Conditional Statements in Python

Conditional statements in Python allow you to execute code based on certain conditions. These conditions are defined using logical expressions, and the result can be either `True` or `False`. Python provides several types of conditional statements, including `if`, `else`, and `elif`.

1. The `if` Statement

The `if` statement allows you to execute a block of code if a certain condition is true.

Syntax:

```
if condition:
    # Block of code to execute if condition is True
```

2. The `else` Statement

The `else` statement allows you to execute a block of code if the condition in the `if` statement evaluates to `False`.

Syntax:

```
if condition:
    # Block of code to execute if condition is True
else:
    # Block of code to execute if condition is False
```

3. The `elif` Statement

The `elif` statement allows you to check multiple conditions and execute a block of code as soon as one of the conditions is `True`. It is short for "else if".

Syntax:

```
if condition1:
    # Block of code to execute if condition1 is True
elif condition2:
    # Block of code to execute if condition2 is True
else:
    # Block of code to execute if both conditions are False
```

4. Nested if Statements

You can nest `if` statements within another `if` statement. This allows you to check additional conditions only if the outer condition is `True`. Nested `if` statements can be used to create complex conditional logic.

Syntax:

```
if outer_condition:
    # Block of code to execute if outer_condition is True
    if inner_condition:
        # Block of code to execute if inner_condition is True
    else:
        # Block of code to execute if inner_condition is False
else:
    # Block of code to execute if outer_condition is False
```

5. One-Line Conditional Statements

Python allows you to write simple conditional statements in a single line using a shorthand syntax. This is particularly useful for short expressions.

Syntax:

```
# One-Line if Statement
if condition: action

# One-Line if-else Statement
action_if_true if condition else action_if_false
```

Note:

When checking for values:

- **Less Than**: Start from the minimum number and work upwards.
- **Greater Than**: Start from the maximum number and work downwards.

Match Statement in Python (Switch-Case Alternative)

In Python 3.10 and above, the `match` statement is introduced as an alternative to switch-case statements found in other languages. It allows for pattern matching against values, offering a more readable and flexible approach than multiple `if-elif` conditions.

Syntax

```
match subject:
    case pattern1:
        # Code block for pattern1
    case pattern2:
```

```
            # Code block for pattern2
        case _:
            # Default case (when no other pattern matches)
```