If Statements

Introduction

The `if` statement is one of the most fundamental control flow tools in Python. It allows a program to make decisions and execute different blocks of code depending on whether a condition is `True` or `False`.

Basic Syntax

```
if condition:
    # code block
```

The colon : marks the start of the block, and indentation defines the code block. **Example:**

```
x = 5
if x > 0:
    print("x is positive")
```

Adding an `else` Clause

```
if condition:
    # if block
else:
    # else block
```

Example:

```
x = -3
if x > 0:
    print("Positive")
else:
    print("Non-positive")
```

Using `elif`

The keyword `elif` stands for "else if" and allows checking multiple conditions.

Syntax:

```
if condition1:
    # block 1
elif condition2:
    # block 2
else:
    # fallback block
```

Example:

```
x = 0
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

Comparison and Logical Operators

Comparison operators:

- `==` equality
- `!=` not equal
- `>` greater than
- `<` less than
- `>=`, `<=`

Logical operators:

- `and` – both conditions must be true
- `or` – at least one condition must be true
- `not` – inverts the truth value

Example:

```
age = 20
if age >= 18 and age < 65:
    print("Adult")
```

Nested `if` Statements

You can place one `if` statement inside another. Example:

```
x = 42
if x > 0:
    if x % 2 == 0:
        print("Positive even number")
```

Common Pitfalls

- Indentation matters—always use consistent spaces.
- Don't forget the colon `:` after `if`, `elif`, or `else`.
- Conditions must evaluate to `True` or `False`.

Conclusion

The `if` statement is essential for writing decision-making logic in Python. Mastering conditional expressions allows for more flexible and dynamic programs.