Indian mobile with +91

Indian mobile in international format.

```
^(\+91[\-\s]?|0)?[6-9]\d{9}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`(`	start of a capturing group
`\+`	literal “+”
`91`	literal text “91”
`[\-\s]?`	optional (zero or one): any of: “-”, whitespace
`|`	OR — try the alternative
`0`	literal text “0”
`)?`	end of group, optional (zero or one)
`[6-9]`	any of: 6–9
`\d{9}`	exactly 9 times: digit (0–9)
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^(\+91[\-\s]?|0)?[6-9]\d{9}$/;
re.test(value);
```

Python

```
import re
re.match(r"^(\+91[\-\s]?|0)?[6-9]\d{9}$", value)
```

Java

```
Pattern.compile("^(\\+91[\\-\\s]?|0)?[6-9]\\d{9}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^(\+91[\-\s]?|0)?[6-9]\d{9}$");
```

Go

```
regexp.MustCompile(`^(\+91[\-\s]?|0)?[6-9]\d{9}$`).MatchString(value)
```

Ruby

```
/^(\+91[\-\s]?|0)?[6-9]\d{9}$/.match?(value)
```

PHP

```
preg_match('~^(\+91[\-\s]?|0)?[6-9]\d{9}$~', $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 indian mobile with +91 against a source of truth (database, API, or checksum) where it matters.