PAN — Individual

PAN issued specifically to an individual (4th char is P).

```
^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`[A-Z]{3}`	exactly 3 times: any of: uppercase letters
`P`	literal text “P”
`[A-Z]`	any of: uppercase letters
`[0-9]{4}`	exactly 4 times: any of: digits
`[A-Z]`	any of: uppercase letters
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript

```
const re = /^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$/;
re.test(value);
```

Python

```
import re
re.match(r"^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$", value)
```

Java

```
Pattern.compile("^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$").matcher(value).matches();
```

C# / .NET

```
Regex.IsMatch(value, @"^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$");
```

Go

```
regexp.MustCompile(`^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$`).MatchString(value)
```

Ruby

```
/^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$/.match?(value)
```

PHP

```
preg_match('~^[A-Z]{3}P[A-Z][0-9]{4}[A-Z]$~', $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.
- **Uppercase only.** Ranges like [A-Z] won't match lowercase. Add a-z (or the i flag) if lowercase input should be accepted.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the pan — individual against a source of truth (database, API, or checksum) where it matters.