Variables y tipos de datos

Lina María Quintero Fonseca

2026-02-21

```
# Ejercicio 1: Asignación de variables y operaciones básicas

nombre_estacion = "Páramo de Santurbán"
latitud = 7.2514
longitud = -72.9069
elevacion_m = 3350
esta_activa = True

coordenadas = [latitud, longitud]

metadatos_estacion = {
    "nombre": nombre_estacion,
    "latitud": latitud,
    "longitud": longitud,
    "elevacion_m": elevacion_m,
    "esta_activa": esta_activa,
    "coordenadas": coordenadas
}

longitud_extraida = metadatos_estacion["coordenadas"][1]

elevacion_m += 12.5
metadatos_estacion["elevacion_m"] = elevacion_m

print(
    f"La estación {nombre_estacion} se encuentra operativa en la longitud {longitud_extraida}
    f"a una altura actualizada de {elevacion_m} msnm."
)
```

```
La estación Páramo de Santurbán se encuentra operativa en la longitud -72.9069 a una altura a
```

```python
# Ejercicio 2: Trabajando con strings

registro_crudo = "   sAnTuaRio de fAuna Y flora iGuaQue   "

registro_sin_espacios = registro_crudo.strip()
registro_titulo = registro_sin_espacios.title()
registro_limpio = registro_titulo.replace(" Y ", " y ")

print(
    "Reporte de Limpieza Toponímica:\n"
    f'\tRegistro Original: "{registro_crudo}"\n'
    f'\tRegistro Limpio: "{registro_limpio}"\n'
    "Estado: Listo para cruce espacial."
)
```

```
Reporte de Limpieza Toponímica:
    Registro Original: "   sAnTuaRio de fAuna Y flora iGuaQue   "
    Registro Limpio: "Santuario De Fauna y Flora Iguaque"
Estado: Listo para cruce espacial.
```

1) Ejercicio 1 — Índice usado y por qué

En Python usé el índice 1 para extraer la longitud porque la lista coordenadas se definió como [latitud, longitud] y Python usa indexación base 0: el primer elemento está en 0 y el segundo en 1.

¿Qué error genera si uso el índice del lenguaje opuesto?

Si alguien se confunde y usa 0 en Python esperando la longitud, obtendrá la latitud (no necesariamente un error, sino un valor incorrecto). En cambio, si intenta usar un índice tipo "base 1" mal aplicado para el primer elemento (por ejemplo, creer que el primero es 1 y no 0), en Python puede terminar apuntando al elemento equivocado. Y si usa un índice fuera del rango (por ejemplo 2), Python lanza: IndexError: list index out of range.

2) Ejercicio 2 — Por qué .strip()/trimws() es crítico antes de joins

En geoprocesamiento y bases relacionales, los joins por texto dependen de igualdad exacta entre claves. Los espacios invisibles al inicio o al final convierten un valor "visualmente igual" en una cadena distinta (por ejemplo "Iguaque" "Iguaque").

Si no se limpia, el resultado es un join con muchos registros sin match (NULLs), generando errores de consistencia y análisis (por ejemplo, entidades que quedan sin atributos oficiales o con clasificaciones equivocadas), lo cual es crítico cuando esos campos son llaves para cruces espaciales o validaciones.

3) Diferencia fundamental entre “ “ ” en Python vs Julia

En Python, “ “ ” … “ “ ” define strings multilínea y también se usa comúnmente para docstrings (documentación asociada a módulos, funciones o clases). En Julia, “ “ ” … “ “ ” también define strings multilínea, pero no cumple el mismo rol de docstring por sí sola: en Julia la documentación se asocia usando un string antes de la definición (macro @doc implícita), y el sistema de documentación de Julia está integrado con esa convención.