BUCLES FOR

Se utiliza para **iterar sobre una secuencia de elementos**, como una lista, una tupla, un diccionario o un conjunto, **y realizar una acción en cada uno** de ellos.

Sintaxis: for elemento in secuencia → hacer algo con el elemento

elemento se ha convertido en una variable

Ejemplo:

estaciones = ['primavera', 'verano', 'otoño', 'invierno']

- Elemento: primavera, verano, otoño, invierno
- Secuencia: lista estaciones

For estación in estaciones

Print(f"estamos accediendo al elemento {estacion}")

Output:

estamos accediendo al elemento primavera

estamos accediendo al elemento verano

estamos accediendo al elemento otoño

estamos accediendo al elemento invierno

Se pueden **usar** las **funciones** aplicables a cada **tipo de secuencia**. Es decir, si la secuencia es una lista se pueden aplicar las funciones de las listas, si es un diccionario se pueden aplicar las funciones de los diccionarios.

Para diccionarios funciona un poco **diferente**. Si le pedimos que imprima el elemento va a imprimir la **key por defecto**. Para que indente las **values** o todos los **ítems** hay que **especificarlo**.

La **sintaxis** es: for elemento in nombre_diccionario.values(): O nombre_diccionario.keys():

Como están formados por clave valor se pueden establecer **dos elementos**, la **sintaxis** es: for k, v in diccionario.items():

Método Range se utiliza para **generar una secuencia de números enteros** para iterar en un bucle for o para crear una lista de números.

Sintaxis: variable = range(start, stop, step)

For x in variable:

    print(x)

Start número en el que empieza o 0, **Stop** número en el que termina pero excluyéndolo, **Step** cuántos salta

List comprehensions con bucles for para crear listas a partir de otras listas o iterables en una sola línea de código.

Son más rápidas que los bules for y con una sintaxis más sencilla

La **sintaxis** lleva [] como una **lista = [acciónquesea for X in Y]**

Ejemplo:

Tenemos la lista → palabras = ["adalab", "python", "data analysts", "sql", "programadoras"]

Queremos crear una lista con el número de caracteres de cada string de la lista palabras → palabras2 = [len(pal) for pal in palabras]

Output → [6, 6, 13, 3, 13]

List comprehensions con if

La **sintaxis** lleva [] como una **lista = [acciónquesea for X in Y if condición]**

Ejemplo: lista = [num ** 2 for num in range(5000000) if num % 2 == 0]

List comprehensions con if...else

Elif no se puede añadir

La **sintaxis** lleva [] como una **lista = [acciónquesea if condición else acciónquesea for X in Y]**

Ejemplo: lista = ["palabracorta " if len(i) <= 6 else "palabra larga" for i in palabras]

List comprehensions con dos for

Listanueva = [num for sublista in listas for num in sublistas