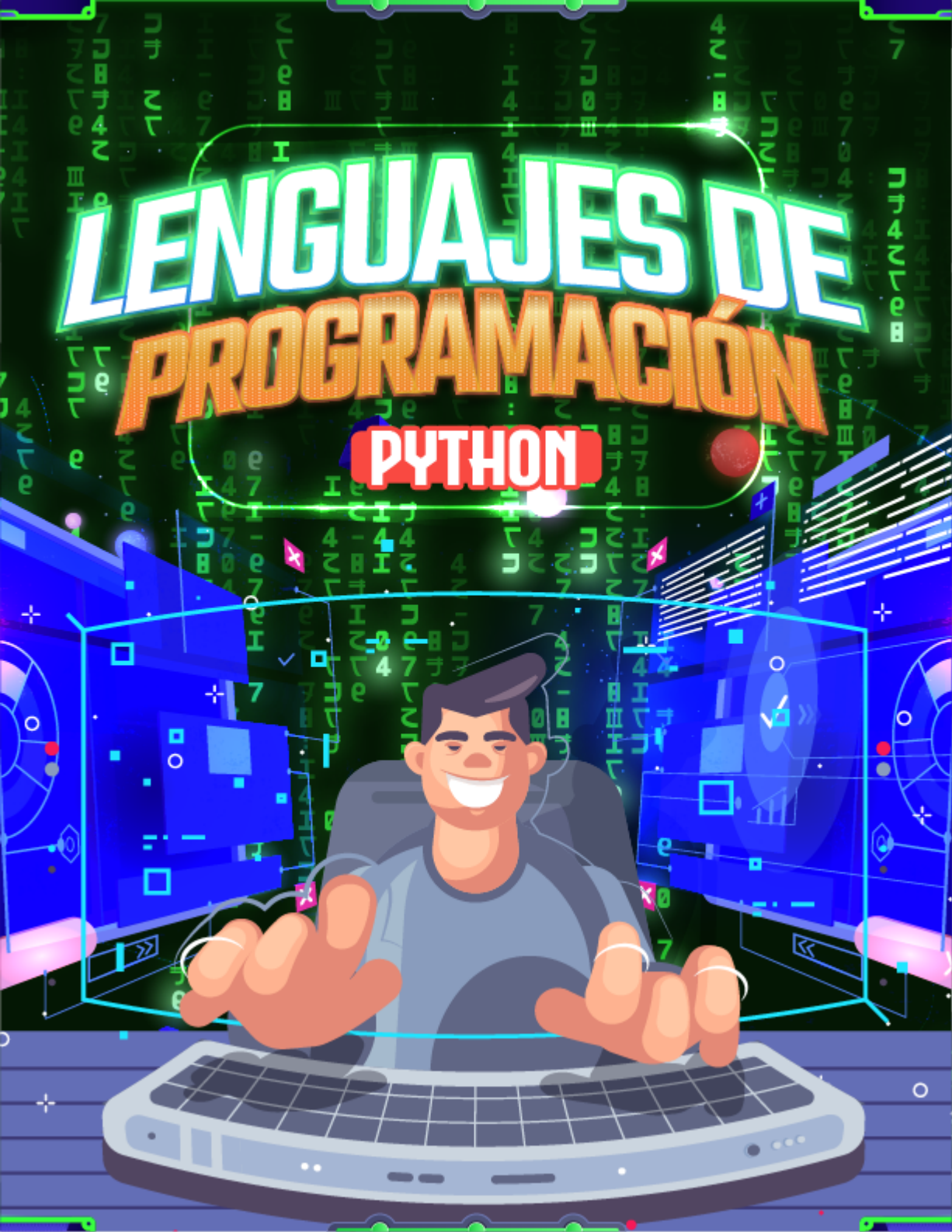
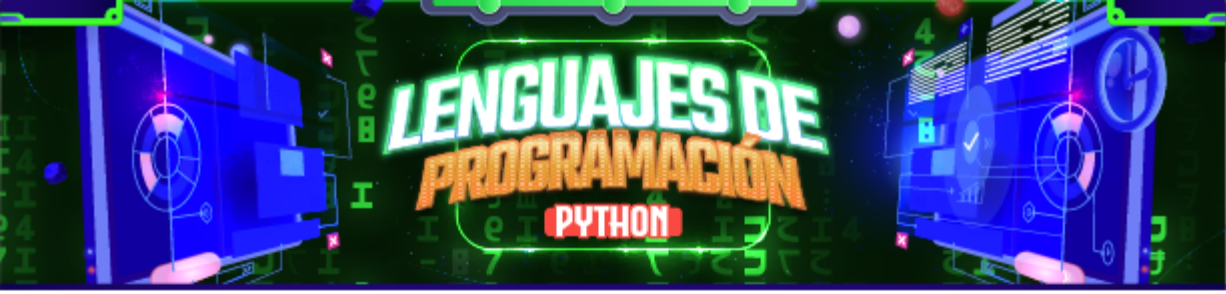
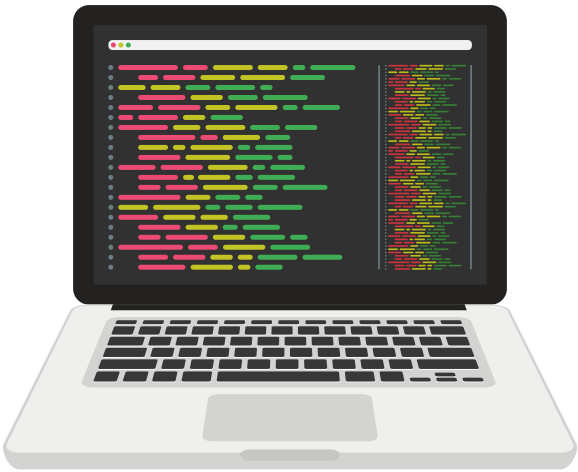
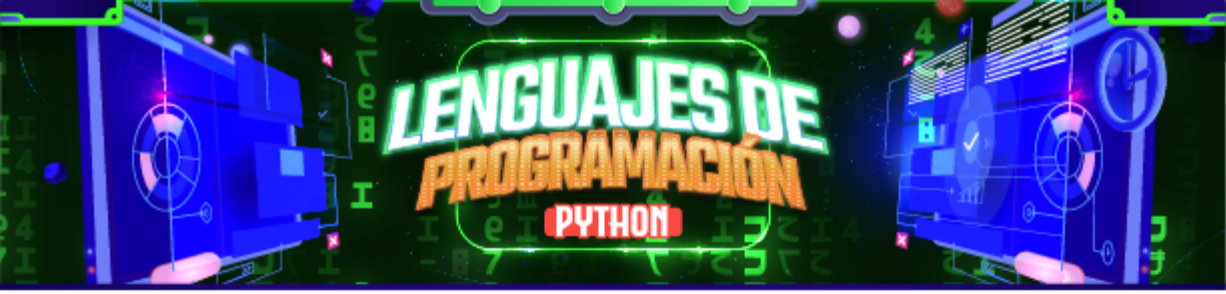

LENGUAJES DE
PROGRAMACIÓN
PYTHON



Gestión de archivos

La gestión de archivos es una parte fundamental de la programación, ya que nos permite leer, escribir y manipular información almacenada en archivos en Python. Esta habilidad es especialmente útil cuando queremos trabajar con datos persistentes, como archivos de texto o bases de datos. En esta lección, exploraremos cómo gestionar archivos en Python, incluyendo la lectura y escritura de archivos, el manejo de rutas de archivo y la manipulación de datos almacenados en archivos. A través de ejemplos y demostraciones, aprenderemos cómo utilizar estas técnicas para organizar y manipular información de manera eficiente.





Lectura y Escritura de Archivos

Para leer o escribir datos en un archivo en Python, necesitamos abrir el archivo en el modo adecuado.

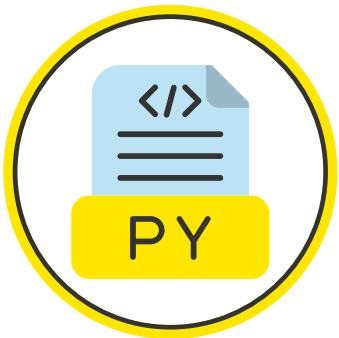
A continuación, se presentan algunos ejemplos:

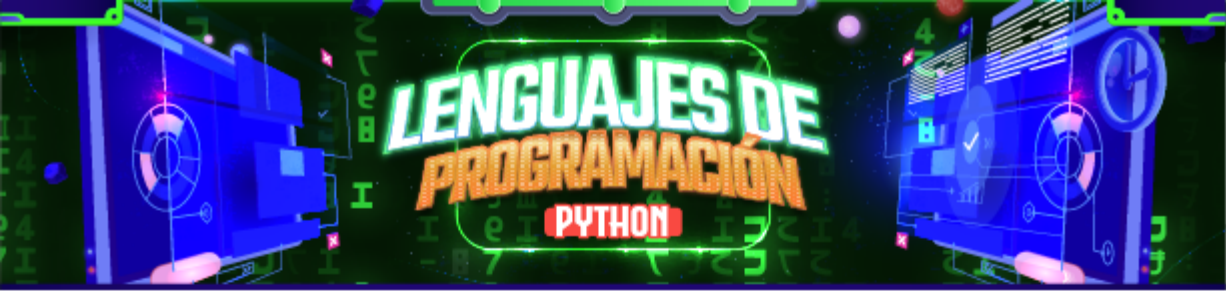
```
 1. fLectura de un archivo:

# Abrir un archivo en modo lectura
archivo = open("datos.txt", "r")

# Leer el contenido del archivo
contenido = archivo.read()

# Cerrar el archivo
archivo.close()
```

En este ejemplo, abrimos el archivo "datos.txt" en modo lectura utilizando la función open() y el modo "r". Luego, utilizamos el método read() para leer todo el contenido del archivo y lo almacenamos en la variable contenido. Finalmente, cerramos el archivo utilizando el método close().

```
2.Escritura en un archivo:

# Abrir un archivo en modo escritura
archivo = open("datos.txt", "w")

# Escribir en el archivo
archivo.write("Hola, mundo!")

# Cerrar el archivo
archivo.close()
```

En este ejemplo, abrimos el archivo "datos.txt" en modo escritura utilizando el modo "w". Luego, utilizamos el método write() para escribir el texto "Hola, mundo!" en el archivo. Finalmente, cerramos el archivo utilizando el método close().



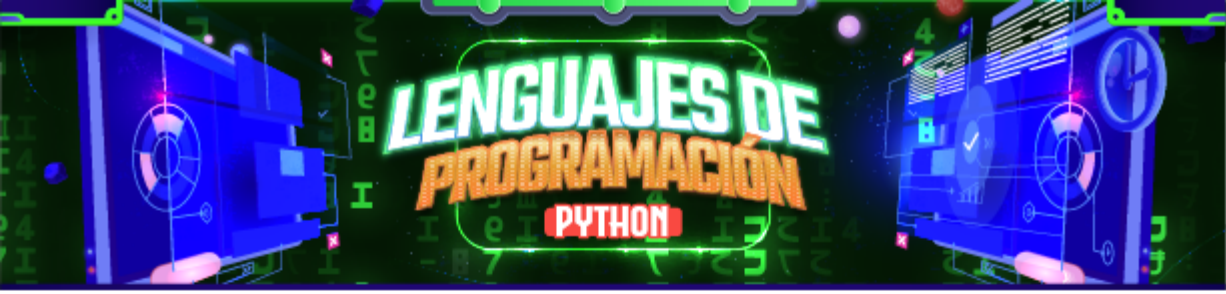


Manejo de Rutas de Archivo

En ocasiones, necesitamos manipular rutas de archivo para trabajar con archivos en diferentes ubicaciones o directorios. Python proporciona la biblioteca os.path para manejar las rutas de archivo de manera eficiente. Veamos un ejemplo:

```
import os

# Obtener la ruta absoluta de un archivo
ruta_absoluta = os.path.abspath("datos.txt")

# Comprobar si un archivo existe
existe = os.path.exists("datos.txt")

# Obtener el nombre base de un archivo
nombre_base = os.path.basename("datos.txt")

# Unir dos rutas de archivo
ruta1 = "ruta1"
ruta2 = "ruta2"
ruta_completa = os.path.join(ruta1, ruta2)
```

En este ejemplo, importamos la biblioteca os y utilizamos las funciones y métodos proporcionados por os.path. Utilizamos abspath() para obtener la ruta absoluta de un archivo, exists() para comprobar si un archivo existe, basename() para obtener el nombre base de un archivo y join() para unir dos rutas de archivo.

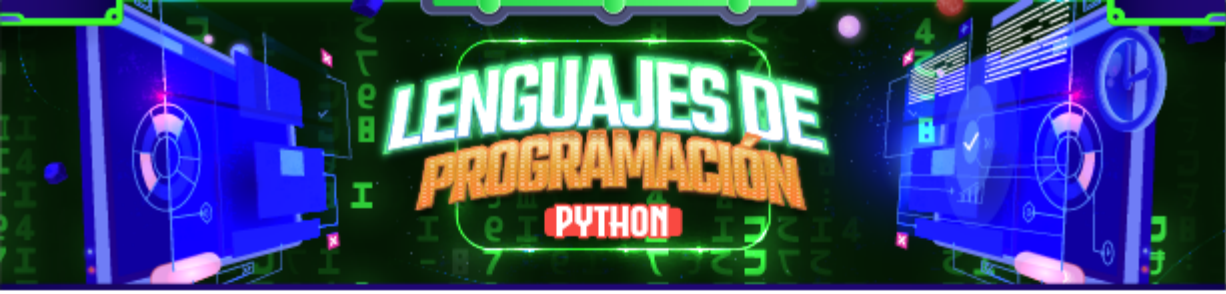


Manipulación de Datos Almacenados en Archivos

En ocasiones, necesitamos manipular rutas de archivo para trabajar con archivos en diferentes ubicaciones o directorios. Python proporciona la biblioteca os.path para manejar las rutas de archivo de manera eficiente. Veamos un ejemplo:

```
# Leer el contenido del archivo
archivo = open("datos.txt", "r")
contenido = archivo.read()
archivo.close()

# Dividir el contenido en líneas
lineas = contenido.split("\n")

# Unir las líneas con un separador
nuevo_contenido = "\n".join(lineas)
```

En este ejemplo, leemos el contenido del archivo "datos.txt" y lo almacenamos en la variable contenido. Luego, utilizamos el método split() para dividir el contenido en líneas utilizando el carácter de salto de línea ("\n"). Después, utilizamos el método join() para unir las líneas nuevamente con el carácter de salto de línea, almacenando el resultado en la variable nuevo_contenido.



La gestión de archivos en Python nos permite organizar y manipular información de forma eficiente.

A través de la lectura y escritura de archivos, podemos almacenar y recuperar datos persistentes. El manejo de rutas de archivo nos permite trabajar con archivos ubicados en diferentes directorios. Además, podemos manipular los datos almacenados en los archivos utilizando las herramientas proporcionadas por Python.

Con estas técnicas, podemos desarrollar aplicaciones que interactúen con archivos de manera eficiente y efectiva. ¡Anímate a explorar y experimentar con la gestión de archivos en Python para ampliar tus habilidades de programación!