Python Cheatsheet – Crash Course on Python

> **Curso:** Crash Course on Python
> **Módulo 3:** Loops
> **Tema:** Bucles y control de iteraciones en Python

Introducción

Los **loops** permiten ejecutar un bloque de código varias veces, según condiciones o secuencias.
Tipos principales:

- **for loop** → para iterar sobre secuencias (listas, tuplas, strings, diccionarios)
- **while loop** → ejecuta mientras una condición sea True

For Loop

```
frutas = ["manzana", "banana", "cereza"]
for fruta in frutas:
    print(fruta)
```

- Itera sobre cada elemento de la secuencia
- No requiere variable de índice, aunque se puede usar `enumerate()`:

```
for i, fruta in enumerate(frutas):
    print(i, fruta)
```

While Loop

```
i = 0
while i < 5:
    print(i)
    i += 1
```

- Ejecuta el bloque mientras la condición sea **True**
- Se debe actualizar la variable de control para evitar **loops infinitos**

Control de Loops

Comando	Función

Comando	Función
break	Sale del loop inmediatamente
continue	Salta a la siguiente iteración
else	Se ejecuta después del loop si no hubo break

Ejemplo:

```
for i in range(5):
    if i == 3:
        break
    print(i)
else:
    print("Terminado")  # No se ejecuta porque hubo break
```

◈ Bucles Anidados

```
adj = ["rojo", "grande"]
frutas = ["manzana", "banana"]

for a in adj:
    for f in frutas:
        print(a, f)
```

- Un bucle dentro de otro → útil para matrices, combinaciones, etc.

◈ Iterar sobre Diccionarios

```
persona = {"nombre": "Ana", "edad": 25}

for clave, valor in persona.items():
    print(clave, valor)
```

- `.items()` → devuelve pares clave-valor
- `.keys()` → devuelve solo claves
- `.values()` → devuelve solo valores

◈ Buenas Prácticas con Loops

- Mantener bloques claros y bien indentados
- Evitar loops infinitos

- Usar `enumerate()` o `items()` para mejorar legibilidad
- Separar lógica compleja en funciones

Referencias

- Python Docs – for statements
- Python Docs – while statements
- W3Schools – Python Loops