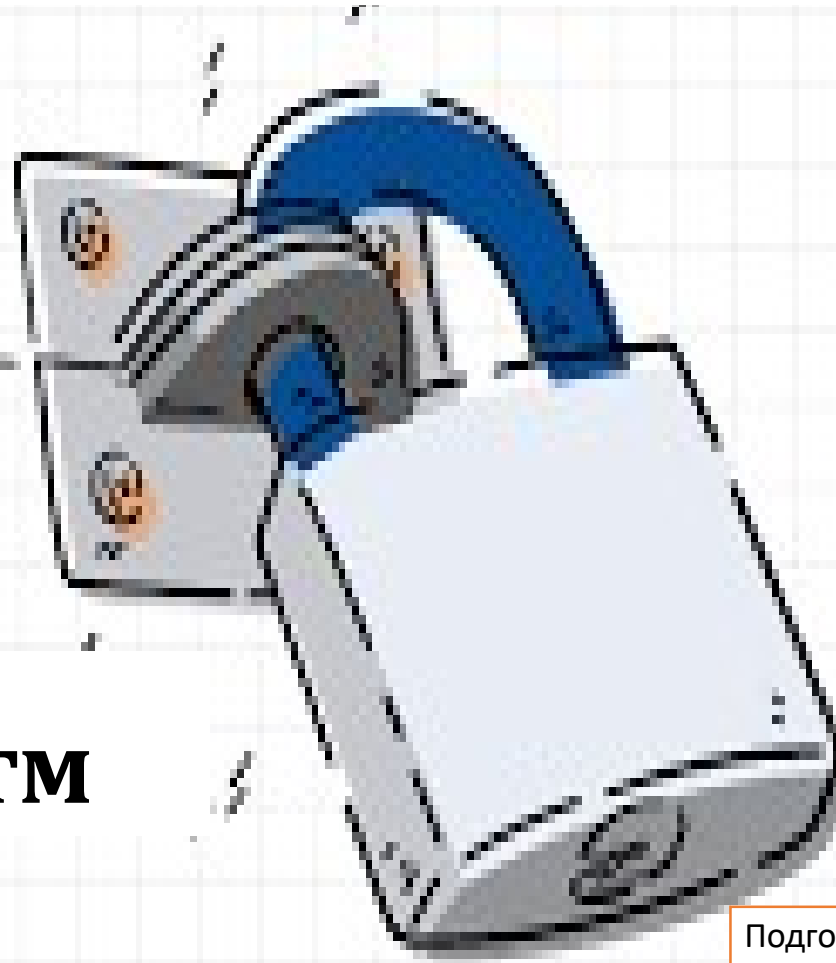


Российский университет дружбы народов Научный факультет

Математические
основы защиты
информации и
информационной
безопасности



Алгоритм

Подготовлено студентом:
Елиенис Санчес Родригес.
Преподаватель: Дмитрий Сергеевич

Алгоритм теста Ферманаерман

Существует множество процедур для проверки того, является ли данное натуральное число n простым или нет.

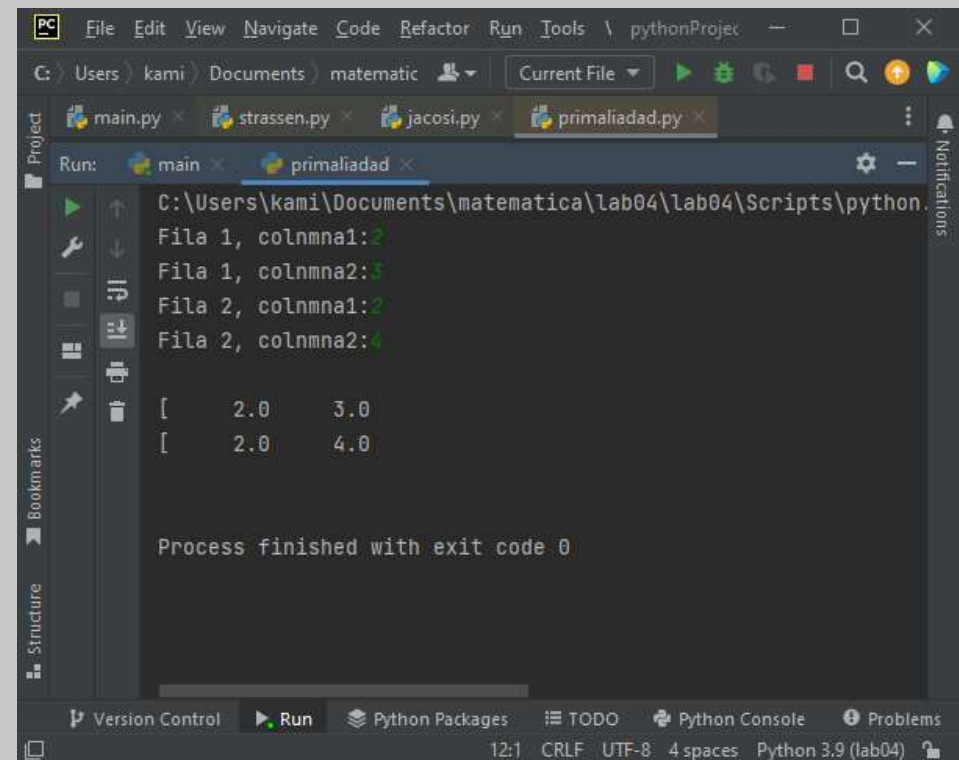
Dado $n \in \mathbb{N}$, verique para $2 \leq a < n$, con $\text{mcd}(a, n) = 1$ si $a^{n-1} \equiv 1 \pmod{n}$.

Пусть будет $n \in \mathbb{N}$: 1. Если $a^{n-1} \not\equiv 1 \pmod{n}$, то с уверенностью n не является простым числом.

Алгоритм теста Ферманаерман

```
from sympy import isprime
from colorama import Fore
from colorama import Style
print(Style.BRIGHT+"Primality Test")
print("")
x = int(input(Fore.YELLOW+"Introduce un numero: "))
y = int(input(Fore.CYAN+"cantidad de veces que se debe probar la
primidad: "))

for n in range(x,y):
    if (2**((n-1)-1) % (n)==0:
        if isprime(n):
            print(n,Fore.LIGHTGREEN_EX+"numero primo")
        else:
            print(n,Fore.GREEN+"pseudoprime")
```



The screenshot shows the PyCharm IDE interface. The top toolbar includes menus like File, Edit, View, Navigate, Code, Refactor, Run, and Tools. The project path is C:\Users\kami\Documents\matematica. The Run console shows the output of the script, which is a 2x2 grid of results for the numbers 2 and 3. The output is as follows:

Fila	colnmna1	colnmna2
Fila 1	2	3
Fila 2	2	4

Below the table, the console indicates that the process finished with exit code 0.

Алгоритм вычисления символа Якоби

Пусть p будет простым нечетным числом. Тогда легко увидеть, что для данного целого числа a конгруэнтность может быть неразрешимой, она может иметь одно решение, если $a \equiv 0$ по модулю p , или она может иметь два решения, в таком случае мы говорим, что a - квадратичный остаток по модулю p . Мы определяем символ Лежандра при p как -1 , если предыдущая конгруэнтность не имеет решения, 0 если $a \equiv 0$ и 1 если a - квадратичный остаток. Число решений по модулю p для приведенной выше конгруэнтности равно от 1 до p .

расширенный алгоритм Евклида

```
import numpy as np
"definimos la funcion de jacobi" \
"a = la matriz que representa el sistema de ecuaciones" \
"b= vector de valores independiente" \
"x0= vector de variables" \
"tol=tolerancia" \
"n= numero de pasos "
def jacobi (A,b,x0,tol,n):
    "matriz diagonal de la matriz"
    D=np.diag(np.diag(A))
    "L MATRIZ triangular inferior de A"
    "U MATRIZ triangular superior de A"
    LU=A-D
    x=x0
    for i in range(n):
        "D es la inversa de la matriz D"
        D_inv=np.linalg.inv(D)
        xtemp = x
        "X Es el producto entre matrices "
        x=np.dot(D_inv,np.dot(-LU,x))+np.dot(D_inv,b)
        print("Iteracion",i,"x =",x)
        "en caso que el valor del vector" \
        "sea menor que la tolerancia, retorno a x"
        if np.linalg.norm(x-xtemp)<tol:
            return x
    return x
"matriz que representa el sistema de ecuaciones a solucionar "
A= np.array([
    [10,-1,2,0],
    [-1,11,-1,3],
    [2,-1,10,.1],
    [0,3,-1,8]
])

import numpy as np
"b =vector de valores independientes"
b=([6,25,-11,15])
x0=np.zeros(4)
tol=1e-3
n= 500
x = jacobi(A,b,x0,tol,n)
```

```
PC File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject2
C:\Users\kami\Documents\matematica\lab05\jacosi.py Current File
main.py strassen.py jacosi.py primalidad.py
40 n= 500
41 x = jacobi(A,b,x0,tol,n)

Run: main jacosi
C:\Users\kami\Documents\matematica\lab04\lab04\Scripts\python.exe C:/Users/kami/
Iteracion 0 : x = [ 0.6      2.27272727 -1.1      1.875    ]
Iteracion 1 : x = [ 1.04727273  1.71590909 -1.01147727  0.88522727]
Iteracion 2 : x = [ 0.97388636  2.03455579 -1.14671591  1.10509943]
Iteracion 3 : x = [ 1.03279876  1.95562474 -1.10237269  0.96870209]
Iteracion 4 : x = [ 1.01603701  2.00221089 -1.1206843   1.00384414]
Iteracion 5 : x = [ 1.02435795  1.98943821 -1.11302475  0.98408538]
Iteracion 6 : x = [ 1.02154877  1.99627973 -1.11576862  0.98983258]
Iteracion 7 : x = [ 1.0227817   1.99420749 -1.11458011  0.98692402]
Iteracion 8 : x = [ 1.02233677  1.99522087 -1.11500483  0.98784968]
Iteracion 9 : x = [ 1.02252305  1.99488936 -1.11482376  0.98741657]

Process finished with exit code 0

PEP 8: W292 no newline at end of file 41:25 CRLF UTF-8 4 spaces Python 3.9 (lab04)
```


Алгоритм доказательства Штрассена

Пусть задано $n \in \mathbb{N}$ нечетных и $k \in \mathbb{N}$. Произвольно и независимо выберите k чисел $a \in \{2, \dots, n-1\}$ с $\gcd(a, n) = 1$, пока не будет найден (если возможно) свидетель для композиции n . Если такое число существует, то n является составным. В противном случае алгоритм завершается ошибкой output

Как и в тесте Миллера - Рабина, можно еще раз доказать, что свидетель состава существует, если n является составным.

Вероятность того, что для составного числа n тест Соловея - Штрассена завершится неудачей, меньше $1/2^k$.

Следовательно, если k велико, то вероятность того, что n будет простым числом, высока, если тест Соловея - Штрассена завершится неудачей

1. Algoritmo de demostración de Strassen

```
import random
import time

#El siguiente Algoritmo de Strassen es sacado de
#https://github.com/stanislvkozlovski/Algoritmos/blob/master/Coursera/algorithms_stanford/Strassen%20Matrix%20Multiplication/python/strassen.py
```

```
def default_matrix_multiplication(a, b):
    """
    Only for 2x2 matrices
    """
    if len(a) != 2 or len(a[0]) != 2 or len(b) != 2 or len(b[0]) != 2:
        raise Exception('Matrices should be 2x2!')
    #print(a[0][0] * b[0][1] + a[0][1] * b[1][1])
    new_matrix = [[a[0][0] * b[0][0] + a[0][1] * b[1][0],
                  a[0][0] * b[0][1] + a[0][1] * b[1][1],
                  a[1][0] * b[0][0] + a[1][1] * b[1][0],
                  a[1][0] * b[0][1] + a[1][1] * b[1][1]]
    return new_matrix
```

```
def matrix_addition(matrix_a, matrix_b):
    # print(matrix_a)
    return [[matrix_a[row][col] +
            matrix_b[row][col]
            for col in range(len(matrix_a[row]))] for
            row in range(len(matrix_a))]
```

```
def matrix_subtraction(matrix_a, matrix_b):
    return [[matrix_a[row][col] -
            matrix_b[row][col]
            for col in range(len(matrix_a[row]))] for
            row in range(len(matrix_a))]
```

```
def split_matrix(a):
    """
    Given a matrix, return the TOP_LEFT,
    TOP_RIGHT, BOT_LEFT and BOT_RIGHT
    quadrant
    """
```

```
if len(a) % 2 != 0 or len(a[0]) % 2 != 0:
    raise Exception('Odd matrices are not supported!')
matrix_length = len(a)
mid = matrix_length // 2
top_left = [[a[i][j] for j in range(mid)] for i in
range(mid)]
bot_left = [[a[i][j] for j in range(mid)] for i in
range(mid, matrix_length)]
```

```
top_right = [[a[i][j] for j in range(mid, matrix_length)]
for i in range(mid)]
bot_right = [[a[i][j] for j in range(mid, matrix_length)]
for i in range(mid, matrix_length)]
```

```
return top_left, top_right, bot_left, bot_right
```

```
def get_matrix_dimensions(matrix):
    return len(matrix), len(matrix[0])
```

```
def strassen(matrix_a, matrix_b):
    """
    Recursive function to calculate the product of two
    matrices, using the Strassen Algorithm.
    Currently only works for matrices of even length (2x2,
    4x4, 6x6...etc)
    """
    if get_matrix_dimensions(matrix_a) !=
    get_matrix_dimensions(matrix_b):
        raise Exception('Both matrices are not the same
dimension! \nMatrix A:{matrix_a} \nMatrix
B:{matrix_b}')
    if get_matrix_dimensions(matrix_a) == (2, 2):
        return default_matrix_multiplication(matrix_a,
matrix_b)
```

```
A, B, C, D = split_matrix(matrix_a)
E, F, G, H = split_matrix(matrix_b)
```

```
p1 = strassen(A, matrix_subtraction(F, H))
p2 = strassen(matrix_addition(A, B), H)
p3 = strassen(matrix_addition(C, D), E)
p4 = strassen(D, matrix_subtraction(G, E))
p5 = strassen(matrix_addition(A, D),
matrix_addition(E, H))
p6 = strassen(matrix_subtraction(B, D),
matrix_addition(G, H))
p7 = strassen(matrix_subtraction(A, C),
matrix_addition(E, F))
```

```
top_left =
matrix_addition(matrix_subtraction(matrix_
addition(p5, p4), p2), p6)
top_right = matrix_addition(p1, p2)
bot_left = matrix_addition(p3, p4)
bot_right =
matrix_subtraction(matrix_subtraction(mat
rix_addition(p1, p5), p3), p7)
```

```
# construct the new matrix from our 4
quadrants
new_matrix = []
for i in range(len(top_right)):
    new_matrix.append(top_left[i] +
top_right[i])
for i in range(len(bot_right)):
    new_matrix.append(bot_left[i] +
bot_right[i])
return new_matrix
```

#Fin de algoritmo de Strassen

```
#Funciones adicionales para generar matriz
aleatoria
def generateMat(f, c):
    M = []
    for i in range(f):
        M.append([])
        for j in range(c):
            M[i].append(random.randint(0,10))
    return M
#Funcion adicional para mostrar la matriz
def mostrarMat(M):
    print("\n")
    for fila in M:
        print(fila)
    print("\n")
```

```
# Algoritmo de multiplicacion de matrices
de forma iterativa
#https://gist.github.com/parzibyte/bb96d0c
5089858b3de2110ec208f55a5#file-
producto-py
```

```
def producto_matrices(a, b):
    filas_a = len(a)
```

```
filas_b = len(b)
columnas_a = len(a[0])
columnas_b = len(b[0])
if columnas_a != filas_b:
    return None
# Asignar espacio al producto. Es decir,
rellenar con "espacios vacíos"
producto = []
for i in range(filas_b):
    producto.append([])
    for j in range(columnas_b):
        producto[i].append(None)
# Rellenar el producto
for c in range(columnas_b):
    for i in range(filas_a):
        suma = 0
        for j in range(columnas_a):
            suma += a[i][j]*b[j][c]
        producto[i][c] = suma
return producto
```

#Fin de algoritmo de producto de 2 matrices

```
#
print(" Programa que multiplica 2
matrices ")
orden = int(input("Generar orden: "))
A = generateMat(orden,orden)
B = generateMat(orden,orden)
```

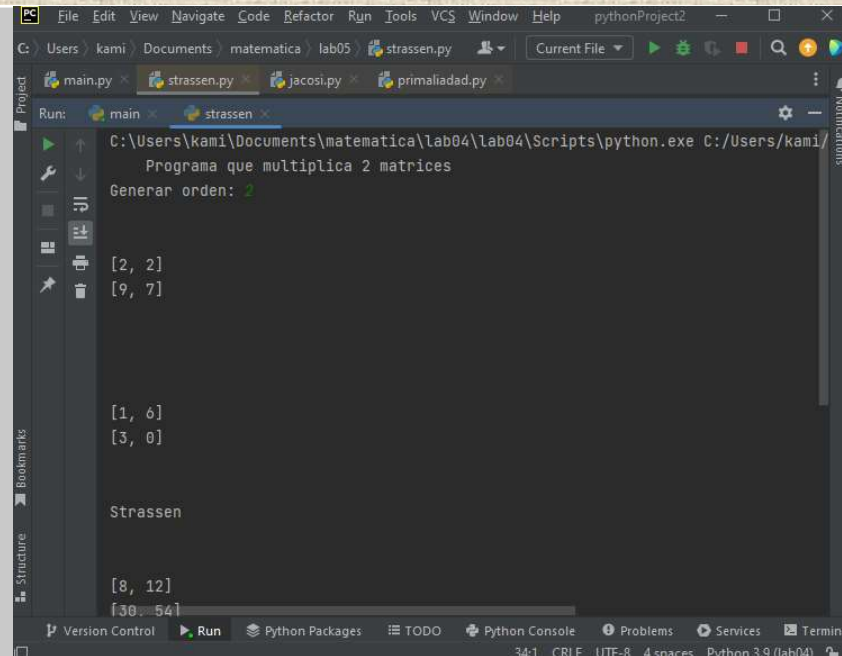
```
mostrarMat(A)
mostrarMat(B)
```

```
print("Strassen")
inicio = time.time()
C = strassen(A,B)
fin = time.time()
mostrarMat(C)
tStrassen = fin -inicio
```

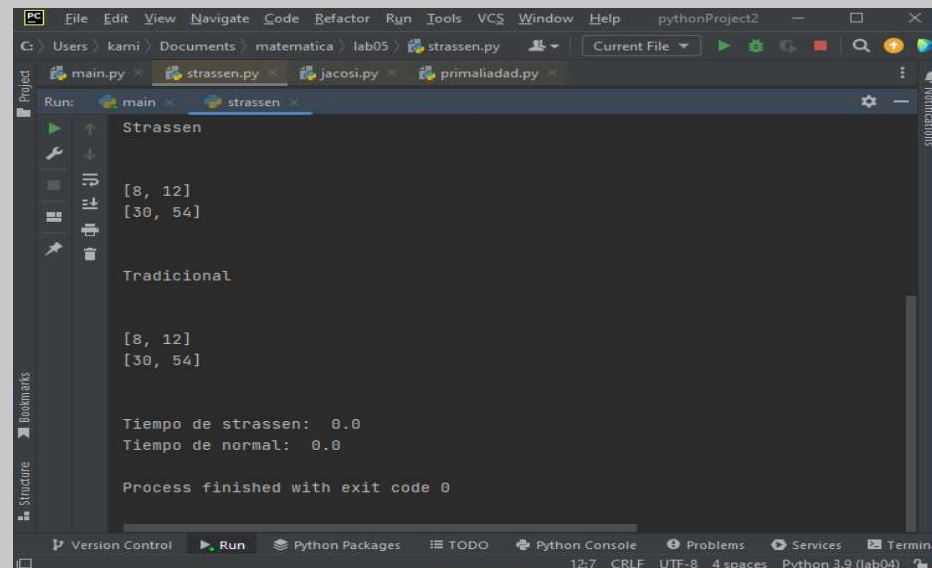
```
print("Tradicional")
inicio = time.time()
D= producto_matrices(A,B)
fin = time.time()
mostrarMat(D)
```

1. Алгоритм доказательства Штрассена

```
tTradicional = fin -inicio  
print("Tiempo de strassen: ", tStrassen)  
print("Tiempo de normal: ", tTradicional)
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject2  
C:\Users\kami\Documents\matematica\lab05\strassen.py Current File  
main.py strassen.py jacosi.py primaliadad.py  
Run: main strassen  
C:\Users\kami\Documents\matematica\lab04\lab04\Scripts\python.exe C:/Users/kami/  
Programa que multiplica 2 matrices  
Generar orden: 2  
  
[2, 2]  
[9, 7]  
  
[1, 6]  
[3, 0]  
  
Strassen  
  
[8, 12]  
[30, 54]  
34:1 CRLF UTF-8 4 spaces Python 3.9 (lab04)
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject2  
C:\Users\kami\Documents\matematica\lab05\strassen.py Current File  
main.py strassen.py jacosi.py primaliadad.py  
Run: main strassen  
C:\Users\kami\Documents\matematica\lab04\lab04\Scripts\python.exe C:/Users/kami/  
Strassen  
[8, 12]  
[30, 54]  
  
Tradicional  
  
[8, 12]  
[30, 54]  
  
Tiempo de strassen: 0.0  
Tiempo de normal: 0.0  
  
Process finished with exit code 0  
12:7 CRLF UTF-8 4 spaces Python 3.9 (lab04)
```


Алгоритм доказательства Рабина Миллера

Предположим, вы хотите определить, является ли данное большое нечетное число $n \in \mathbb{N}$ простым или составным числом. 1. Произвольно и независимо выберите k чисел a таким образом, чтобы $2 \leq a \leq n-1$. 2. Определите с помощью алгоритма Евклида $\text{mcd}(a, n)$. Если $\text{mcd}(a, n) \neq 1$, то n является составным. 3. Проверьте, находится ли подвыбранной буквой a контрольная лампа для композиции. При первом столкновении алгоритм завершается, и n не является простым числом. Если невозможно найти такое число a , то алгоритм завершается с ошибкой.

Алгоритм доказательства Рабина Миллера

```
import random

# p is the number to check, a is the base, d is odd, r is the power
def Check_If_Composite(p, a, d, r):

    # If the below is true, then the number is likely to be prime.
    # Condition 1: If ( a ^ d ) = 1 ( mod p ) i.e find out if ( a ^ d ) % p
    = 1
    # Condition 2: If ( a ^ ( ( 2 ^ 0 ) . d ) = - 1 ( mod p )
    # i.e find out if ( a ^ d ) = -1 (mod p) which is same as finding
    out if ( a ^ d ) % p = p - 1
    remainder = pow(a, d, p)

    if (remainder == 1 or remainder == p - 1):
        return False

    # Note : Remainder is already calculated above. It is (a ^ d) % p.
    # Below loop would calculate if (a ^ (( 2 ^ t ).d)) == -1 (mod p) for
    some 1 <= t <= r-1
    # And ( a ^ d ) = -1 (mod p) is same as ( a ^ d ) % p = p - 1
    # Example ( a ^ d . a ^ d ) % p = (a ^ 2.d) mod p
    for t in range(1, r):
        if ((remainder * remainder) % p == p - 1):
            return False
    return True

def Check_If_Prime(p, iterations):

    if (p == 2 or p == 3):
        return True

    # p is even
    if (p % 2 == 0):
        return False
```

```
# p - 1 has to be even as p is odd
d = p - 1
r = 0
while ((d % 2) == 0):
    d = int(d / 2)
    r += 1

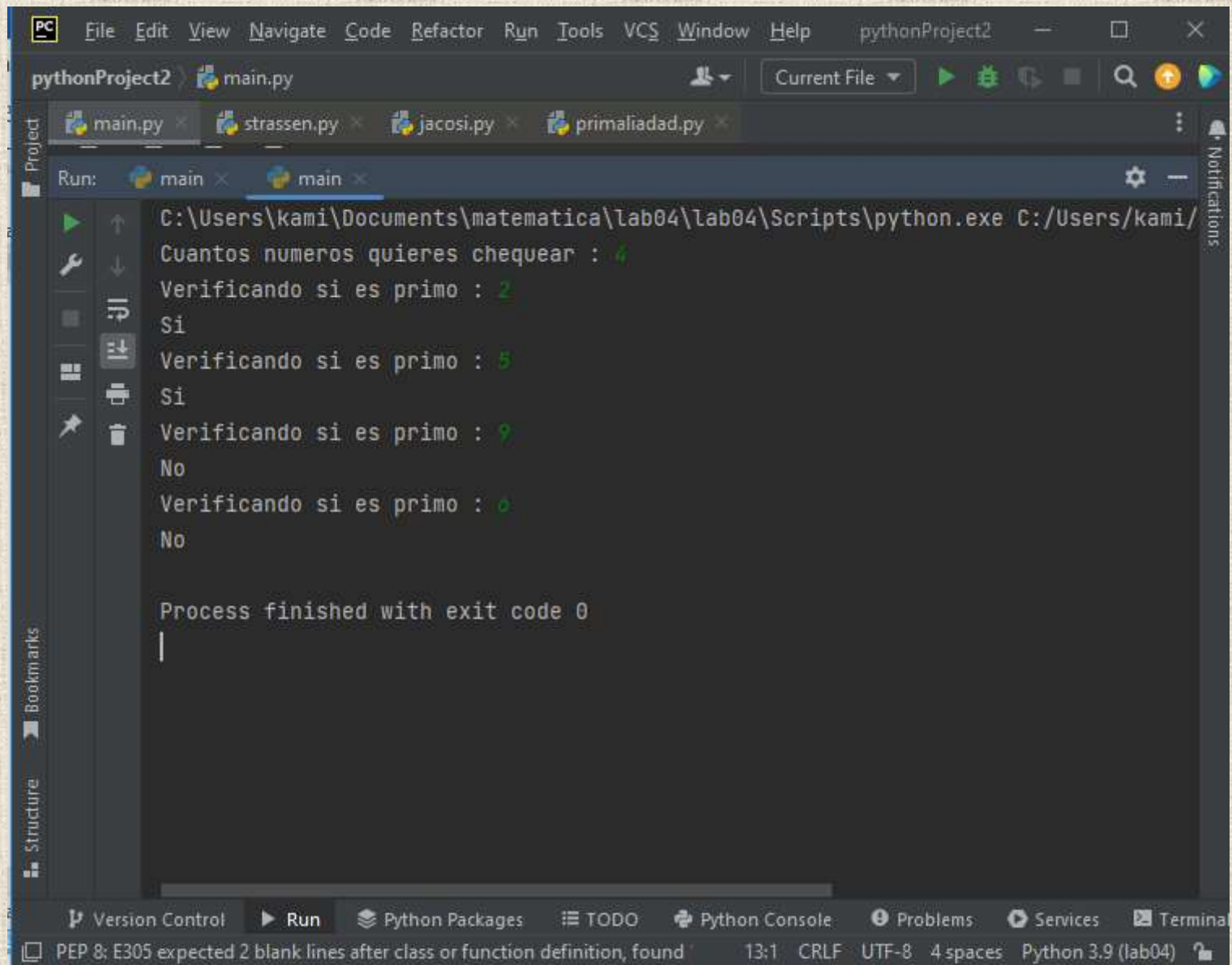
for i in range(iterations + 1):
    # Choose a random a ∈ { 1, 2, ..., p - 1 }.
    a = random.randrange(2, p)
    # If it is definitely composite, then it is not prime.
    if (Check_If_Composite(p, a, d, r) == True):
        return False
    return True

def main():

    tests = int(input("Cuantos numeros quieres chequear : "))

    for i in range(tests):
        p = int(input("Verificando si es primo : "))
        if (Check_If_Prime(p, 5) == True):
            print("Si")
        else:
            print("No")

if __name__ == "__main__":
    main()
```



The screenshot shows an IDE window titled "pythonProject2" with a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help) and a toolbar. The editor displays four Python files: main.py, strassen.py, jacosi.py, and primaliudad.py. The "Run" panel is active, showing the execution of "main.py". The command executed is "C:\Users\kami\Documents\matematica\lab04\lab04\Scripts\python.exe C:/Users/kami/". The output is as follows:

```
C:\Users\kami\Documents\matematica\lab04\lab04\Scripts\python.exe C:/Users/kami/  
Cuantos numeros quieres chequear : 4  
Verificando si es primo : 2  
Si  
Verificando si es primo : 5  
Si  
Verificando si es primo : 9  
No  
Verificando si es primo : 0  
No  
  
Process finished with exit code 0
```

The bottom status bar indicates a PEP 8 error: "PEP 8: E305 expected 2 blank lines after class or function definition, found 1" at line 13, column 1. The status bar also shows "CRLF", "UTF-8", "4 spaces", and "Python 3.9 (lab04)".


```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>
C:\Users\kami>
```

ВЫВОД

Проблема простоты состоит в том, чтобы выяснить, является ли число простым или составным. Существуют очень древние методы определения простоты числа, такие как Сито Эратосфена (2000 г. до н.э.), но это невозможно для анализа большого числа n . Ферма в 1636 году представил свою знаменитую Малую теорему Ферма, в которой он определяет характеристику, которой удовлетворяют все простые числа. Теорема утверждает, что, когда n является простым числом и при совместном числе a выполняется при $a \equiv 1 \pmod n$.

В 1770 году Джон Уилсон нашел характеристику простых чисел, которая полезна для теоретического развития, но на практике не часто используется в качестве доказательства простоты, поскольку для вычисления $(n-1)! \pmod n$ для большого числа n имеет высокие вычислительные затраты.

C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>
C:\Users\kami>

Библиография

Rios P, B., & Salcedo A, R. (2020). Algoritmo . Ciencia Digital, 2(3), 61-74
https://hmong.es/wiki/Jacobi_symbol

Cabrera R, Juan , Estructuras de Datos en Python(2020). Python
<https://www.programarya.com/Cursos/Python/estructuras-de-datos#:~:text=Las%20estructuras%20de%20datos%20m%C3%A1s,y%20los%20arreglos%20in dexados%2C%20respectivamente.>

C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Versión 10.0.19042.1706]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\kami>

C:\Users\kami>Большое спасибо