Control Structures in Python

Control structures are fundamental building blocks in programming. They allow for the execution of code based on specific conditions or in a repetitive manner.

by Adamya Neupane

Conditional Statements

Conditional statements determine the flow of execution in your Python code. They allow you to execute different blocks of code based on whether a specific condition is true or false.

if Statements

The 'if' statement is used to execute a block of code if a given condition is true.

elif Statements

The 'elif' statement is used to check another condition if the preceding 'if' or 'elif' conditions are false.

else Statements

The 'else' statement is used to execute a block of code if all preceding 'if' and 'elif' conditions are false.

if, elif, else Statements

These statements work together to create flexible conditional logic. They allow you to handle multiple scenarios based on different conditions.

if Statement

The 'if' statement evaluates a condition. If the condition is true, the code block within the 'if' statement is executed.

elif Statement

The 'elif' statement provides an alternative condition to check if the preceding 'if' condition is false. If the 'elif' condition is true, its code block is executed.

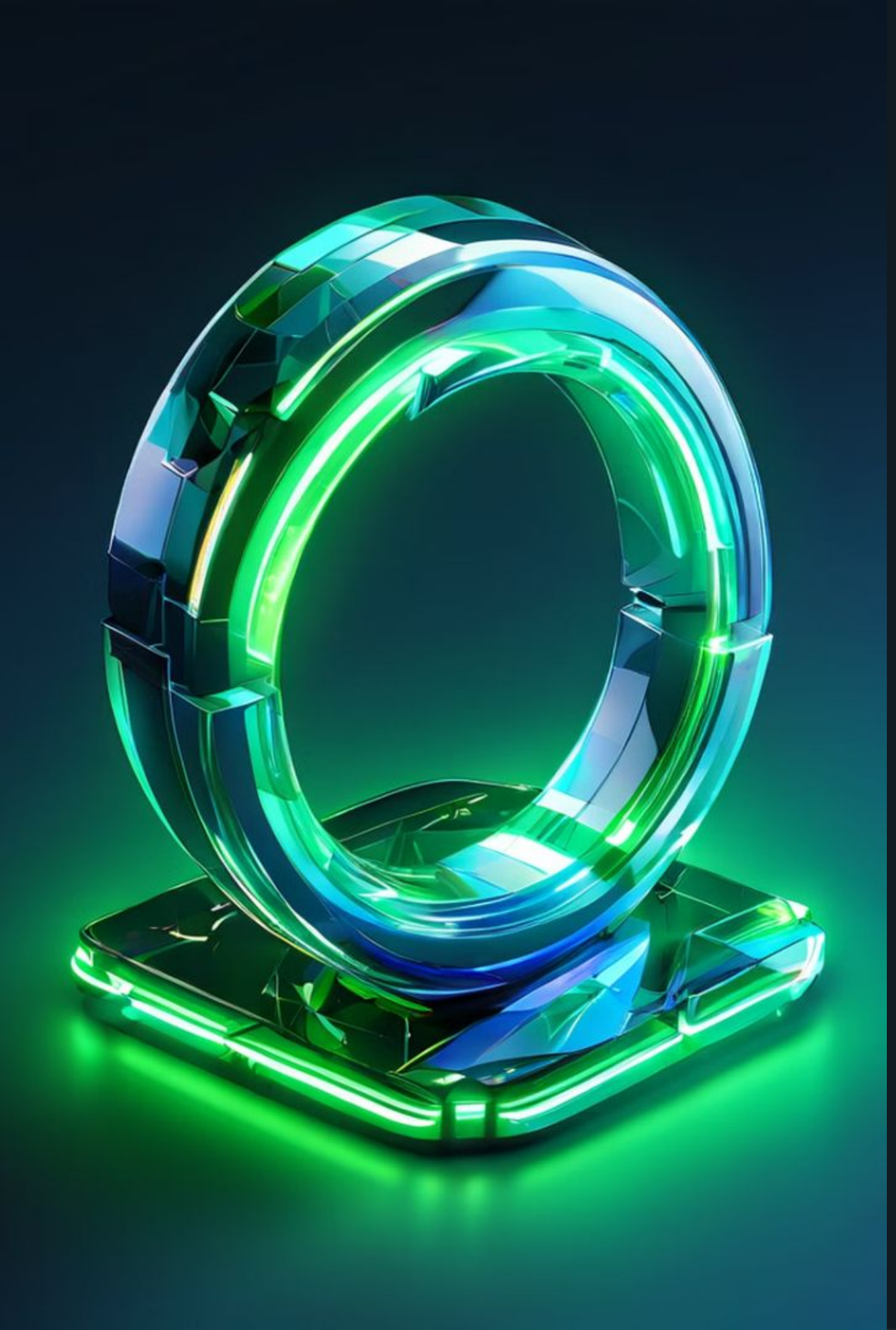
else Statement

The 'else' statement is executed only if all preceding 'if' and 'elif' conditions are false. It provides a default code block.

Comparison Operators

Comparison operators are used to compare values and determine the truth or falsity of a condition.

Operator	Description	Example
==	Equal to	5 == 5 (True)
!=	Not equal to	5 != 3 (True)
>	Greater than	10 > 5 (True)
<	Less than	5 < 10 (True)
>=	Greater than or equal to	5 >= 5 (True)
<=	Less than or equal to	5 <= 10 (True)



Loops

Loops are used to repeatedly execute a block of code as long as a certain condition is met.

1

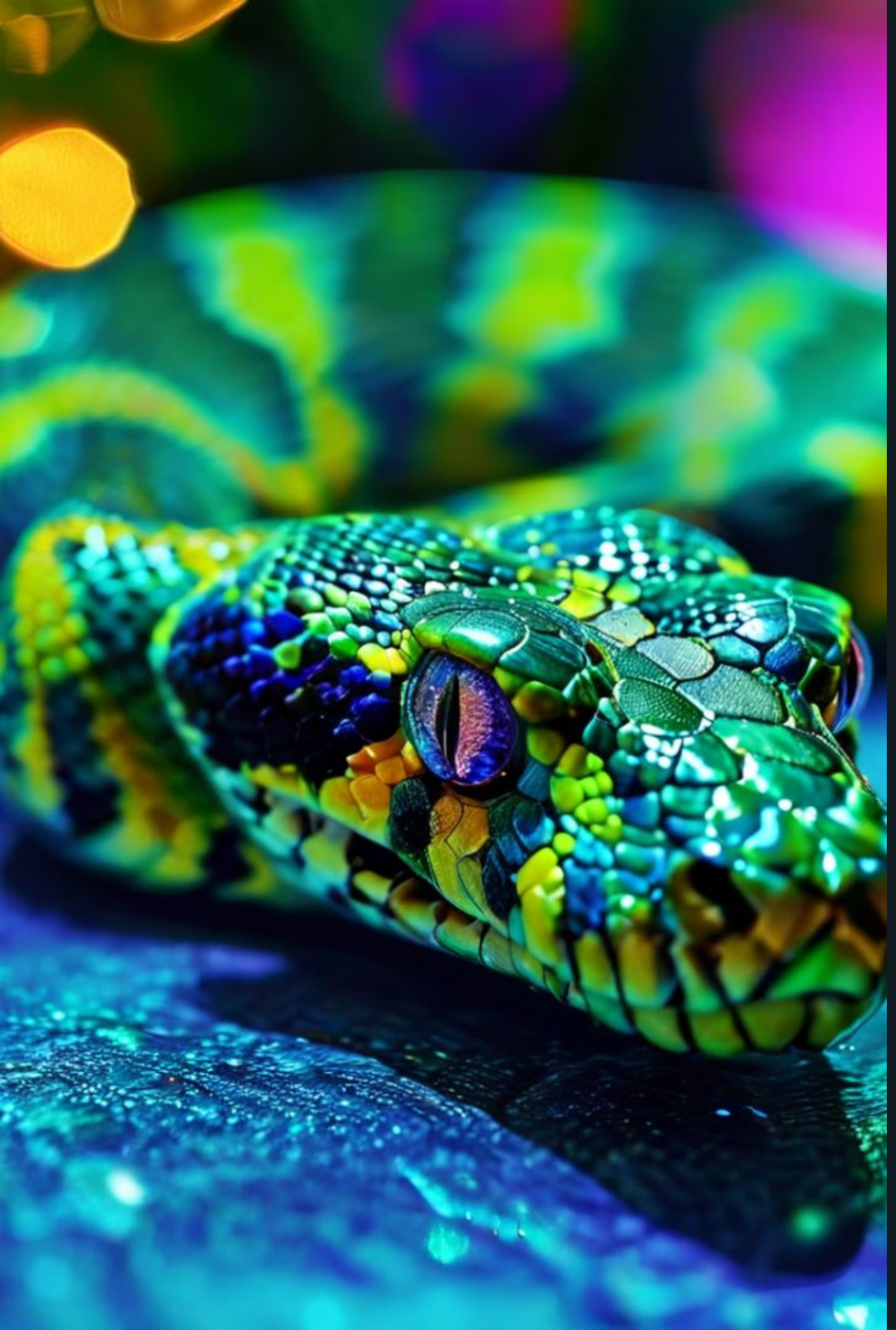
For Loops

Iterate over a sequence of items, executing the code block for each item.

2

While Loops

Execute the code block as long as a specified condition remains true.



for Loops

For loops are used to iterate over a sequence of items, such as lists, tuples, strings, or ranges.

1

Initialization

The loop starts with an initialization step. It sets up a counter variable and initializes it to the first value in the sequence.

2

Condition

The loop continues as long as the counter variable satisfies the specified condition. This condition usually involves checking if the counter is within the bounds of the sequence.

3

Iteration

After each iteration, the loop updates the counter variable, typically by incrementing it to the next value in the sequence.

while Loops

While loops continue executing as long as a condition is true.

1

Condition

The 'while' loop starts by evaluating a condition. If the condition is true, the code block within the loop is executed.

2

Execution

The code block inside the 'while' loop is executed repeatedly.

3

Re-evaluation

After each execution of the code block, the condition is re-evaluated. The loop continues to execute as long as the condition remains true.

Loop Control (break, continue)

Break and continue statements provide more control over the execution of loops.

Break

The 'break' statement immediately exits the loop, regardless of the loop's condition. It's often used to terminate a loop prematurely based on a specific condition.

Continue

The 'continue' statement skips the remaining code in the current iteration and moves to the next iteration of the loop. It's useful for skipping specific iterations based on a condition.

Nested Loops

A nested loop is a loop within another loop. They are useful for iterating over multi-dimensional data structures or performing complex calculations.

Outer Loop

The outer loop controls the overall iterations. For each iteration of the outer loop, the inner loop runs completely.

Inner Loop

The inner loop is executed for each iteration of the outer loop. It often operates on a subset of data determined by the outer loop.

Conclusion

Control structures are fundamental components of Python programming. Understanding how to use them effectively allows you to build complex and powerful applications.