**ScopeTreasure Courses**

DART COURSES · LESSON D.7 · CODE SCOPE SERIES

# Lists, Sets & Maps

Three ways to hold many values: List (ordered), Set (no duplicates) and Map (key → value) — with add, remove and lookup.

## Introduction

One variable holds one value — but real data comes in crowds: 40 students, 200 products, a whole class's marks. Dart gives you three **collections** for crowds, and each has a personality: the **List** keeps items in order, the **Set** refuses duplicates, and the **Map** stores every value under a name (a key) so you can fetch it instantly. Almost every app screen you will ever build shows data from one of these three. *(Roman Urdu: List = tartib wali line, Set = duplicate-mukt dabba, Map = naam ke sath value.)*

## Key points

- A `List` is an ordered row of values of one type: `List<String> names = ['Ali', 'Sara'];`. You reach an item by its **index** — and counting starts at **0**, so the first item is `names[0]`, the second is `names[1]`.
- List tools you will use daily: `length` (how many), `add(x)` (one more at the end), `remove(x)` (take this value out), `contains(x)` (is it here? true/false). A list pairs perfectly with the `for-in` loop from Lesson D.5.

  *(Roman Urdu: List mein ginti 0 se shuru hoti hai — pehli cheez hamesha [0] hai.)*
- A `Set` is written with `{ }` and keeps every value **only once**: `{5, 12, 5}` quietly becomes `{5, 12}`. There is no order promise and no index — a Set answers questions like `contains`, it does not line items up.
- A `Map` stores **key** → **value** pairs: `Map<String, int> marks = {'Ali': 92};`. Fetch by key: `marks['Ali']` gives 92. Add or change a pair with the same move: `marks['Bilal'] = 78;`.
- A Map lookup for a **missing key gives** `null` — not a crash, not zero: `null`, meaning "nothing here". That is your first meeting with null; Lesson D.9 is fully about handling it safely (for example with `??`).
- Which one do I pick? Order matters and duplicates are fine → **List**. Duplicates must be impossible → **Set**. Every value needs a name or ID to fetch it by → **Map**.

## Code examples

### Example 1 — The List: order, index, add, remove

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

  print(students.length); // 3
  print(students[0]); // Ali — ginti 0 se shuru
  print(students[2]); // Bilal

  students.add('Zara');
  students.remove('Bilal');
  print(students); // [Ali, Sara, Zara]
}
```

What it does: The list starts with three names. Index 0 fetches the first, index 2 the third. `add` puts Zara at the end, `remove` takes Bilal out by value, and printing the list itself shows the whole row.

**What you will see:** five lines — `3`, `Ali`, `Bilal`, and finally `[Ali, Sara, Zara]`. If you try `students[3]` *before* adding Zara, Dart throws a RangeError: index 3 does not exist in a 3-item list whose last index is 2. Index mistakes are the most common collection bug — always think "last index = length − 1".

### Example 2 — The Set: duplicates quietly disappear

```
void main() {
  Set<int> rollNumbers = {5, 12, 5, 9, 12};

  print(rollNumbers); // {5, 12, 9} — duplicate gayab
  print(rollNumbers.contains(9)); // true
  print(rollNumbers.contains(7)); // false

  rollNumbers.add(20);
  print(rollNumbers.length); // 4
}
```

What it does: You *wrote* five numbers, but the Set kept only the three different ones — the second 5 and second 12 were refused silently. `contains` answers membership questions, and `add` grows the set with a genuinely new value.

**What you will see:** four lines — `{5, 12, 9}`, `true`, `false`, `4`. Use a Set whenever "the same thing twice" would be a bug: roll numbers, usernames, booked seat numbers.

**Example 3 — The Map: every value has a name**

```
void main() {
  Map<String, int> marks = {
    'Ali': 92,
    'Sara': 85,
  };

  print(marks['Ali']); // 92
  marks['Bilal'] = 78; // naya pair add
  print(marks.length); // 3

  int? zaraMarks = marks['Zara'];
  print(zaraMarks ?? 'Not found'); // Not found
}
```

What it does: Keys are names, values are marks. `marks['Ali']` fetches by name — no counting, no index. Writing `marks['Bilal'] = 78` adds a brand-new pair. The last part asks for Zara, who is not in the map: the lookup gives `null`, the variable is typed `int?` (it is *allowed* to be empty — D.9 explains), and `??` prints 'Not found' instead of the null.

**What you will see:** three lines — `92`, `3`, `Not found`. A list of maps like this — one map per student — is exactly the data shape behind the Flutter project in Lesson F.5.

## Quick recap

- List: ordered, index from 0, duplicates allowed — `add`, `remove`, `length`, `contains`.
- Set: `{ }`, every value kept once, no index — perfect for "no duplicates" data.
- Map: key → value pairs, fetch by name — `marks['Ali']`. Missing key → `null`.
- Last index of a list = `length - 1`. Most collection bugs live right there.

## Common mistakes

1. **Counting list items from 1** — `students[1]` is the *second* student. Dart, like almost every language, counts from 0. When the wrong name prints, check the index first.
2. **Expecting a Set to keep order or duplicates** — a Set makes no order promise and drops repeats on purpose. If order or repeats matter to you, you wanted a List.
3. **Trusting a Map lookup blindly** — `marks['Zara']` is `null` when Zara is absent, and printing or calculating with that null surprises beginners. Always decide what happens when the key is missing: check it, or give a fallback with `??`.

## Practice

1. Make a list of 5 fruit names. Print the first and the last (use `length - 1` for the last index), add one more fruit, remove one, and print the final list.
2. Build a `Map<String, String>` of 4 friends with their cities. Print one friend's city by key, then look up a name that is not in the map and print a friendly fallback message using `??`.

## Practice quiz

**1. What is the index of the first item in a Dart list?**

- 1
- 0
- -1
- It has no index

**Answer:** 0

**2. What does the Set {5, 12, 5} become?**

- {5, 12, 5}, sets keep duplicates
- {5, 12}, every value is kept only once
- {5, 5, 12}
- An error, sets cannot hold numbers

**Answer:** {5, 12}, every value is kept only once

**3. What does a Map lookup for a missing key give?**

- 0
- An empty string
- null, meaning nothing here
- An error and a crash

**Answer:** null, meaning nothing here

---

**Watch the video lesson:** https://youtu.be/AD0Y5SYrmqM

**Read online:** https://scopetreasure.com/notes/dart-lists-sets-maps · More free tools and notes at https://scopetreasure.com