Jump Statements: `break` and `continue`

Python offers two jump statements (within loops) to jump out of the loop iterations. These are `break` and `continue` statements. Let us see how these statements work.

`break` Statement

The `break` statement enables a program to skip over a part of the code. A `break` statement terminates the very loop it lies within.

The working of a `break` statement is as follows:

```
while <Expression/Condition/Statement>:
      #Statement1
      if <condition1>:
            break #If "condition" is true, then code breaks out
      #Statement2
      #Statement3
#Statement4: This statement is executed if it breaks out of the loop
#Statement5
```

In the given code snippet, if `condition1` is true, then the flow of execution will break out of the loop. The next statement to be executed will be `Statement4`.

Note: This loop can terminate in 2 ways:

1. Throughout the iterations of the loop, if `condition1` remains false, the loop will go through its normal course of execution and stop when its `condition/expression` becomes false.
2. If during any iteration, the `condition1` becomes true, the flow of execution will break out of the loop and the loop will terminate.

The following code fragment shows you an example of the `break` statement:

```
for val in "string":
    if val == "i":
        break
    print(val)
print("The end")
```

The output will be:

```
s
t
r
```

Here, as soon as the value of `val` becomes equal to `"i"`, the loop terminates.

`continue` Statement

The `continue` statement jumps out of the current iteration and forces the next iteration of the loop to take place.

The working of a `continue` statement is as follows:

```
while <Expression/Condition/Statement>:
      #Statement1
      if <condition1>:
            continue
      #Statement2
      #Statement3
#Statement4: This statement is executed if it breaks out of the loop
#Statement5
```

In the given code snippet if `condition1` is true, the `continue` statement will cause the skipping of `Statement2` and `Statement3` in the current iteration, and the next iteration will start.

The following code fragment shows you an example of the `continue` statement:

```
for val in "string":
    if val == "i":
        continue
    print(val)
print("The end")
```

The output will be:

```
s
t
r
n
g
The end
```

Here, the iteration with `val = "i"`, gets skipped and hence `"i"` is not printed.