

TECNOLÓGICO DE ESTUDIOS SUPERIORES CHALCO

CARRERA: INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA: INTRODUCCIÓN A LA CIENCIA DE DATOS

DOCENTE: MTRO. IVAN AZAMAR PALMA

ALUMNOS: VAZQUEZ GARCIA LUZ LIZETH

MATRÍCULA: 202227005

SEMESTRE: SEPTIMO SEMESTRE

GRUPO: 4771 SEMI-PRESENCIAL

TEMA: PRÁCTICA 2

FECHA: OCTUBRE 2025

INDICE

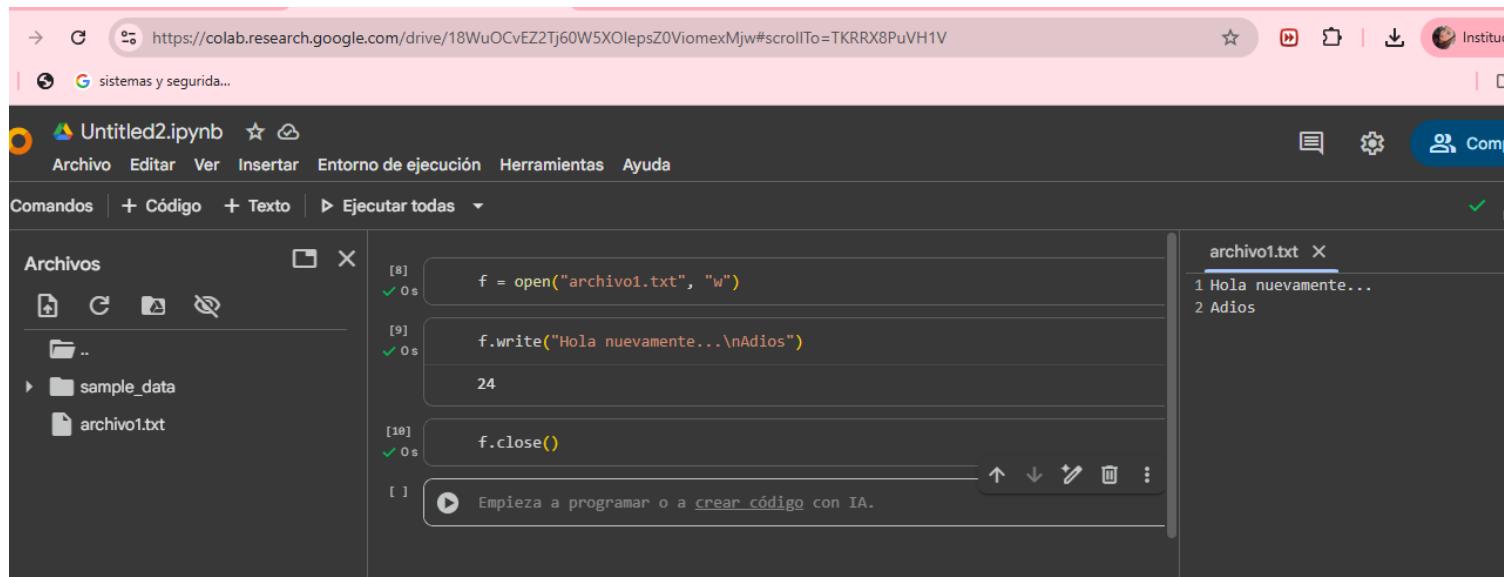
INTRODUCCIÓN	1
DESARROLLO DE LA PRÁCTICA.....	2
1. OPERACIÓN BÁSICA DE ESCRITURA:	2
2. ESCRITURA DE NÚMEROS:	3
3. LECTURA DE ARCHIVOS LÍNEA POR LÍNEA	4
4. SUMATORIA.....	5
5. ARCHIVOS CSV EN MODO ESCRITURA.....	6
6. ARCHIVOS CSV CONTENIDO LÍNEA POR LÍNEA	7
7. ARCHIVOS CSV LEER CON DICTREADER	8
CONCLUSIÓN	9
BIBLIOGRAFÍA:	10

INTRODUCCIÓN

En esta práctica se exploraron conceptos fundamentales del manejo de archivos y procesamiento de datos utilizando el lenguaje de programación Python. Python es una herramienta poderosa y versátil, ampliamente utilizada en el análisis de datos, automatización y desarrollo de aplicaciones. A través de ejercicios prácticos, se trabajó con la creación, lectura y escritura de archivos de texto y archivos CSV, lo cual es esencial para la manipulación de grandes volúmenes de información y su posterior análisis. Además, se aplicaron técnicas para generar datos aleatorios y estructurarlos en formatos estándar, como archivos CSV, facilitando su lectura y procesamiento con módulos especializados. Estas actividades permitieron comprender la importancia del manejo eficiente de datos y cómo Python facilita estas tareas con su sintaxis sencilla y bibliotecas integradas, consolidando así conocimientos clave para el desarrollo de proyectos en ciencia de datos y programación general.

DESARROLLO DE LA PRÁCTICA

1. OPERACIÓN BÁSICA DE ESCRITURA:



The screenshot shows a Google Colab notebook titled "Untitled2.ipynb". The code cell contains the following Python script:

```
[8]: f = open("archivo1.txt", "w")
[9]: f.write("Hola nuevamente...\nAdios")
[10]: f.close()
```

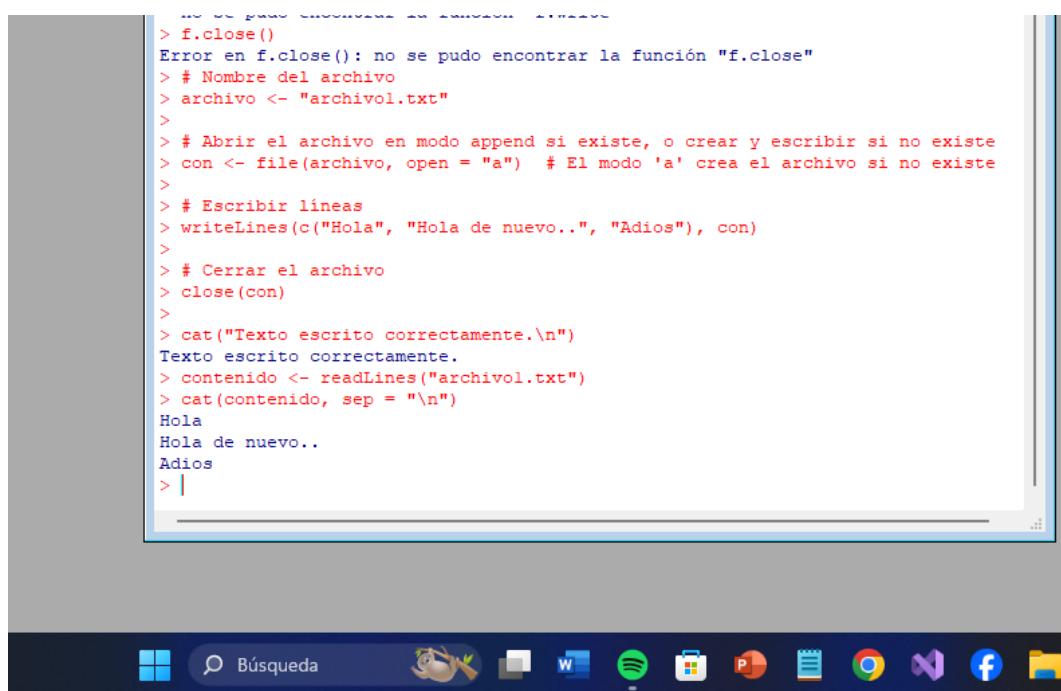
The output cell shows the contents of the file "archivo1.txt":

```
1 Hola nuevamente...
2 Adios
```

- `open()`: Abre un archivo.
- "archivo1.txt": Es el nombre del archivo que se va a abrir.
- "a": Es el modo en que se abre el archivo:
 - "a" significa append (agregar). Esto quiere decir que:
 - Si el archivo no existe, Python lo crea.
 - Si el archivo ya existe, el contenido nuevo se agrega al final sin borrar lo anterior.

El archivo se abre y se guarda en la variable `f`.

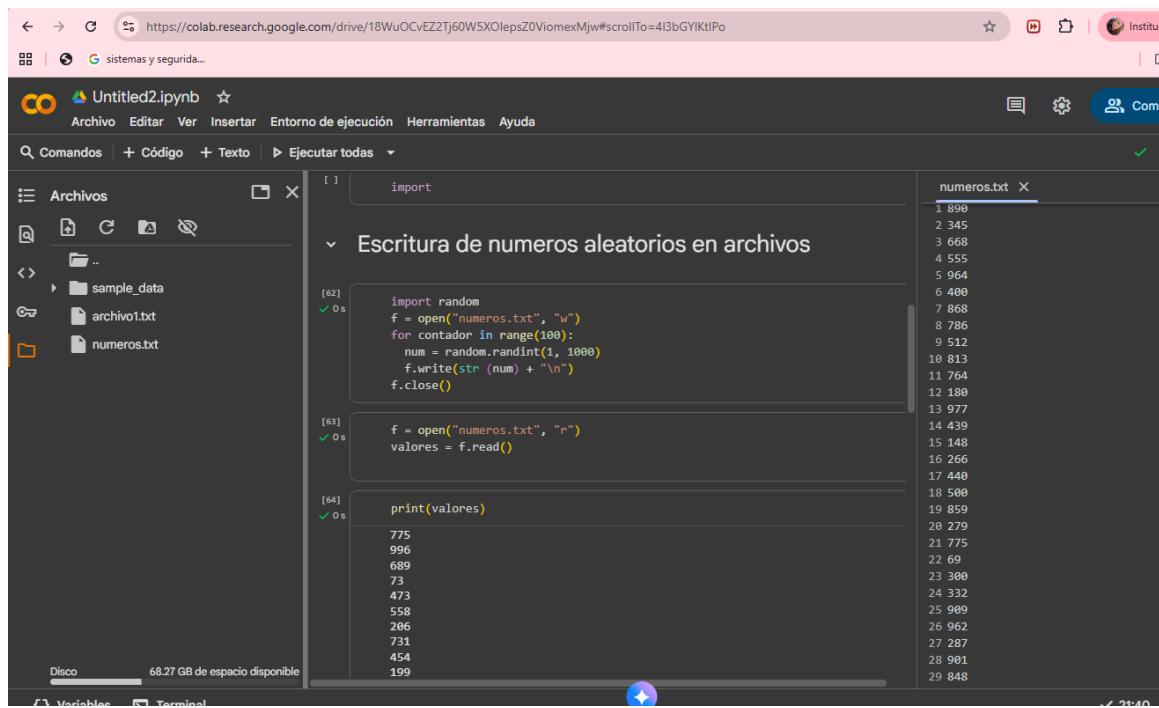
En lenguaje R



The screenshot shows an R console window. The user has run the following commands:

```
> f.close()
Error en f.close(): no se pudo encontrar la función "f.close"
> # Nombre del archivo
> archivo <- "archivol.txt"
>
> # Abrir el archivo en modo append si existe, o crear y escribir si no existe
> con <- file(archivo, open = "a") # El modo 'a' crea el archivo si no existe
>
> # Escribir lineas
> writeLines(c("Hola", "Hola de nuevo...", "Adios"), con)
>
> # Cerrar el archivo
> close(con)
>
> cat("Texto escrito correctamente.\n")
Texto escrito correctamente.
> contenido <- readLines("archivol.txt")
> cat(contenido, sep = "\n")
Hola
Hola de nuevo...
Adios
> |
```

2. ESCRITURA DE NÚMEROS:



The screenshot shows a Google Colab notebook titled "Untitled2.ipynb". The code cell contains the following Python script:

```

import random
f = open("numeros.txt", "w")
for contador in range(100):
    num = random.randint(1, 1000)
    f.write(str(num) + "\n")
f.close()

f = open("numeros.txt", "r")
valores = f.read()

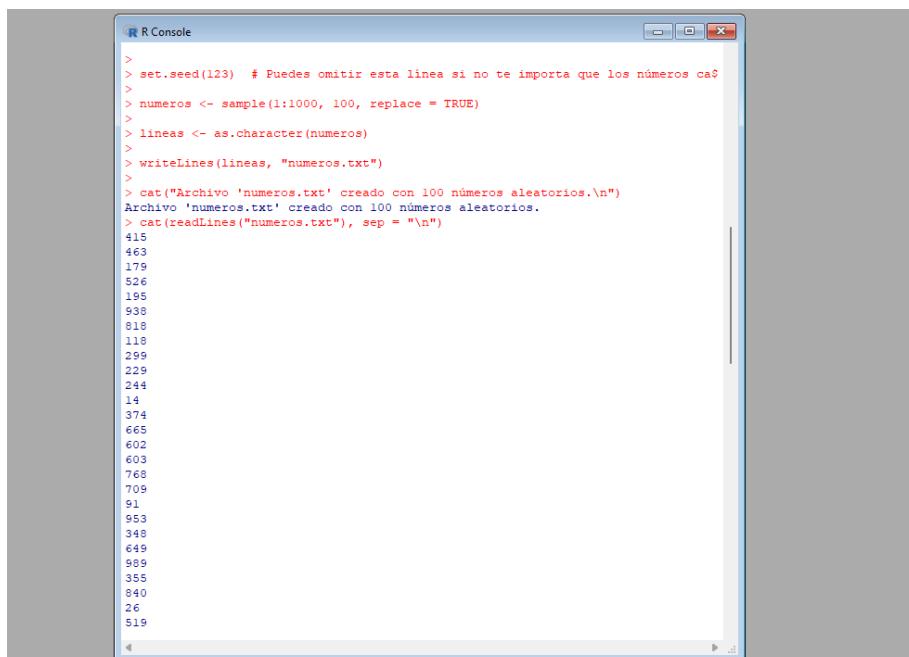
print(valores)

```

The output of the code is displayed in the next cell, showing 100 random integers from 1 to 1000, each on a new line. The file "numeros.txt" is also visible in the sidebar.

- Este código genera **100** números aleatorios entre 1 y 1000 y los guarda en un archivo llamado numeros.txt, escribiendo un número por línea.
- Primero se importa el módulo random para poder generar números aleatorios. Luego se abre el archivo en modo escritura ("w"), lo que borra su contenido si ya existe.
- Dentro de un bucle for que se repite 100 veces, se genera un número aleatorio con random.randint(1, 1000), se convierte a texto y se escribe en el archivo seguido de un salto de línea.
- Finalmente, se cierra el archivo con f.close() para guardar los cambios.

Lenguaje R



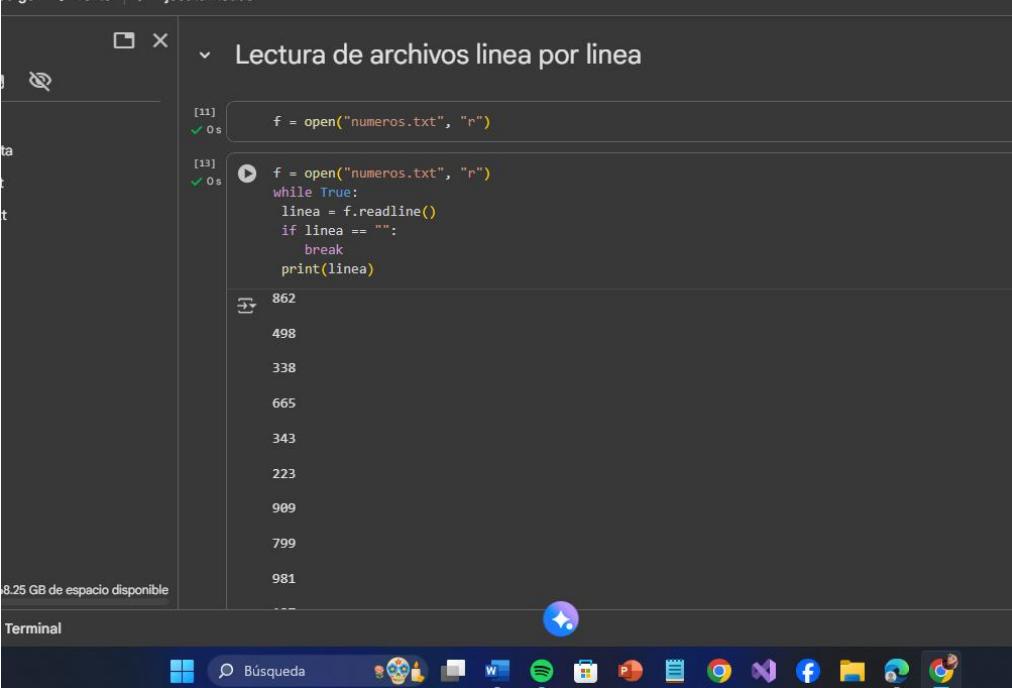
The screenshot shows an R console window. The code entered is:

```

> set.seed(123) # Puedes omitir esta linea si no te importa que los números ca$>
> numeros <- sample(1:1000, 100, replace = TRUE)
>
> lineas <- as.character(numeros)
>
> writeLines(lineas, "numeros.txt")
>
> cat("Archivo 'numeros.txt' creado con 100 números aleatorios.\n")
Archivo 'numeros.txt' creado con 100 números aleatorios.
> cat(readLines("numeros.txt"), sep = "\n")
415
463
179
526
195
935
818
118
299
229
244
14
374
665
602
603
768
709
91
953
348
649
989
355
840
26
519

```

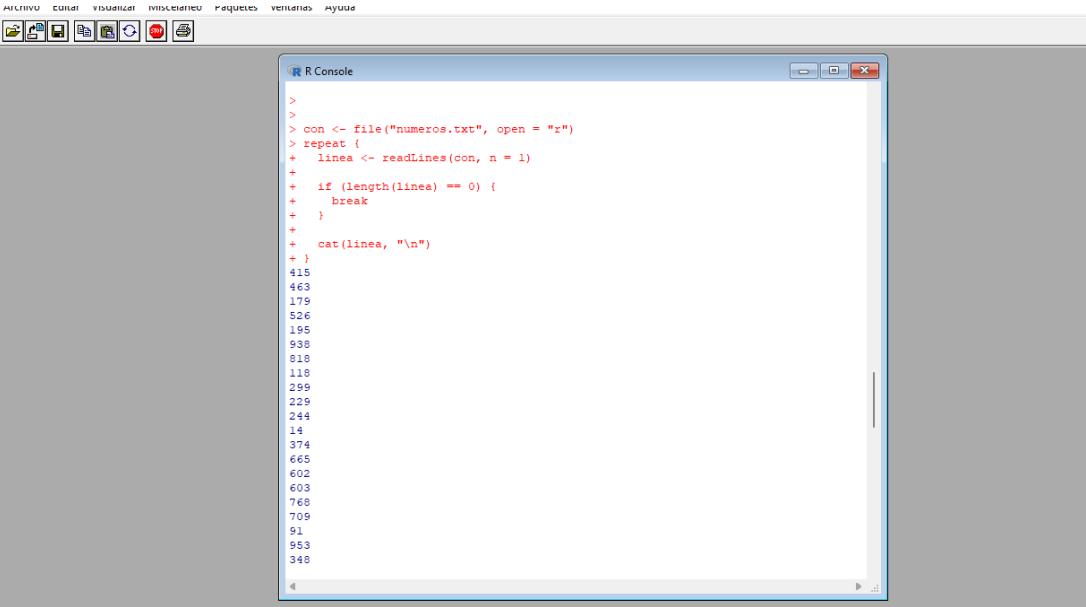
3. LECTURA DE ARCHIVOS LÍNEA POR LÍNEA



```
[11] f = open("numeros.txt", "r")
[13] f = open("numeros.txt", "r")
> while True:
>     linea = f.readline()
>     if linea == "":
>         break
>     print(linea)
862
498
338
665
343
223
999
799
981
--
```

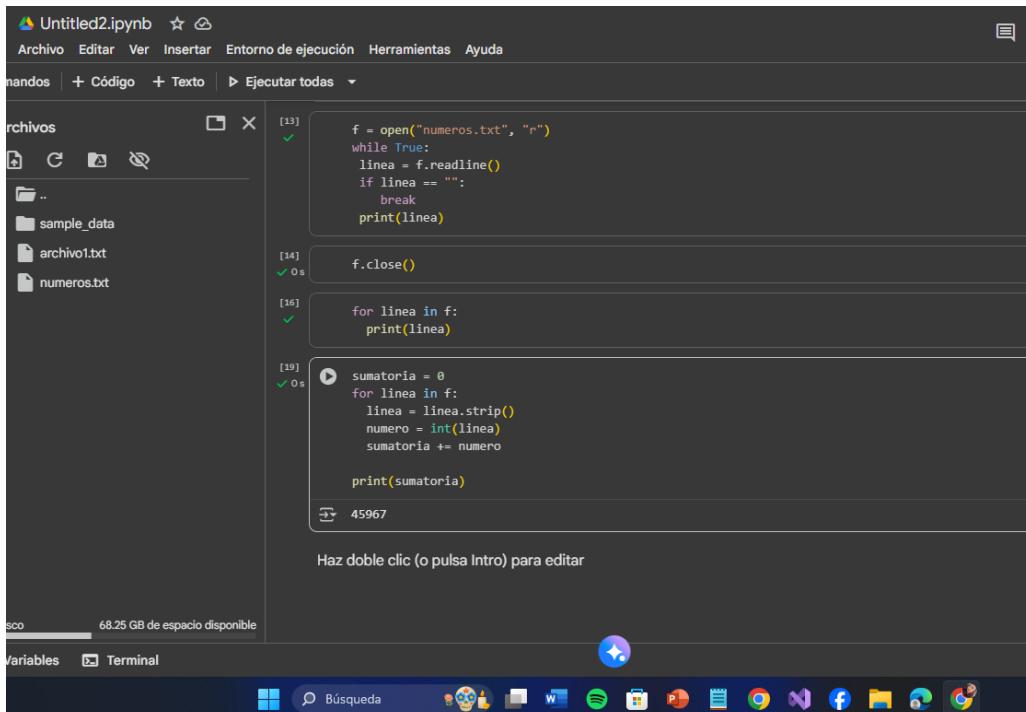
- Este código abre el archivo numeros.txt en modo lectura ("r") y lee su contenido línea por línea. Usa un bucle while True que se ejecuta indefinidamente hasta que ya no hay más líneas por leer.
- Dentro del bucle, f.readline() lee una línea del archivo; si la línea está vacía (""), significa que se llegó al final del archivo y se sale del bucle con break.
- Si no, imprime la línea con print(linea).
- Como cada línea ya incluye un salto de línea (\n), el resultado puede mostrarse con líneas en blanco entre ellas, a menos que se use print(linea.strip()) para limpiar los saltos.

Lenguaje R



```
>
>
> con <- file("numeros.txt", open = "r")
> repeat {
+   linea <- readLines(con, n = 1)
+
+   if (length(linea) == 0) {
+     break
+   }
+
+   cat(linea, "\n")
+ }
415
463
179
526
195
938
818
118
299
229
244
14
374
665
602
603
768
709
91
953
348
```

4. SUMATORIA



```

Untitled2.ipynb
Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda
+ Código + Texto ▶ Ejecutar todas
Archivos
f = open("numeros.txt", "r")
while True:
    linea = f.readline()
    if linea == "":
        break
    print(linea)

f.close()

for linea in f:
    print(linea)

sumatoria = 0
for linea in f:
    linea = linea.strip()
    numero = int(linea)
    sumatoria += numero

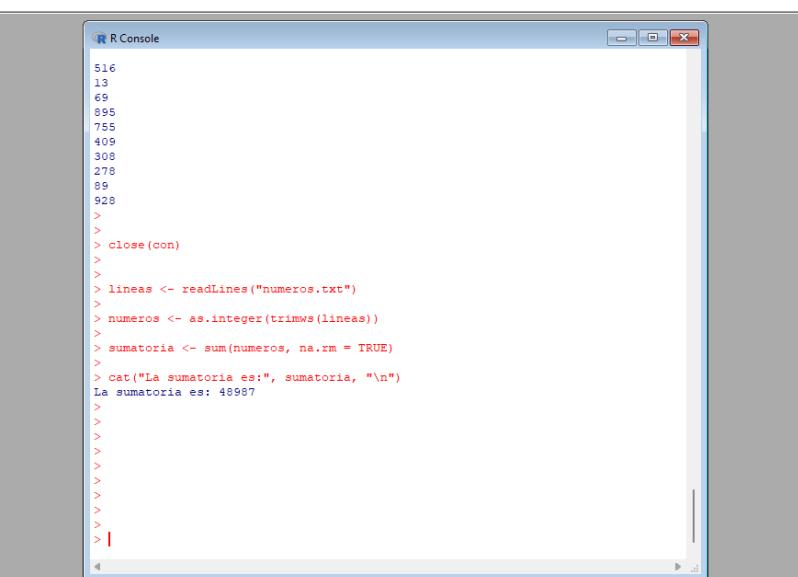
print(sumatoria)
45967

```

Haz doble clic (o pulsa Intro) para editar

- Este código calcula la suma de todos los números guardados en el archivo numeros.txt. Supone que f ya está abierto en modo lectura.
- Recorre cada línea del archivo con un bucle for, elimina los espacios en blanco y saltos de línea con strip(), convierte el texto a número entero con int(), y lo va sumando en la variable sumatoria.
- Al final, imprime el total de la suma de todos los números del arch

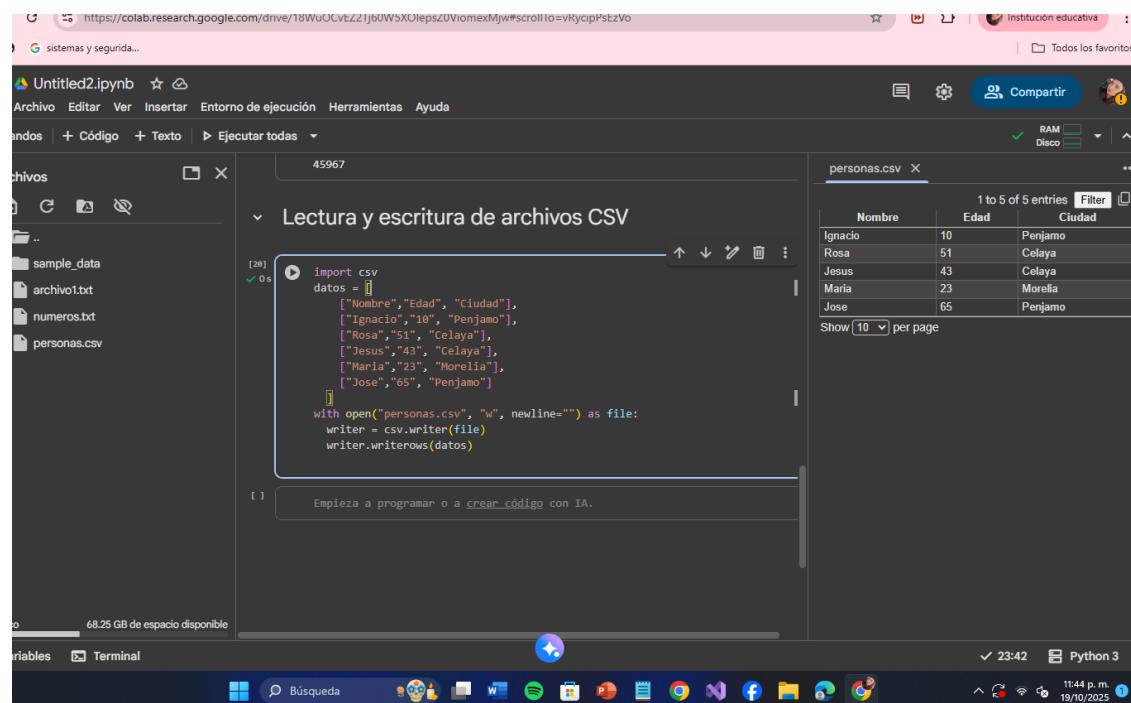
Lenguaje R



```

516
13
69
895
755
409
308
278
69
928
>
>
> close(con)
>
>
> lineas <- readLines("numeros.txt")
>
> numeros <- as.integer(trimws(lineas))
>
> sumatoria <- sum(numeros, na.rm = TRUE)
>
> cat("La sumatoria es:", sumatoria, "\n")
La sumatoria es: 45967
>
>
>
>
>
>
>
> |
```

5. ARCHIVOS CSV EN MODO ESCRITURA



The screenshot shows a Google Colab notebook titled "Untitled2.ipynb". The code cell contains the following Python script:

```

import csv
datos = [
    ["Nombre", "Edad", "Ciudad"],
    ["Ignacio", "10", "Penjamo"],
    ["Rosa", "51", "Celaya"],
    ["Jesus", "43", "Celaya"],
    ["Maria", "23", "Morelia"],
    ["Jose", "65", "Penjamo"]
]
with open("personas.csv", "w", newline="") as file:
    writer = csv.writer(file)
    writer.writerows(datos)

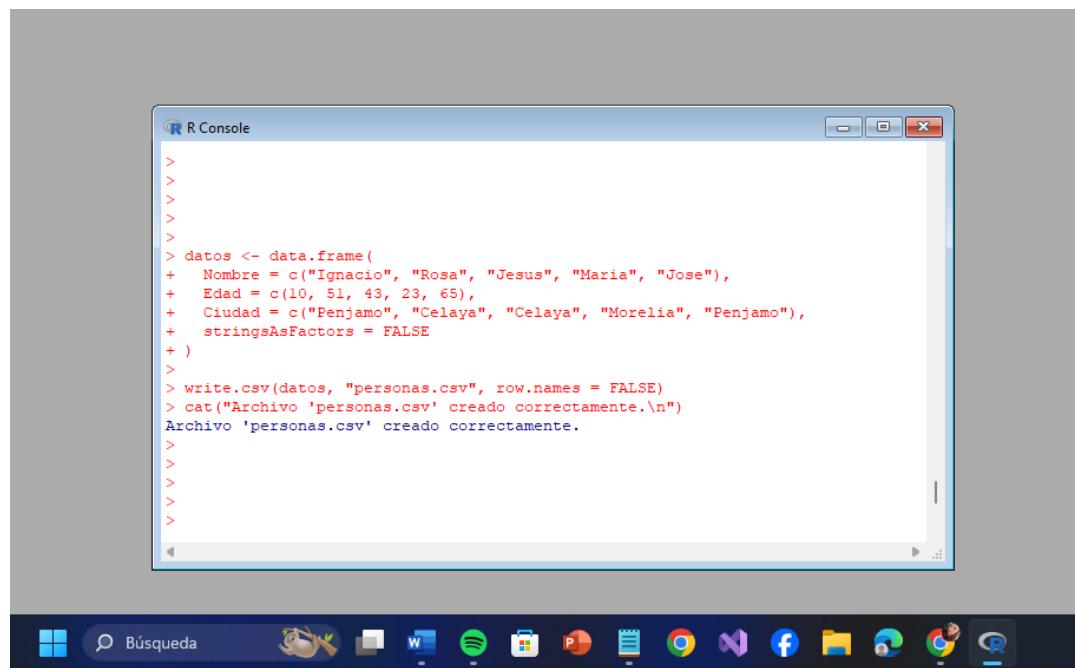
```

To the right of the code cell, there is a preview of the "personas.csv" file. The table has three columns: Nombre, Edad, and Ciudad. The data is as follows:

Nombre	Edad	Ciudad
Ignacio	10	Penjamo
Rosa	51	Celaya
Jesus	43	Celaya
Maria	23	Morelia
Jose	65	Penjamo

- Abre (o crea) el archivo personas.csv en modo escritura ("w").
- Usa csv.writer() para escribir datos en formato CSV.
- writer.writerows(datos) escribe todas las filas del arreglo datos en el archivo.
- newline="" evita que se inserten líneas vacías extra entre cada fila

Lenguaje R:



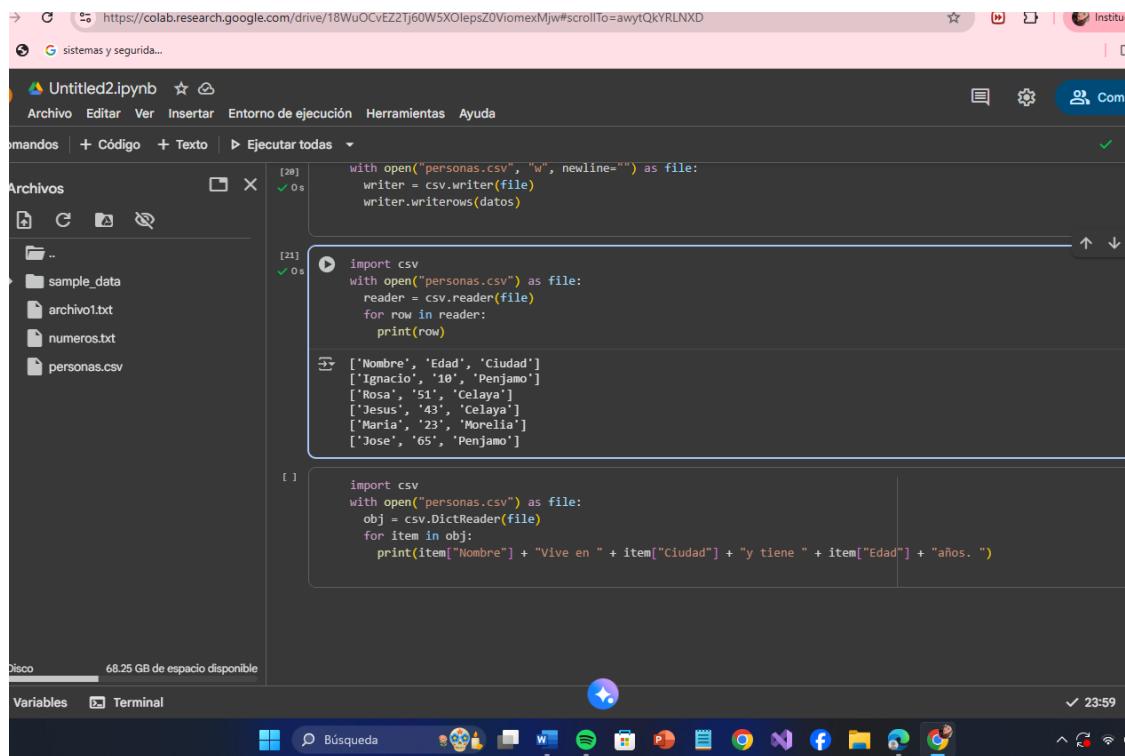
The screenshot shows an R Console window. The R code entered is:

```

>
>
>
>
>
> datos <- data.frame(
+   Nombre = c("Ignacio", "Rosa", "Jesus", "Maria", "Jose"),
+   Edad = c(10, 51, 43, 23, 65),
+   Ciudad = c("Penjamo", "Celaya", "Celaya", "Morelia", "Penjamo"),
+   stringsAsFactors = FALSE
+ )
>
> write.csv(datos, "personas.csv", row.names = FALSE)
> cat("Archivo 'personas.csv' creado correctamente.\n")
Archivo 'personas.csv' creado correctamente.
>
>
>
>
>

```

6. ARCHIVOS CSV CONTENIDO LÍNEA POR LÍNEA



The screenshot shows a Google Colab notebook titled "Untitled2.ipynb". The code cell contains the following Python script:

```

import csv
with open("personas.csv") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)

['Nombre', 'Edad', 'Ciudad']
['Ignacio', '10', 'Penjamo']
['Rosa', '51', 'Celaya']
['Jesus', '43', 'Celaya']
['Maria', '23', 'Morelia']
['Jose', '65', 'Penjamo']

```

Below the code cell, there is another cell with the following code:

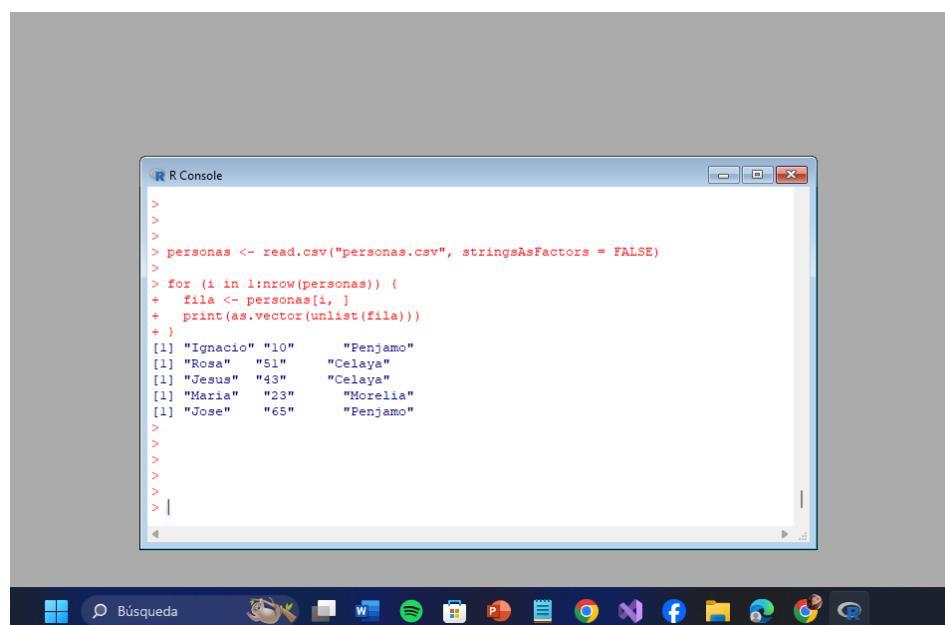
```

import csv
with open("personas.csv") as file:
    obj = csv.DictReader(file)
    for item in obj:
        print(item["Nombre"] + " Vive en " + item["Ciudad"] + " y tiene " + item["Edad"] + " años. ")

```

- Este código lee el archivo personas.csv y muestra su contenido línea por línea.
- Primero se importa el módulo csv, que permite trabajar con archivos CSV de forma sencilla.
- Luego se abre el archivo personas.csv en modo lectura usando with open(...), lo que asegura que el archivo se cierre automáticamente al terminar.
- Con csv.reader(file) se crea un lector que interpreta cada línea del archivo como una lista de valores separados por comas.
- Finalmente, se recorre cada fila del archivo con un bucle for y se imprime cada fila como una lista, mostrando por pantalla el contenido completo del archivo CSV.

Lenguaje R:



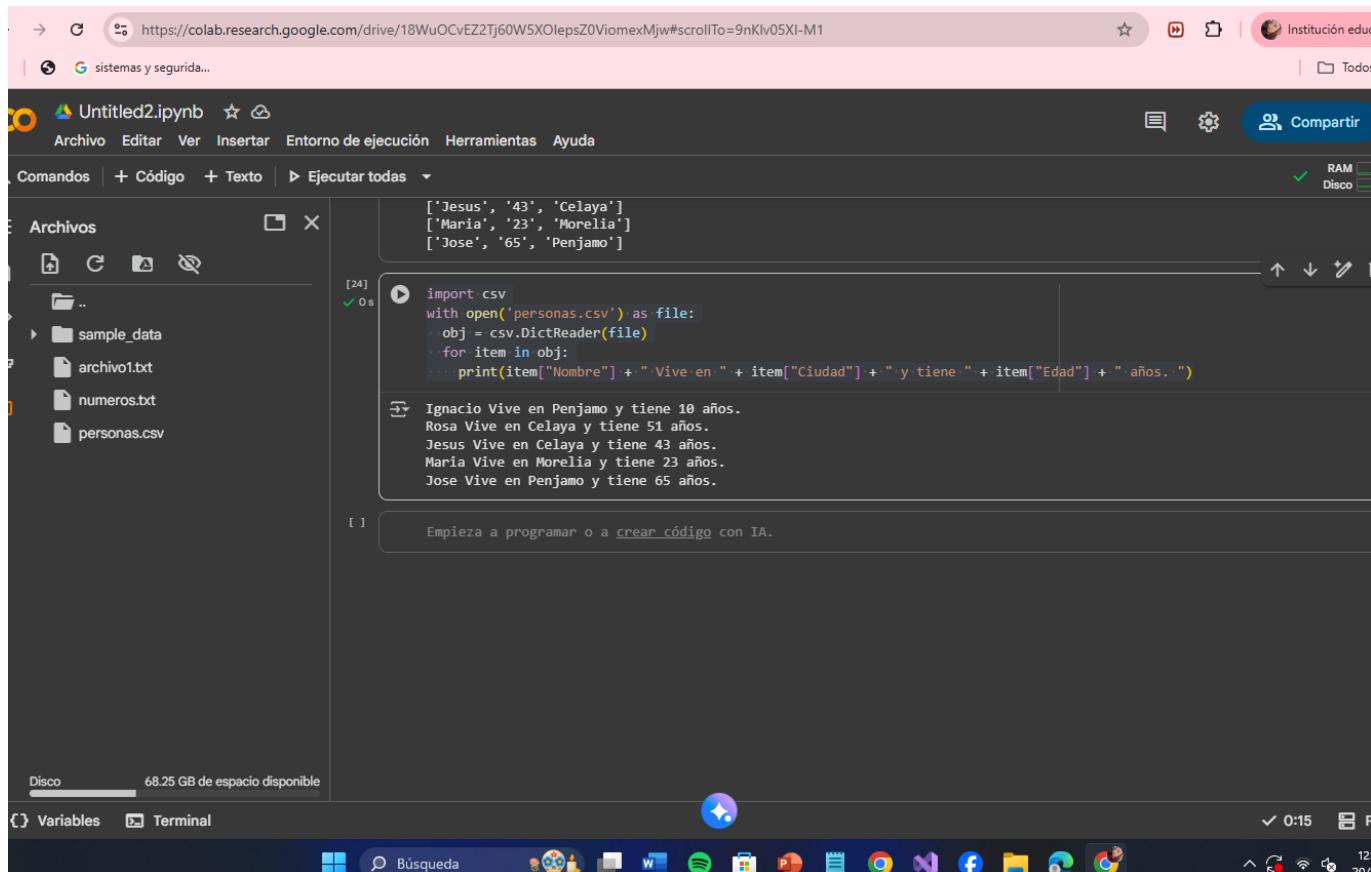
The screenshot shows an R Console window with the following R script and output:

```

>
>
>
> personas <- read.csv("personas.csv", stringsAsFactors = FALSE)
>
> for (i in 1:nrow(personas)) {
+   fila <- personas[i, ]
+   print(as.vector(unlist(fila)))
+ }
[1] "Ignacio" "10"      "Penjamo"
[1] "Rosa"     "51"      "Celaya"
[1] "Jesus"    "43"      "Celaya"
[1] "Maria"    "23"      "Morelia"
[1] "Jose"     "65"      "Penjamo"
>
>
>
>
>
> |

```

7. ARCHIVOS CSV LEER CON DICTREADER



```

['Jesus', '43', 'Celaya']
['Maria', '23', 'Morelia']
['Jose', '65', 'Penjamo']

[24]  ✓ 0s
import csv
with open('personas.csv') as file:
    obj = csv.DictReader(file)
    for item in obj:
        print(item["Nombre"] + " Vive en " + item["Ciudad"] + " y tiene " + item["Edad"] + " años.")

Ignacio Vive en Penjamo y tiene 10 años.
Rosa Vive en Celaya y tiene 51 años.
Jesus Vive en Celaya y tiene 43 años.
Maria Vive en Morelia y tiene 23 años.
Jose Vive en Penjamo y tiene 65 años.

[ ] Empieza a programar o a crear código con IA.

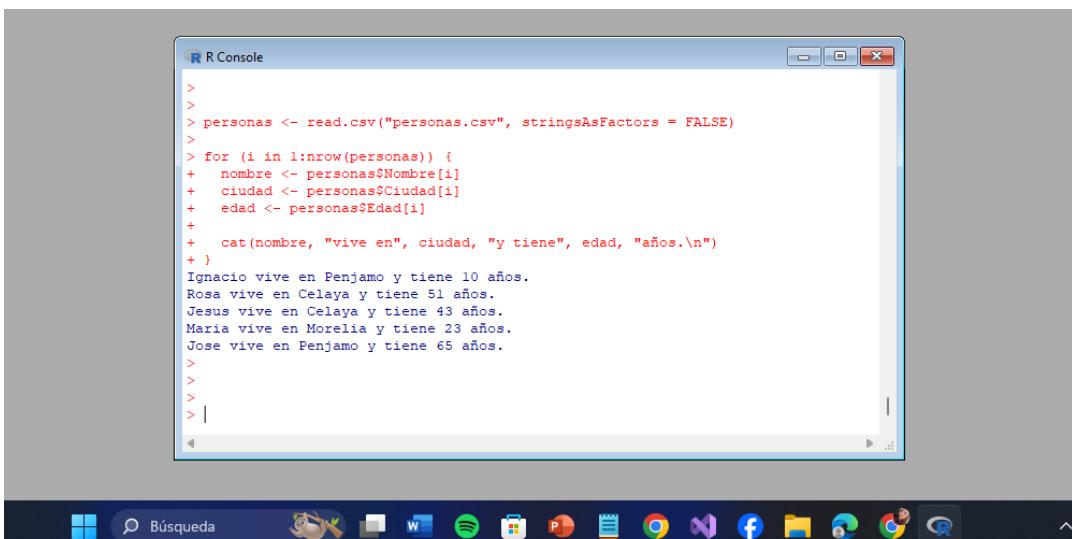
```

Disco 68.25 GB de espacio disponible

Variables Terminal 0:15 20/2

- Este código lee el archivo personas.csv y muestra un mensaje personalizado para cada persona usando los datos del archivo.
- Primero se importa el módulo csv. Luego se abre el archivo con with open(...), y se utiliza csv.DictReader(file) para leer cada fila del CSV como un diccionario, donde las claves son los encabezados de la primera fila del archivo (por ejemplo: "Nombre", "Edad", "Ciudad").
- En el bucle for, cada fila se guarda en la variable item, y se accede a los datos por su nombre de columna. Finalmente, se imprime una frase para cada persona como: "Ignacio Vive en Penjamo y tiene 10 años.", usando los valores del diccionario.

Lenguaje R



```

R Console
>
>
> personas <- read.csv("personas.csv", stringsAsFactors = FALSE)
>
> for (i in 1:nrow(personas)) {
+   nombre <- personas$Nombre[i]
+   ciudad <- personas$Ciudad[i]
+   edad <- personas$Edad[i]
+
+   cat(nombre, "vive en", ciudad, "y tiene", edad, "años.\n")
+ }
Ignacio vive en Penjamo y tiene 10 años.
Rosa vive en Celaya y tiene 51 años.
Jesus vive en Celaya y tiene 43 años.
Maria vive en Morelia y tiene 23 años.
Jose vive en Penjamo y tiene 65 años.
>
>
>
> | pág. 8

```

Búsqueda

CONCLUSIÓN

Python y R son dos lenguajes de programación ampliamente utilizados en el análisis de datos, la ciencia de datos y la programación en general, cada uno con sus fortalezas particulares. Python se destaca por su sintaxis clara y sencilla, su gran comunidad, y la gran cantidad de bibliotecas que facilitan tareas de manipulación de datos, automatización y desarrollo web, entre otras. En contraste, R es especialmente fuerte en análisis estadístico y visualización avanzada, siendo muy popular en la comunidad académica y de investigación debido a su enfoque en la estadística y las gráficas.

Durante las actividades realizadas, se exploraron varias funciones básicas de Python, especialmente orientadas al manejo de archivos y la generación de datos aleatorios. Se aprendió a abrir, escribir y leer archivos de texto, cómo manipular datos línea por línea y cómo trabajar con archivos CSV usando el módulo csv, incluyendo tanto lectura básica como lectura avanzada con DictReader para un acceso más semántico a los datos. Estas actividades son fundamentales para entender cómo Python facilita el procesamiento y análisis de datos en formatos comunes y cómo se puede automatizar el trabajo con grandes volúmenes de información.

Estas tareas demuestran la versatilidad de Python para actividades relacionadas con la manipulación de datos, un campo donde R también es muy competente pero con una orientación más estadística. Aprender a manejar archivos y a utilizar bibliotecas estándar de Python como csv es un paso esencial para cualquier persona que quiera desenvolverse en programación orientada a datos y análisis. En resumen, el uso de Python en estas actividades confirma su rol como una herramienta poderosa y accesible para programadores de todos los niveles, complementando las capacidades que ofrece R en entornos donde el análisis estadístico riguroso es prioridad.

BIBLIOGRAFÍA:

- Python Software Foundation. (2023). *Python documentation.* <https://docs.python.org/3/tutorial/inputoutput.html>
- Real Python. (2023). *Working with CSV Files in Python.* <https://realpython.com/python-csv/>
- The R Project for Statistical Computing. (2023). *R Documentation.* <https://cran.r-project.org/manuals.html>
- Welling, L. (2021). *Python vs R for Data Science.* Towards Data Science. <https://towardsdatascience.com/python-vs-r-for-data-science-4d348e401dd4>
- GeeksforGeeks. (2023). *File Handling in Python.* <https://www.geeksforgeeks.org/file-handling-python/>