Japanese postal code

7-digit Japanese postal code, formatted NNN-NNNN.

```
^\d{3}-\d{4}$
```

Token-by-token breakdown

Token	Meaning
^	start of string (or line in multiline mode)
\d{3}	exactly 3 times: digit (0–9)
-	literal text “-”
\d{4}	exactly 4 times: digit (0–9)
$	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^\d{3}-\d{4}$/;
re.test(value);
```

Python

```
import re
re.match(r"^\d{3}-\d{4}$", value)
```

Java

```
Pattern.compile("^\\d{3}-\\d{4}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^\d{3}-\d{4}$");
```

Go

```
regexp.MustCompile(`^\d{3}-\d{4}$`).MatchString(value)
```

Ruby

```
/^\d{3}-\d{4}$/.match?(value)
```

PHP

```
preg_match('~^\d{3}-\d{4}$~', $value);
```

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and $, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the japanese postal code against a source of truth (database, API, or checksum) where it matters.