

## **Defensa proyecta final Programacion**

El programa desarrollado es una aplicación de línea de comandos en Python llamada Organizador cuyo objetivo es permitir la gestión de tareas mediante operaciones básicas: crear, listar, marcar como completadas y eliminar tareas. La aplicación utiliza persistencia de datos con SQLite y fue diseñada aplicando los conceptos fundamentales vistos durante el año.

### **Implementacion de metodos y funciones:**

El diseño del programa se fundamenta en métodos y funciones que tienen tareas claramente definidas.

- En tareas.py, la clase Tarea reúne la información de una tarea y su funcionamiento a través de métodos como mark\_completed() y mark\_undone(), que se alinea con el método de programación orientada a objetos.
- En storage.py, los métodos add\_tarea(), list\_tareas(), get\_tarea(), update\_tarea() y delete\_tarea() llevan a cabo las funciones CRUD sobre la base de datos.
- En cli.py, cada instrucción (add, list, done, delete) se desarrolla como una función separada, lo que facilita su comprensión y mantenimiento.

Esta división previene que las funciones sean largas y complicadas, y permite que cada sección del sistema asuma una única tarea.

### **Uso de docstrings y comentarios**

Todo el código está documentado mediante docstrings, conforme al estándar PEP 257.

Los docstrings clarifican:

- La finalidad de cada módulo.
- La tarea de las clases.
- El funcionamiento de métodos y funciones, que incluye parámetros y valores de retorno cuando es necesario.

Esto facilita que cualquier persona entienda de manera rápida cómo opera el código sin tener que estudiarlo detenidamente.

### **Buenas prácticas: código limpio, modular y legible**

Con el fin de garantizar que el código sea claro y fácil de leer, se implementaron las siguientes buenas prácticas:

- Utilizar nombres que describan claramente variables, funciones y métodos.
- Crear funciones que sean breves y con un solo propósito.
- Mantener una sangría e indentación uniforme.
- Prevenir la repetición de código.

- Establecer una distinción clara entre la lógica del negocio, la persistencia y la presentación.

Estas elecciones ayudan a que el código sea más fácil de leer y a que su mantenimiento sea más sencillo en el futuro.

### **Organización del proyecto y modulación**

El proyecto se estructuró en módulos separados dentro del paquete Organizador

- cli.py: interfaz para la línea de comandos.
- tareas.py: definición del modelo de información.
- storage.py: almacenamiento en una base de datos SQLite.
- utils.py: funciones auxiliares que se pueden reutilizar.

Esta estructura sigue el principio de dividir responsabilidades, aumenta la capacidad de escalar el proyecto y hace más sencillo escribir pruebas automáticas.

### **Aplicación del Zen de Python y PEP 8**

En el proceso de desarrollo, se destacó el principio del Zen de Python:

**“Simple is better than complex.”**

Este principio se refleja en:

- La utilización de SQLite como una solución básica y adecuada.
- La selección de una interfaz de línea de comandos en lugar de una opción más compleja.
- El estructurado de funciones que son claras y fáciles de comprender.

Además, el código respeta las normas de estilo de la PEP 8, observando:

- Longitudes de línea apropiadas.
- Normas de nomenclatura.
- Espacio y sangrías consistentes.

### **Uso del entorno virtual y desarrollo colaborativo**

Se empleó un entorno virtual (venv) para separar las dependencias del proyecto

Esto permite:

- Prevenir problemas con otras versiones de bibliotecas.
- Asegurar que el proyecto opere de la misma manera en diferentes entornos.
- Hacer más sencillo el trabajo en equipo, puesto que cualquier persona puede reproducir el entorno instalando las dependencias desde requirements.txt.

Para resumir, el Organizador combina los principios esenciales aprendidos en la materia y muestra la habilidad de crear un proyecto Python que funcione correctamente, esté bien estructurado y documentado, usando prácticas adecuadas, el Zen de Python y las normas de estilo sugeridas.