REGEX CHEAT SHEET
regexguide.com/patterns/currency-eur.html

Euro amount

```
^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`€?`	optional (zero or one): the literal “€”
`\s?`	optional (zero or one): whitespace
`-?`	optional (zero or one): the literal “-”
`(`	start of a capturing group
`\d{1,3}`	between 1 and 3 times: digit (0–9)
`(`	start of a capturing group
`[.,]`	any of: “.”, “,”
`\d{3}`	exactly 3 times: digit (0–9)
`)+`	end of group, one or more
`|`	OR — try the alternative
`\d+`	one or more: digit (0–9)
`)`	end of group
`(`	start of a capturing group
`[.,]`	any of: “.”, “,”
`\d{2}`	exactly 2 times: digit (0–9)
`)?`	end of group, optional (zero or one)
`\s?`	optional (zero or one): whitespace
`€?`	optional (zero or one): the literal “€”
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$/;
re.test(value);
```

Python

```
import re
re.match(r"^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$", value)
```

Java

```
Pattern.compile("^€?\\s?-?(\\d{1,3}([.,]\\d{3})+|\\d+)([.,]\\d{2})?\\s?€?$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$");
```

Go

```
regexp.MustCompile(`^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$`).MatchString(value)
```

Ruby	`/^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$/.match?(value)`
PHP	`preg_match('~^€?\s?-?(\d{1,3}([.,]\d{3})+|\d+)([.,]\d{2})?\s?€?$~', $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.

• **Watch for backtracking.** This pattern nests quantifiers; on adversarial input that can cause exponential backtracking (ReDoS). Test with long non-matching strings, or use an RE2-based engine.

• **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 euro amount against a source of truth (database, API, or checksum) where it matters.