**Aplicación del Paradigma Funcional en la Solución Proporcionada**

**Aciertos**

**Funciones Definidas:** La solución define varias funciones para realizar tareas específicas, como mostrar el menú principal, mostrar opciones de cultivos, mostrar horario de gestión, mostrar etapas, y calcular información contable. Estas funciones encapsulan funcionalidad relacionada y pueden considerarse como aplicaciones del paradigma funcional.

**Inmutabilidad:** En general, se utilizan variables para almacenar datos que no se modifican, como los datos de cultivo y los costos/gastos mensuales. Esto es consistente con el concepto de inmutabilidad en la programación funcional.

**Falencias y Oportunidades de Mejora**

**Recursión:** Una característica importante de la programación funcional es la recursión en lugar de los bucles. En esta solución, se utilizan bucles for para iterar sobre los meses y realizar cálculos. Podría mejorarse utilizando recursión o funciones de orden superior como map o reduce.

**Funciones de Orden Superior:** En la solución actual, no se utilizan funciones de orden superior, como map o filter, que son herramientas comunes en la programación funcional para operar en colecciones de datos. Podrían aplicarse para simplificar el código.

**Efectos Secundarios:** La solución actual usa efectos secundarios, como la impresión directa en la consola y la entrada del usuario (input). La programación funcional tiende a minimizar los efectos secundarios, favoreciendo la pureza de las funciones.

**División de Tareas en Funciones:** Aunque se definen funciones, algunas de ellas podrían dividirse aún más en funciones más pequeñas para lograr una mayor modularidad y reutilización de código.

**Sugerencias para Aplicar el Paradigma Funcional**

Para aplicar el paradigma funcional de manera más efectiva en esta solución, se podrían considerar las siguientes acciones:

**Dividir en Funciones más Pequeñas:** Descomponer las funciones existentes en funciones más pequeñas y especializadas que realicen tareas únicas. Por ejemplo, podría haber una función que calcule los costos totales, otra para calcular la ganancia, y así sucesivamente.

**Evitar Efectos Secundarios:** Reducir la cantidad de impresiones directas en la consola y la entrada del usuario en las funciones. En lugar de imprimir resultados directamente, las funciones podrían devolver datos que luego se impriman en una función principal.

**Recursión:** Reemplazar los bucles for con funciones recursivas donde sea posible, como en los cálculos mensuales. Por ejemplo, se podría calcular el costo total de los meses utilizando una función recursiva.

**Usar Funciones de Orden Superior:** Donde sea relevante, aplicar funciones de orden superior como map o reduce para operar en colecciones de datos, como la lista de gastos mensuales.

**Evitar Variables Mutables:** Evitar el uso de variables mutables cuando no sea necesario. Por ejemplo, en lugar de modificar la lista gastos\_mensuales, se podría utilizar una función recursiva para calcularla sin modificarla.

**Encapsular Datos:** Utilizar estructuras de datos inmutables para representar los cultivos y los datos contables, en lugar de diccionarios mutables.

**Manejar Errores con Excepciones:** En lugar de utilizar valores especiales para manejar errores (como el valor de -1 para meses sin gastos), considerar el uso de excepciones para manejar casos excepcionales.

**Usar Programación Funcional de Bibliotecas:** Explorar bibliotecas de programación funcional en Python, como functools, para aprovechar herramientas que facilitan la programación funcional.