\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
**Topic 13 - Nested if Statements**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Introduction**

In Python, if statements are often used to control the flow of a program based on conditions. Sometimes, however, you may encounter scenarios that require evaluating conditions within other conditions. This approach, known as *nesting*, allows for more complex decision structures by placing if statements inside other if, elif, or else statements.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**What is a Nested if Statement?**

A nested if statement is an if statement that appears inside another if (or elif/else) block. With nested if statements, you can perform additional checks within a single decision structure to make nuanced decisions based on multiple layers of conditions.

Example of Nested if Statement:

*if c == d:*

*if x == y:*

*g = h*

*elif a == b:*

*g = h*

*else:*

*e = f*

*else:*

*e = f*

In this code, g = h is assigned if c == d is true and one of the nested conditions is also true. If none of these nested conditions is met, e = f is assigned instead.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Why Use Nested if Statements?**

* Control: Nested if statements allow for more refined control over complex logical conditions.
* Clarity in Complexity: When multiple conditions need to be evaluated in a specific order, nested if statements can provide a clear structure for complex decision trees.
* Handling Multiple Layers: Nested if structures help manage scenarios where a decision depends on several layers of conditions, making it easier to logically separate them.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**How to Use Nested if Statements in Python**

1. Starting with the First-Level Condition  
   Write the top-level if statement as the first condition:

*if c == d:*

# Nested conditions will go here

1. Adding Nested Conditions  
   If the first condition is true, the code proceeds to the next level of nested if statements:

*if c == d:*

*if x == y:*

*g = h*

*elif a == b:*

*g = h*

*else:*

*e = f*

1. Including an else Block  
   If the top-level if condition is false, the outer else handles the alternative action:

*if c == d:*

*# Nested conditions are here*

*else:*

*e = f*

1. Using Indentation for Clarity  
   Proper indentation in nested statements is critical in Python since it distinguishes between different levels of conditions:

*if c == d:*

*if x == y:*

*g = h*

*elif a == b:*

*g = h*

*else:*

*e = f*

*else:*

*e = f*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Conclusion**

While it is sometimes more efficient to use logical operators (and, or) for simpler conditions, nested if statements are valuable for complex scenarios where multiple conditions depend on each other. They enable clear control over intricate decision-making processes, making them a powerful tool for Python programmers.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_