¿Tu Código es un Tesoro... o un Laberinto?

La Importancia de la Refactorización en Python

El Problema: "Deuda Técnica"

Es el costo oculto del código "rápido" o "desordenado".

Hoy funciona, pero mañana te costará **horas** (o días) entenderlo, arreglarlo o añadirle nuevas funciones.

¿Qué es Refactorizar?

NO ES:

- Arreglar bugs (aunque puede ayudar).
- Añadir nuevas funciones.
- Reescribir todo desde cero (¡pánico!).

NO cambia el **comportamiento externo**.

SÍ ES:

- Mejorar la estructura interna.
- Aumentar la legibilidad (para humanos).
- Facilitar el mantenimiento futuro.

SÍ cambia la **implementación interna**.

Caso de Estudio: El "ANTES"

Este código **funciona**, pero es difícil de mantener.

```
# ¡Problema! Una variable global peliculas = []
def agregar_pelicula(nombre):
peliculas.append(nombre) print(f"'{nombre}'
agregada.") def mostrar_peliculas(): print("--- Mi
Catálogo ---") for p in peliculas: print(p) #
Lógica mezclada en el script while True: opcion =
input("1. Agregar 2. Ver ... ") if opcion == '1':
nombre = input("Nombre: ")
agregar_pelicula(nombre) elif opcion == '2':
mostrar_peliculas() ...
```

¿Por Qué es "Malo" si Funciona?

- **Variables Globales (`peliculas = []`):** ¡Peligro! Cualquier función, en cualquier parte, puede modificar esta lista. Es impredecible y una fuente común de bugs.
- **Baja Cohesión:** La lógica del menú (`while True`) está mezclada con la lógica de negocio (las funciones `agregar_...`).
- **Datos Pobres:** Es solo una lista de strings. ¿Y si queremos guardar el **Año** o el **Director**? No podemos.
- **No es Reutilizable:** ¿Y si queremos tener DOS catálogos (ej. "Terror" y "Comedia")? No podemos.

La Solución: Refactorizar a Clases

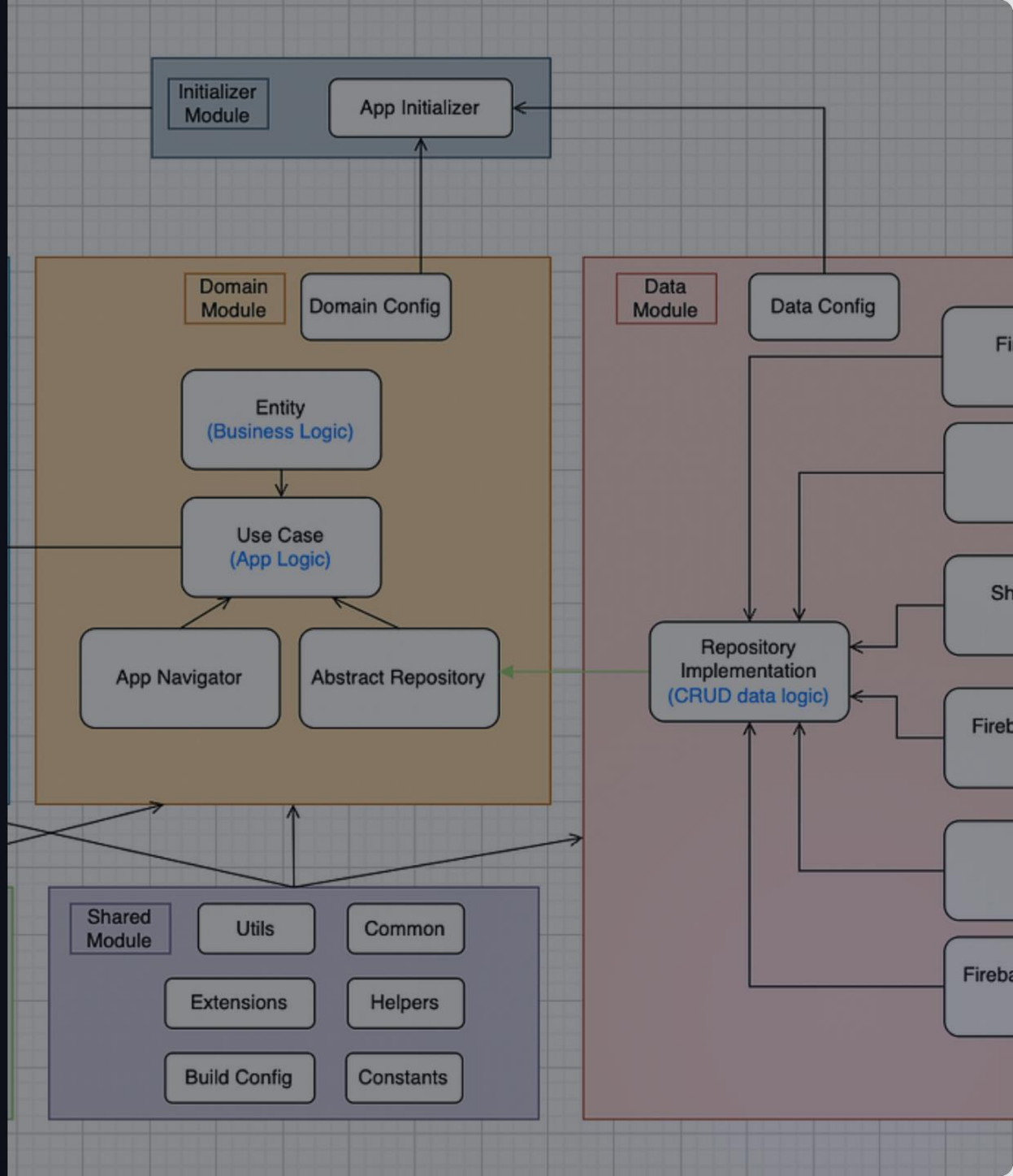
Usamos **Programación Orientada a Objetos (OOP)**.

Agrupamos los **Datos** (la lista) y las **Funciones** (agregar, mostrar) en un solo "molde": una **Clase**.

Caso de Estudio: El "DESPUÉS"

El código ahora está encapsulado, limpio y es reutilizable.

```
class CatalogoPeliculas: def __init__(self, nombre): self.nombre = nombre # ¡Encapsulado! La lista vive DENTRO self.peliculas = [] def agregar_pelicula(self, nombre): self.peliculas.append(nombre) print(f"'{nombre}' agregada a {self.nombre}") def mostrar_peliculas(self): print(f"--- {self.nombre} ---") for p in self.peliculas: print(p) # La lógica de uso está SEPARADA catalogo_estrenos = CatalogoPeliculas("Estrenos") catalogo_terror = CatalogoPeliculas("Terror") catalogo_estrenos.agregar_pelicula("Dune 2") catalogo_terror.agregar_pelicula("Hereditary")
```



¿Qué Ganamos con Esto?

Encapsulación

Los datos (`self.peliculas`) están protegidos dentro de cada "objeto". No más variables globales peligrosas.

Separación de Intereses

La **definición** de la clase (el molde) está separada de su **uso** (el menú o script principal).

Reutilización

¡Podemos crear múltiples catálogos! `catalogo_estrenos` y `catalogo_terror` son objetos separados con sus propias listas.

Un Principio Clave

> "Cualquier tonto puede escribir código que una computadora entienda. Los buenos programadores escriben código que los humanos entiendan."
>
> — Martin Fowler (Pionero de la Refactorización)

Infografía: Principios del Código Limpio

Claves para Refactorizar:

- **La Regla del Boy Scout:** Deja el código (campamento) siempre un poco más limpio de como lo encontraste.
- **DRY (Don't Repeat Yourself):** ¡No te repitas! Si escribes el mismo código dos veces, probablemente debería ser una función.
- **Separación de Intereses:** Cada clase o función debe hacer UNA sola cosa, y hacerla bien.

Refactorizar No Es Un Lujo

Es una parte **necesaria** del desarrollo de software profesional.

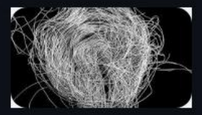
Es la diferencia entre un código que se desmorona y un código que puede crecer y adaptarse.

¿Preguntas?

El código limpio es un hábito. ¡Empiecen a practicarlo hoy!

¡A refactorizar!

Image Sources



https://images.unsplash.com/photo-1756940781530-42149559c922?fm=jpg&q=60&w=3000&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxwaG90by1yZWxhdGVkfDIyfHx8ZW58MHx8fHx8

Source: unsplash.com

https://miro.medium.com/v2/resize:fit:1400/1*C2Euk2Dc9H3Pyf7I9WY3cw.png

Source: medium.com

https://static.vecteezy.com/system/resources/previews/060/522/126/non_2x/code-edit-line-icon-clean-and-minimalist-icon-for-websites-and-mobile-apps-vector.jpg

Source: www.vecteezy.com