Control flow statements in Programming

Control flow refers to the order in which statements within a program execute. While programs typically follow a **sequential flow** from top to bottom, there are scenarios where we need more flexibility. This article provides a clear understanding about everything you need to know about Control Flow Statements.

What are Control Flow Statements in Programming?

Control flow statements *are fundamental components of programming languages that allow developers to control the order in which instructions are executed in a program. They enable execution of a block of code multiple times, execute a block of code based on conditions, terminate or skip the execution of certain lines of code, etc.*

Types of Control Flow statements in Programming:

Control Flow Statements Type	Control Flow Statement	Description
Conditional Statements	if-else	Executes a block of code if a specified condition is true, and another block if the condition is false.
Looping Statements	for	Executes a block of code a specified number of times, typically iterating over a range of values.
	while	Executes a block of code as long as a specified condition is true.
Jump Statements	break	Terminates the loop or switch statement and transfers control to the statement immediately following the loop or switch.
	continue	Skips the current iteration of a loop and continues with the next iteration.
	return	Exits a function and returns a value to the caller.

Conditional Statements in Programming:

Conditional statements in programming are used to execute certain blocks of code based on specified conditions. They are fundamental to decision-making in programs. Here are some common types of conditional statements:

1. If Statement in Programming:

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

C++CJavaC#JavaScriptPython3

```
a = 5
if a == 5:
    print("a is equal to 5")
```

Output

```
a is equal to 5
```

2. if-else Statement in Programming:

The if-else statement is used to execute one block of code if a specified condition is true, and another block of code if the condition is false.

C++CJavaC#JavaScriptPython3

```
a = 10
if a == 5:
    print("a is equal to 5")
else:
    print("a is not equal to 5")
```

Output

```
a is not equal to 5
```

3. if-else-if Statement in Programming:

The if-else-if statement is used to execute one block of code if a specified condition is true, another block of code if another condition is true, and a default block of code if none of the conditions are true.

Python3

```
a = 15
if a == 5:
    print("a is equal to 5")
elif a == 10:
    print("a is equal to 10")
else:
    print("a is not equal to 5 or 10")
```

Output

```
a is not equal to 5 or 10
```

4. Ternary Operator or Conditional Operator in Programming:

In some programming languages, a ternary operator is used to assign a value to a variable based on a condition.

Python3

```
a = 10
print("a is equal to 5" if a == 5 else "a is not equal to 5")
```

Output

a is not equal to 5

Each programming language may have its own syntax and specific variations of these conditional statements.

Looping Statements in Programming:

Looping statements, also known as iteration or repetition statements, are used in programming to repeatedly execute a block of code. They are essential for performing tasks such as iterating over elements in a list, reading data from a file, or executing a set of instructions a specific number of times. Here are some common types of looping statements:

1. For Loop in Programming:

The for loop is used to iterate over a sequence (e.g., a list, tuple, string, or range) and execute a block of code for each item in the sequence.

```
for i in range(5):
    print(i)
```

Output

```
0
1
2
3
4
```

2. While Loop in Programming:

The while loop is used to repeatedly execute a block of code as long as a specified condition is true.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Output

```
0
1
2
3
4
```

4. Nested Loops in Programming:

Loops can be nested within one another to perform more complex iterations. For example, a for loop can be nested inside another for loop to create a two-dimensional iteration.

```
for i in range(2):
    for j in range(2):
        print(f"i={i} j={j}")
```

Output

```
i=0 j=0

i=0 j=1

i=1 j=0

i=1 j=1
```

Each programming language may have its own syntax and specific variations of these looping statements.

Jump Statements in Programming:

Jump statements in programming are used to change the flow of control within a program. They allow the programmer to transfer program control to different parts of the code based on certain conditions or requirements. Here are common types of jump statements:

1. Break Statement in Programming:

The break statement is primarily used to exit from loops prematurely. When encountered inside a loop, it terminates the loop's execution and transfers control to the statement immediately following the loop.

```
for i in range(10):

    if i == 5:

        break

    print(f"{i} ", end="")
```

Output

```
0 1 2 3 4
```

2. Continue Statement in Programming:

The continue statement is used to skip the current iteration of a loop and proceed to the next iteration.

```
for i in range(10):

    if i % 2 == 1:

        continue

    print(f"{i} ", end="")
```

Output

```
0 2 4 6 8
```

3. Return Statement in Programming:

The return statement is used to exit a function and optionally return a value to the caller.