SESIÓN CONTROL DE FLUJO EN PYTHON

CONTENIDOS:

- ¿Qué es una sentencia condicional y por qué se necesitan?
- Operadores Booleanos: and, or, negación.
- Operadores de comparación: mayor qué, mayor o igual que, menor que, menor o igual que, igual que, distinto que.
- Paréntesis y expresiones Booleanas.
- La sentencia if.
- La sentencia if else.
- La sentencia if elif else.
- Expresiones ternarias.
- ¿Qué es una sentencia iterativa y por qué se necesitan?
- La sentencia while.
- La sentencia for.
- Iterando listas de elementos.
- Iterando diccionarios de elementos.
- La función range.
- Iterando con la función range.

INTRODUCCIÓN AL CONTROL DE FLUJO

El control de flujo es un concepto fundamental en programación, ya que permite definir la secuencia de ejecución del código en función de condiciones o repeticiones. Existen dos tipos principales de control de flujo: **sentencias condicionales**, que permiten al programa tomar decisiones, y **sentencias iterativas**, que permiten repetir acciones múltiples veces. Estos conceptos son esenciales para escribir programas más flexibles y útiles, ya que permiten crear estructuras complejas en lugar de secuencias lineales de comandos.

¿QUÉ ES UNA SENTENCIA CONDICIONAL Y POR QUÉ SE NECESITAN?

Una sentencia condicional evalúa una condición y, según si esta es verdadera o falsa, ejecuta una determinada sección de código. Estas sentencias son necesarias para que un programa pueda reaccionar a diferentes situaciones. Por ejemplo, en un sistema de inicio de sesión, se utiliza una sentencia condicional para verificar si la contraseña ingresada es correcta.

OPERADORES BOOLEANOS

Los operadores booleanos en Python (and, or, y not) permiten combinar condiciones y tomar decisiones más complejas.

- **and**: Devuelve True solo si ambas condiciones son verdaderas.
- **or**: Devuelve True si al menos una de las condiciones es verdadera.
- **not**: Invierte el valor de una condición (True se convierte en False, y viceversa).

```python
edad = 20
pais = "Chile"

if edad >= 18 and pais == "Chile":
    print("Eres mayor de edad en Chile")
```

Ilustración 1 Ejemplo de evaluación condicional con operador booleano

OPERADORES DE COMPARACIÓN

Los operadores de comparación se usan para comparar valores y devolver un resultado booleano (True o False).

Operador	Función
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
==	Igual que
!=	Distinto que

```python
a = 10
b = 5

if a > b:
    print("a es mayor que b")
```

Ilustración 2 Ejemplo uso operadores de comparación

PARÉNTESIS Y EXPRESIONES BOOLEANAS

Al usar múltiples condiciones, los paréntesis ayudan a agruparlas y definir el orden de evaluación.

```python
x = 10
y = 20
z = 30

if (x < y and y < z) or (z < x):
    print("La condición es verdadera")
```

Ilustración 3 Ejemplo uso de paréntesis en expresiones condicionales

LA SENTENCIA IF

La sentencia **if** es una estructura de control en programación que permite ejecutar un bloque de código solo si una condición específica es verdadera. Si la condición es falsa, el programa ignora ese bloque y sigue ejecutando el código siguiente.

```python
temperatura = 25

if temperatura > 20:
    print("Hace calor")
```

Ilustración 4 Uso de sentencia if

LA SENTENCIA IF ELSE

La sentencia if else permite ejecutar un bloque de código si una condición es verdadera y otro bloque alternativo si es falsa. Es una estructura de control que permite tomar decisiones entre dos caminos diferentes en la ejecución del programa.

```
temperatura = 15

if temperatura > 20:
    print("Hace calor")
else:
    print("Hace frío")
```

Ilustración 5 Sentencia if else

LA SENTENCIA IF ELIF ELSE

La sentencia if elif else permite evaluar múltiples condiciones en secuencia. Si la primera condición (if) es verdadera, ejecuta un bloque de código; si no, evalúa condiciones adicionales (elif), y si ninguna es verdadera, ejecuta el bloque else.

```
temperatura = 15

if temperatura > 30:
    print("Hace mucho calor")
elif temperatura > 20:
    print("Hace calor")
else:
    print("Hace frío")
```

Ilustración 6 Sentencia if elif else

EXPRESIONES TERNARIAS

Las expresiones ternarias son una forma concisa de la sentencia if else en una sola línea. Evalúan una condición y devuelven un valor si es verdadera y otro si es falsa. Son útiles para simplificar decisiones simples en el código.

```
edad = 18
mensaje = "Mayor de edad" if edad >= 18 else "Menor de edad"
print(mensaje)
```

Ilustración 7 Uso de expresión ternaria

¿QUÉ ES UNA SENTENCIA ITERATIVA Y POR QUÉ SE NECESITAN?

Una sentencia iterativa permite repetir un bloque de código varias veces. Esto es útil para realizar operaciones repetitivas o para procesar elementos de una lista o colección.

LA SENTENCIA WHILE

La sentencia while ejecuta un bloque de código mientras una condición sea verdadera. Es importante asegurar que la condición cambie en algún momento para evitar un bucle infinito.

```
contador = 1

while contador <= 5:
    print(f"Contador: {contador}")
    contador += 1
```

Ilustración 8 Uso de While

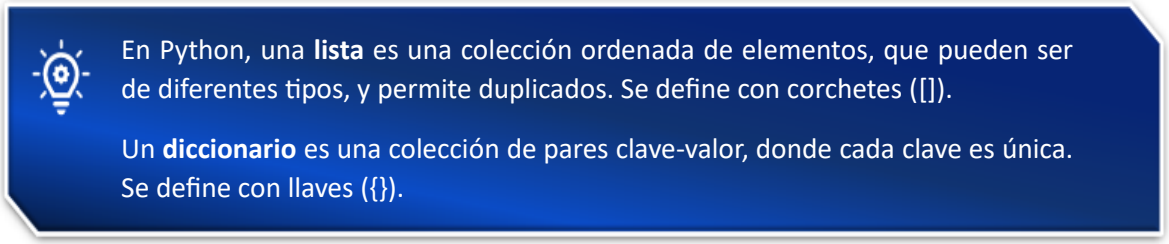
LA SENTENCIA FOR

La sentencia for permite recorrer elementos de una lista, diccionario o rango de números.

```python
nombres = ["Ana", "Luis", "Juan"]

for nombre in nombres:
    print(f"Hola, {nombre}")
```

Ilustración 9 Sentencia for



En Python, una **lista** es una colección ordenada de elementos, que pueden ser de diferentes tipos, y permite duplicados. Se define con corchetes ([]).

Un **diccionario** es una colección de pares clave-valor, donde cada clave es única. Se define con llaves ({}).

ITERANDO LISTAS DE ELEMENTOS

Puedes usar for para recorrer listas y trabajar con cada elemento individualmente.

```python
frutas = ["manzana", "banana", "naranja"]

for fruta in frutas:
    print(f"Fruta: {fruta}")
```

Ilustración 10 Iterando una lista con for

ITERANDO DICCIONARIOS DE ELEMENTOS

Para iterar un diccionario, se usa **.items()** para acceder tanto a las claves como a los valores.

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

for clave, valor in persona.items():
    print(f"{clave}: {valor}")
```

Ilustración 11 Iterando un diccionario

LA FUNCIÓN RANGE

range() genera una secuencia de números, comúnmente usada para repetir acciones un número específico de veces.

```python
for i in range(5):
    print(i)  # Imprime números del 0 al 4
```

Ilustración 12 Función range()

ITERANDO CON LA FUNCIÓN RANGE

Usar range() con un valor de inicio, fin y paso permite más control sobre la secuencia de números generados.

```python
for i in range(1, 10, 2):
    print(i)  # Imprime números 1, 3, 5, 7, 9
```

Ilustración 13 Iteración con range()