

```
got <d is under the bar in the >
```

That's because `.*` was greedy, so you get everything between the *first* "foo" and the *last* "bar". Here it's more effective to use minimal matching to make sure you get the text between a "foo" and the first "bar" thereafter.

```
if ( /foo(.*?)bar/ ) { print "got <$1>\n" }
got <d is under the >
```

Here's another example: let's say you'd like to match a number at the end of a string, and you also want to keep the preceding part of the match. So you write this:

```
$_ = "I have 2 numbers: 53147";
if ( /(.*)(\d*)/ ) {                               # Wrong!
    print "Beginning is <$1>, number is <$2>.\n";
}
```

That won't work at all, because `.*` was greedy and gobbled up the whole string. As `\d*` can match on an empty string the complete regular expression matched successfully.

```
Beginning is <I have 2 numbers: 53147>, number is <>.
```

Here are some variants, most of which don't work:

```
$_ = "I have 2 numbers: 53147";
@pats = qw{
    (.*)(\d*)
    (.*)(\d+)
    (.*?)(\d*)
    (.*?)(\d+)
    (.*)(\d+)$
    (.*?)(\d+)$
    (.*)\b(\d+)$
    (.*\D)(\d+)$
};

for $pat (@pats) {
    printf "%-12s ", $pat;
    if ( /$pat/ ) {
        print "<$1> <$2>\n";
    } else {
        print "FAIL\n";
    }
}
```

That will print out:

```
(.*)(\d*)    <I have 2 numbers: 53147> <>
(.*)(\d+)    <I have 2 numbers: 5314> <7>
(.*?)(\d*)   <> <>
(.*?)(\d+)   <I have > <2>
(.*)(\d+)$   <I have 2 numbers: 5314> <7>
(.*?)(\d+)$  <I have 2 numbers: > <53147>
(.*)\b(\d+)$ <I have 2 numbers: > <53147>
(.*\D)(\d+)$ <I have 2 numbers: > <53147>
```