

PYTHON KEYWORDS

continue Exercises

Python Keywords

REFERENCE · CONTINUE · EXERCISES

💻 Practice: Skipping to the Next Pass

The companion exercise set for the `continue` reference page. Ten exercises that use `continue` in the situations that page covers: filtering out unwanted items, guard clauses, validation loops, the `while`-loop counter trap, `try` / `except`, batch records, nested loops, and `continue` working together with `break`. Try each one yourself before opening its solution. Every solution was run, and its sample output is the real output.

Exercise 1: Add Up Only the Positive Numbers

Given `readings = [3, -1, 8, -5, 4, 0, -2]`, add up only the numbers greater than zero and print the total. Print each number that is counted (`counted 3`) and each one that is ignored (`ignored -1`), so you can see the loop working.

Goal: Use `continue` to dismiss unwanted items early, so the line that does the real work (`total += n`) stays at the top indentation level of the loop body with no `if` around it.

→ Solution

Exercise 2: Count Without the Multiples of 3 or 5

Print the numbers from 1 to 20 on a single line, separated by spaces, leaving out every number that is a multiple of 3 or a multiple of 5. Then print how many numbers were left out.

Goal: Practise a `continue` with a combined condition (`or`), counting what is skipped as well as what is shown, and printing on one line with `end=" "`.

→ Solution

Exercise 3: Parse a Settings File and Report What Was Skipped

Given a list of text lines (some blank, some comments starting with `#`, some malformed with no `=`), build a dictionary of `key` to `value` from the valid lines. Ignore blank lines and comments silently, but report each malformed line by its line number. Print the dictionary at the end.

Goal: Use several `continue` statements in a row as guard clauses, each dealing with one kind of bad line, so the code that handles a good line has no nesting.

→ Solution

Exercise 4: Ask Again Until the Age Is Valid

Ask for an age with `input()`. If the text is not made of digits, or the number is not between 0 and 120, print an error and ask again. When the age is valid, print `Age recorded: N` and stop asking.

Goal: Combine `continue` (reject and ask again) with `break` (accept and leave) in a `while True` validation loop.

→ Solution

Exercise 5: The Odd-Number Sum That Must Not Loop Forever

Using a `while` loop (not `for`) and `continue`, add up the odd numbers from 1 to 99 and print the total. The loop must skip the even numbers with `continue`. Then explain, in a comment, what would happen if the counter were updated at the *end* of the loop body instead of the start.

Goal: Learn the trap: in a `while` loop, a `continue` skips everything after it, including the line that moves the loop variable on. Place the update *before* the `continue`.

→ Solution

Exercise 6: Total a List of Messy Values

Given `raw = ["12", "x", "7.5", "", "3", "1e2", "n/a"]`, convert each item to a float and add it to a total. If an item cannot be converted, skip it and remember it. At the end print the total and the list of skipped items.

Goal: Use `try` / `except ValueError` with `continue`, so the code after the `try` only ever sees values that converted successfully.

→ Solution

Exercise 7: Guessing Game: Typos and Repeats Do Not Cost an Attempt

Build a guessing game for 1 to 20 with five attempts, using `while`. Text that is not a whole number from 1 to 20 must be rejected with a message, and so must a number that was already guessed; neither of these may use up an attempt. A correct guess ends the game with the number of attempts used. If the attempts run out, say so.

Goal: Use two `continue` statements and one `break` in the same loop, and put the attempts counter after both checks so rejected input is never counted.

→ Solution

Exercise 8: Load Records, Skipping and Reporting the Bad Ones

Each string in `rows` should look like `name,age,city`. Build a list of `(name, age, city)` tuples from the valid rows, with the age as an integer. A row is skipped if it does not have exactly three fields or if its age is not a whole number. Print the good records, then one error message per bad row that includes the row number.

Goal: Practise batch processing: one bad row must never stop the others, so each problem is recorded and the loop carries on with `continue`.

→ Solution

Exercise 9: A Grid Without the Diagonal

Print the coordinates of every cell in a 4 by 4 grid as `(row,col)`, one row per line, but leave out the cells on the diagonal (where row and column are equal). Each row of output must end with a line break, including rows that lose a cell.

Goal: See that `continue` in an *inner* loop skips only one cell and leaves the outer loop completely unaffected.

→ Solution

Exercise 10: Numbered Long Words Until stop

Given `sentence = "the quick brown fox and a lazy dog stop jumps never"`, go through the words in order. Ignore every word shorter than 4 letters, number the remaining words starting from 1, and print each as `1. quick`. As soon as the word `stop` appears, end the loop; the words after it are not processed. Finish by printing how many words were numbered.

Goal: Use `continue` and `break` together, and decide where the counter belongs: it must only count words that are actually printed.

→ Solution