CURSO PYTHON FULL STACK IMPARTIDO POR BEJOB
Alumna: Erika Samara Alvares Angelim
Este código encuentra-se en Github:
https://github.com/ea-analisisdatos/Python-Full-Stack/tree/main/caso-practico-final

Acerca de esta actividad formativa

Caso práctico final donde el alumno deberá de entregar la evidencia de que ha realizado la actividad.

Enunciado:

Escribe un programa en Python utilizando Programación Orientada a Objetos que gestione una lista de tareas pendientes. El programa deberá permitir al usuario realizar las siguientes operaciones:
• Agregar una nueva tarea: El programa deberá permitir al usuario agregar una nueva tarea a la lista de tareas pendientes.
• Marcar una tarea como completada: El programa deberá permitir al usuario marcar una tarea como completada, dada su posición en la lista.
• Mostrar todas las tareas: El programa deberá imprimir en pantalla todas las tareas pendientes, numeradas y mostrando su estado (completada o pendiente).
• Eliminar una tarea: El programa deberá permitir al usuario eliminar una tarea de la lista, dada su posición.

El programa deberá incluir las siguientes características:
• Manejo de excepciones: El programa deberá manejar excepciones en caso de que el usuario ingrese una opción no válida o una posición que no exista en la lista.
• Comentarios explicativos: El código deberá estar comentado para explicar su funcionamiento en cada parte relevante

CÓDIGO DESARROLLADO EN PYTHON - Archivo: gestor-de-tareas.py
Para ejecutarlo, copie y pegue el código abajo en el archivo de Python "gestor-de-tareas.py" y luego, ejecutarlo:

```
import sys
# Reconfiguración de la salida estándar para usar la codificación UTF-8,
# lo cual asegura que los caracteres acentuados se impriman correctamente.
sys.stdout.reconfigure(encoding='utf-8')

class Tarea:
    def __init__(self, descripcion, completada=False):
        # Constructor de la clase Tarea, inicializa una tarea con una descripción y estado de completado (opcional).
        self.descripcion = descripcion
        self.completada = completada
```

```
class GestorTareas:
    def __init__(self):
        # Constructor de la clase GestorTareas, inicializa una lista vacía de
tareas.
        self.tareas = []

    def agregar_tarea(self, descripcion):
        # Método para agregar una nueva tarea a la lista de tareas.
        tarea = Tarea(descripcion)
        self.tareas.append(tarea)

    def marcar_completada(self, posicion):
        # Método para marcar una tarea como completada dado su posición en la
lista.
        if 0 <= posicion < len(self.tareas):
            self.tareas[posicion].completada = True
        else:
            raise IndexError("La posición ingresada no es válida.")

    def mostrar_tareas(self):
        # Método para imprimir todas las tareas pendientes en la lista.
        for i, tarea in enumerate(self.tareas):
            estado = "Completada" if tarea.completada else "Pendiente"
            print(f"{i + 1}. [{estado}] {tarea.descripcion}")

    def eliminar_tarea(self, posicion):
        # Método para eliminar una tarea de la lista, solicitando confirmación al
usuario.
        if 0 <= posicion < len(self.tareas):
            while True:
                confirmacion = input(f"¿Está seguro que desea eliminar la tarea
'{self.tareas[posicion].descripcion}'? (s/n): ").lower()
                if confirmacion == 's':
                    del self.tareas[posicion]
                    break
                elif confirmacion == 'n':
                    print("Acción cancelada.")
                    break
                else:
                    print("Opción no válida. Por favor, ingrese 's' para eliminar o
'n' para cancelar.")
        else:
            raise IndexError("La posición ingresada no es válida.")

if __name__ == "__main__":
    gestor = GestorTareas()

    while True:
        print("\n--- Gestor de Tareas ---")
        # Menú de opciones para que el usuario interactúe con el gestor de tareas.
```

```
        print("1. Agregar nueva tarea")
        print("2. Marcar tarea como completada")
        print("3. Mostrar todas las tareas")
        print("4. Eliminar tarea")
        print("5. Salir")

        opcion = input("Ingrese el número de la opción deseada: ")

        if opcion == "1":
            descripcion = input("Ingrese la descripción de la nueva tarea: ")
            gestor.agregar_tarea(descripcion)
        elif opcion == "2":
            try:
                posicion = int(input("Ingrese la posición de la tarea que desea
marcar como completada: ")) - 1
                gestor.marcar_completada(posicion)
            except (ValueError, IndexError) as e:
                print("Error:", e)
        elif opcion == "3":
            gestor.mostrar_tareas()
        elif opcion == "4":
            try:
                posicion = int(input("Ingrese la posición de la tarea que desea
eliminar: ")) - 1
                gestor.eliminar_tarea(posicion)
            except (ValueError, IndexError) as e:
                print("Error:", e)
        elif opcion == "5":
            print("¡Hasta luego!")
            break
        else:
            print("Opción no válida. Por favor, ingrese un número del 1 al 5.")
```