Documento Analítico – Taller 2 de Programación para SIG

Kevin Orlando Reyes Mosquera

Table of contents

1 1. Introducción **1**

2 2. Ejercicio 1: Variables, estructuras y extracción de datos **2**
2.1 2.1 Código desarrollado 2
2.2 2.2 Análisis técnico integral 2
2.2.1 Escenarios de error 3
2.2.2 Implicaciones en Sistemas de Información Geográfica 3

3 3. Ejercicio 2: Limpieza de Strings y Normalización Toponímica **3**
3.1 3.1 Código desarrollado 3
3.2 3.2 Importancia en bases de datos espaciales 4

4 4. Diferencia entre triples comillas en Python y Julia **4**

5 5. Conclusión **5**

1 1. Introducción

El presente documento examina los fundamentos conceptuales y técnicos aplicados en los ejercicios desarrollados sobre variables, estructuras de datos y manipulación de cadenas de texto, contextualizados en aplicaciones geoespaciales y gestión de bases de datos relacionales.

El análisis no se limita a la ejecución del código, sino que profundiza en:

- La lógica de indexación.
- La estructura interna de almacenamiento.
- La consistencia semántica en bases de datos.
- Las implicaciones prácticas en entornos SIG.

---

2 2. Ejercicio 1: Variables, estructuras y extracción de datos

2.1 2.1 Código desarrollado

```
# Asignación de variables
Nombre_estacion = "Páramo_de_Santurbán"
Latitud = 7.2514
Longitud = -72.9069
Elevación_metros = 3350
Esta_activa = True

# Creación de lista
Variables_estacion = [Nombre_estacion, Latitud, Longitud, Elevación_metros, Esta_activa]

# Creación de diccionario
metadatos_estacion = {
    "Nombre": Nombre_estacion,
    "Coordenadas": [Latitud, Longitud],  # Lista (Base 0)
    "Elevación_metros": Elevación_metros,
    "Esta_activa": Esta_activa
}

# Extracción de longitud
Longitud_estacion = metadatos_estacion["Coordenadas"][1]

Longitud_estacion
```

```
-72.9069
```

2.2 2.2 Análisis técnico integral

Para extraer la longitud se utilizó el índice numérico **1**, debido a que Python emplea **indexación en Base 0**.

La organización interna de la lista fue:

- Posición 0 → Latitud

- Posición 1 → Longitud

Por lo tanto:

`metadatos_estacion["Coordenadas"][1]` devuelve correctamente la Longitud.

2.2.1 Escenarios de error

- En **R**, usar `[0]` produciría un error, ya que R trabaja en Base 1.
- En **Python**, usar `[0]` devolvería la Latitud (error lógico).
- Acceder a `[2]` generaría:

```
IndexError: list index out of range
```

2.2.2 Implicaciones en Sistemas de Información Geográfica

Un error de indexación puede provocar:

- Inversión de coordenadas.
- Desplazamientos espaciales incorrectos.
- Geometrías mal ubicadas.
- Resultados analíticos erróneos.

---

3 3. Ejercicio 2: Limpieza de Strings y Normalización Toponímica

3.1 3.1 Código desarrollado

```
# Creación de variable string
registro_crudo = "   sAnTuaRio de fAuna Y flora iGuaQue   "

# Limpieza
registro_sin_espacios = registro_crudo.strip()

# Formateo
registro_formateado = registro_sin_espacios.title()

# Reemplazo
```

```python
registro_final = registro_formateado.replace("Y", "y")

# Reporte
reporte = f"""Reporte de Limpieza Toponímica:
Registro original: {registro_crudo}
Registro limpio: {registro_sin_espacios}
Registro formateado: {registro_formateado}
Registro final: {registro_final}
Estado: Listo para cruce espacial.
"""

print(reporte)
```

```
Reporte de Limpieza Toponímica:
Registro original:    sAnTuaRio de fAuna Y flora iGuaQue
Registro limpio: sAnTuaRio de fAuna Y flora iGuaQue
Registro formateado: Santuario De Fauna Y Flora Iguaque
Registro final: Santuario De Fauna y Flora Iguaque
Estado: Listo para cruce espacial.
```

3.2 3.2 Importancia en bases de datos espaciales

Los JOINs en bases relacionales requieren coincidencias exactas.

Inconsistencias mínimas pueden generar:

- Registros no emparejados.
- Valores NULL inesperados.
- Pérdida de integridad referencial.
- Mapas incompletos.
- Resultados estadísticos sesgados.

Por ello, la normalización es un paso técnico obligatorio.

---

4 4. Diferencia entre triples comillas en Python y Julia

En Python, las triples comillas permiten:

- Cadenas multilínea.

- Docstrings.
- Documentación interna.

Se almacenan en el atributo:

`__doc__`

En Julia:

- Permiten cadenas multilínea.
- Se integran al sistema de ayuda interactiva.
- Forman parte de su ecosistema científico.

La diferencia fundamental es que en Python son metadatos del objeto, mientras que en Julia están más integradas al entorno interactivo.

---

5 5. Conclusión

La indexación correcta garantiza precisión espacial.
La limpieza de datos asegura integridad relacional.
El dominio de estos fundamentos es esencial en análisis geoespacial profesional.