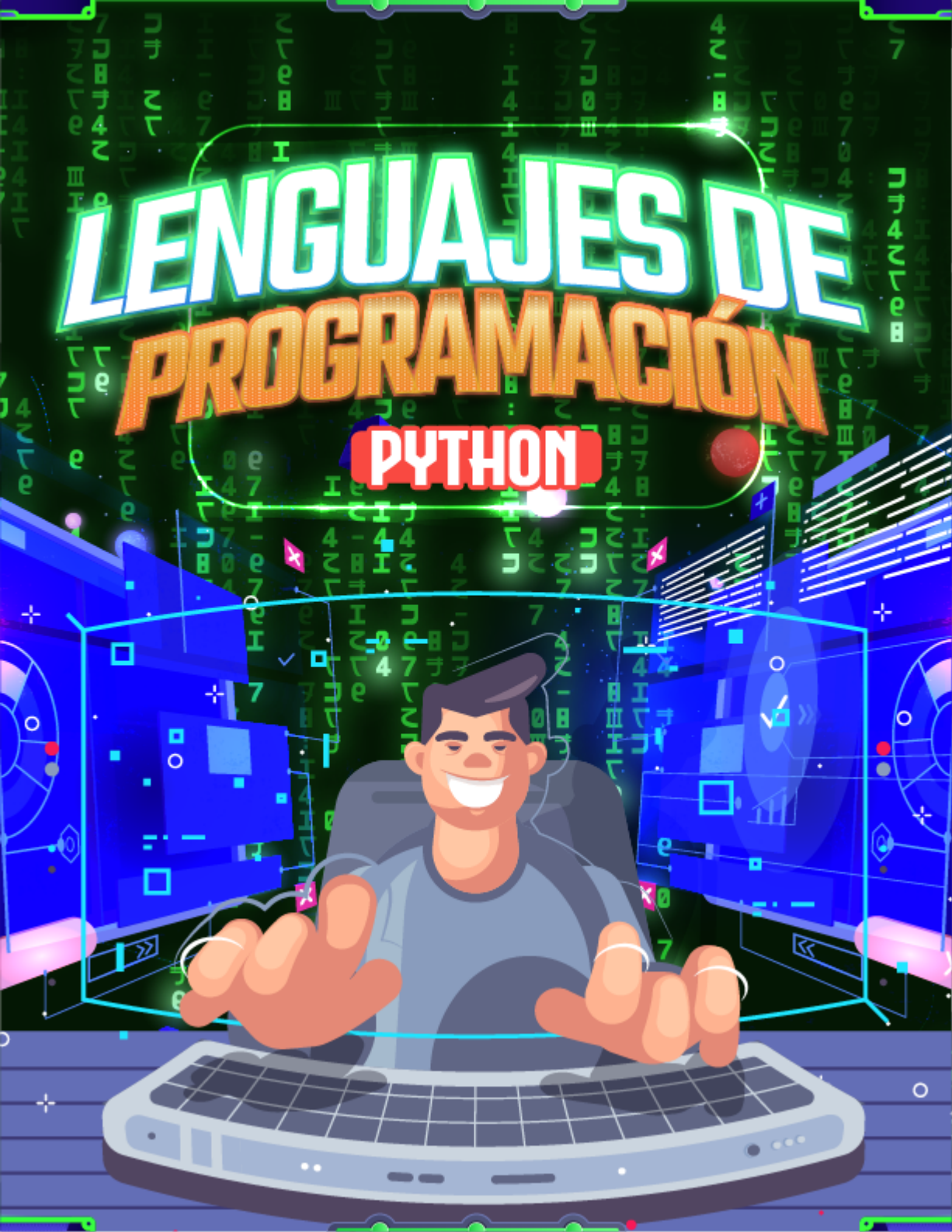
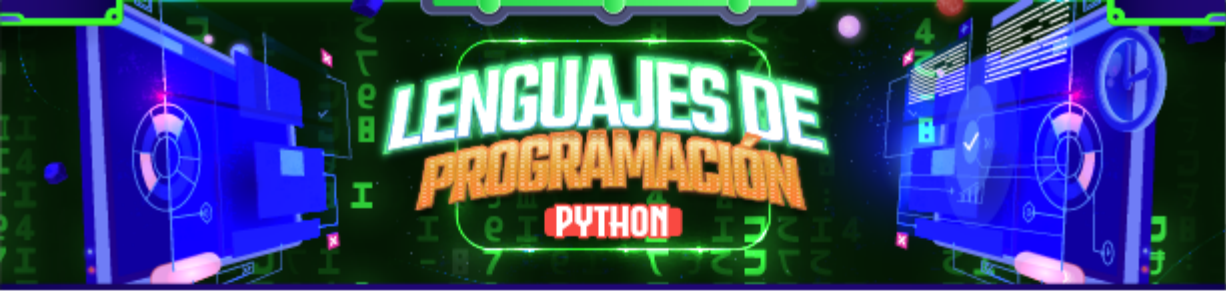
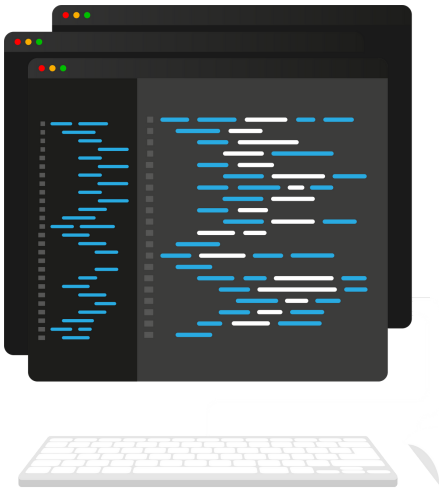
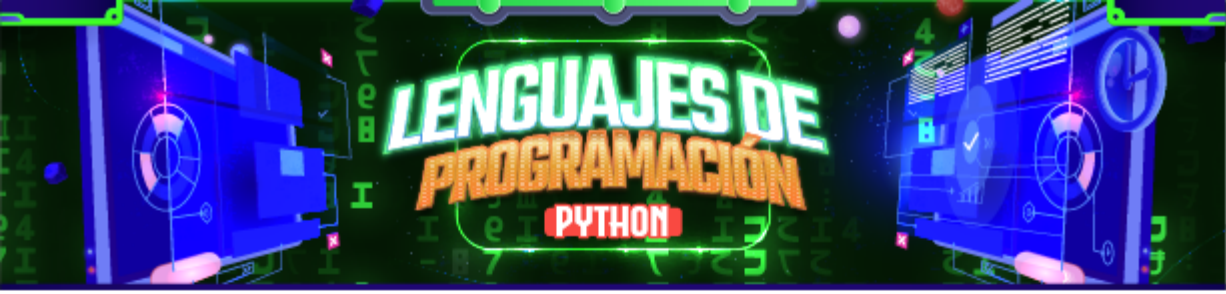

LENGUAJES DE
PROGRAMACIÓN
PYTHON



Conexión a Bases de Datos

En el mundo de la programación, es común trabajar con bases de datos para almacenar y acceder a grandes cantidades de información. En Python, podemos conectar y manipular bases de datos utilizando bibliotecas específicas que proporcionan interfaces sencillas y eficientes. En esta lección, exploraremos cómo establecer conexiones a bases de datos en Python, realizar consultas y manipular los datos almacenados en ellas. A través de ejemplos y demostraciones, aprenderemos cómo utilizar estas técnicas para acceder y gestionar datos de forma eficiente.



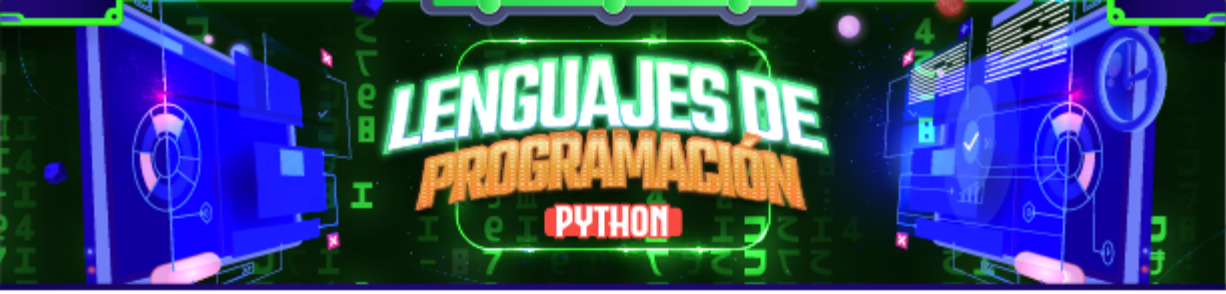


Establecimiento de Conexiones a Bases de Datos

Para conectarnos a una base de datos en Python, necesitamos utilizar una biblioteca específica que admita el motor de base de datos que estamos utilizando. Algunas bibliotecas populares para la conexión a bases de datos en Python son sqlite3, MySQL Connector, psycopg2 (para PostgreSQL), entre otras. A continuación, presentaremos un ejemplo utilizando sqlite3:

```
import sqlite3

# Establecer una conexión a la base de datos
conexion = sqlite3.connect('basedatos.db')

# Realizar operaciones en la base de datos

# Cerrar la conexión a la base de datos
conexion.close()
```

En este ejemplo, importamos la biblioteca sqlite3 y utilizamos la función connect() para establecer una conexión a la base de datos basedatos.db. Luego, podemos realizar operaciones en la base de datos, como consultas y modificaciones. Finalmente, cerramos la conexión utilizando el método close().

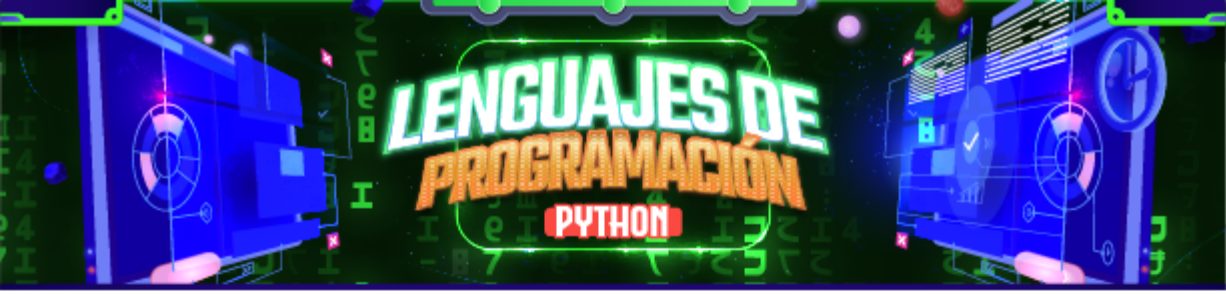


Consultas y Manipulación de Datos

Una vez establecida la conexión a la base de datos, podemos realizar consultas y manipulaciones de datos utilizando sentencias SQL. A continuación, presentaremos un ejemplo básico de consulta utilizando sqlite3:

```
import sqlite3

# Establecer una conexión a la base de datos
conexion = sqlite3.connect('basedatos.db')

# Crear un cursor para ejecutar las consultas
cursor = conexion.cursor()

# Ejecutar una consulta SELECT
cursor.execute("SELECT * FROM usuarios")

# Obtener los resultados de la consulta
resultados = cursor.fetchall()

# Mostrar los resultados
for resultado in resultados:
    print(resultado)

# Cerrar el cursor y la conexión a la base de datos
cursor.close()
conexion.close()
```

En este ejemplo, después de establecer la conexión a la base de datos, creamos un cursor utilizando el método cursor() de la conexión. Luego, ejecutamos una consulta SELECT utilizando el método execute() del cursor y obtenemos los resultados utilizando el método fetchall(). Finalmente, recorremos los resultados y los mostramos en la consola.



Bibliotecas ORM

Además de trabajar directamente con sentencias SQL, también existen bibliotecas ORM (Object-Relational Mapping) en Python que simplifican el acceso y manipulación de datos en bases de datos.

Estas bibliotecas mapean las tablas de la base de datos a objetos en Python, lo que facilita la manipulación de los datos. Ejemplos de bibliotecas ORM populares son SQLAlchemy y Django ORM.

La conexión a bases de datos en Python nos permite acceder y manipular datos de manera eficiente. Utilizando las bibliotecas adecuadas, podemos establecer conexiones a bases de datos, realizar consultas y modificar los datos almacenados en ellas.

Ya sea trabajando con sentencias SQL directamente o utilizando bibliotecas ORM, Python nos brinda herramientas poderosas para interactuar con bases de datos. ¡Anímate a explorar y experimentar con la conexión a bases de datos en Python y amplía tus habilidades de programación!

python™