LOOPS

John Karuitha, PhD

February 27, 2025

1 Introduction to Loops in Python

Loops are a fundamental concept in programming that allow us to repeat a block of code multiple times. In Python, there are two main types of loops: `for` loops and `while` loops. Understanding how to use these loops will help you automate repetitive tasks, process data, and solve problems more efficiently.

2 `for` Loop

A `for` loop is used to iterate over a sequence (like a list, tuple, string, or range) and execute a block of code for each element in that sequence (Downey 2024; McKinney 2022).

Syntax:

```
for item in sequence:
    # Code block to execute
```

Example 1: Iterating Over a List

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

Output:

```
apple
banana
cherry
```

Example 2: Using `range()` to Repeat Code

```
for i in range(5):
    print("Hello, world!")
```

Output:

```
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
```

- `range(5)` generates a sequence of numbers from 0 to 4.

3 2. `while` Loop

A `while` loop repeats a block of code as long as a condition is true. This loop is useful when you don't know in advance how many times you need to repeat the code.

Syntax:

```
while condition:
    # Code block to execute
```

Example 3: Counting Down

```
count = 5
while count > 0:
    print(count)
    count -= 1
```

Output:

```
5
4
3
2
1
```

4 3. Loop Controls: `break` and `continue`

Sometimes, you may want to alter the flow of a loop. Python provides two keywords for this: `break` and `continue`.

4.1 `break` Statement

The `break` statement is used to exit the loop immediately, regardless of the loop's condition.

Example 4: Stop the Loop When a Condition is Met

```
for number in range(10):
    if number == 5:
        break
    print(number)
```

Output:

```
0
1
2
3
4
```

- The loop stops when `number` is 5.

4.2 `continue` Statement

The `continue` statement skips the rest of the code inside the loop for the current iteration and moves on to the next iteration.

Example 5: Skip Even Numbers

```
for number in range(10):
    if number % 2 == 0:
        continue
    print(number)
```

Output:

```
1
3
5
7
9
```

- The loop skips even numbers and prints only odd numbers.

5 4. Practical Examples

5.1 Example 6: Summing Numbers Using a `for` Loop

```
numbers = [1, 2, 3, 4, 5]
total = 0
for num in numbers:
    total += num
print("Total:", total)
```

Output:

```
Total: 15
```

5.2 Example 7: Simple Password Checker Using a `while` Loop

```python
password = ""
while password != "python123":
    password = input("Enter the password: ")
print("Access granted!")
```

Output: (If user enters `python123`)

```
Enter the password: python123
Access granted!
```

6 Summary

- Use `for` loops when you know how many times you want to iterate.
- Use `while` loops when you need to keep repeating as long as a condition is true.
- Use `break` to exit a loop early.
- Use `continue` to skip to the next iteration.

By mastering loops, you can write more efficient and less repetitive code! A central mantra in programming is "Don't repeat yourself (DRY)".

References

Downey, Allen B. 2024. *Think Python*. 3rd ed. Green Tea Press.
McKinney, Wes. 2022. *Python for Data Analysis*. " O'Reilly Media, Inc.".