

Python If Condition Notes for Beginners

Conditional statements in Python allow you to make decisions based on certain conditions. The most common conditional statement is the **if statement**. It helps control the flow of a program by executing different blocks of code based on conditions.

1. Simple If Statement

The `if` statement executes a block of code only if the given condition is true.

Example:

```
num = 10
if num > 5:
    print("Number is greater than 5")
```

2. If-Else Statement

The `if-else` statement allows you to execute one block of code if the condition is true, and another block if the condition is false.

Example:

```
num = 3
if num > 5:
    print("Number is greater than 5")
else:
    print("Number is less than or equal to 5")
```

3. If-Elif-Else Statement

The `if-elif-else` statement is used when you have multiple conditions to check. Only the first true condition block will execute.

Example:

```
marks = 85
if marks >= 90:
    print("Grade A")
elif marks >= 75:
    print("Grade B")
elif marks >= 50:
    print("Grade C")
else:
    print("Fail")
```

4. Nested If Statement

A nested `if` is an `if` statement inside another `if` statement. It allows checking multiple levels of conditions.

Example:

```
num = 15
```

```
if num > 0:  
    print("Positive number")  
    if num % 2 == 0:  
        print("Even number")  
    else:  
        print("Odd number")  
    else:  
        print("Negative number")
```

5. Using Logical Operators in If Conditions

Logical operators like `and`, `or`, and `not` are used to combine multiple conditions in a single `if` statement.

Example 1: Using 'and'

```
age = 25  
if age > 18 and age < 30:  
    print("Eligible for the program")
```

Example 2: Using 'or'

```
day = "Sunday"  
if day == "Saturday" or day == "Sunday":  
    print("It's weekend!")
```

Example 3: Using 'not'

```
is_raining = False  
if not is_raining:  
    print("You can go outside!")
```

■ **Summary:** - Use `if` to execute a block when a condition is true. - Use `if-else` to handle true/false outcomes. - Use `if-elif-else` for multiple condition checks. - Use nested `if` for complex condition checks. - Combine conditions with logical operators ('and', 'or', 'not'). These concepts form the foundation of decision-making in Python programming.