# Ejercicio Práctico: Gestor de Tareas Pendientes (To-Do List)

## Objetivo

Crear un sistema de gestión de tareas pendientes que utilice archivos para persistencia de datos, implementando programación orientada a objetos y estructuras de control.

## Estructura Requerida

### Clase `Tarea`

•**Atributos**:

•`descripcion` (str): Descripción de la tarea
•`completada` (bool): Estado de la tarea (False por defecto)
•`fecha_creacion` (datetime): Fecha y hora de creación
•**Métodos**:

•`__init__(self, descripcion)`: Constructor
•`marcar_completada(self)`: Cambia el estado a completada
•`__str__(self)`: Retorna representación en string (ej: `"[ ] Comprar pan"` o `"[x] Hacer ejercicio"`)

### Clase `GestorTareas`

•**Atributos**:

•`archivo` (str): Nombre del archivo de almacenamiento (ej: `"tareas.txt"`)
• `tareas` (list): Lista de objetos `Tarea`
•**Métodos**:

- `__init__(self, nombre_archivo)`: Constructor
- `cargar_tareas(self)`: Lee el archivo y carga las tareas en memoria
- `guardar_tareas(self)`: Guarda todas las tareas en el archivo

- `agregar_tarea(self, descripcion)`: Crea y agrega una nueva tarea
- `marcar_completada(self, indice)`: Marca una tarea como completada
- `mostrar_tareas(self)`: Muestra todas las tareas numeradas
- `eliminar_tarea(self, indice)`: Elimina una tarea específica

## Archivo de Almacenamiento

Formato para cada línea del archivo `tareas.txt`:

```
[estado]|descripcion|fecha_creacion
```

Ejemplo:

```
[x]|Comprar leche|2023-10-25 10:30:15
[ ]|Estudiar Python|2023-10-25 09:15:22
```

## Interfaz de Usuario

El programa principal debe mostrar un menú con las siguientes opciones (usando un ciclo `while`):

```
=== GESTOR DE TAREAS ===
1. Ver todas las tareas
2. Agregar nueva tarea
3. Marcar tarea como completada
4. Eliminar tarea
5. Salir
Seleccione una opción:
```