Programación con python

For Loops Parte 2

For Loop

```
for i in range(10):
    for j in range(10):
        print(j)
```

For Loop

Un nested for loop (o bucle for anidado) es cuando dentro de un bucle for se coloca otro bucle for.
Esto significa que por cada iteración del bucle externo, se ejecuta completamente el bucle interno.

Es como tener dos niveles de repetición:

1. El bucle externo controla la cantidad de veces que se repite todo el ciclo.

2. El bucle interno ejecuta una serie de instrucciones cada vez que el externo da una vuelta.

Se usa mucho para trabajar con tablas, matrices, combinaciones de valores o patrones repetitivos.

For Loop

```python
for i in range(3):          # bucle externo (3 veces)
    for j in range(3):      # bucle interno (3 veces)
        print("i =", i, "j =", j)
```

For Loop

```
i = 0 j = 0
i = 0 j = 1
i = 0 j = 2
i = 1 j = 0
i = 1 j = 1
i = 1 j = 2
i = 2 j = 0
i = 2 j = 1
i = 2 j = 2
```

For Loop

Correrá 200 veces

```
for i in range(10):
    for j in range(10):
        for w in range(2):
            print(i, j, w)
```

Esto es útil para problemas con 3 dimensiones (como coordenadas en un cubo, matrices 3D, o pruebas de todas las combinaciones de valores posibles).

For Loop

Imprimir los valores individuales de listas dentro de una lista

```
lst = [[1, 2], [3, 4], [5, 6], [7, 8]]

for i in range(len(lst)):
    interior_lst = lst[i]

    for j in range(len(interior_lst)):
        print(interior_lst[j])
```

For Loop

Ejemplo usando elementos directamente

```
lst = [[1, 2], [3, 4], [5, 6], [7, 8]]

# Recorre cada sublista directamente
for interior_lst in lst:
    # Recorre cada valor de la sublista
    for elemento in interior_lst:
        print(elemento)
```

For Loop

1 o 2 bucles anidados → muy comunes (listas, tablas, combinaciones simples).

3 bucles anidados → se usan en casos reales, especialmente cuando hay tres dimensiones de datos o combinaciones.

Más de 3 → casi nunca directamente, ahí se buscan algoritmos más eficientes.

Actividad

1. Escribe un programa en Python que:

   Tenga una cadena de texto almacenada en una variable (por ejemplo: "hello world!").

   Recorra la cadena caracter por caracter.

   Cuando encuentre la letra "w", imprima la posición (índice) en la que aparece dentro de la cadena.

Actividad

2. Escribe un programa en Python que:

Tenga una cadena de texto almacenada en una variable (por ejemplo: "hello world!").

Recorra la cadena carácter por carácter.

Busque todas las ocurrencias de una letra específica (por ejemplo "l").

Imprima todas las posiciones (índices) en las que aparece esa letra.

Actividad

3. Tenga una cadena de texto almacenada en una variable (por ejemplo: "hello world!").

Pida al usuario que ingrese una letra que desee buscar en la cadena.

Recorra la cadena carácter por carácter.

Imprima todas las posiciones (índices) en las que aparece esa letra.

Si la letra no aparece, muestra un mensaje indicando que no fue encontrada.

Actividad

4. Dada la cadena "python programming", recorre cada letra e imprime únicamente las vocales.

Actividad

5. Usa bucles anidados para imprimir las tablas de multiplicar del 1 al 5.

1 x 1 = 1
1 x 2 = 2
…
5 x 10 = 50

Actividad

6. Usa un for anidado para imprimir un triángulo como este:

```
*
**
***
****
*****
```

Actividad

7. Genera todas las combinaciones posibles de una letra (A, B, C) con un número (1, 2, 3).

Actividad

8. Escribe un programa que use bucles anidados para imprimir la palabra "Python" 3 veces en 4 filas.

Python Python Python
Python Python Python
Python Python Python
Python Python Python