В результате выполнения работы я освоил на практике применение алгоритмов проверки чисел на простоту.

## Список литературы

1. Методические материалы курса