Fundamentos de Computación Científica

Lógica Algorítmica y Vectorización para IA

Semana 02

IA Aplicada al Agro

11 de enero de 2026

”El código no solo debe funcionar, debe escalar.”

Hoja de Ruta

1. El Principio ”Fail Fast”

En ingeniería, un error silencioso es peor que un error ruidoso.

Problema: Arrow Code

Anidar `if` dentro de `if` hace el código ilegible y difícil de mantener.

```
# MAL: Dificil de leer
def sistema_riego(sensor):
    if sensor.activo:
        if sensor.humedad < 30:
            activar_bombas()
```

Guard Clauses (Cláusulas de Guardia)

Validamos lo negativo primero y retornamos. El código ”feliz” queda plano.

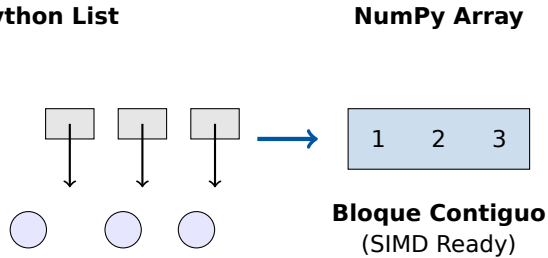
```python
# BIEN: Ingenieria de Software
def sistema_riego(sensor):
    # 1. Validacion (Fail Fast)
    if not sensor.activo:
        return "Error: Sensor inactivo"

    # 2. Logica de Negocio
    if sensor.humedad < 30:
        activar_bombas()
        return "Regando..."

    return "En espera"
```

2. Listas vs. Arrays (La verdad de la RAM)

¿Por qué Python puro es lento? Porque las listas son punteros dispersos.



Complejidad Computacional (Big O)

Analizar 1 millón de plantas.

Python Loop (Iterativo)

- Verifica tipo de dato en cada paso.
- No usa caché del CPU.
- Tiempo: $\approx 300ms$

NumPy (Vectorizado)

- Operación en C optimizado.
- Usa instrucciones SIMD.
- Tiempo: $\approx 5ms$

¡NumPy es 60x más rápido!

3. Pensar en Matrices, no en Bucles

Objetivo: Encontrar zonas de sequía en un campo de 100×100.

```python
import numpy as np

# Generar datos simulados
campo = np.random.rand(100, 100)

# MALA PRACTICA (Estilo C antiguo)
# for i in range(100):
#     for j in range(100): ...

# BUENA PRACTICA (Vectorizacion)
# Crea una mascara booleana instantanea
zonas_sequia = campo < 0.3

print(f"Hectareas afectadas: {np.sum(zonas_sequia)}")
```

Lógica Condicional: np.where

El equivalente a `=SI()` de Excel, pero masivo.

$$R = \text{np.where}(\text{condición}, \text{Valor SI}, \text{Valor NO})$$

```
# Calculo de riego necesario
# Si humedad < 0.4, echar agua hasta llegar a 0.5
# Si no, echar 0 agua.

agua = np.where(campo < 0.4, 0.5 - campo, 0)
```

Próximos Pasos

Semana 03: Big Data

- **Pandas:** Manejo de archivos CSV gigantes.
- **Matplotlib:** Visualización de datos.
- **Limpieza:** Manejo de datos corruptos (NaN).

¿Preguntas?