Huawei HCCDA-AI certification

Trainer: Fawad Bahadur Marwat

4

Introduction to Loops in Python

What Are Loops?

- Programming structures that repeat instructions until a condition is met.
- Enable iteration over data structures or repetitive task automation.

Why Use Loops?

- Reduce code redundancy
- Automate repetitive tasks
- Improve code efficiency and readability

Types of Loops

- **while Loop:** Repeats while a condition is `True`
- **for Loop**: Iterates over a sequence (e.g., lists, tuples, strings)

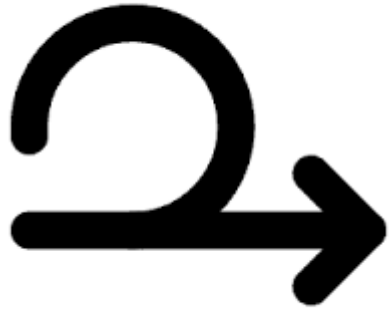
CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

Iterating with Python Loops

What Is Iteration?

- Accessing each item in a collection (e.g., list, tuple, string) one by one.



How Loops Work

- Iterate over Python data structures: lists, dictionaries, sets, strings.
- Enable sequential processing of items in an iterable.

Key Points

- Loops provide sequential access to elements.
- Use loops to read or modify items in sequences (e.g., lists, strings).

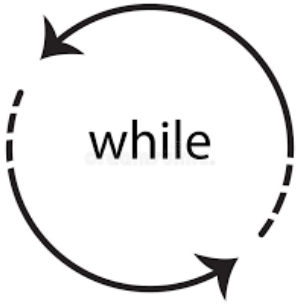
CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The while Loop in Python

How It Works

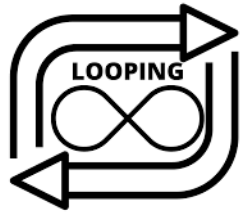
- Executes a code block as long as a condition is True.
- Stops when the condition becomes False.



Useful Scenarios

- When the number of iterations is unknown (e.g., reading a file until its end).

Example




```
count = 0
while count < 3:
    print("Counting:", count)
    count += 1
```

Explanation

- Checks if count < 3 (True initially).
- Prints count and increments by 1.
- Stops when count reaches 3 (False).

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The for Loop in Python

How It Works

Iterates over each item in an iterable
(e.g., list, string, dictionary) until all items are processed.

Useful Scenarios

When the number of iterations is known or iterating over a collection (e.g., lists, strings).

Example

```
colors = ["red", "green", "blue"]
for color in colors:
    print("Color:", color)
```

Explanation

- List: Loops through colors, printing each item.

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The range() Function in Python

Purpose

Generates a sequence of numbers for iterating a specific number of times.

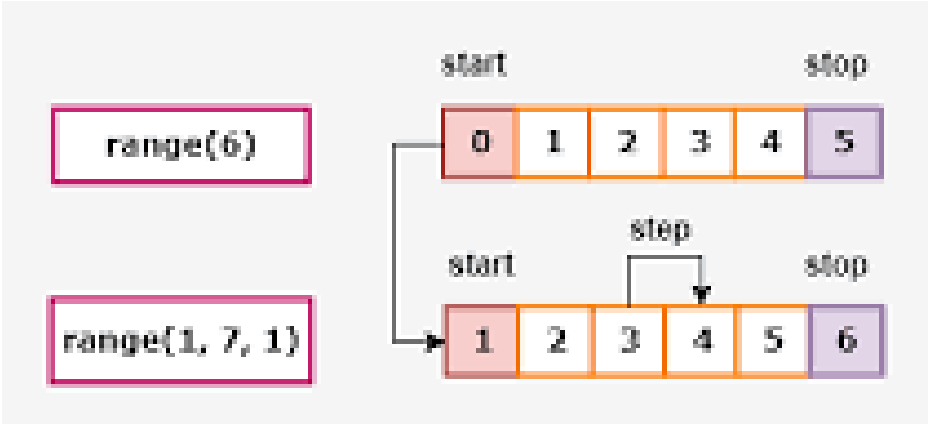
Example

```
for i in range(1, 6):
    print(i)
```

```
for i in range(5, 0, -1):
    print(i)
```

Parameters

Start: starting number (default: 0)
Stop: end number (non-inclusive)
Step: increment/decrement (default: 1)



CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The break Statement in Python

Purpose

Exits a loop prematurely when a specific condition is met.

When to Use

To stop a loop immediately (e.g., when finding a target element in a list).

Example

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

Explanation

Loops through numbers 0 to 9.
When number == 5, break stops the loop.
Output: Prints 0, 1, 2, 3, 4.

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The continue Statement in Python

Purpose

Skips the current iteration and proceeds to the next one without exiting the loop.

When to Use

To bypass specific iterations (e.g., skipping unwanted values in a dataset).

Example

```
for i in range(5):
    if i == 2:
        continue
    print(i)
```

Explanation

- Loops through numbers 0 to 4.
- When i == 2, continue skips the print statement.

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat

The Pass Statement in Python

Purpose

- A null operation; acts as a placeholder when no action is needed.
- Used where code is syntactically required but not yet implemented.

When to Use

As a temporary placeholder (e.g., in empty loops, functions, or try blocks).

Example

```
for i in range(5):
    if i < 3:
        pass
    print(i)
```

Explanation

- Loops through numbers 0 to 4.
- When i < 3, pass does nothing, and the loop continues.
- Output: Prints 0, 1, 2, 3, 4.

CORVIT
Reaching end to end

Trainer: Fawad Bahadur Marwat