

Mejora tu código: Evita anti-patrones en Python

Noviembre 2024 - Alejandro López



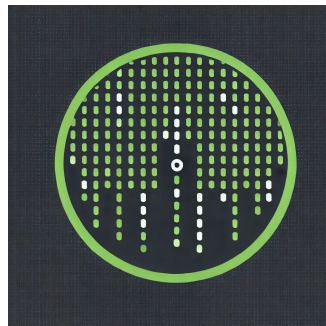
Un Poco Sobre Mí



Hobbie



Deporte



Series y Películas



Comida Favorita

¿Qué son los Anti-Patrones?

1. Los anti-patrones son soluciones comunes a problemas de desarrollo que parecen correctas, pero en realidad son ineficaces o contraproducentes.
2. En lugar de mejorar el código, estos enfoques suelen aumentar la complejidad, dificultar el mantenimiento y reducir la legibilidad.
3. Objetivo: Identificar y evitar estas malas prácticas para escribir código más limpio y efectivo.

El camino fácil no siempre es el correcto

"En Python, escribir código puede parecer sencillo, pero eso no significa que lo estemos haciendo de la mejor manera."

"Hay una y preferiblemente sólo una manera obvia de hacerlo." - Zen python

El Atajo en la Montaña

Imagina que estás en una montaña y decides tomar un atajo para llegar más rápido a la cima.

Al principio parece una buena idea, pero pronto te das cuenta de que el camino es rocoso, lleno de obstáculos y termina siendo más difícil y peligroso.

Anti-patrones = Atajos en el código que parecen rápidos, pero nos llevan a problemas complejos.

Código Pythonic = El sendero marcado: más largo al inicio, pero seguro y más fácil de mantener en el largo plazo.

Categorías



Correctness

Se refiere a evitar errores y asegurar que el código haga lo que se espera. Los anti-patrones de esta categoría suelen introducir errores inesperados o fallos en tiempo de ejecución debido a prácticas inadecuadas.



Readability

La legibilidad es fundamental para que cualquier desarrollador pueda entender y trabajar con el código. Los anti-patrones de esta categoría hacen el código difícil de leer y comprender, lo que afecta la colaboración y el mantenimiento.



Maintainability

Un código fácil de mantener es esencial para facilitar cambios y nuevas funcionalidades. Los anti-patrones de mantenibilidad complican la actualización del código, aumentando el riesgo de introducir errores.



Performance and security

Optimizar el rendimiento y asegurar la protección del código contra vulnerabilidades es clave. Los anti-patrones en esta área generan ineficiencias, problemas de seguridad y potenciales vulnerabilidades en el sistema.

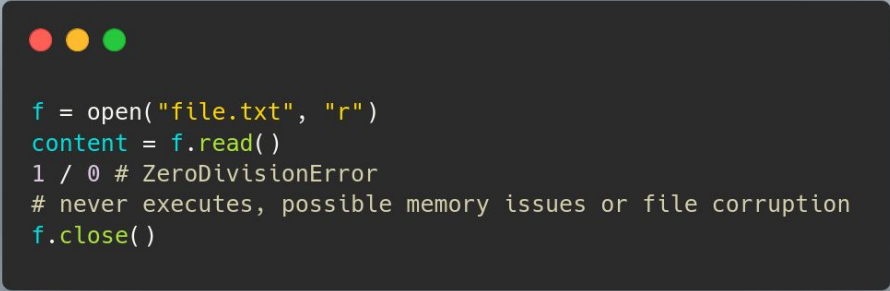
Correctness: Evita errores inesperados



```
def divide(a, b):  
    try:  
        result = a / b  
    except:  
        result = None  
    return result
```

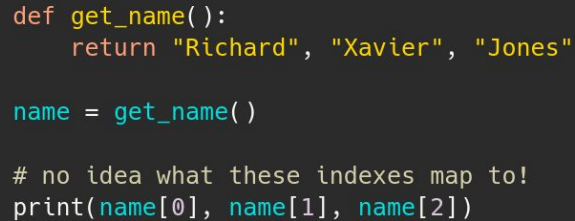
Maintainability:

Código fácil de actualizar



```
f = open("file.txt", "r")
content = f.read()
1 / 0 # ZeroDivisionError
# never executes, possible memory issues or file corruption
f.close()
```


Readability: Código que se entiende a simple vista



```
def get_name():  
    return "Richard", "Xavier", "Jones"  
  
name = get_name()  
  
# no idea what these indexes map to!  
print(name[0], name[1], name[2])
```

Performance & Security: Código eficiente y seguro

```
l = [1, 2, 3, 4]

# iterates over three elements in the list
if 3 in l:
    print("The number 3 is in the list.")
else:
    print("The number 3 is NOT in the list.")
```

Thanks!

Q&A

CREDITS: This presentation template was created by [Slidesgo](#), and includes icons by [Flaticon](#), and infographics & images by [Freepik](#)

References

- <https://docs.quantifiedcode.com/python-anti-patterns/index.html>
- <https://github.com/quantifiedcode/python-anti-patterns>