

**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

<b>INTRODUCCIÓN .....</b>	<b>1</b>
<b>DESARROLLO DE LA PRÁCTICA.....</b>	<b>2</b>
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
<b>CONCLUSIÓN .....</b>	<b>9</b>
<b>BIBLIOGRAFÍA: .....</b>	<b>10</b>



## 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 Jupyter Notebook titled 'Untitled2.ipynb' in a web browser. The code is as follows:

```
[8]: f = open("archivo1.txt", "w")
✓ 0 s

[9]: f.write("Hola nuevamente...\nAdios")
✓ 0 s
24

[10]: f.close()
✓ 0 s

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

The file explorer on the left shows a folder named 'sample\_data' containing a file named 'archivo1.txt'. The output area on the right shows the contents of 'archivo1.txt':

```
archivo1.txt X
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 with the following code:

```
> f.close()
Error en f.close(): no se pudo encontrar la función "f.close"
> # Nombre del archivo
> archivo <- "archivo1.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 líneas
> writeLines(c("Hola", "Hola de nuevo..", "Adios"), con)
>
> # Cerrar el archivo
> close(con)
>
> cat("Texto escrito correctamente.\n")
Texto escrito correctamente.
> contenido <- readLines("archivo1.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 is as follows:

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

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

[64] print(valores)
```

The output of the code is displayed in the right-hand pane, showing 100 random numbers written to the file 'numeros.txt'.

- 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 with the following code:

```
> 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
938
818
110
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

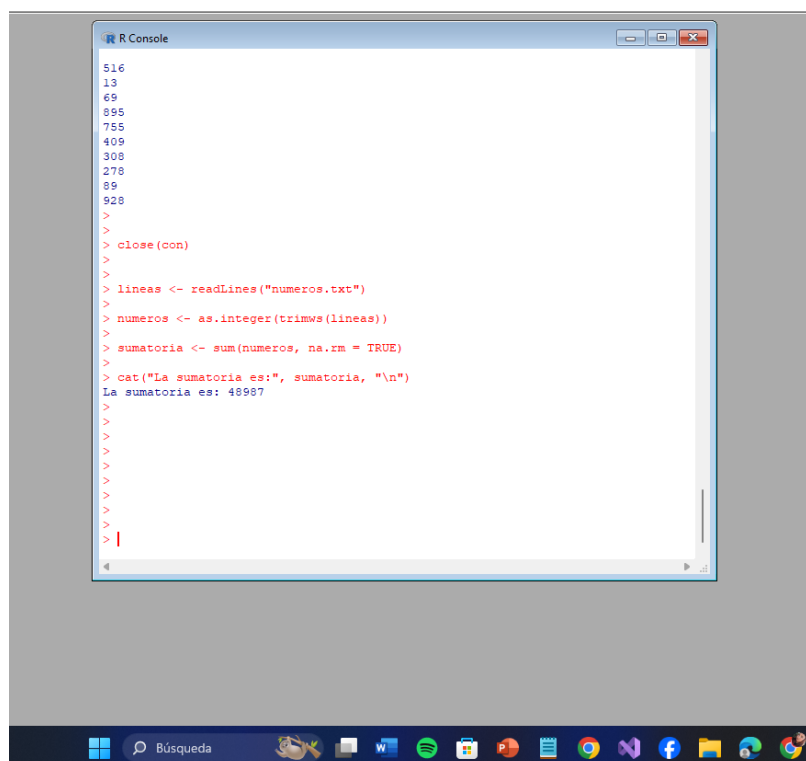
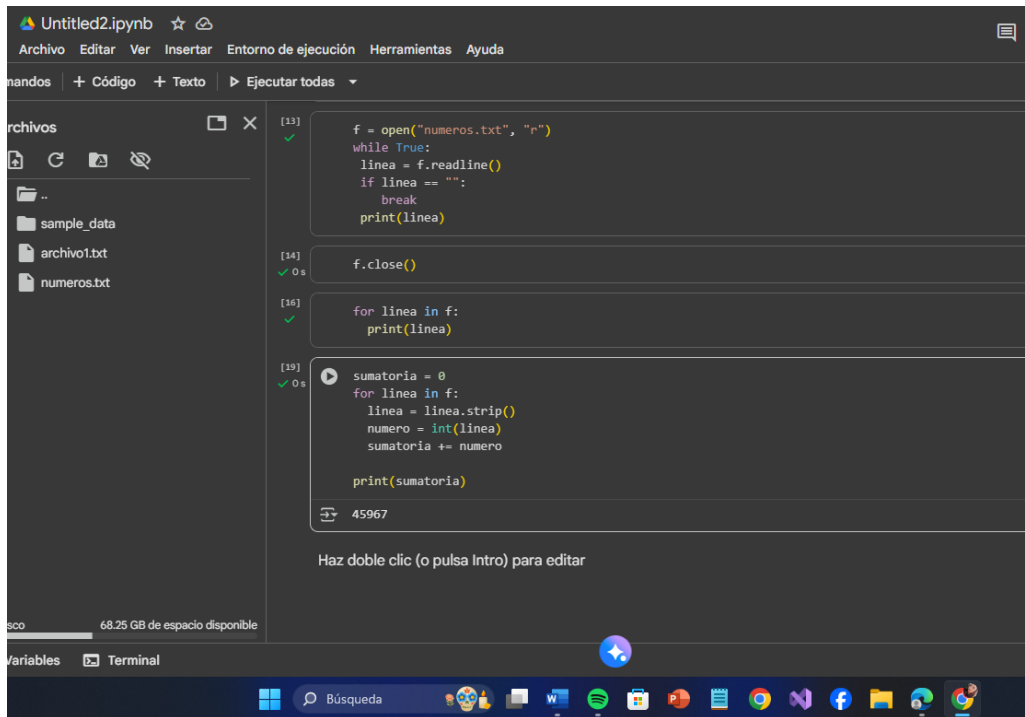
```
[11] f = open("numeros.txt", "r")
[13] f = open("numeros.txt", "r")
while True:
    linea = f.readline()
    if linea == "":
        break
    print(linea)
862
498
338
665
343
223
909
799
981
```

- 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
```







## 5. ARCHIVOS CSV EN MODOS ESCRITURA

The screenshot shows a Google Colab environment. On the left, a file explorer shows a folder named 'sample\_data' containing 'archivo1.txt', 'numeros.txt', and 'personas.csv'. The main code editor contains the following Python code:

```
[28] 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)
```

On the right, a preview of the 'personas.csv' file is shown with the following data:

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 with the following R code:

```
>
>
>
>
>
> 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 Jupyter Notebook with the following code cells:

```
[20] with open("personas.csv", "w", newline="") as file:
      writer = csv.writer(file)
      writer.writerow(datos)

[21] 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']
```

The interface also shows a file explorer on the left with files like 'sample\_data', 'archivo1.txt', 'numeros.txt', and 'personas.csv'. The bottom status bar indicates 68.25 GB of available space.

- 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 code:

```
> 
> 
> 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"
> 
> 
> 
> |
```

The console output shows the contents of the CSV file being printed line by line. The bottom status bar shows the Windows taskbar with various application icons.



## 7. ARCHIVOS CSV LEER CON DICTREADER

The screenshot shows a Google Colab notebook titled 'Untitled2.ipynb'. The left sidebar displays a file explorer with a folder 'sample\_data' containing files 'archivo1.txt', 'numeros.txt', and 'personas.csv'. The main area shows a code cell with the following Python code:

```
[24] 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.")
```

The output of the code cell shows the following personalized messages:

```
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.
```

Below the code cell, there is a prompt: 'Empieza a programar o a crear código con IA.'

- 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

The screenshot shows an R Console window with the following R code:

```
>
>
> 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.
>
>
> |
```



## 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/>