Python Loop Control Statements (break, continue, pass)

These statements modify the behavior of loops.

break: Terminates the loop entirely. **continue:** Skips the current iteration and moves to the next one. **pass:** Does nothing, often used as a placeholder.

break

Exits the loop prematurely.

```
for item in sequence:
    if some_condition:
        break  # exit the loop
```

continue

Skips the current iteration and proceeds to the next iteration of the loop.

```
for item in sequence:
    if some_condition:
        continue  # skip the rest of the code in this iteration
    # code to execute if some_condition is False
```

pass

A null statement, used as a placeholder.

```
if condition:
    pass  # do nothing
```

- Video: How to Effectively Use Break and Continue Statements
- Video: Using Python break statement with a while loop

Example #: Using Break Statement in a Loop with Range

```
for x in range(3):
    if x == 1:
        break
```

Example 1: Exit a loop when a number is found

```
# Search for the number 5 and exit the loop when found
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for num in numbers:
    print(f"Checking {num}...")
    if num == 5:
        print("Number 5 found! Exiting the loop.")
        break  # Exit the loop
```

Output:

```
Checking 1...
Checking 2...
Checking 3...
Checking 4...
Checking 5...
Number 5 found! Exiting the loop.
```

Example 2: Password validation with `while` loop

```
# Keep asking for a password until the correct one is entered
correct_password = "python123"

while True:
    user_input = input("Enter the password: ")
    if user_input == correct_password:
        print("Access granted!")
        break  # Exit the loop
    else:
        print("Wrong password. Try again!")
```

Example 1: Skip even numbers using `continue` Statement

Skips the **current iteration** and moves to the next loop cycle.

```
# Print only odd numbers (skip even numbers)
for num in range(1, 11):
    if num % 2 == 0:
        continue  # Skip even numbers
    print(num)
```

Output:

```
1
3
5
7
9
```

Example 2: Temperature Monitoring

Stop monitoring if temperature exceeds a safe limit.

```
temperatures = [25, 30, 32, 28, 45, 29, 33]  # Sensor readings

for temp in temperatures:
    if temp > 40:
        print(f"ALERT: Temperature {temp}°C is unsafe! Shutting down.")
        break  # Exit immediately
    print(f"Temperature {temp}°C is safe.")
```

Output:

```
Temperature 25°C is safe.
Temperature 30°C is safe.
Temperature 32°C is safe.
Temperature 28°C is safe.
ALERT: Temperature 45°C is unsafe! Shutting down.
```

Example 1: **Data Cleaning using `continue` Statement**

Skip invalid entries when processing a dataset.

```
user_ages = [20, 15, "unknown", 30, -5, 25]  # Some invalid data

print("Valid ages:")
for age in user_ages:
    if not isinstance(age, int) or age < 0 or age > 120:
        continue  # Skip invalid entries
    print(f"- {age} years old")
```

Output:

```
Valid ages:
- 20 years old
- 15 years old
- 30 years old
- 25 years old
```

Task: Coffee Machine Stock Check using `break` Statement

Simulate a coffee machine that stops serving when a drink is out of stock.

- **Input List**: `["latte", "cappuccino", "espresso", "mocha", "out_of_stock", "latte"]`
- **Goal**: Loop through the list and stop when "out_of_stock" is found.
- **Example Output**:

```
Serving latte...
Serving cappuccino...
Serving espresso...
Serving mocha...
Out of stock! Machine stopping.
```

Task: Skip Negative Numbers
Calculate the sum of positive numbers in a list, ignoring negatives.

- **Input**: `[5, -2, 10, -8, 3]`
- **Goal**: Use `continue` to skip negative values.
- **Output**: `Sum of positive numbers: 18`

Task: Simple Calculator with Exit using `continue` and `break`
Create a loop that:

- Lets users type numbers to add.
- Skips non-numeric inputs (use `continue`).
- Exits when the user types "quit" (use `break`).
- **Example Output**:

```
Enter a number (or 'quit'): 5
Enter a number (or 'quit'): ten
Invalid input!
Enter a number (or 'quit'): 3
Enter a number (or 'quit'): quit
Total: 8
```

7. **Task: Movie Ticket Checker**
   Loop through a list of ages and:

- Skip ages < 0 (invalid).
- Stop if a "VIP" (age 100+) is found.
- **Input**: `[25, -5, 30, 105, 40]`
- **Output**:

```
Valid age: 25
Skipped invalid age.
Valid age: 30
VIP detected! Stopping sales.
```

8. **Task: Flight Booking System**
   Check seat availability in a list. Stop when a seat is found, or skip "reserved" seats.
   - **Input**: `["reserved", "reserved", "available", "reserved"]`
   - **Output**:

```
Seat 1: Reserved.
Seat 2: Reserved.
Seat 3: Available! Booked successfully.
```