Conferencia 04

Repitiendo código, pero bien

Repaso Rápido

- Ejecución Secuencial: Una instrucción tras otra.
- Bifurcación: `if`/`elif`/`else`.
- El código se ejecuta **una sola vez**.

El Siguiente Paso: Repetición

¿Cómo procesar colecciones de datos o automatizar tareas a gran escala?

Necesitamos ejecutar bloques de código múltiples veces.

Estructuras de Control Iterativas (Ciclos)

- `while`: Itera mientras una condición sea verdadera.
- `for`: Itera sobre los elementos de una secuencia.

1. El Ciclo `while`: Iteración Condicional

- El ciclo más fundamental.
- Se ejecuta **mientras** una condición sea `True`.

Sintaxis:

```
1  while condicion:
2      # Bloque de código`
```

Ciclo `while`

Ejemplo de código

¡Cuidado! El Ciclo Infinito

- Ocurre si la condición de terminación **nunca** se alcanza.
- El programa se “cuelga”/“traba”.
- Hay que asegurar que la condición pueda volverse `False`, *¿o no?*.

Terminación Prematura de Ciclos

- `break`: Sale inmediatamente del ciclo.
- `continue`: Salta a la siguiente iteración.

2. El Ciclo `for`: Iteración sobre Secuencias

- Diseñado para recorrer colecciones (listas, tuplas, etc.).
- Más simple y seguro que `while` para estos casos.
- Termina automáticamente.

Sintaxis:

```
1  for elemento in secuencia:
2      # Bloque de código`
```

Ciclo `for`

Ejemplo de código

Generando Secuencias: `range()`

- Para iterar un número específico de veces.
- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

`range()` en acción

Ejemplo de código

3. Patrón de Iteración: Acumulador

Un patrón clásico en programación:

1. **Inicializar** una variable.
2. **Actualizarla** dentro de un ciclo.
3. Usar el **resultado** final.

Acumulador: Sumatoria

$$S = \sum_{i=1}^{n} x_i$$

- **Inicializar:** `sumatoria = 0.0`
- **Actualizar:** `sumatoria += x`
- **Resultado:** `sumatoria` contiene el total.

Patrón Acumulador

Ejemplo de código

Búsqueda Secuencial

- ¿Cómo encontrar un elemento en una lista?
- Recorrerla uno por uno hasta hallarlo.
- `break` es ideal para optimizarla.

Búsqueda Secuencial

Ejemplo de código

4. ¿Cuán Eficiente es Nuestro Código?

- El tiempo de ejecución importa.
- Medimos la eficiencia por el número de operaciones principal.
- En la búsqueda, la operación principal es la **comparación**.

Análisis de la Búsqueda Secuencial

Para una lista de tamaño N:

- **Mejor caso:** 1 comparación (el primer elemento).
- **Peor caso:** N comparaciones (el último elemento o no está).

Coste Lineal

- El tiempo de ejecución es **proporcional** al tamaño de los datos.
- Si N se duplica, el tiempo (en el peor caso) se duplica.
- Concepto clave: **Escalabilidad**.

Resumen

- `while`: Iteración condicional.
- `for`: Iteración sobre secuencias.
- `range()`: Generador de secuencias numéricas.
- **Patrón Acumulador**: Para calcular resultados agregados.
- `break`: Para optimizar búsquedas.
- **Coste Lineal**: Una primera mirada a la eficiencia.