Why Do We Need Conditional Statements?

Conditional statements are used to make decisions in a program based on specific conditions. They allow the program to execute different blocks of code depending on whether a condition is true or false.

Real-Life Examples:

- If it rains, take an umbrella.
- If the traffic light is green, drive; if it is red, stop.
- If the user enters the correct password, allow login; otherwise, deny access.

Key Insight: Conditional statements change the sequence of program execution based on conditions.

Basic Conditional Statements

Python provides three main conditional statements:

- if
- else
- elif

if Statement

Syntax:

```
if condition:
    # Code to execute if the condition is true
```

Example:

```
traffic_light = "green"
if traffic_light == "green":
    print("You can go")
```

Output:

```
You can go
```

Explanation:

- The condition checks if the value of traffic_light is "green".
- If the condition is true, the message "You can go" is printed.

else Statement

Syntax:

```
if condition:
    # Code to execute if the condition is true
```

else:

```
# Code to execute if the condition is false
```

Example:

```
traffic_light = "red"
if traffic_light == "green":
    print("You can go")
else:
    print("You must stop")
Output:
You must stop
```

elif Statement

Syntax:

```
if condition1:
    # Code to execute if condition1 is true
elif condition2:
    # Code to execute if condition2 is true
else:
    # Code to execute if all conditions are false
```

Example:

```
traffic_light = "yellow"
if traffic_light == "green":
    print("You can go")
elif traffic_light == "yellow":
    print("Slow down and prepare to stop")
else:
    print("You must stop")
```

Output:

Slow down and prepare to stop

Nested Conditions

Nested conditions are used when you need to check multiple conditions within a condition.

Example:

```
car_speed = 60
seat_belt = True
if car_speed <= 60:
    if seat_belt:
        print("You are driving safely")
    else:
        print("Please wear your seat belt")
else:
    print("Slow down, you are exceeding the speed limit")
```

Output:
You are driving safely

Logical Operators in Conditional Statements

Logical operators (and, or, not) are used to combine multiple conditions.

AND Operator

Example:

```
pedestrian_signal = "walk"
is_flashing = False
if pedestrian_signal == "walk" and not is_flashing:
    print("Pedestrian can cross safely")
else:
    print("Pedestrian should wait")
```

Output:
Pedestrian can cross safely

OR Operator

Example:

```
traffic_light = "red"
vehicle_type = “ambulance”
if traffic_light == "green" or vehicle_type == "ambulance":
    print("You can go")
```

```
else:
    print("You must stop")
```

Output:
You can go

NOT Operator

Example:

```
pedestrian_signal = “don't walk”
if not pedestrian_signal == "walk":
    print("Do not cross")
else:
    print("You can cross")
```

Output:
Do not cross

Case Sensitivity in Conditions

Python is case-sensitive, so conditions must match the exact case of the string.

Example:

```
traffic_light = "Red"
if traffic_light == "red":
    print("You must stop")
else:
    print("Slow down and prepare to stop")
```

Output:
Slow down and prepare to stop

Best Practices for Using Conditional Statements

- Use indentation correctly (4 spaces or a tab).
- Keep conditions simple and readable.
- Use logical operators to combine multiple conditions.
- Handle case sensitivity when comparing strings.

Final Thoughts

Conditional statements are a fundamental part of Python programming, allowing you to control the flow of your program based on specific conditions. By mastering if, else, elif, and nested conditions, you can write efficient and logical code. Additionally, understanding logical operators and case sensitivity is crucial for handling complex conditions.

Key Takeaways:

- Use if to check a single condition.
- Use else to handle the opposite of the if condition.
- Use elif to check multiple conditions.
- Use nested conditions for complex decision-making.
- Combine conditions using logical operators (and, or, not).
- Always handle case sensitivity when comparing strings.