Coordinación de
Educación Abierta y a Distancia
VICERRECTORADO ACADÉMICO

Unach
UNIVERSIDAD NACIONAL DE CHIMBORAZO Online

CULTURA DIGITAL Y SOCIEDAD

Actividad Autónoma 4

Unidad 2: Herramientas y Metodologías en Ciencia de Datos

Tema 2: Buenas Prácticas en Programación para Ciencia de Datos

FACULTAD DE
Ingeniería

Nombres: Maria Jose Vimos Lluay

Fecha: 19/11/2025

Carrera: Ingeniería en Ciencia de Datos e Inteligencia Artificial

Periodo académico: 2025-2S

Semestre: Tercer Semestre "A"

OPTIMIZACION DE CODIGO EN PYTHON

El siguiente proyecto es un código en Python que busca números primos en el rango de 1 a 100.000 donde el código original realizaba muchas operaciones innecesarias ya ue comprobaba las divisiones desde 2 hasta n eso hacia que el tiempo de ejecución sea muy lento .

El código original su tiempo fue de :

15.15 segundos

Para lo que se realizo una optimización del algoritmo aplicando técnicas sencillas para su rendimiento

OPTIMIZACION DEL CODIGO

para mejorar el tiempo de ejecución se aplicaron las siguientes técnicas

- Uso e la raíz cuadrada

En vez de revisar divisores desde 2 hasta n solo aplicamos la raíz cuadrada del numero esto reduce la cantidad e operaciones.

- Lista de Primos

Se crea una lista de primos en una sola linea para mejorar el rendimiento del código

```
# Usamos list comprehension para hacerlo más rápido
primos = [num for num in range(1, 100000) if es_primo_rapido(num)]
```

- Lógica del algoritmo

Se reorganizo la función para evitar pasos innecesarios y cortar el proceso

RESULTADOS DE LA OPTIMIZACION

Tiempo del Código Original:

15.10 segundos

Tiempo del Código Optimizado:

0.16 segundos

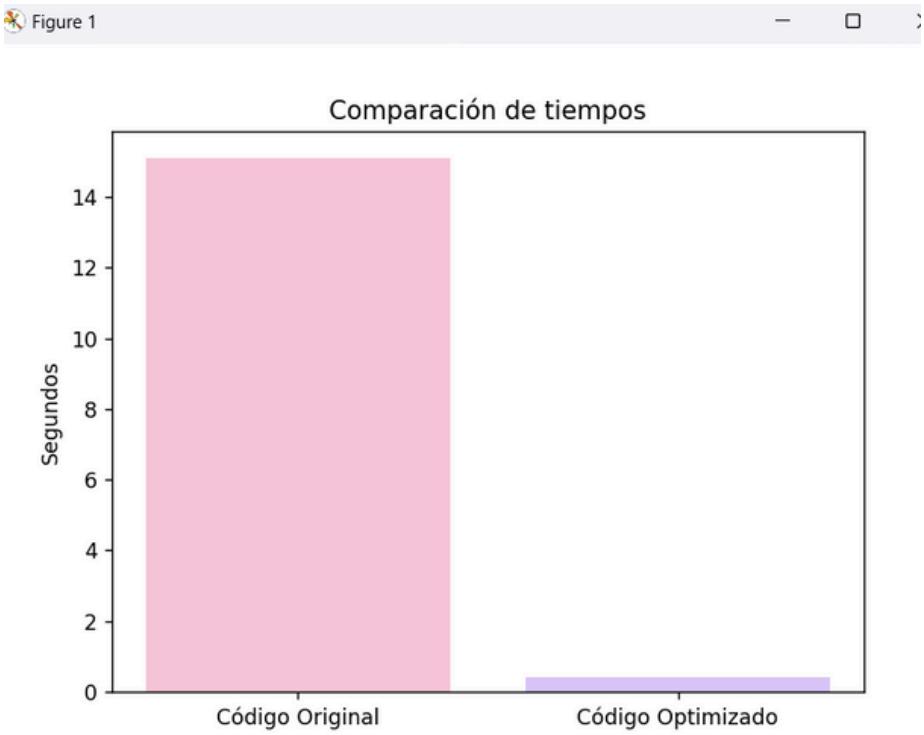
ANALISIS CON CPTOFILE

Con el análisis de c Profile se pudo observa que la función es_primo del codigo original consumía mayor parte del tiempo mientras que el codigo optimizado la carga disminuyo gracias al uso de la raíz cuadrada , el archivo profiling:optimizado muestra estas funciones ordenadas por tiempo.

Graficas

Se generaron graficas usando la libreria Matplotlib para mostrar:

- La comparación entre el tiempo del código original y optimizado
- La distribución del tiempo de ejecución



Lo que nos permite visualizar la mejoria optenida enytre el codigo original y el optimizado , se observa una diferencia significativa en los tiewmpos

CONCLUSIONES

- La optimización del algoritmo se redujo significativamente al tiempo de ejecución , las técnicas fueron simples solo usar la raíz cuadrada y list comprehensions para mejorar el rendimiento del código .
- Mientras que el C Profile nos permitio identificar las funciones que consumían mas tiempo , con esa ayuda la optimizacion del código la velocidad y la escalabilidad mejoraron mucho .
- Un código mas limpio y bien documentado es mas facil de mantener

ANEXOS

Fig 1. Código original

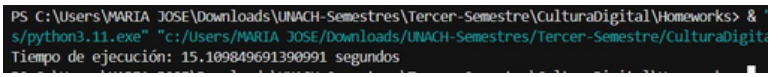


Fig 2. Tiempo código original

Fig 2. Tiempo código optimizado

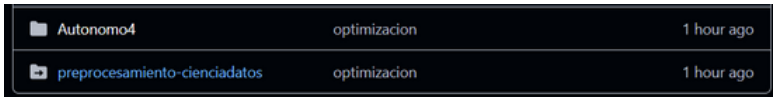
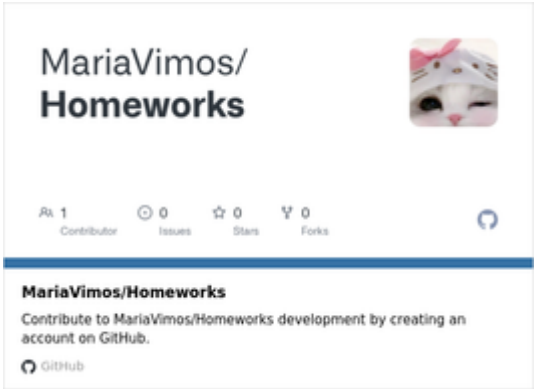


Fig 3. Repositorio GitHub

REPOSITORIO EN GITHUB.



https://github.com/MariaVimos/Homeworks.git

BIBLIOGRAFIA

- Python Software Foundation. Python 3 Documentation. Disponible en: https://docs.python.org/3/
- Van Rossum, G., & Drake, F. L. (2010). The Python Language Reference Manual. Network Theory Ltd.
- McKinney, W. (2018). Python for Data Analysis. O'Reilly Media.
- Grus, J. (2019). Data Science from Scratch: First Principles with Python. O'Reilly Media.