

Toll-free 1800 number

1800-series toll-free number — free to caller within India.

```
^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}$
```

Token-by-token breakdown

Token	Meaning
^	start of string (or line in multiline mode)
1800	literal text "1800"
[\s\-]?	optional (zero or one): any of: whitespace, "-"
\d{2,3}	between 2 and 3 times: digit (0–9)
[\s\-]?	optional (zero or one): any of: whitespace, "-"
\d{4,7}	between 4 and 7 times: digit (0–9)
\$	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	<pre>const re = /^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$/; re.test(value);</pre>
Python	<pre>import re re.match(r"^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$", value)</pre>
Java	<pre>Pattern.compile("^1800[\\s\\-]?\\d{2,3}[\\s\\-]?\\d{4,7}\$").matcher(value).matches();</pre>
C# / .NET	<pre>Regex.IsMatch(value, @"^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$");</pre>
Go	<pre>regexp.MustCompile(`^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$`).MatchString(value)</pre>
Ruby	<pre>/^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$/ .match?(value)</pre>
PHP	<pre>preg_match('~^1800[\s\-]?\d{2,3}[\s\-]?\d{4,7}\$~', \$value);</pre>

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 toll-free 1800 number against a source of truth (database, API, or checksum) where it matters.