Gestión de tareas

El objetivo es desarrollar una aplicación de **gestión de tareas** en línea de comandos usando el paradigma de *Programación Orientada a Objetos* con Python junto al módulo sqlite para el *acceso a datos*.

1. Funciones globales

Implementa, *al menos*, las siguientes **funciones** globales:

```
def create_db(db_path: str) -> None:
```

- Crea la base de datos en la ruta `db_path`
- Crea la tabla `tasks`:

`id`	Clave primaria (identificador numérico)
`name`	Nombre/Descripción de la tarea
`done`	Indica si la tarea está hecha o no

2. Clase `Task`

Escribe una clase `Task` que represente una tarea.

2.1. Atributos de clase

Crea, *al menos*, los siguientes **atributos de clase**:

`con`

- Conexión a la base de datos `DB_PATH`.
- *Resultados de consulta* en **modo fila**.

`cur`

- Cursor creado desde la conexión a la base de datos.

2.2. Métodos

Implementa, *al menos*, los siguientes **métodos**:

`def __init__(self, name: str, done: bool = False, id: int = -1):`

- Crea los atributos *homónimos* a los parámetros.

`def save(self) -> None:`

- Inserta la tarea `self` en la base de datos.
- Actualiza el atributo `id` de `self` desde el asignado en la base de datos.

`def update(self) -> None:`

- Actualiza la tarea `self` en la base de datos (*nombre* y *estado*).

`def check(self) -> None:`

- Marca la tarea como completada.
- Haz uso también de `update()`

`def uncheck(self) -> None:`

- Marca la tarea como no completada.
- Haz uso también de `update()`

`def __repr__(self):`

- Devuelve formato de tarea.
- Si la tarea está completada: `[X] <name> (id=<id>)`
- Si la tarea no está completada: `[ ] <name> (id=<id>)`

`def from_db_row(cls, row: sqlite3.Row) -> Task:`

- Es un **método de clase**.
- Construye (y devuelve) una nueva tarea.
- Los datos de la tarea están en `row` (*fila de consulta*).

`def get(cls, task_id: int) -> Task:`

- Es un **método de clase**.
- Construye (y devuelve) una nueva tarea.
- Los datos de la tarea se obtienen consultando la base de datos desde `task_id`.
- Haz uso también de `from_db_row()`

3. Clase `ToDo`

Escribe una clase `ToDo` que represente el "controlador" de la gestión de tareas.

3.1. Atributos de clase

Crea, *al menos*, los siguientes **atributos de clase**:

`con`

- Conexión a la base de datos `DB_PATH`.
- *Resultados de consulta* en **modo fila**.

`cur`

- Cursor creado desde la conexión a la base de datos.

3.2. Métodos

Implementa, *al menos*, los siguientes **métodos**:

```
def get_tasks(self, done: int = -1):
```

- Es una **función generadora**.
- Devuelve tareas (desde la base de datos) como **objetos de tipo** `Task`:

$$done = \begin{cases} -1 & \text{Devuelve } \textbf{todas las tareas} \\ 0 & \text{Devuelve } \textbf{las tareas pendientes} \\ 1 & \text{Devuelve } \textbf{las tareas completadas} \end{cases}$$

```
def add_task(self, name: str) -> None:
```

- Añade a la base de datos una tarea con nombre `name`.
- Haz uso de la clase `Task` y *sus métodos*.

```
def complete_task(self, task_id: int) -> None:
```

- Marca la tarea con identificador `task_id` como **completada**.
- Haz uso de la clase `Task` y *sus métodos*.

```
def reopen_task(self, task_id: int) -> None:
```

- Marca la tarea con identificador `task_id` como **pendiente**.
- Haz uso de la clase `Task` y *sus métodos*.