

PYTHON KEYWORDS

break Exercises

Python Keywords

REFERENCE · BREAK · EXERCISES

💻 Practice: Leaving Loops Early

The companion exercise set for the `break` reference page. Ten exercises that use `break` in the situations that page covers: stopping a search at the first match, `while True` loops with one or two exits, the loop `else`, nested loops, a menu, a retry loop and a safety limit. Try each one yourself before opening its solution. Every solution was run, and its sample output is the real output.

Exercise 1: Find the First Freezing Reading

Given `temps = [4, 7, 2, -3, 5, -8, 1]`, print the **first** reading below zero together with its position in the list. If there is no such reading, print `No freezing readings`. Test it with a list that has no negative numbers too, such as `[4, 7, 2]`.

Goal: Stop a `for` loop at the first match with `break`, and use the loop's `else` block for the "not found" case, without a flag variable.

→ Solution

Exercise 2: Guessing Game With Hints, Attempts and a Quit Option

Build a number-guessing game for 1 to 100 using `while True:`. Pick the secret number with `random.randint`, give "Too low!" or "Too high!" hints, and count the guesses so the win message can say `You got it in N tries!`. Typing `q` gives up: print the secret number and stop.

Goal: Use **two different** `break` statements to leave the same loop for two different reasons, and put the counter in the right place so the winning guess is counted.

→ Solution

Exercise 3: Three Password Attempts

Ask for a password up to **three** times. On a correct password (`open sesame`) print `Access granted` and stop asking. If all three attempts are wrong, print `Too many attempts. Account locked.`, but only in that case. Run it once with a correct answer and once with three wrong ones.

Goal: Combine a `for` over `range(1, 4)` (the end is exclusive), a `break` on success and a loop `else` for failure.

→ Solution

Exercise 4: Is It Prime? Stop at the First Divisor

For each number in `[2, 15, 17, 21, 97, 100]`, print whether it is prime. For a number that is not prime, also print the first factor pair you find (such as `15 is not prime (3 x 5)`). Only test divisors from 2 up to the square root of the number.

Goal: Put a `break` inside an *inner* loop, so it ends only that loop and the outer loop moves on to the next number, and use the inner loop's `else` to mean “no divisor found”.

→ Solution

Exercise 5: Average of Numbers Until the User Types done

Keep asking for numbers until the user types `done`. Then print how many numbers were entered, their sum and their average (to two decimal places). If `done` is typed straight away, print `No numbers entered` and do not divide by zero.

Goal: The classic sentinel loop: `while True` with a `break` on a special value, followed by a check that protects the calculation after the loop.

→ Solution

Exercise 6: The First Repeated Word

Given `sentence = "The cat saw the dog and the bird"`, find the first word that appears for the second time (ignoring upper and lower case) and print it together with how many different words came before it. If no word repeats, print `No repeated word`.

Goal: Stop scanning the moment the answer is known, and keep a variable that records what the loop found so the code after it can report either outcome.

→ Solution

Exercise 7: Find a Pair That Adds Up, Leaving Both Loops

Given `numbers = [8, 3, 11, 5, 2, 9]` and `target = 14`, find the first pair of different positions whose values add up to the target and print the pair. Write it **twice**: once with a variable that the outer loop checks after the inner loop, and once as a function that uses `return`.

Goal: A `break` leaves only one loop. Practise the two standard ways of leaving two nested loops at once, and compare them.

→ Solution

Exercise 8: A Shopping-List Menu

Write a menu that repeats until the user quits: `1` adds an item (ask for its name), `2` shows the list numbered from 1 (or `(empty)`), `3` quits, and anything else prints `Please choose 1, 2 or 3`. When the loop ends, print how many items were added.

Goal: Use `break` as the only exit from a `while True` menu, and notice which loop each `break` belongs to when a `for` loop sits inside the menu loop.

→ Solution

Exercise 9: Retry a Flaky Call

A function `fetch(attempt)` raises `ConnectionError` on attempts 1 and 2 and succeeds from attempt 3. Call it in a loop of at most `MAX_ATTEMPTS = 5` tries: print each failure, stop at

the first success and print the result, and print `Giving up after N attempts` if every attempt fails. Then change `MAX_ATTEMPTS` to 2 and check the give-up message.

Goal: Use `try` / `except` inside a `for` loop, with `break` after the successful call and a loop `else` for giving up. This is the shape of real retry code.

→ Solution

Exercise 10: How Long Until the Money Doubles?

Ask for a starting balance and a yearly interest rate (in percent). Add a year of interest at a time until the balance has at least doubled, then print the number of years and the balance to two decimal places. Add a safety limit: if 50 years pass without doubling, stop and say so. Try rates of 7 and 1.

Goal: Use `while True` with **two** `break` conditions, one for the goal and one as a safety net so the loop can never run forever.

→ Solution