

If we want to evaluate the code below more than once then we can use the `repeat` and `again` as follows:

```

repeat
    code block
    again

```

Note: Here we are getting the output "Hello to Gauri" 10 times but it is not a recommended way to write loops. In fact we are encouraged to use a loop with multiple blocks as long as the condition is true.

For loop

The `for` loop is used to repeat a block of code for a fixed number of times. It has the following syntax:

```

for variable in range:
    code block

```

Example:

```

for i in range(10):
    print("Hello to Gauri")

```

In this example, the variable `i` will take values from 0 to 9. The loop will execute the code block 10 times.

Break Statement

The `break` statement is used to exit a loop before it reaches its end. It has the following syntax:

```

break

```

Example:

```

for i in range(10):
    print("Hello to Gauri")
    break

```

In this example, the loop will exit after the first iteration because when it reaches the condition, it will break out of the loop.

Continue Statement

The `continue` statement is used to skip the current iteration of a loop and continue with the next one. It has the following syntax:

```

continue

```

Example:

```

for i in range(10):
    print("Hello to Gauri")
    continue

```

In this example, the loop will skip the current iteration and move to the next one.

Nested Loops

Nested loops are loops inside another loop. They are used to perform operations on multiple levels. For example, if we want to print a grid of numbers, we can use nested loops.

```

for i in range(3):
    for j in range(3):
        print(j)

```

The output of the above code will be:

```

0
1
2
0
1
2
0
1
2

```

Note: In the while condition, Any value having odd case like `if value` on the body of while loop will result in an infinite loop. It is a common mistake.

While loop

The `while` loop is used to repeat a block of code until a certain condition is met. It has the following syntax:

```

while condition:
    code block

```

Example:

```

x = 0
while x < 10:
    print(x)
    x += 1

```

In this example, the loop will continue to print the value of `x` until it reaches 10.

Break Statement

The `break` statement is used to exit a loop before it reaches its end. It has the following syntax:

```

break

```

Example:

```

x = 0
while x < 10:
    print(x)
    if x == 5:
        break
    x += 1

```

In this example, the loop will exit after the fifth iteration because when it reaches the condition, it will break out of the loop.

Continue Statement

The `continue` statement is used to skip the current iteration of a loop and continue with the next one. It has the following syntax:

```

continue

```

Example:

```

x = 0
while x < 10:
    print(x)
    if x == 5:
        continue
    x += 1

```

In this example, the loop will skip the current iteration when `x` is 5 and move to the next one.

Pass Statement

The `pass` statement is used to do nothing. It is used to write code that needs to be written but does not have any logic. It has the following syntax:

```

pass

```

Example:

```

for i in range(10):
    pass

```

In this example, the loop will do nothing for each iteration.

For Each Loop in Python

The `for` loop can be enhanced for lists, tuples, sets, and dictionaries. It can be used to iterate over the collection like array, tuple, list, set, dictionary etc. The `for` loop can be converted to the `foreach` loop for arrays.

Syntax:

```

for variable in collection:
    code block

```

Example:

```

for item in [10, 20, 30, 40, 50]:
    print(item)

```

Output:

```

10
20
30
40
50

```

Example:

```

for item in {"name": "Gauri", "age": 100, "language": "Python"}:
    print(item)

```

Output:

```

name
age
language

```