Conditionals (IF Statements)

John Karuitha, PhD

February 27, 2025

1 Conditionals in Python: `if`, `elif`, and Ternary Operator

Conditionals allow programs to execute specific code blocks based on certain conditions. In addition to basic `if` and `else` statements, Python provides other useful tools like `elif` (short for "else if") for checking multiple conditions, and the **ternary operator** for writing concise conditional expressions (Downey 2024; McKinney 2022).

---

2 1. Basic `if` Statement

An `if` statement allows you to execute a block of code if a specific condition evaluates to `True`.

2.1 Syntax:

```
if condition:
    # Code to execute if condition is True
```

2.2 Example:

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

- Since `x = 10` is greater than `5`, the condition evaluates to `True`, and the message `"x is greater than 5"` is printed.

---

3 2. `if-else` Statement

The `if-else` statement allows you to run an alternative block of code if the condition is `False`.

3.1 Syntax:

```
if condition:
    # Code to execute if condition is True
else:
    # Code to execute if condition is False
```

3.2 Example:

```
x = 3
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")
```

- Since `x = 3` is not greater than `5`, the message `"x is not greater than 5"` is printed.

---

4 3. The `elif` Statement

The `elif` statement, short for "else if", allows you to check multiple conditions sequentially. Once Python finds a condition that evaluates to `True`, it will execute the corresponding block of code and ignore the rest.

4.1 Syntax:

```
if condition1:
    # Code to execute if condition1 is True
elif condition2:
    # Code to execute if condition2 is True
else:
    # Code to execute if none of the above conditions are True
```

4.2 Example:

```
x = 7

if x > 10:
    print("x is greater than 10")
elif x > 5:
    print("x is greater than 5 but less than or equal to 10")
else:
    print("x is less than or equal to 5")
```

- In this case, `x = 7` is greater than `5` but less than `10`, so Python prints `"x is greater than 5 but less than or equal to 10"`.
- If no conditions are `True`, the `else` block will be executed.

4.3 How it works:

- Python checks each condition in order. If a condition is `True`, its block is executed, and the rest of the conditions are ignored.
- If none of the `if` or `elif` conditions are `True`, the `else` block is executed (if present).

4.4 Multiple `elif` statements:

You can include as many `elif` statements as needed to check more than two conditions.

```
age = 20

if age < 13:
    print("Child")
elif age < 18:
    print("Teenager")
elif age < 60:
    print("Adult")
else:
    print("Senior")
```

---

5 4. Ternary Operator (Conditional Expression)

The **ternary operator** in Python provides a concise way to write simple `if-else` conditions in a single line. This is also known as a **conditional expression**.

5.1 Syntax:

```
value_if_true if condition else value_if_false
```

- If the `condition` is `True`, the expression evaluates to `value_if_true`.
- If the `condition` is `False`, it evaluates to `value_if_false`.

5.2 Example:

```
age = 18
status = "Adult" if age >= 18 else "Minor"
print(status)
```

- In this example, the condition `age >= 18` is `True`, so the expression evaluates to `"Adult"`, and the output is `"Adult"`.
- If the condition were `False`, the output would have been `"Minor"`.

5.3 Another Example:

```
x = 10
result = "Even" if x % 2 == 0 else "Odd"
print(result)
```

- Here, `x % 2 == 0` checks if `x` is even. If `x` is even, it prints `"Even"`, otherwise it prints `"Odd"`.

---

6 5. Combining `if`, `elif`, and Ternary Operator

You can use `if`, `elif`, and the ternary operator together to write more readable and concise code. While `if-elif-else` statements are better for complex conditions, the ternary operator can be useful for simple conditional checks.

6.1 Example of using `if`, `elif`, and `else`:

```
temperature = 30

if temperature > 35:
    print("It's really hot!")
elif temperature > 25:
    print("It's a warm day.")
else:
    print("It's a bit chilly.")
```

6.2 Example of using Ternary Operator:

```
temperature = 30
status = "Really hot" if temperature > 35 else "Warm" if temperature > 25 else "Chilly"
print(status)
```

- In this case, `temperature = 30` is greater than 25 but not greater than 35, so the output is `"Warm"`. The ternary operator here simplifies the `if-elif-else` structure into a single line.

---

7 6. Best Practices for Using Conditionals

- **Use `elif` for multiple conditions**: When you need to check more than two conditions, `elif` makes your code more readable than nesting multiple `if` statements.
- **Limit the use of ternary operators**: While ternary operators are concise, they can become hard to read when the logic is complex. Stick to the regular `if-elif-else` structure for more intricate conditions.
- **Keep conditions simple**: If your conditions are getting too complicated, break them down into smaller, more manageable parts for better readability.

---

8 Conclusion

Conditionals are a powerful tool in Python, allowing programs to make decisions based on different conditions. With `if`, `elif`, `else`, and the ternary operator, you can control the flow of your program and execute code only when necessary. Whether using the full structure of `if-elif-else` for complex logic or the ternary operator for concise conditional expressions, mastering these tools is essential for writing dynamic and flexible Python programs.

References

Downey, Allen B. 2024. *Think Python.* 3rd ed. Green Tea Press.
McKinney, Wes. 2022. *Python for Data Analysis.* " O'Reilly Media, Inc.".