Python para Ciencia de Datos Agroindustrial

Fundamentos Prácticos para la Inteligencia Artificial

Curso de IA Aplicada al Agro

11 de enero de 2026

Índice general

1 Capítulo I: Automatizar Decisiones Agronómicas **3**
- 1.1 Variables: Etiquetas para tus datos . . . 3
- 1.2 Decisiones automáticas: if / elif / else . . . 3
- 1.3 Repetición eficiente: bucles `for` . . . 3
- 1.4 Funciones: Empaqueta tu lógica . . . 3

2 Capítulo II: Trabajar con Datos Reales **5**
- 2.1 Listas: Series de mediciones . . . 5
- 2.2 Diccionarios: Registros estructurados . . . 5
- 2.3 Archivos CSV: Leer datos del mundo real . . . 5

3 Capítulo III: Hacia la Eficiencia — Introducción a NumPy **6**
- 3.1 ¿Por qué NumPy? Un ejemplo con datos reales . . . 6
- 3.2 Operaciones vectorizadas: sin bucles . . . 6
- 3.3 Lógica condicional vectorizada: `np.where` . . . 7

Prefacio: Python como Asistente Inteligente del Campo

Tras dominar la terminal y Git en la Semana 1, ahora usarás **Python para dar inteligencia a tus datos**. Este lenguaje es la herramienta estándar en IA porque es legible, flexible y tiene librerías poderosas.

Al finalizar, podrás:

- Escribir scripts que evalúen condiciones de cultivo.
- Procesar mediciones de sensores y archivos CSV.
- Entender por qué NumPy es esencial para la IA.
- Guardar tu trabajo en Git, como un verdadero ingeniero.

> **Recuerda**: No se trata de memorizar sintaxis, sino de resolver problemas reales del agro — y versionar tus soluciones.

1 Capítulo I: Automatizar Decisiones Agronómicas

La programación es simplemente **automatizar reglas**. En agroindustria, ya usas reglas todos los días: *"Si la humedad es baja y la temperatura alta, riego."* Python te permite codificar esas reglas.

1.1 Variables: Etiquetas para tus datos

Una variable guarda un valor. Piensa en ellas como etiquetas en frascos de laboratorio.

Listing 1: Datos de una parcela

```
temperatura = 28.5   # en grados Celsius
humedad = 72         # en porcentaje
parcela = "Zona Norte"
activo = True        # Sensor activo
```

1.2 Decisiones automáticas: if / elif / else

Python evalúa condiciones y actúa según reglas.

Listing 2: Sistema de riego automático

```
if humedad < 40:
    print("⚠ Riego necesario en", parcela)
elif humedad > 80:
    print("□ Drenaje recomendado")
else:
    print("□ Condiciones óptimas")
```

Aplicación en IA

Los modelos de Machine Learning (como árboles de decisión) toman miles de decisiones como esta, pero aprendidas de los datos, no escritas a mano.

1.3 Repetición eficiente: bucles `for`

Cuando tienes múltiples parcelas, no repites código: usas un bucle.

Listing 3: Análisis de varias parcelas

```
rendimientos = [25, 30, 28, 32]  # qq/ha

for i in range(len(rendimientos)):
    r = rendimientos[i]
    if r > 30:
        print(f"Parcela {i+1}: Alta productividad ({r} qq/ha)")
    else:
        print(f"Parcela {i+1}: Revisar manejo ({r} qq/ha)")
```

1.4 Funciones: Empaqueta tu lógica

Una función es una receta reutilizable.

Listing 4: Evaluación de suelo

```
def evaluar_suelo(ph):
    """Devuelve una recomendación según el pH del suelo."""
```

```
    if ph < 5.5:
        return "Ácido — necesita cal"
    elif ph > 7.5:
        return "Alcalino — necesita azufre"
    else:
        return "pH óptimo"

print("Recomendación:", evaluar_suelo(6.2))
```

2 Capítulo II: Trabajar con Datos Reales

En la agroindustria, los datos viven en archivos. Aprenderás a leerlos y organizarlos.

2.1 Listas: Series de mediciones

Ideal para temperaturas diarias, rendimientos por parcela, etc.

```
temperaturas = [22.1, 23.5, 25.0, 24.8, 26.2]
promedio = sum(temperaturas) / len(temperaturas)
print(f"Temp. promedio: {promedio:.1f} C")
```

2.2 Diccionarios: Registros estructurados

Útil para representar sensores, parcelas o lotes.

Listing 5: Sensor inteligente

```
sensor_zona1 = {
    "nombre": "Zona A",
    "temperatura": 26.3,
    "humedad": 68,
    "activo": True
}

print(f"{sensor_zona1['nombre']}: {sensor_zona1['temperatura']} C")
```

2.3 Archivos CSV: Leer datos del mundo real

Un archivo CSV es una tabla. Puedes leerlo línea por línea:

Listing 6: Lectura manual de cosecha.csv

```
with open('cosecha.csv', 'r') as f:
    lineas = f.readlines()

# Saltar encabezado
for linea in lineas[1:]:
    parcela, rend, hum = linea.strip().split(',')
    print(f"{parcela}: {rend} qq/ha, humedad {hum}%")
```

3 Capítulo III: Hacia la Eficiencia — Introducción a NumPy

Cuando tus datos crecen, el código común se vuelve lento. Aquí entra *NumPy*.

3.1 ¿Por qué NumPy? Un ejemplo con datos reales

Supón que tienes un archivo `temperaturas.txt` con 365 valores.

Listing 7: Con listas (lento)

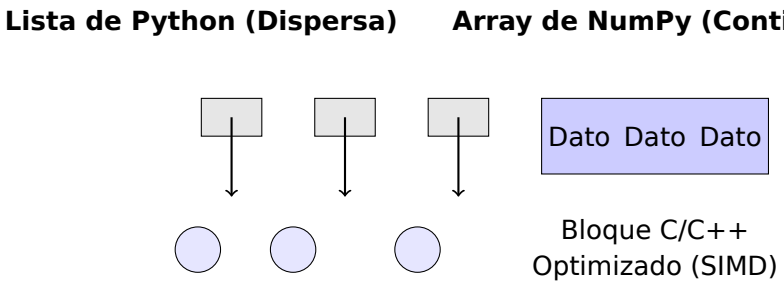
```
temps = []
with open('temperaturas.txt') as f:
    for linea in f:
        temps.append(float(linea))

dias_calurosos = sum(1 for t in temps if t > 30)
```

Listing 8: Con NumPy (rápido y limpio)

```
import numpy as np
temps = np.loadtxt('temperaturas.txt')
dias_calurosos = np.sum(temps > 30)
```

¿Qué cambia?

NumPy almacena los números de forma compacta y usa operaciones optimizadas en C. Para IA, siempre usamos NumPy.



3.2 Operaciones vectorizadas: sin bucles

Todas las operaciones en NumPy se aplican a todo el array a la vez.

Listing 9: Procesamiento de imágenes satelitales (simulado)

```
import numpy as np

# Matriz 100x100: humedad del suelo en una finca
humedad = np.random.rand(100, 100)

# Zonas con riesgo de sequía (humedad < 0.3)
sequia = humedad < 0.3

# Contar celdas en sequía
area_sequia = np.sum(sequia)
print(f"Area en sequia: {area_sequia} m2")
```

3.3 Lógica condicional vectorizada: `np.where`

Para tomar decisiones complejas sobre una matriz:

Listing 10: Presupuesto hídrico automático

```
agua_necesaria = np.where(humedad < 0.4, (0.5 - humedad), 0)
print(f"Agua necesaria: {np.sum(agua_necesaria):.2f}")
```

Reto Final: Sistema de Alerta para Cultivos

Crea un script `alerta_cultivo.py` que defina 3 zonas y evalúe sequía.

```
zonas = [
    {"nombre": "Norte", "humedad": 35},
    {"nombre": "Sur", "humedad": 60},
    {"nombre": "Este", "humedad": 28}
]

def alerta_sequia(humedad):
    return "RIESGO" if humedad < 40 else "OK"

for z in zonas:
    print(f"{z['nombre']}: {alerta_sequia(z['humedad'])}")
```