Loops (for / while / for-in)

Repeat work without copy-paste: the for loop, the while loop, for-in over a list, and break vs continue.

Introduction

Nobody writes `print` a hundred times. When work repeats — a table of 2, a countdown, every name in a list — programmers write a **loop**: a block of code Dart runs again and again until a condition says stop. This lesson gives you Dart's three loops, the rule that keeps a loop from running forever, and the two escape words, `break` and `continue`. *(Roman Urdu: Loop wohi kaam baar baar karta hai — aap aik dafa likhte hain, Dart sau dafa chalata hai.)*

Key points

- The **`for` loop** is for a *known count* of rounds. Its head has three parts: **start** (`int i = 1`), **condition** (`i <= 10` — the loop runs while this is true), **update** (`i++` — add 1 after every round).
  (Roman Urdu: for loop ki teen cheezein — kahan se shuru, kab tak, har dafa kitna barhe.)
- The **`while` loop** is for an *unknown count*: it checks the condition first, then runs, again and again. If the condition never becomes false, the loop **never stops** — this is the famous infinite loop, and it freezes your program.
- **The golden rule of while:** something inside the loop must move the condition towards false (like `i--` or `i++`). Forget the update and the loop runs forever.
- **`for-in`** walks through a list item by item — no counting, no indexes: `for (String name in names) { ... }`. Use it whenever you have a list (Lesson D.7) and want every item once.
- **`break`** leaves the loop completely, right now. **`continue`** skips only the *rest of this round* and jumps to the next one. Break = exit the room; continue = skip this turn, stay in the room.
- Loops are where interpolation shines: `print('2 x $i = ${2 * i}')` prints a fresh line every round because `i` changes every round.

Code examples

Example 1 — The for loop prints the table of 2

```
void main() {
  for (int i = 1; i <= 10; i++) {
    print('2 x $i = ${2 * i}');
  }
  print('Table khatam!');
}
```

What it does: `i` starts at 1. Dart checks `i <= 10` (true), prints the line, then `i++` makes it 2. This repeats until `i` becomes 11 — the check fails, the loop ends, and the line after the loop prints.

What you will see: ten lines — `2 x 1 = 2` up to `2 x 10 = 20` — then `Table khatam!`. You wrote one print line; the loop produced ten. That is the whole idea.

Example 2 — while: a countdown that cannot run forever

```
void main() {
  int i = 5;

  while (i > 0) {
    print('Counting: $i');
    i--; // yeh line bhool gaye to loop kabhi nahi rukega
  }
  print('Start!');
}
```

What it does: The condition `i > 0` is checked before every round. Inside, `i--` makes the number smaller each time (5, 4, 3, 2, 1) until `i` becomes 0 — the condition turns false and the loop stops by itself.

What you will see: `Counting: 5` down to `Counting: 1`, then `Start!`. Delete the `i--;` line and run it in DartPad to feel the infinite loop once — then bring the line back. Every programmer learns this lesson exactly this way.

Example 3 — for-in with break and continue

```
void main() {
  List<String> names = ['Ali', 'Sara', 'Bilal', 'Zara'];

  for (String name in names) {
    if (name == 'Bilal') {
      continue; // Bilal ko skip — agle naam par jao
    }
    if (name == 'Zara') {
      break; // Zara aate hi loop band
    }
    print(name);
  }
  print('Loop khatam');
}
```

What it does: The loop picks each name in order. Ali prints. Sara prints. Bilal hits `continue`, so his print is skipped. Zara hits `break` *before* her print line, so she never prints and the loop ends at once.

What you will see: three lines — `Ali`, `Sara`, `Loop khatam`. No Bilal (skipped by continue), no Zara (stopped by break). The position of `break` and `continue` inside the loop decides everything.

Quick recap

- `for` = known rounds (start / condition / update in one head).
- `while` = unknown rounds; you MUST update the condition's value inside, or it never stops.
- `for-in` = every item of a list, one by one, no indexes.
- `break` leaves the loop; `continue` skips only this round.

Common mistakes

1. **Forgetting the update in a while loop** — `i--` or `i++` missing means the condition stays true forever and the program hangs. In DartPad, use the stop/reset and fix the line.
2. **Off-by-one in for** — `i < 10` runs 1 to 9 (nine rounds), `i <= 10` runs 1 to 10. When the count looks one short, this is the first place to look.
3. **Putting break where continue belongs** — if you want to *skip one item and keep going*, `break` is wrong; it kills the whole loop. Ask: "rukna hai ya sirf skip karna hai?" — skip means `continue`.

Practice

1. Use a for loop to print the table of 7 (7 x 1 to 7 x 10). Then print the same table backwards (10 down to 1) by changing the start, condition and update.

2. Make a list of 5 student names and print only the names longer than 4 letters, using for-in and an if inside (hint: `name.length` tells you the length).

Practice quiz

1. When do you use a for loop?

- When the number of rounds is unknown
- When the number of rounds is known, with start, condition and update in one head
- Only for lists, never for counting
- Only inside functions

Answer: When the number of rounds is known, with start, condition and update in one head

2. What happens if a while loop never updates its condition?

- It stops after 10 rounds by itself
- It never stops, an infinite loop that freezes the program
- It skips to the next line
- It prints an error and continues

Answer: It never stops, an infinite loop that freezes the program

3. What is the difference between break and continue in a loop?

- They are the same
- break leaves the loop completely, continue skips only this round and goes to the next
- break skips one round, continue leaves the loop
- break works only in for loops

Answer: break leaves the loop completely, continue skips only this round and goes to the next

---

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

Read online: https://scopetreasure.com/notes/dart-loops · More free tools and notes at https://scopetreasure.com