Email address

Standard email validation.

```
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[a-zA-Z0-9._%+-]+`	one or more: any of: lowercase letters, uppercase letters, digits, “.”, “_”, “%”, “+”, “-”
`@`	literal text “@”
`[a-zA-Z0-9.-]+`	one or more: any of: lowercase letters, uppercase letters, digits, “.”, “-”
`\.`	literal “.”
`[a-zA-Z]{2,}`	2 or more times: any of: lowercase letters, uppercase letters
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;
re.test(value);
```

Python

```
import re
re.match(r"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$", value)
```

Java

```
Pattern.compile("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$");
```

Go

```
regexp.MustCompile(`^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$`).MatchString(value)
```

Ruby

```
/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/.match?(value)
```

PHP

```
preg_match('~^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$~', $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.
- **ASCII letters only.** [a-zA-Z] excludes accented and non-Latin letters (é, ü, ß, ñ, and non-Latin scripts). For international input use Unicode properties like \p{L} with the u 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 email address against a source of truth (database, API, or checksum) where it matters.