Credit card (any major)

Match any major credit card brand.

```
^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`(?:`	start of a non-capturing group
`4`	literal text “4”
`\d{12}`	exactly 12 times: digit (0–9)
`(?:`	start of a non-capturing group
`\d{3}`	exactly 3 times: digit (0–9)
`)?`	end of group, optional (zero or one)
`|`	OR — try the alternative
`(?:`	start of a non-capturing group
`5`	literal text “5”
`[1-5]`	any of: 1–5
`\d{2}`	exactly 2 times: digit (0–9)
`|`	OR — try the alternative
`222`	literal text “222”
`[1-9]`	any of: 1–9
`|`	OR — try the alternative
`22`	literal text “22”
`[3-9]`	any of: 3–9
`\d`	any digit (0–9)
`|`	OR — try the alternative
`2`	literal text “2”
`[3-6]`	any of: 3–6
`\d{2}`	exactly 2 times: digit (0–9)
`|`	OR — try the alternative
`27`	literal text “27”
`[01]`	any of: “0”, “1”
`\d`	any digit (0–9)

Token	Meaning
`|`	OR — try the alternative
`2720`	literal text “2720”
`)`	end of group
`\d{12}`	exactly 12 times: digit (0–9)
`|`	OR — try the alternative
`3`	literal text “3”
`[47]`	any of: “4”, “7”
`\d{13}`	exactly 13 times: digit (0–9)
`|`	OR — try the alternative
`6`	literal text “6”
`(?:`	start of a non-capturing group
`011`	literal text “011”
`|`	OR — try the alternative
`5`	literal text “5”
`\d{2}`	exactly 2 times: digit (0–9)
`|`	OR — try the alternative
`4`	literal text “4”
`[4-9]`	any of: 4–9
`\d`	any digit (0–9)
`|`	OR — try the alternative
`22`	literal text “22”
`[1-9]`	any of: 1–9
`\d{2}`	exactly 2 times: digit (0–9)
`)`	end of group
`\d{12}`	exactly 12 times: digit (0–9)
`|`	OR — try the alternative
`(?:`	start of a non-capturing group
`508`	literal text “508”
`[5-9]`	any of: 5–9
`|`	OR — try the alternative
`6069`	literal text “6069”
`|`	OR — try the alternative
`607`	literal text “607”
`[0-9]`	any of: digits
`|`	OR — try the alternative
`608`	literal text “608”

Token	Meaning
`[0-7]`	any of: 0–7
`|`	OR — try the alternative
`6521`	literal text “6521”
`|`	OR — try the alternative
`6522`	literal text “6522”
`)`	end of group
`\d{12}`	exactly 12 times: digit (0–9)
`)`	end of group
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$/;
re.test(value);
```

Python

```
import re
re.match(r"^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$", value)
```

Java

```
Pattern.compile("^(?:4\\d{12}(?:\\d{3})?|(?:5[1-5]\\d{2}|222[1-9]|22[3-9]\\d|2[3-6]\\d{2}|27[01]\\d|2720)\\d{12}|3[47]\\d{13}|6(?:011|5\\d{2}|4[4-9]\\d|22[1-9]\\d{2})\\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\\d{12})$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$");
```

Go

```
regexp.MustCompile(`^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$`).MatchString(value)
```

Ruby

```
/^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$/.match?(value)
```

PHP

```
preg_match('~^(?:4\d{12}(?:\d{3})?|(?:5[1-5]\d{2}|222[1-9]|22[3-9]\d|2[3-6]\d{2}|27[01]\d|2720)\d{12}|3[47]\d{13}|6(?:011|5\d{2}|4[4-9]\d|22[1-9]\d{2})\d{12}|(?:508[5-9]|6069|607[0-9]|608[0-7]|6521|6522)\d{12})$~', $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 credit card (any major) against a source of truth (database, API, or checksum) where it matters.