ThePower business school

Bucles

Python proporciona dos tipos principales de bucles: `for` y `while`.

Bucles for

Bucles `for`: Estos bucles se utilizan para iterar sobre una secuencia (como una lista, una tupla, un diccionario, un conjunto o una cadena) o cualquier otro objeto iterable.

```
# Iterar sobre una lista
nombres = ["Ana", "Juan", "Carlos"]
for nombre in nombres:
    print(nombre)

# Iterar sobre una cadena
s = "Hola Mundo"
for letra in s:
    print(letra)
```

Bucles while

Bucles `while`: Un bucle `while` se ejecuta repetidamente mientras una condición dada sea verdadera.

```
# Imprimir números del 1 al 5
i = 1
while i <= 5:
    print(i)
    i += 1
```

Elementos iterables

En cuanto a los objetos **iterables**, estos son objetos que contienen un número contable de valores y pueden ser iterados. Python tiene varios tipos de objetos iterables, incluyendo:

- **Cadenas**: Las cadenas son una secuencia de caracteres, por lo que pueden ser iteradas carácter por carácter.
- **Listas y tuplas**: Estos son ejemplos de colecciones ordenadas en Python. Pueden contener elementos de cualquier tipo y pueden ser iteradas elemento por elemento.
- **Diccionarios y conjuntos**: Aunque no están ordenados, estos tipos de colecciones también son iterables. Los diccionarios son iterados a través de sus claves.
- **Archivos**: Los archivos en Python también son iterables. Cuando iteras sobre un archivo, obtienes una línea de texto en cada iteración.

Ejemplos adicionales

vamos a ver algunos casos de uso para entender mejor el funcionamiento de los bucles, cuando comenzamos a desarrollar suele ser un concepto que tardamos en encajar.

Iterar a través de una lista de números y calcular su suma:

```
numeros = [1, 2, 3, 4, 5]
suma = 0

for numero in numeros:
    suma += numero

print("La suma es:", suma)
# Salida: La suma es: 15
```

Utilizar el bucle `for` con la función `range()` para generar una secuencia de números:

```
for i in range(5):
    print(i)
# Salida: 0 1 2 3 4
```

Aquí, `range(5)` genera una secuencia de números del 0 al 4, y el bucle `for` itera sobre esta secuencia.

Usar un bucle `for` para iterar a través de los caracteres de una cadena:

```
cadena = "Hola mundo"

for caracter in cadena:
    print(caracter)
```

Este bucle imprimirá cada caracter de la cadena en una nueva línea.

Usar un bucle `while` para encontrar el primer número en una lista que sea divisible por 5:

```
numeros = [12, 16, 18, 20, 25]
i = 0

while numeros[i] % 5 != 0:
    i += 1

print(numeros[i])
# Salida: 20
```

Aquí, el bucle `while` se ejecuta hasta que encuentra un número divisible por 5.

Iterar a través de un diccionario para imprimir las claves y los valores:

```
estudiante = {
    "nombre": "Juan",
    "edad": 20,
    "curso": "Python"
}
```

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

Aquí, `.items()` devuelve pares de clave-valor en el diccionario, y el bucle `for` itera sobre estos pares.

Uso de la declaración `break` para salir anticipadamente de un bucle:

```
numeros = [1, 3, 9, 16, 20, 22, 25]

for numero in numeros:
    if numero % 8 == 0:
        print("El primer número divisible por 8 encontrado es:", numero)
        break
```

En este caso, una vez que se encuentra un número que es divisible por 8, el bucle se interrumpe debido al `break`.

Estos son solo algunos ejemplos del uso de bucles en Python. Existen muchas otras formas en las que puedes utilizar bucles para realizar tareas repetitivas de manera eficiente.

Combinación Bucles y Condicionales

¡Por supuesto! Aquí te dejo algunos ejemplos más donde los bucles se combinan con las declaraciones `if` para realizar tareas más complejas:

Encontrar los números pares en una lista:

```
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for numero in numeros:
    if numero % 2 == 0:
        print(numero)
```

En este ejemplo, el bucle `for` recorre cada número en la lista. La declaración `if` comprueba si el número actual es par, y si es así, se imprime.

Encontrar las vocales en una cadena de texto:

```
texto = "Hola Mundo"
vocales = "aeiouAEIOU"

for letra in texto:
    if letra in vocales:
        print(letra)
```

Aquí, el bucle `for` recorre cada carácter en la cadena de texto. La declaración `if` comprueba si el carácter actual es una vocal (es decir, si se encuentra en la cadena `vocales`), y si es así, se imprime.

Usar un bucle `while` para solicitar al usuario que ingrese un número positivo:

```
numero = -1

while numero < 0:
    numero = int(input("Por favor, ingresa un número positivo: "))
```

En este caso, el bucle `while` se ejecutará hasta que el usuario ingrese un número positivo.

Encontrar el primer número negativo en una lista:

```
numeros = [5, 8, 10, -3, 2, 1]

for numero in numeros:
    if numero < 0:
        print("El primer número negativo encontrado es:", numero)
        break
```

En este ejemplo, el bucle se rompe tan pronto como se encuentra el primer número negativo.

Contar la cantidad de letras minúsculas en una cadena:

```
cadena = "Hola Mundo Python"
count = 0

for letra in cadena:
    if letra.islower():
        conteo += 1

print("Cantidad de letras minúsculas:", conteo)
```

Este ejemplo recorre cada carácter en la cadena y utiliza el método `.islower()` para verificar si el carácter es una letra minúscula. Si es así, incrementa el contador.

Estos ejemplos muestran cómo se pueden combinar los bucles con las declaraciones condicionales para realizar tareas más complejas en Python.