ScopeTreasure Courses

PYTHON COURSES · LESSON PY.6 · CODE SCOPE SERIES

If, Else and Decisions

Programs make decisions with if, elif and else — learn conditions, the difference between = and ==, and combining tests with and, or and not.

Introduction

So far, every program ran every line from top to bottom. Real programs do not work like that. A program must decide: if the marks are 50 or more, print "Pass"; otherwise, print "Fail". If the user is logged in, show the page; otherwise, show a login message. Decisions are what make programs feel smart.

Python makes decisions with `if`, `elif` (short for "else if") and `else`. You write a **condition** — a test that is either `True` or `False` — and Python runs only the block that matches. The block is the indented lines under the condition. Indentation is not style here; it is how Python knows which lines belong to which decision.

In this lesson you will write your first decisions, chain several conditions with `elif`, and combine tests with `and`, `or` and `not`.

(Roman Urdu: If ka matlab hai "agar" — agar shart poori ho to aik kaam karo, warna doosra. Indentation batati hai ke kaun si lines kis shart se judi hain.)

Decision code goes wrong in two quiet ways. First, a condition that is almost right: `if age == 18 or 20:` looks sensible in English but means something else entirely in Python, and it is True for everyone. Second, an `elif` chain checked in the wrong order, so a loose test catches everyone before the strict test is ever reached. Read each condition out loud as Python reads it, not as a person would.

Key points

- **A condition is a test with a True or False answer.** Examples: `marks >= 50`, `age == 18`, `name == "Ali"`.
- **Comparison operators:** `==` (equal), `!=` (not equal), `>` (greater), `<` (smaller), `>=` (greater or equal), `<=` (smaller or equal).
- **`=` stores, `==` compares.** This is the most common beginner mix-up. `age = 20` stores 20; `age == 20` asks "is age 20?".
- **`if` runs its block only when the condition is True.** Otherwise Python skips to the next part.
- **`else` catches everything the `if` did not catch.** It has no condition of its own.
- **`elif` chains several conditions in order.** Python checks them one by one and runs only the first block that is True.
- **`and` needs both tests True.** `or` needs at least one test True. `not` flips True to False and False to True.
- **Indentation defines the block.** Every line that belongs to an `if`, `elif` or `else` must be indented by the same amount, usually 4 spaces.

- **A colon `:` ends the condition line.** Forgetting it is a common syntax error.
- **Chained comparisons read naturally.** `75 <= marks < 90` means "marks is at least 75 and less than 90" — Python checks both sides.
- **Nested `if` means a decision inside a decision.** Indent one more level, and the inner test only runs when the outer test already passed.

Examples

Example 1 — Pass or fail

```
marks = 65

if marks >= 50:
    print("Pass")
else:
    print("Fail")
```

What you will get:

```
Pass
```

Change `marks` to 40 and run it again — you will get `Fail`, because the condition `marks >= 50` is now False and Python runs the `else` block instead.

Example 2 — Grades with an elif chain

```
marks = 82

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

What you will get:

```
Grade: B
```

Python checks from the top: 82 is not 90 or more, so it skips the first block; 82 is 75 or more, so it runs the second block and then skips the rest. Only one block ever runs in a chain.

Example 3 — Combining conditions with and, or and not

```
age = 20
has_ticket = True

if age >= 18 and has_ticket:
    print("You can enter")

if age < 18 or not has_ticket:
    print("You cannot enter")

day = "Sunday"
if day == "Saturday" or day == "Sunday":
    print("It is the weekend")
```

What you will get:

```
You can enter
It is the weekend
```

The second `if` is False here (age is 20 and the ticket exists), so its message does not print. The weekend test is True because `day` equals `"Sunday"`, even though it does not equal `"Saturday"` — `or` needs only one side True.

Example 4 — A nested decision: teen with or without ID

Sometimes one decision depends on another. A teenager may enter only with an ID, while an adult enters freely and a child does not enter at all. Indentation shows the nesting:

```
age = 16
has_id = False
if age >= 18:
    print("Adult")
elif age >= 13:
    if has_id:
        print("Teen with ID")
    else:
        print("Teen without ID")
else:
    print("Child")
```

What you will get:

```
Teen without ID
```

Age 16 reaches the `elif`, and inside it the inner `if` checks the ID. Change `has_id` to `True` and run it again to see the other branch. Keep nesting shallow — two levels is plenty; deeper nesting gets hard to read.

Example 5 — Ranges with chained comparisons

Grades are ranges, and Python lets you write a range the way you say it: "between 75 and 90":

```
marks = 85
if 75 <= marks < 90:
    print("Grade: B")
temperature = 35
if temperature > 30:
    print("It is hot")
elif temperature > 20:
    print("It is pleasant")
else:
    print("It is cold")
```

What you will get:

```
Grade: B
It is hot
```

`75 <= marks < 90` is exactly the same as `marks >= 75 and marks < 90`, but shorter and harder to get wrong. Notice the temperature chain is also order-sensitive: the strictest test (`> 30`) comes first.

Example 6 — Real-world case: a shop discount

A shop gives a 10% discount to students, and to any bill of 5000 or more. The program checks both conditions, works out the saving, and prints what the customer actually pays:

```
bill = 6000
is_student = True
if bill >= 5000 or is_student:
    print("Discount: 10%")
    print("You pay:", bill - bill * 10 / 100)
else:
    print("No discount")
```

What you will get:

```
Discount: 10%
You pay: 5400.0
```

The bill is 6000, 10% of it is 600, and 6000 − 600 = 5400. Change `bill` to 3000 and `is_student` to `False`, and the program correctly prints "No discount" — test both paths, not just the happy one.

Quick recap

- `if` runs a block when its condition is True; `else` runs when it is not.
- `elif` chains several conditions; only the first True block runs.
- `==` compares; `=` stores. Never mix them up.
- `and` needs both True; `or` needs one True; `not` flips a value.
- Indentation and the colon are part of the syntax, not decoration.
- Chained comparisons like `75 <= marks < 90` test a range in one line.
- A nested `if` runs only when the outer condition already passed.

Common mistakes

1. **Writing `=` instead of `==` in a condition** — `if age = 18:` is an error. Fix: always use `==` when comparing.
2. **Forgetting the colon** — `if marks >= 50` without `:` gives a syntax error. Fix: end every `if`, `elif` and `else` line with a colon.
3. **Forgetting the indent** — an unindented line after `if` is not part of the block and will either error or run always. Fix: indent the whole block by 4 spaces.
4. **Checking conditions in the wrong order** — in a grade chain, testing `>= 50` before `>= 90` means everyone passes the first test and never reaches A. Fix: test the strictest (highest) condition first.
5. **Writing `if age == 18 or 20:`** — Python reads this as two separate tests, and `20` alone is always truthy, so the test is always True. Fix: write the full test twice: `if age == 18 or age == 20:`.

1. **Comparing text with a number** — `if "50" >= 50:` gives a TypeError, because Python will not compare a string with a number. Fix: convert first, for example `if int("50") >= 50:`.
2. **Giving `else` a condition** — `else marks < 50:` is an error; `else` catches everything left over and takes no test of its own. Fix: use `elif marks < 50:` if you need a test, or plain `else:`.

Practice

1. Write a program that asks for marks in a variable and prints "Pass" or "Fail" using 50 as the pass mark. Test it with 45 and with 75.
2. Extend it into a grade chain: A for 90+, B for 75+, C for 50+, F below 50. Test it with 92, 80, 60 and 30.
3. Write a test for a discount: a shopper gets a discount if their age is under 18 OR their total bill is over 5000. Test three different shoppers.
4. Deliberately write `if age = 20:` and read the error. Then fix it to `==` and write down, in one line, what the error taught you.
5. Ticket prices: under 12 pays 100, 12 to 59 pays 300, 60 and over pays 150. Write the chain, and test it with ages 8, 30 and 65.

6. Write a nested decision: if a number is positive, check inside whether it is even or odd; if it is not positive, print "Not positive". Test with 7, 8 and -3.
7. A library fines a late book 20 rupees per day, but students pay half. Write a program with variables for days late and student status, and print the fine for 5 days late, student and non-student.

Practice quiz

1. What is the difference between = and == in Python?

- No difference
- = stores a value, == compares two values
- == stores a value, = compares
- Both are for maths only

Answer: = stores a value, == compares two values

2. What does elif do?

- Ends the program
- Checks the next condition when the earlier one was false
- Repeats the code
- Asks the user a question

Answer: Checks the next condition when the earlier one was false

3. When is "A and B" true?

- When either one is true
- Only when both A and B are true
- When one is false
- Never

Answer: Only when both A and B are true

Watch the video lesson: https://youtu.be/o3AsaGg11qE

Read online: https://scopetreasure.com/notes/py-if-else · More free tools and notes at https://scopetreasure.com