

CONEXIÓN PYTHON

MariaDB | PostgreSQL | OracleDB



Gestión de bases de datos | I.E.S Gonzalo Nazareno

Concepción Guisado Jurado

1º A.S.I.R (Administración de sistemas informáticos en red)

2025-2026

Jesús Figueroa Roldán



Índice

Dependencias necesarias.....	3
Conector Python - MariaDB.....	4
Conector Python - PostgreSQL.....	4
Conector Python - Oracle.....	5
Funcionamiento del programa.....	7
MariaDB.....	7
PostgreSQL.....	10
Oracle.....	12
Bibliografía.....	14



Dependencias necesarias

Para que este programa en Python nos funcione correctamente primeramente tendremos que instalar unas dependencias necesarias.

```
jesusfigueroa ASIR ~/SGBD-Python-Jesús-Figueroa-Roldán sudo apt install python3-mysqldb python3-psycopg2
python3-dev default-libmysqlclient-dev build-essential libpq-dev build-essential pkg-config
[sudo] contraseña para jesusfigueroa:
python3-mysqldb ya está en su versión más reciente (1.4.6-2+b5).
python3-psycopg2 ya está en su versión más reciente (2.9.10-1+b1).
python3-dev ya está en su versión más reciente (3.13.5-1).
default-libmysqlclient-dev ya está en su versión más reciente (1.1.1).
build-essential ya está en su versión más reciente (12.12).
libpq-dev ya está en su versión más reciente (17.7-0+deb13u1).
pkg-config ya está en su versión más reciente (1.8.1-4).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0

jesusfigueroa ASIR ~/SGBD-Python-Jesús-Figueroa-Roldán
```

Una vez que tengamos las dependencias necesarias instaladas vamos a crearnos un entorno virtual en python para trabajar desde ahí y tenerlo aislado del sistema. También esto es bueno para que no haya fallos entre las versiones de los repositorios y demás. Una vez que nos hagamos el entorno virtual lo vamos a activar y voy a instalar el conector de MariaDB y PostgreSQL aunque realmente con las dependencias que se han instalado antes con los repositorios se podría hacer pero por comodidad en mi caso lo voy a instalar todo en un entorno virtual.

```
jesusfigueroa ASIR ~/python python3 -m venv proyecto

jesusfigueroa ASIR ~/python source proyecto/bin/activate

jesusfigueroa ASIR ~/python "proyecto" pip install mysqlclient psycopg2
Collecting mysqlclient
  Using cached mysqlclient-2.2.7-cp313-cp313-linux_x86_64.whl
Collecting psycopg2
  Using cached psycopg2-2.9.11-cp313-cp313-linux_x86_64.whl
Installing collected packages: psycopg2, mysqlclient
Successfully installed mysqlclient-2.2.7 psycopg2-2.9.11

jesusfigueroa ASIR ~/python "proyecto"
```



Conector Python - MariaDB

Para que este programa en python nos funcione correctamente con MariaDB tendremos que tener instaladas las dependencias que he mencionado anteriormente, además de la base de datos creada con sus respectivas tablas. Una vez que tenemos la base de datos desplegada nos vamos a centrar en la función de conexión. Tenemos que asignar a una variable el modulo connect de MySQLdb y a esta función del módulo MySQLdb le vamos a pasar los parámetros de conexión, el host en mi caso es mi máquina pero si fuese un servidor remoto pues la dirección IP de ese servidor, el usuario de la base de datos con su respectiva contraseña y la base de datos, si todo esto lo tenemos bien cuando ejecutemos el programa nos vamos a poder conectar a MariaDB.

```
def conectar_mysql():  
    try:  
        db = MySQLdb.connect(host="localhost", user="hotel", passwd="passwd", db="gestionhotel")  
        print(">> Conectado a MySQL/MariaDB correctamente.")  
        return db  
    except MySQLdb.Error as error:  
        print("Error al conectar a MySQL:", error)
```

Conector Python - PostgreSQL

Para que este programa en python nos funcione correctamente con el sistema gestor de base de datos PostgreSQL tendremos que tener las dependencias que he mencionado anteriormente, además de la base de datos creada con sus respectivas tablas. Para este sistema gestor de base de datos es similar a MariaDB. Para la conectividad vamos a usar el módulo psycopg2 y el método connect al cuál le vamos a pasar los parámetros de conexión, host en mi caso mi máquina pero si fuese un servidor la dirección IP, la base de datos, el usuario y la contraseña de dicho usuario.

Si la base de datos la hemos desplegado bien y hemos asignado bien los parámetros a la función de conexión nos podremos conectar a la base de datos.

```
def conectar_postgresql():  
    try:  
        db = psycopg2.connect(host="localhost", database="gestionhotel", user="hotel", password="password")  
        print(">> Conectado a PostgreSQL correctamente.")  
        return db  
    except psycopg2.Error as error:  
        print("Error al conectar a PostgreSQL:", error)
```

Conector Python - Oracle

Una vez que tengamos las dependencias necesarias instaladas y el entorno virtual creado vamos a instalar el conector de Oracle. He probado varios conectores de Oracle pero el único que me ha funcionado correctamente es este que lo he conseguido de la documentación oficial.

IMPORTANTE: la ruta en la que creamos el entorno virtual no puede tener espacios ya que si tiene espacios nos va a dar error y no vamos a poder activarlo.

```
jesusfigueroa ASIR ~/Descargas/conector "proyecto" python3 -m pip install oracledb --upgrade
Collecting oracledb
  Downloading oracledb-3.4.2-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl.metadata (7.7 kB)
Collecting cryptography>=3.2.1 (from oracledb)
  Downloading cryptography-46.0.4-cp311-abi3-manylinux_2_34_x86_64.whl.metadata (5.7 kB)
Collecting typing_extensions>=4.14.0 (from oracledb)
  Using cached typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Collecting cffi>=2.0.0 (from cryptography>=3.2.1->oracledb)
  Using cached cffi-2.0.0-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.whl.metadata (2.6 kB)
Collecting pycparser (from cffi>=2.0.0->cryptography>=3.2.1->oracledb)
  Downloading pycparser-3.0-py3-none-any.whl.metadata (8.2 kB)
Downloading oracledb-3.4.2-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.manylinux_2_28_x86_64.whl (2.4 MB)
2.4/2.4 MB 44.8 MB/s eta 0:00:00
Downloading cryptography-46.0.4-cp311-abi3-manylinux_2_34_x86_64.whl (4.5 MB)
4.5/4.5 MB 63.9 MB/s eta 0:00:00
Using cached cffi-2.0.0-cp313-cp313-manylinux2014_x86_64.manylinux_2_17_x86_64.whl (219 kB)
Using cached typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Downloading pycparser-3.0-py3-none-any.whl (48 kB)
Installing collected packages: typing_extensions, pycparser, cffi, cryptography, oracledb
Successfully installed cffi-2.0.0 cryptography-46.0.4 oracledb-3.4.2 pycparser-3.0 typing_extensions-4.15.0

jesusfigueroa ASIR ~/Descargas/conector "proyecto" █
```

Una vez que tenemos las dependencias necesarias y el entorno virtual creado, vamos a irnos a la máquina que tenemos instalado Oracle y vamos a poner el siguiente comando para ver si está en escucha y si lo está vamos a ver el puerto que está usando ya que esto va a ser importante para poner en el fichero de funciones para la conexión. Normalmente Oracle usa el puerto 1521 pero para asegurarnos ponemos este comando y lo miramos manualmente.

```
usuario@debian:~$ lsnrctl status

LSNRCTL for Linux: Version 21.0.0.0.0 - Production on 31-ENE-2026 12:13:38

Copyright (c) 1991, 2021, Oracle. All rights reserved.

Conectándose a (ADDRESS=(PROTOCOL=tcp)(HOST=)(PORT=1521))
ESTADO del LISTENER
-----
Alias                LISTENER
Versión              TNSLSNR for Linux: Version 21.0.0.0.0 - Production
Fecha de Inicio      31-ENE-2026 11:21:01
Tiempo Actividad     0 días 0 hr. 52 min. 37 seg.
Nivel de Rastreo     off
Seguridad            ON: Local OS Authentication
SNMP                 OFF
Parámetros del Listener /opt/oracle/homes/OraDBHome21cXE/network/admin/listener.ora
Log del Listener     /opt/oracle/diag/tnslsnr/debian/listener/alert/log.xml
Recibiendo Resumen de Puntos Finales...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=debian)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=EXTPROC1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=127.0.0.1)(PORT=5500))(Security=(my_wallet_directory=/opt/oracle/homes/OraDBHome21cXE/admin/XE/xdw_wall
et))(Presentation=HTTP)(Session=RAW))
Resumen de Servicios...
El servicio "47311402cc3e129ee0630101007f72f4" tiene 1 instancia(s).
  La instancia "XE", con estado READY, tiene 1 manejador(es) para este servicio...
El servicio "XE" tiene 1 instancia(s).
  La instancia "XE", con estado READY, tiene 1 manejador(es) para este servicio...
El servicio "XEXDB" tiene 1 instancia(s).
  La instancia "XE", con estado READY, tiene 1 manejador(es) para este servicio...
El servicio "xepdb1" tiene 1 instancia(s).
  La instancia "XE", con estado READY, tiene 1 manejador(es) para este servicio...
El comando ha terminado correctamente
usuario@debian:~$ █
```



IMPORTANTE: una vez que hemos instalado las dependencias en el entorno virtual para que el editor de código te interprete la librería hay que reiniciarlo normalmente. Dependiendo del editor de código que uses en mi caso he usado Thony y Visual Studio Code y en ambos tuve que reiniciarlo.

Aquí vemos la parte importante de la conexión con Oracle, en la variable dsn_str (Data source name) que es como la he sacado de la documentación oficial, tenemos que indicarle la dirección IP, el puerto y la versión de Oracle que está usando la máquina que lo tiene instalado. Si Oracle lo tienes en tu máquina física puedes poner localhost pero en mi caso lo tengo en una máquina virtual en Virt-Maganer por lo tanto indicamos la dirección IP y el puerto seguido de la versión. Una vez que tenemos eso pasamos a el usuario, que es el que tengamos creado con su contraseña y para la variable dsn (Data source name) indicamos la variable de arriba que la hemos definido previamente con los valores de conexión.

IMPORTANTE: Una vez que ejecutamos el programa para hacer operaciones debemos estar dentro del entorno virtual.

```
def conectar_oracle():  
    try:  
        dsn_str = "192.168.122.167:1521/xe"  
  
        db = oracledb.connect(  
            user="hotel",  
            password="passwd",  
            dsn=dsn_str  
        )  
        print(">> Conectado a Oracle correctamente.")
```

Si todo esto funciona correctamente cuando elegimos la opción de Oracle ya estaremos conectado correctamente y podremos ejecutar las operaciones.

```
1. MySQL / MariaDB  
2. PostgreSQL  
3. Oracle  
4. Salir
```

```
Introduce una opción: 3  
>> Conectado a Oracle correctamente.
```



Funcionamiento del programa

MariaDB

Opción 1

Con esta opción vamos a listar los tipos de habitación con una consulta sencilla (función de agregación).

```
Introduce una opción: 1
--- Resumen de Habitaciones ---
Tipo: Doble | Cantidad: 17
Tipo: Individual | Cantidad: 16
Tipo: Suite | Cantidad: 17
```

Opción 2

Esta opción del menú podemos filtrar por precio de habitaciones entre un precio mínimo y un máximo.

```
Introduce una opción: 2
Precio mínimo: 5
Precio máximo: 70
--- Habitaciones Disponibles ---
Habitación: 101 | Tipo: Individual | Precio: 55.00
Habitación: 104 | Tipo: Individual | Precio: 55.00
Habitación: 201 | Tipo: Individual | Precio: 55.00
Habitación: 204 | Tipo: Individual | Precio: 55.00
Habitación: 301 | Tipo: Individual | Precio: 60.00
Habitación: 304 | Tipo: Individual | Precio: 60.00
Habitación: 401 | Tipo: Individual | Precio: 65.00
Habitación: 404 | Tipo: Individual | Precio: 65.00
Habitación: 501 | Tipo: Individual | Precio: 70.00
Habitación: 504 | Tipo: Individual | Precio: 70.00
```

Opción 3

Con esta opción vamos a buscar las reservas de un cliente por DNI.

```
Introduce una opción: 3
Introduce el DNI del cliente: 58596061J
--- Reservas encontradas ---
Nombre: Paula Moreno | Fecha entrada: 2024-04-03 | Habitación: 604
```



Opción 4

Esta opción nos va a permitir dar de alta a clientes en nuestra base de datos, nos pedirá los campos de la tabla clientes y lo iremos rellenando.

```
Introduce una opción: 4
--- Registro de Nuevo Cliente ---
ID Cliente: 58
Nombre: Jesús
Email: jesusfigueroa@iesgn
Teléfono: 690122021
DNI: 12334458P
>> Cliente registrado correctamente.
```

Una vez que lo añadimos si nos vamos al sistema gestor de base de datos que estabamos usando en el programa de python podemos observar que se ha añadido correctamente.

45		María Fernanda		maria.fernanda@mail.com		610555666		02030405U		2024-02-14
46		Raúl Pérez		raul.perez@mail.com		610777888		06070809V		2024-02-15
47		Alicia Campos		alicia.campos@mail.com		611111222		10111213W		2024-02-16
48		Antonio Moreno		antonio.moreno@mail.com		611333444		14151617X		2024-02-17
49		Cristina Navarro		cristina.navarro@mail.com		611555666		18192022Y		2024-02-18
50		David Herrera		david.herrera@mail.com		611777888		22232426Z		2024-02-19
58		Jesús		jesusfigueroa@iesgn		690122021		12334458P		2026-02-03

Para la opción 5 vamos a borrar por id de reserva, antes de borrar nada vamos a mirar los datos de la tabla de nuestro SGBD correspondiente.

```
MariaDB [gestionhotel]> select * from servicios_reserva;
+-----+-----+-----+-----+-----+
| id_servicio | id_reserva | servicio | precio | fecha_servicio |
+-----+-----+-----+-----+-----+
|          2 |          2 | Desayuno |   10.00 | 2024-03-03      |
|          3 |          3 | Cena     |   15.00 | 2024-03-04      |
```

Opción 5

Nos vamos al programa en python y borramos por ejemplo el id de reserva 2 y ya estaría borrada de nuestra tabla del SGBD correspondiente.

```
Introduce una opción: 5
Introduce ID de reserva para borrar sus servicios: 2
>> Servicios eliminados. Filas afectadas: 1
```



Para la opción 6 del menú vamos a aumentar el precio de una noche en tanto por ciento para ello antes de actualizar nada vamos a mirar que precio tienen.

```
MariaDB [gestionhotel]> select * from habitaciones where piso = 1;
+-----+-----+-----+-----+-----+-----+-----+
| id_habitacion | numero | tipo      | piso | capacidad | precio_noche | disponible |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 101 | Individual | 1 | 1 | 55.00 | S |
| 2 | 102 | Doble     | 1 | 2 | 77.00 | S |
| 3 | 103 | Suite     | 1 | 3 | 132.00 | S |
| 4 | 104 | Individual | 1 | 1 | 55.00 | S |
| 5 | 105 | Doble     | 1 | 2 | 77.00 | S |
| 6 | 106 | Suite     | 1 | 3 | 132.00 | S |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0,000 sec)
```

Opción 6

Nos vamos a nuestro programa en python y vamos a aumentar un 20% por ejemplo, nos dice que se han actualizado 6 filas.

```
Introduce una opción: 6
Número de piso a actualizar: 1
Porcentaje de incremento (ej: 10): 20
>> Precios actualizados. Filas actualizadas: 6
```

Volvemos a MariaDB y como vemos se ha actualizado correctamente.

```
MariaDB [gestionhotel]> select * from habitaciones where piso = 1;
+-----+-----+-----+-----+-----+-----+-----+
| id_habitacion | numero | tipo      | piso | capacidad | precio_noche | disponible |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | 101 | Individual | 1 | 1 | 66.00 | S |
| 2 | 102 | Doble     | 1 | 2 | 92.40 | S |
| 3 | 103 | Suite     | 1 | 3 | 158.40 | S |
| 4 | 104 | Individual | 1 | 1 | 66.00 | S |
| 5 | 105 | Doble     | 1 | 2 | 92.40 | S |
| 6 | 106 | Suite     | 1 | 3 | 158.40 | S |
+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0,000 sec)
```



PostgreSQL

Primeramente tendremos que montar nuestra base de datos con el esquema correctamente y si todo funciona bien estaremos conectados.

```
Introduce una opción: 2
>> Conectado a PostgreSQL correctamente.
```

Opción 1

Es lo mismo que en MariaDB lista por tipo de habitaciones.

```
Introduce una opción: 1
--- Resumen de Habitaciones ---
Tipo: Suite | Cantidad: 17
Tipo: Individual | Cantidad: 16
Tipo: Doble | Cantidad: 17
```

Opción 2

Buscamos pisos en un rango entre precio mínimo y precio máximo y nos lo muestra.

```
Introduce una opción: 2
Precio mínimo: 10
Precio máximo: 50
--- Habitaciones Disponibles ---
Habitación: 101 | Tipo: Individual | Precio: 50.00
Habitación: 104 | Tipo: Individual | Precio: 50.00
```

Opción 3

Buscamos las reservas de un cliente por DNI.

```
Introduce una opción: 3
Introduce el DNI del cliente: 34567890C
--- Reservas encontradas ---
Nombre: María Pérez | Fecha entrada: 2024-03-03 | Habitación: 103
```

Opción 4

Podemos dar de alta a clientes con todos los campos de la tabla clientes.

```
Introduce una opción: 4
--- Registro de Nuevo Cliente ---
ID Cliente: 57
Nombre: Jesús Figueroa
Email: jesusfigueroa@iesgn
Teléfono: 671915815
DNI: 22796049P
>> Cliente registrado correctamente.
```

Si nos vamos a la tabla clientes vemos que se ha añadido el cliente correctamente.



49	Cristina Navarro	cristina.navarro@mail.com	611555666	18192022Y	2024-02-18
50	David Herrera	david.herrera@mail.com	611777888	22232426Z	2024-02-19
57	Jesús Figueroa	jesusfigueroa@iesgn	671915815	22796049P	2026-02-03

Para la opción 5 vamos a borrar servicios de reserva por id de reserva.

```
gestionhotel⇒ select * from servicios_reserva where id_reserva = 1;
id_servicio | id_reserva | servicio | precio | fecha_servicio
-----+-----+-----+-----+-----
1 | 1 | Desayuno | 10.00 | 2024-03-02
(1 fila)
```

Indicamos la opción 5 en el menú e indicamos el ID de reserva en este caso 1 por ejemplo y nos dice 1 filas afectadas.

```
Introduce una opción: 5
Introduce ID de reserva para borrar sus servicios: 1
>> Servicios eliminados. Filas afectadas: 1
```

Si nos vamos a nuestra base de datos vemos que nos lo ha borrado correctamente.

```
gestionhotel⇒ select * from servicios_reserva where id_reserva = 1;
id_servicio | id_reserva | servicio | precio | fecha_servicio
-----+-----+-----+-----+-----
(0 filas)
```

Para la opción 6 vamos a actualizar el precio en porcentaje para ello vamos a ver como esta el piso 1 antes de ejecutar esta opción.

```
gestionhotel⇒ select * from habitaciones where piso = 1;
```

id_habitacion	numero	tipo	piso	capacidad	precio_noche	disponible
1	101	Individual	1	1	50.00	S
2	102	Doble	1	2	70.00	S
3	103	Suite	1	3	120.00	S
4	104	Individual	1	1	50.00	S
5	105	Doble	1	2	70.00	S
6	106	Suite	1	3	120.00	S

(6 filas)

Ponemos un 15% por ejemplo de incremento y nos lo ha actualizado correctamente.

```
Introduce una opción: 6
Número de piso a actualizar: 1
Porcentaje de incremento (ej: 10): 15
>> Precios actualizados. Filas actualizadas: 6
```

Si nos vamos a la base de datos vemos que nos lo ha subido en un 15% el piso 1.



```
gestionhotel⇒ select * from habitaciones where piso = 1;
id_habitacion | numero | tipo | piso | capacidad | precio_noche | disponible
-----+-----+-----+-----+-----+-----+-----
1 | 101 | Individual | 1 | 1 | 57.50 | S
2 | 102 | Doble | 1 | 2 | 80.50 | S
3 | 103 | Suite | 1 | 3 | 138.00 | S
4 | 104 | Individual | 1 | 1 | 57.50 | S
5 | 105 | Doble | 1 | 2 | 80.50 | S
6 | 106 | Suite | 1 | 3 | 138.00 | S
(6 filas)
```

Oracle

Para Oracle tendremos que tenerlo instalado ya sea en local, en una máquina virtual o en un servidor remoto como puede ser un VPS.

```
Introduce una opción: 3
>> Conectado a Oracle correctamente.
```

Para la opción 1 tenemos lo mismo que es los 2 sistemas, nos filtra por tipo y nos muestra cuanto hay de cada tipo.

```
Introduce una opción: 1
--- Resumen de Habitaciones ---
Tipo: Individual | Cantidad: 16
Tipo: Doble | Cantidad: 17
Tipo: Suite | Cantidad: 17
```

Para la opción 2 buscamos habitaciones en un rango, si no hay ninguno no nos muestra ninguna.

```
Introduce una opción: 2
Precio mínimo: 10
Precio máximo: 40
--- Habitaciones Disponibles ---
No se encontraron registros.
```

Para la opción 3 vamos a poder buscar las reservas filtrando por el DNI del cliente.

```
Introduce una opción: 3
Introduce el DNI del cliente: 1011213W
--- Reservas encontradas ---
Nombre: Alicia Campos | Fecha entrada: 2024-04-16 00:00:00 | Habitación: 805
```

Para la opción 4 vamos a dar de alta a un nuevo cliente, una vez que lo hayamos añadido vamos a ver que se haya añadido correctamente a nuestro sistema gestor de bases de datos.



```
Introduce una opción: 4
--- Registro de Nuevo Cliente ---
ID Cliente: 51
Nombre: Jesús Figueroa Roldán
Email: jesusfigueroa@iesgn
Teléfono: 69118501
DNI: 12671919P
>> Cliente registrado correctamente.
```

Y como vemos ahí tenemos la entrada que hemos añadido.

```
SQL> select * from clientes;
```

ID_CLIENTE	NOMBRE	EMAIL	TELEFONO	DNI	FECHA_RE
51	Jesús Figueroa Roldán	jesusfigueroa@iesgn	69118501	12671919P	03/02/26

Ahora para la opción 5 vamos a poder eliminar entradas de la tabla servicios_reserva por id reserva.

```
SQL> select * from servicios_reserva;
```

ID_SERVICIO	ID_RESERVA	SERVICIO	PRECIO	FECHA_SE
1	1	Desayuno	10	02/03/24
2	2	Desayuno	10	03/03/24
3	3	Cena	15	04/03/24

Como vemos nos pedirá un id de reserva y lo borramos de nuestra tabla.

```
Introduce una opción: 5
Introduce ID de reserva para borrar sus servicios: 1
>> Servicios eliminados. Filas afectadas: 1
```

Si nos vamos a Oracle vemos como lo ha borrado correctamente y que ya no tenemos esa entrada en la tabla servicios_reserva.

```
SQL> select * from servicios_reserva where id_reserva = 1;

ninguna fila seleccionada
```

Para la opción 6 vamos a poder incrementar el precio por piso en un tanto por ciento que le indiquemos al programa. Esto sería una muestra antes de ejecutar esta opción.



```
SQL> select * from habitaciones where piso = 1;
```

ID_HABITACION	NUMERO	TIPO	PISO	CAPACIDAD	PRECIO_NOCHE	D
1	101	Individual	1	1	50	S
2	102	Doble	1	2	70	S
3	103	Suite	1	3	120	S
4	104	Individual	1	1	50	S
5	105	Doble	1	2	70	S
6	106	Suite	1	3	120	S

6 filas seleccionadas.

Nos vamos a nuestro programa vamos a elegir por ejemplo el piso 1 y vamos a incrementar un 16%.

```
Introduce una opción: 6
Número de piso a actualizar: 1
Porcentaje de incremento (ej: 10): 16
>> Precios actualizados. Filas actualizadas: 6
```

Cuando nos vamos a Oracle y vamos los precios vemos como ha incrementado un 16%.

ID_HABITACION	NUMERO	TIPO	PISO	CAPACIDAD	PRECIO_NOCHE	D
1	101	Individual	1	1	58	S
2	102	Doble	1	2	81,2	S
3	103	Suite	1	3	139,2	S
4	104	Individual	1	1	58	S
5	105	Doble	1	2	81,2	S
6	106	Suite	1	3	139,2	S

6 filas seleccionadas.

Bibliografía

https://python-oracledb.readthedocs.io/en/latest/user_guide/installation.html

<https://www.josedomingo.org/pledin/2021/12/python3-mysql/>

<https://www.psycopg.org/docs/>

<https://mariadb.com/resources/blog/how-to-connect-python-programs-to-mariadb/>