Puntos Clave

1. Existen dos tipos de bucles en Python: `while` y `for`:

- El bucle `while` ejecuta una sentencia o un conjunto de sentencias siempre que una condición booleana especificada sea verdadera, por ejemplo:

```
# Ejemplo 1
while True:
    print("Atascado en un bucle infinito.")

# Ejemplo 2
counter = 5
while counter > 2:
    print(counter)
    counter -= 1
```

- El bucle `for` ejecuta un conjunto de sentencias muchas veces; se usa para iterar sobre una secuencia (por ejemplo, una lista, un diccionario, una tupla o un conjunto; pronto aprenderás sobre ellos) u otros objetos que son iterables (por ejemplo, cadenas). Puedes usar el bucle `for` para iterar sobre una secuencia de números usando la función incorporada `range`. Mira los ejemplos a continuación:

```
# Ejemplo 1
word = "Python"
for letter in word:
    print(letter, end="*")

# Ejemplo 2
for i in range(1, 10):
    if i % 2 == 0:
        print(i)
```

2. Puedes usar las sentencias `break` y `continue` para cambiar el flujo de un bucle:

- Utiliza `break` para salir de un bucle, por ejemplo:

```
text = "OpenEDG Python Institute"
for letter in text:
    if letter == "P":
        break
    print(letter, end="")
```

- Utiliza `continue` para omitir la iteración actual, y continuar con la siguiente iteración, por ejemplo:

```
text = "pyxpyxpyx"
for letter in text:
    if letter == "x":
        continue
    print(letter, end="")
```

3. Los bucles `while` y `for` también pueden tener una cláusula `else` en Python. La cláusula `else` se ejecuta después de que el bucle finalice su ejecución siempre y cuando no haya terminado con `break`, por ejemplo:

```
n = 0

while n != 3:
    print(n)
    n += 1
else:
    print(n, "else")

print()

for i in range(0, 3):
    print(i)
else:
    print(i, "else")
```

4. La función `range()` genera una secuencia de números. Acepta enteros y devuelve objetos de rango. La sintaxis de `range()` tiene el siguiente aspecto: `range(start, stop, step)`, donde:

- `start` es un parámetro opcional que especifica el número de inicio de la secuencia (`0` por defecto)
- `stop` es un parámetro opcional que especifica el final de la secuencia generada (no está incluido).
- y `step` es un parámetro opcional que especifica la diferencia entre los números en la secuencia es (`1` por defecto.)

Código de ejemplo:

```
for i in range(3):
    print(i, end=" ")  # Salidas: 0 1 2

for i in range(6, 1, -2):
    print(i, end=" ")  # Salidas: 6, 4, 2
```