**Python break and continue**

In programming, the break and continue statements are used to alter the flow of loops:

* break exits the loop entirely
* continue skips the current iteration and proceeds to the next one

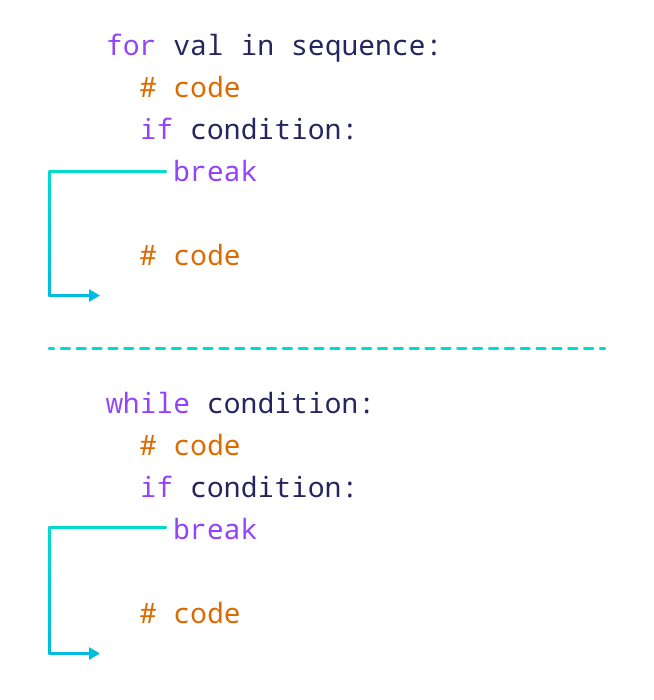
**Python break Statement**

The break statement terminates the loop immediately when it's encountered.

**Syntax**

break

**Working of Python break Statement**

Working of break Statement in Python

The above image shows the working of break statements in [for](https://www.programiz.com/python-programming/for-loop) and [while](https://www.programiz.com/python-programming/while-loop) loops.

**Note:** The break statement is usually used inside decision-making statements such as [if...else](https://www.programiz.com/python-programming/if-elif-else).

**Example: break Statement with for Loop**

We can use the break statement with the for loop to terminate the loop when a certain condition is met. For example,

for i in range(5):

if i == 3:

break

print(i)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

0

1

2

In the above example,

if i == 3:

break

terminates the loop when i is equal to **3.** Hence, the output doesn't include values after **2**.

**Note:** We can also terminate the while loop using a break statement.

break Statement with while Loop

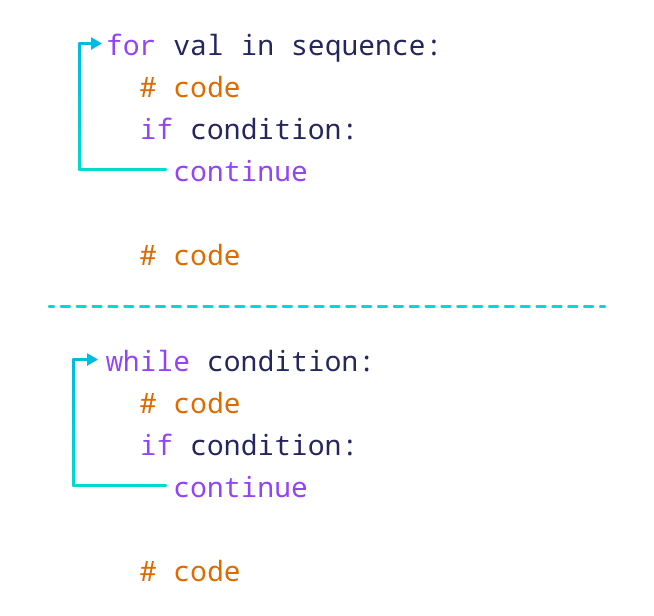
**Python continue Statement**

The continue statement skips the current iteration of the loop and the control flow of the program goes to the next iteration.

**Syntax**

continue

**Working of continue Statement in Python**



**Example: continue Statement with for Loop**

We can use the continue statement with the for loop to skip the current iteration of the loop and jump to the next iteration. For example,

for i in range(5):

if i == 3:

continue

print(i)

[Run Code](https://www.programiz.com/python-programming/online-compiler)

**Output**

0

1

2

4

In the above example,

if i == 3:

continue

skips the current iteration when i is equal to **3**, and continues the next iteration. Hence, the output has all the values except **3**.

**Note:** We can also use the continue statement with a while loop.

continue Statement with while Loop

**Also Read:**

* [Python pass Statement](https://www.programiz.com/python-programming/pass-statement)
* [Python range()](https://www.programiz.com/python-programming/methods/built-in/range)

Before we wrap up, let’s put your knowledge of Python break and continue to the test! Can you solve the following challenge?

Challenge:

Write a function to calculate the sum of elements in a list that are greater than a given number.

* Return the sum of the numbers greater than the given number.
* If numbers is [1, 2, 3, 4, 5] and n is **3**, the return value should be **9** because 4 + 5 is **9**.

1

2

def sum\_greater\_than(numbers, n):

Check Code