CSI-160 Python Programming
Python Conditional Statements

Vikas Thammanna Gowda

07/15/2025

Contents

1 Introduction

In programming, we often need to make decisions: "If it's raining, take an umbrella; otherwise, enjoy the sunshine." In Python, if-else statements let your program choose between different actions based on whether a condition is true or false. This ability to control the flow of your code makes your programs dynamic and responsive. Conditional statements in python are of 4 types

- if statement
- if else statement
- if elif statement
- Nested if else

2 if Statement

Syntax:

```
if (condition):
    <statement block>
```

The Anatomy

- `if` **-** Keyword
- **condition -** an expression that evaluates to `True` or `False`
- **colon (:) -** marks the start of the block
- **statement block -** one or more indented lines that run only if the condition is `True`

if statement is used to run a statement conditionally i.e. if given condition is `True` then only the statement given in if block will be executed.

2.1 Illustration

Example:

```
temperature = 85

if (temperature > 80):
    print("It's hot outside!")
```

Output:

```
It's hot outside!
```

- Here, `temperature > 80` is the condition.
- The condition translates to `85 > 80`, which is `True`.
- `print` line runs because the condition is `True`

3 if else Statement

Syntax:

```
if (condition):
    <statement block 1>
else:
    <statement block 2>
```

The Anatomy

- **if** - Keyword
- **else** - Keyword
- **condition** - an expression that evaluates to `True` or `False`
- **colon (:)** - marks the start of the block
- **statement block** - one or more indented lines that run only if the condition is `True`

if else statement is used to run any one statement conditionally i.e. if given condition is `True` then the statement given in if block will be executed, and if given condition is `False` then the statement given in else block will be executed.

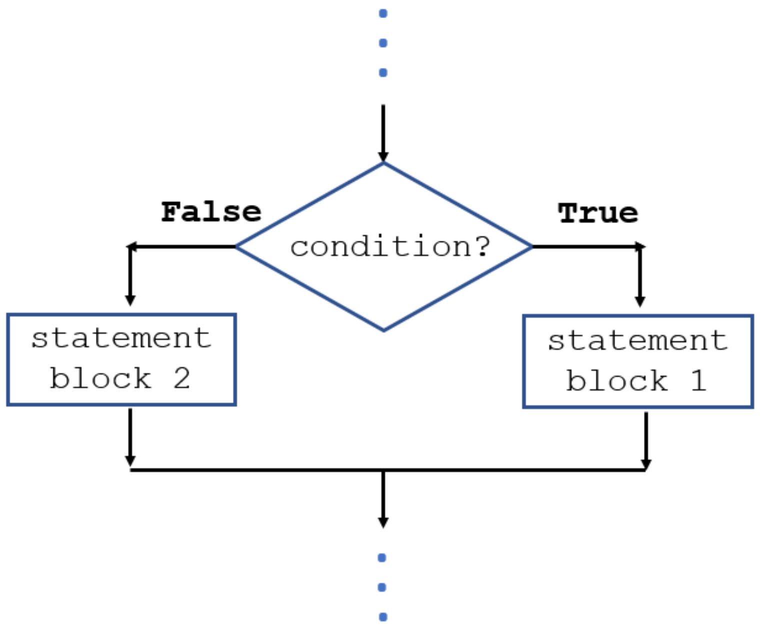


Figure 1: Flow chart for `if else`

3.1 Illustration

Example 1:

```
temperature = 75

if (temperature > 80):
    print("It's hot outside!")
else:
    print("The weather is comfortable.")
```

Output:

```
The weather is comfortable.
```

- Here, `temperature > 80` is the condition.
- The condition translates to `75 > 80`, which is `False`.
- `print` statement under `else` will execute.

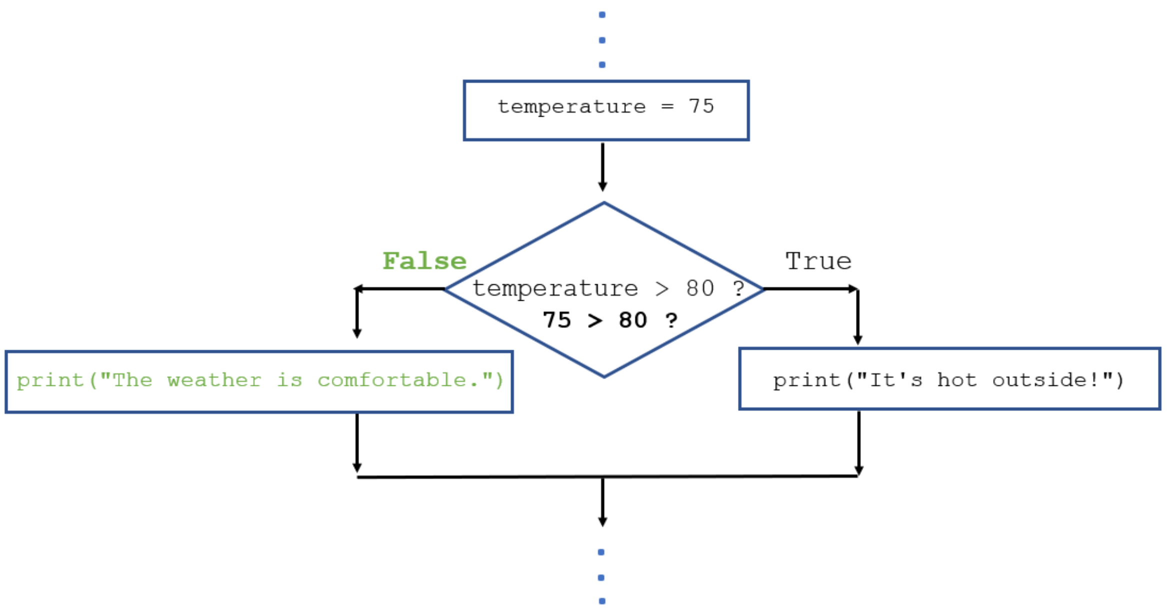

temperature = 75
False
True
temperature > 80 ?
75 > 80 ?
print("The weather is comfortable.")
print("It's hot outside!")

Figure 2: Flow chart for Example 1

Example 2:

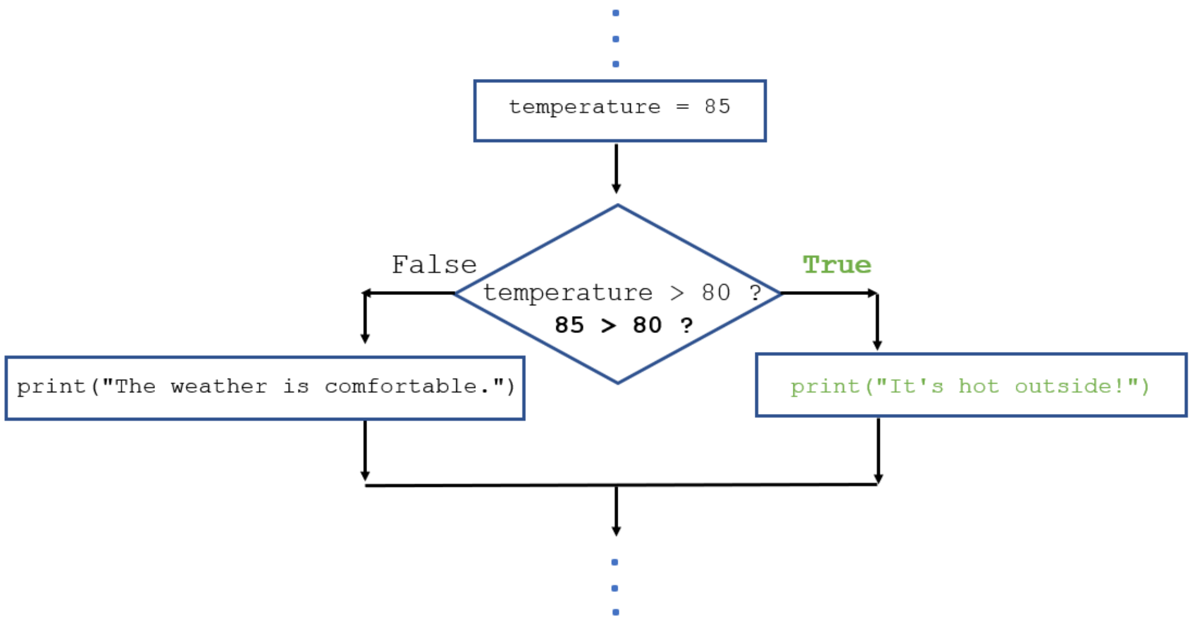
```
temperature = 85

if (temperature > 80):
    print("It's hot outside!")
else:
    print("The weather is comfortable.")
```

Output:

```
It's hot outside!
```

- Here, `temperature > 80` is the condition.
- The condition translates to `85 > 80`, which is `True`.
- `print` statement under `if` will execute.



temperature = 85
False
True
temperature > 80 ?
85 > 80 ?
print("The weather is comfortable.")
print("It's hot outside!")

Figure 3: Flow chart for Example 2

4 if elif Statement

Syntax:

```
if (condition 1):
    <statement block 1>
elif (condition 2):
    <statement block 2>
elif (condition 3):
    <statement block 3>
.
.
.
else:
    <statement block last>
```

if elif statement is used when you have more than two possibilities.

- Python evaluates each condition in order.
- As soon as one is `True`, its block runs and the rest are skipped.
- if none of the conditions are `True`, `else` block will execute.

4.1 Illustration

Example:

```
temperature = 50

if (temperature > 80):
    print("It's hot!")
elif (temperature >= 60):
    print("It's nice and warm.")
elif (temperature >= 40):
    print("It's a bit chilly.")
else:
    print("Brrr... it's cold!")
```

Output:

```
It's a bit chilly.
```

- Here, first the condition `temperature > 80` is checked.
- The condition translates to `50 > 80`, which is `False`.
- The execution moves on to check the next condition `temperature >= 60`
- The condition translates to `50 >= 60`, which is `False`.
- The execution moves on to check the next condition `temperature >= 40`
- The condition translates to `50 >= 40`, which is `True`.
- `print` statement under this condition will execute.

5 Nested if else

You can put an `if-else` structure inside another to handle more complex logic

5.1 Illustration

Example:

```
score = 85

if (score >= 60):
    if (score >= 90):
        print("Grade: A")
    else:
        print("Grade: B, C, or D depending on exact score.")
else:
    print("Student failed.")
```

Output:

```
Grade: B, C, or D depending on exact score.
```

- First, check if the student passed `score >= 60`.
- The condition translates to `85 >= 60`, which is `True`.
- Now, it is passes to check the other if condition.
- Check the condition `score >= 90`.
- The condition translates to `85 >= 90`, which is `False`.
- `print` statement under else will execute.

6 Key Points for Conditional Statements in Python

- **Indentation matters:** Python uses spaces (or tabs) to group statements under `if`, `elif`, and `else`.
- **Boolean conditions:** Conditions must result in `True` or `False`.
- **Comparison operators:** Use `==`, `!=`, `<`, `>`, `<=`, `>=` to form conditions.
- **Logical operators:** Combine conditions with `and`, `or`, `not`.
- **Flow of execution:**
  1. Evaluate the `if` condition.
  2. If false, check each `elif` in order.
  3. If none match, execute the `else` block (if present).