Black Magic with Regular Expressions

Jakob Westhoff

Zend Webinar
20. November 2012

Qafoo
passion for software quality

About Me

Jakob Westhoff

About Me

Jakob Westhoff

Qafoo

passion for software quality

About Me

Jakob Westhoff

- PHP Professional since 2001
- JavaScript Professional since 2006
- Trainer and Consultant
- Author of articles and a book
- Regular speaker at technology conferences

Terminology

Terminology

- 🟥 RegExp
- 🟧 Subject
- 🟩 Match

Engine Flavors

PCRE

PCRE

=

Perl Compatible Regular Expressions

PCRE

PCRE

=

Perl Compatible Regular Expressions

- Library that PHP utilizes

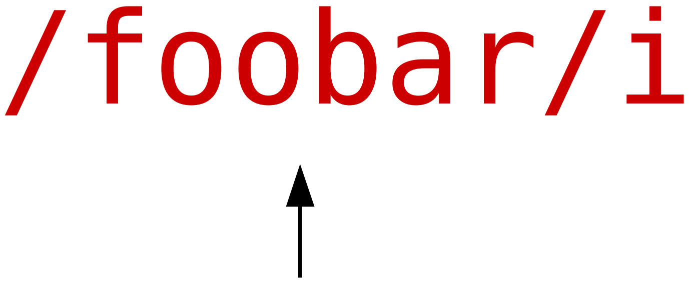
RegExp

passion for software quality

Basic structure of a RegExp

```
/foobar/i
```

Basic structure of a RegExp



```
/foobar/i
```

- Pattern
  - Description of the matching Strings

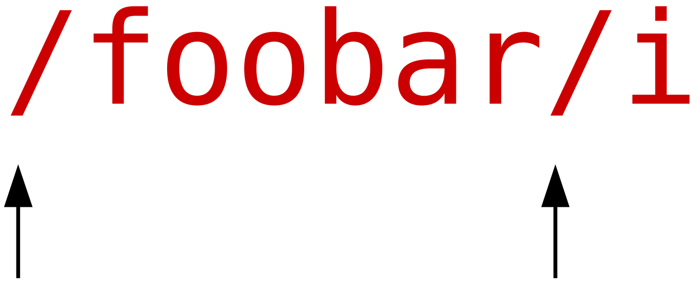
Basic structure of a RegExp

```
/foobar/i
```

↑

- Modifier
  - Additional Options

Basic structure of a RegExp



```
/foobar/i
```

- Delimiter
  - Enclosure of Pattern
  - Divider between Pattern and Modifier

Basic structure of a RegExp

```
(foobar)i
```

- Delimiter
  - PCRE allows arbitrary Brackets
    - () [] {}

Metacharacters

Metacharacters

- Certain characters inside a RegExp Pattern have got a special meaning

```
([We]b \s* Te+c.no)
```

Quantifier

- Quantifiers specify Repetitions of the previous character or group

```
(We*b Te+ch?n{1,3}o)
```

Quantifier

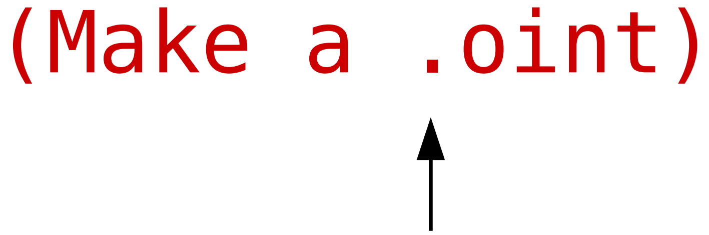
- Quantifiers specify Repetitions of the previous character or group

```
(We*b Te+ch?n{1,3}o)
```

- `*` Any number of occurrences (0 → ∞)
- `+` One occurrence minimum (1 → ∞)
- `?` Not at all or one time (0 → 1)
- `{x,y}` Between x and y (x → y)

The Dot

- The Dot (.) matches any character
  - Everything except newline



```
(Make a .oint)
```

Character Classes

```
([abcdef]+)
```

- Character classes define a Set of arbitrary characters

Character Classes

```
([a-cd-f]+)
```

- Ranges can be defined
- One Character Class may contain multiple Ranges

Character Classes

```
([^abcdef]+)
```

- A Character Class can be negated
- The newline character is part of the negation

Alternatives

- Logical OR

```
(Open|Source)
```

Alternatives

- Logical OR

```
(Open|Source)
```

```
Open
```

Alternatives

- Logical OR

`(Open|Source)`

`Open` ✓

Alternatives

- Logical OR

```
(Open|Source)
```

`Open` ✓

`Source` ✓

Alternatives

- Logical <u>OR</u>

```
(Open|Source)
```

`Open`	✓
`Source`	✓
`Open Source`	

Alternatives

- Logical OR

↓

`(Open|Source)`

`Open` ✓

`Source` ✓

`Open Source` ✓

Subpattern

Subpattern

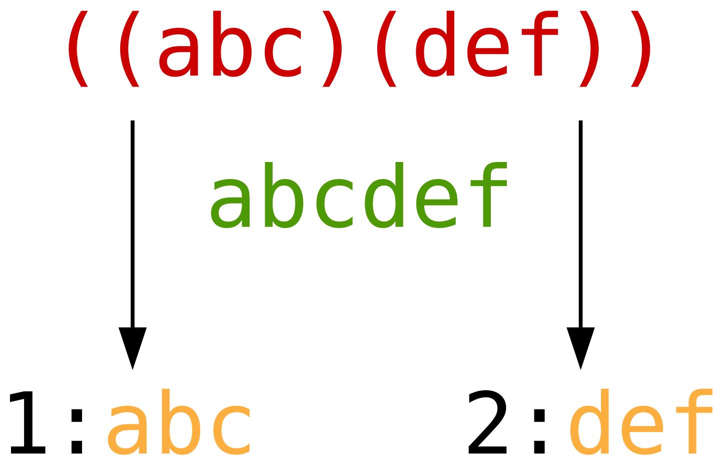
- Pattern can be divided using parenthesis

```
((abc)(def))
```

```
abcdef
```

Subpattern

- Pattern can be divided using parenthesis



```
((abc)(def))
abcdef
1:abc    2:def
```

- Subpatterns may be used to extract parts of the match

Subpattern Options

- Subpattern may be used to set options/modifiers for a certain area of the Regular Expression

```
((?#I am a comment subpattern.))
```

Subpattern Options

- Setting options for a subpattern

```
(?OptionPattern)
```

    - Abstract syntax for any option

Subpattern Options

- Setting the case-insensitive modifier using a subpattern option

```
(((?i)[a-z]+) [a-z]+)
```

Subpattern Options

- Setting the case-insensitive modifier using a subpattern option

```
(((?i)[a-z]+) [a-z]+)
```

```
Jakob westhoff
```

Subpattern Options

- Setting the case-insensitive modifier using a subpattern option

```
(((?i)[a-z]+) [a-z]+)
```

```
Jakob westhoff
```
✓

Subpattern Options

- Setting the case-insensitive modifier using a subpattern option

```
(((?i)[a-z]+) [a-z]+)
```

```
Jakob westhoff  ✓

Jakob Westhoff  ✗
```

Named Subpattern

Named Subpattern

- Subpatterns may be named

```
((?P<firstname>Jakob))
```

  - The P Option is used for naming subpatterns

Named Subpattern

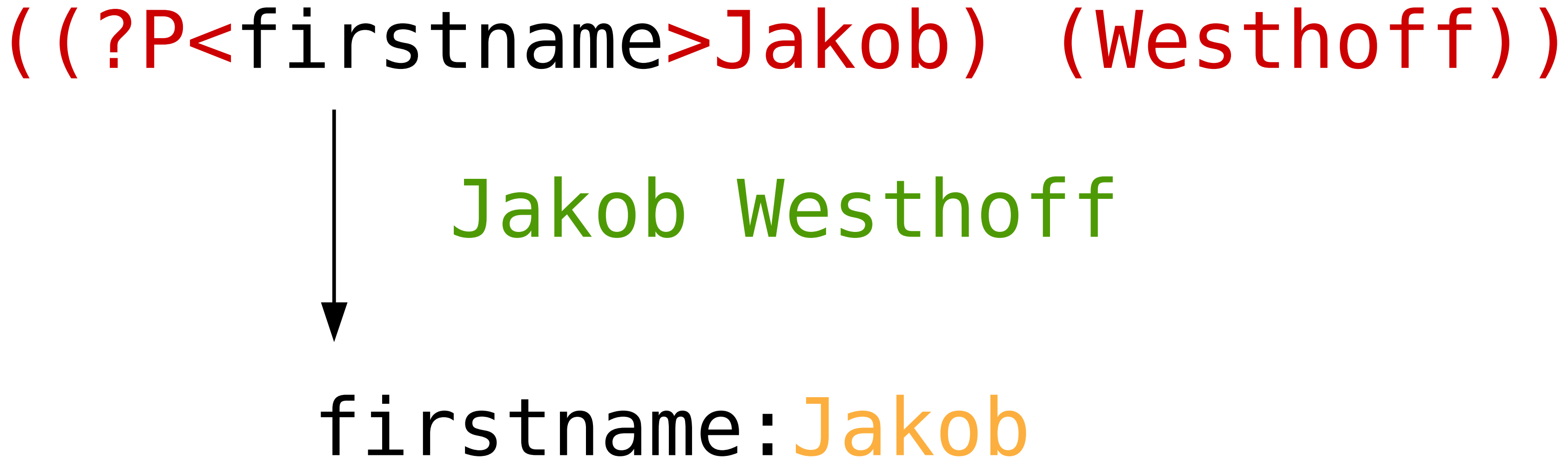
```
((?P<firstname>Jakob) (Westhoff))
```

Named Subpattern

```
((?P<firstname>Jakob) (Westhoff))
```

```
Jakob Westhoff
```

Named Subpattern



```
((?P<firstname>Jakob) (Westhoff))
```

Jakob Westhoff

firstname:Jakob

- Access to extraction using the subpatterns name is possible

Non grouping Subpattern

- Subpattern can be used without being a group

```
((?:Jakob))
```

↑

- The question mark followed by a colon (?:) creates a non grouping subpattern

Non grouping Subpattern

- Why are non grouping subpatterns useful?

Non grouping Subpattern

- Why are non grouping subpatterns useful?

```
((?:Jakob|Veronika) Westhoff)
```

Non grouping Subpattern

- Why are non grouping subpatterns useful?

```
((?:Jakob|Veronika) Westhoff)
```

```
  Jakob Westhoff      ✓
Veronika Westhoff     ✓
```

Non grouping Subpattern

- Why are non grouping subpatterns useful?

```
((?:Jakob|Veronika) Westhoff)
```

```
  Jakob Westhoff      ✓
Veronika Westhoff     ✓
```

  - No clobbering of extracted matches

Assertions

Anchors

- Anchors are part of the family of Assertions in Regular Expressions
- They are used to assert certain conditions without affecting the match
- Anchors: Beginning and end of the Subject

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

Anchors

- `^` Beginning of the Subject
- `$` End of the Subject

```
(Apple)i
```

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(Apple)i
```

```
Apple
```

Anchors

- `^` Beginning of the Subject
- `$` End of the Subject

```
(Apple)i
```

```
Apple
```

✓

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(Apple)i
```

```
Apple       ✓

Pineapple
```

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

`(Apple)i`

`Apple` ✓

`Pineapple` ✓

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

`(^Apple)i`

`Apple`	✓
`Pineapple`	✗

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(Apple$)i
```

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(Apple$)i

Apple        ✓
```

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(Apple$)i

Apple        ✓

Apple-pie
```

Anchors

- `^` Beginning of the Subject
- `$` End of the Subject

↓

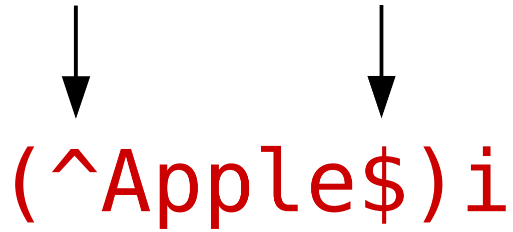
```
(Apple$)i
```

```
Apple      ✓

Apple-pie  ✗
```

Anchors

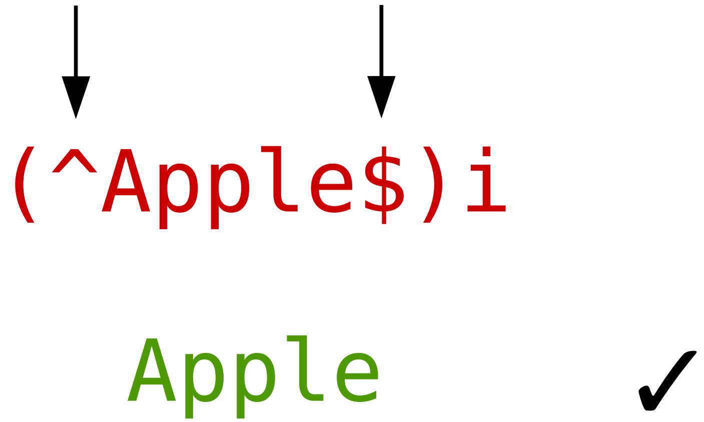
- ^ Beginning of the Subject
- $ End of the Subject



```
(^Apple$)i
```

Anchors

- ^ Beginning of the Subject
- $ End of the Subject



```
(^Apple$)i
```

Apple ✓

Anchors

- ^ Beginning of the Subject
- $ End of the Subject

```
(^Apple$)i
```

Apple	✓
Apple-pie	✗
Pineapple	✗

Other Assertions

- Custom assertions can be created
- The ?= Option is used for this

Other Assertions

- Custom assertions can be created
- The ?= Option is used for this

```
([a-z]+(?=,))i
```

Other Assertions

- Custom assertions can be created
- The ?= Option is used for this

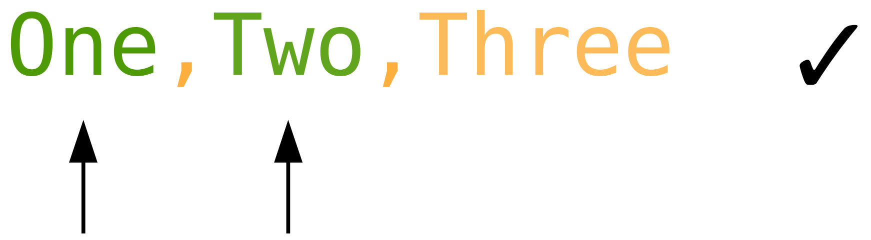
```
([a-z]+(?=,))i
```

```
One,Two,Three   ✓
   ↑
```

Other Assertions

- Custom assertions can be created
- The ?= Option is used for this

```
([a-z]+(?=,))i
```

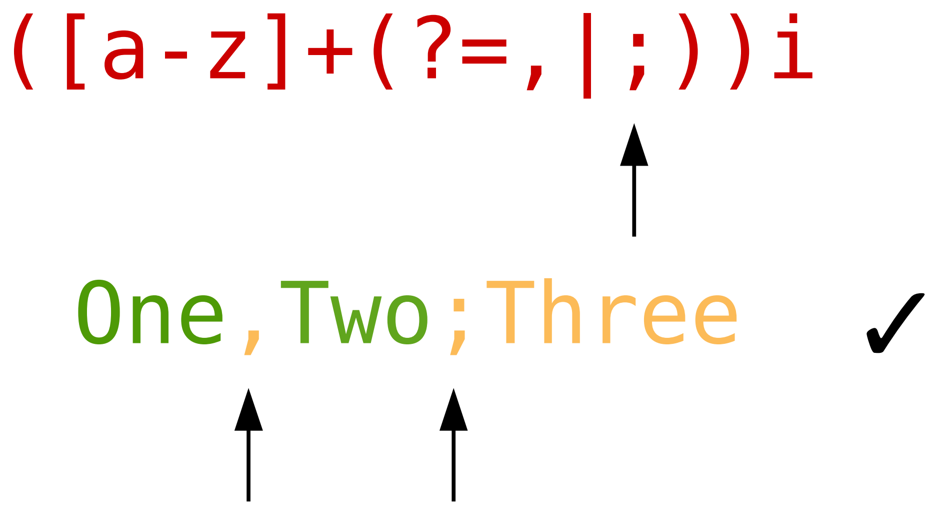


One,Two,Three ✓

- Useful in combination with `preg_match_all`

Other Assertions

- Custom assertions can be created
- The ?= Option is used for this



```
([a-z]+(?=,|;))i

One,Two;Three   ✓
```

- Alternatives may be used

Other Assertions

- Negative Assertions are possible
- The `?!` Option is used for this

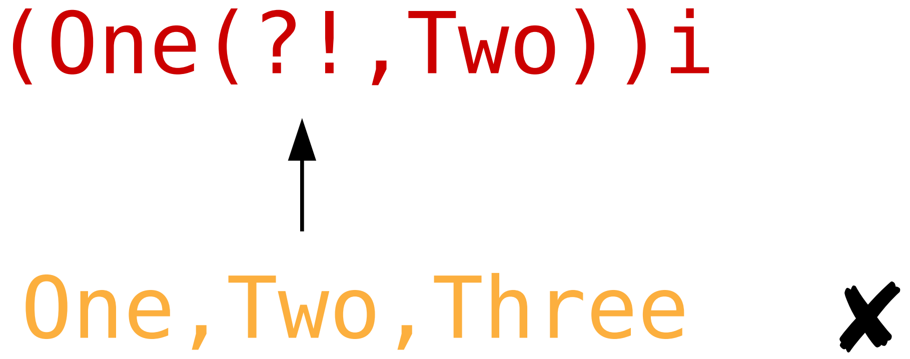
Other Assertions

- Negative Assertions are possible
- The ?! Option is used for this

```
(One(?!,Two))i
```

Other Assertions

- Negative Assertions are possible
- The ?! Option is used for this



```
(One(?!,Two))i
```

```
One,Two,Three
```

✗

Other Assertions

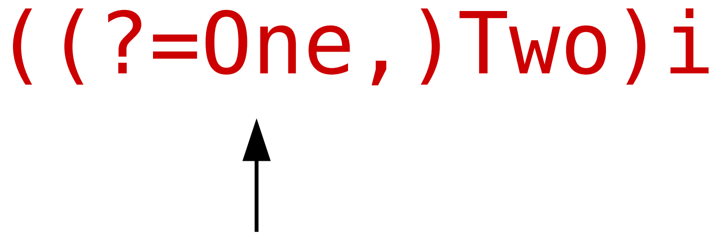
- Negative Assertions are possible
- The `?!` Option is used for this

```
(One(?!,Two))i
     ↑
One,Two,Three   ✗

One,Three   ✓
```

Other Assertions

- Assert on something before the cursor



```
((?=One,)Two)i
```

Other Assertions

- Assert on something before the cursor

```
((?=One,)Two)i
```

```
Three,Two,One
```

Other Assertions

- Assert on something before the cursor

```
((?=One,)Two)i
```

```
Three,Two,One
```

✗

Other Assertions

- Assert on something before the cursor

```
((?=One,)Two)i

Three,Two,One    ✘
One,Two,Three
```

Other Assertions

- Assert on something before the cursor

```
((?=One,)Two)i
```

```
Three,Two,One     ✗
One,Two,Three     ✗
```

Other Assertions

- Assert on something before the cursor

```
((?=One,)Two)i

Three,Two,One     ✗
One,Two,Three     ✗

Why?
```

passion for software quality

Other Assertions

- Assert on something before the cursor

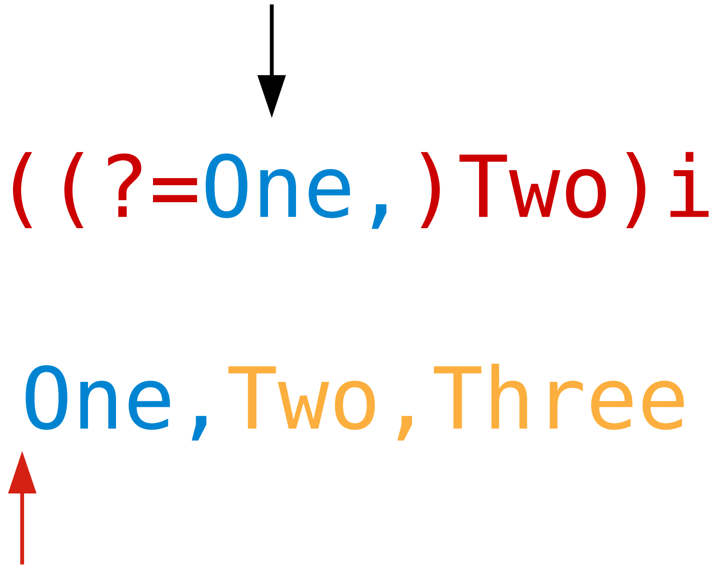
```
((?=One,)Two)i

One,Two,Three
↑
```

passion for software quality

Other Assertions

- Assert on something before the cursor

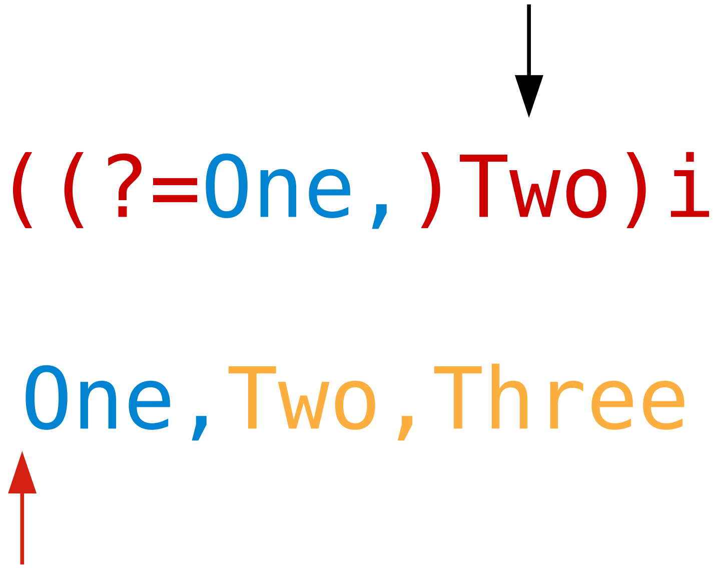


```
((?=One,)Two)i

One,Two,Three
```

Other Assertions

- Assert on something before the cursor

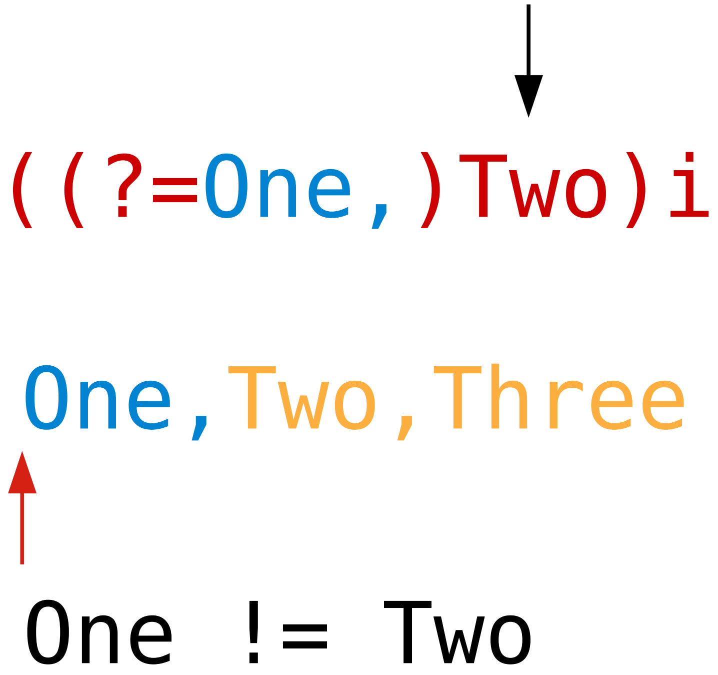


```
((?=One,)Two)i

One,Two,Three
```

Other Assertions

- Assert on something before the cursor



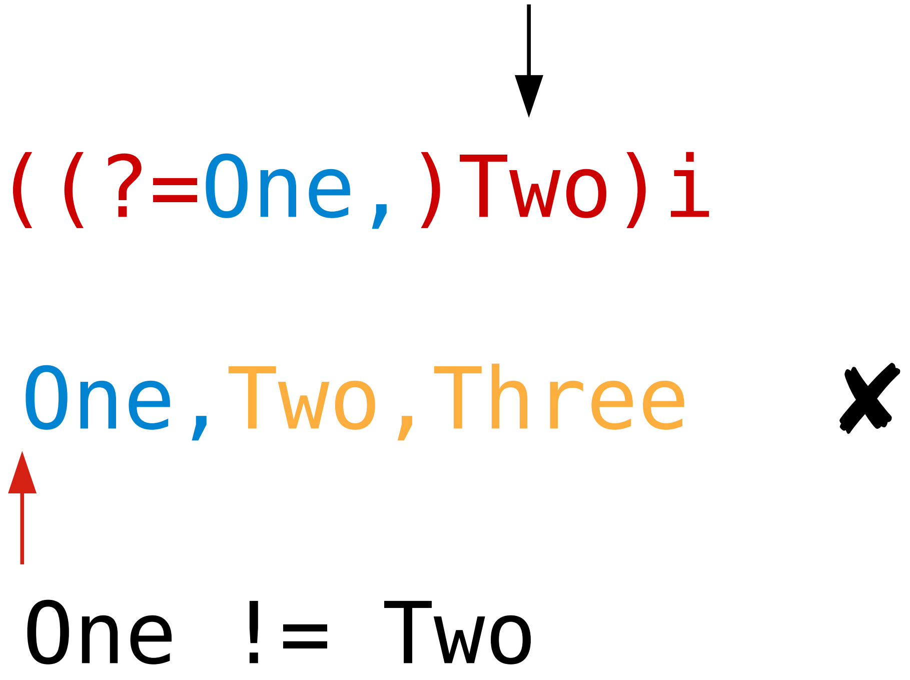
```
((?=One,)Two)i

One,Two,Three

One != Two
```

Other Assertions

- Assert on something before the cursor



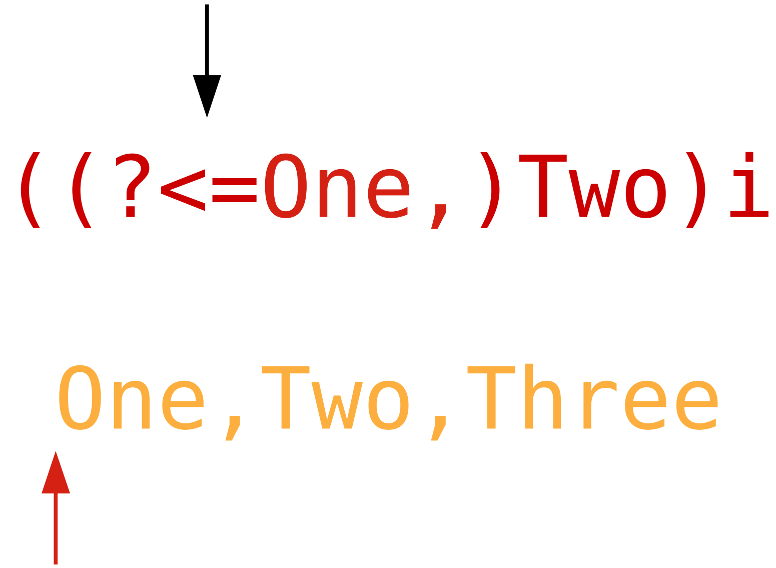
```
((?=One,)Two)i

One,Two,Three     ✗

One != Two
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: ?<=

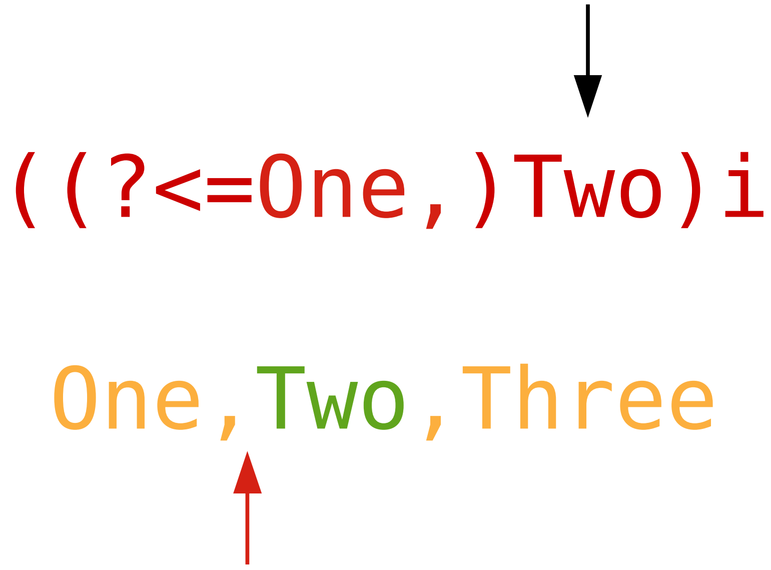


```
((?<=One,)Two)i
```

```
One,Two,Three
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: ?<=

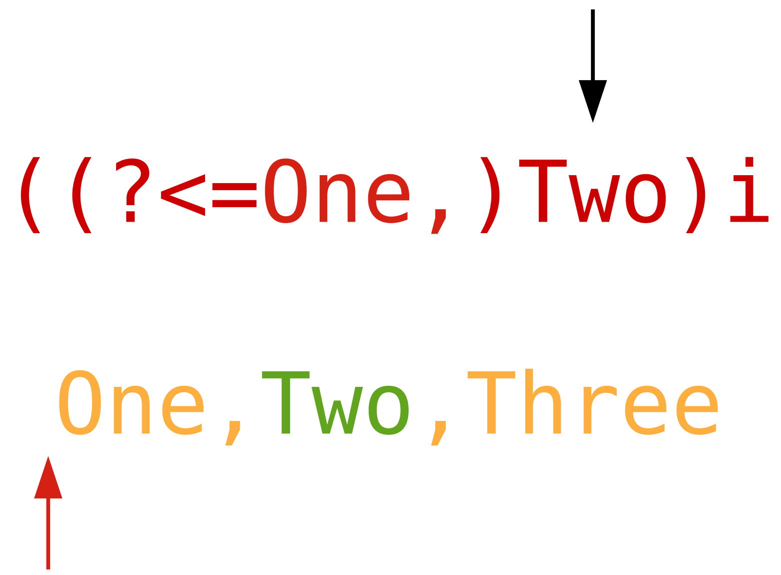


```
((?<=One,)Two)i

One,Two,Three
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: `?<=`

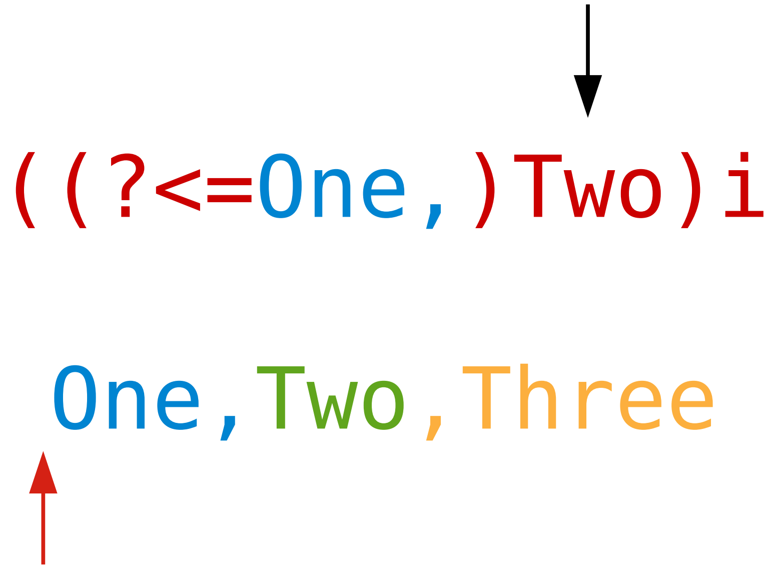


```
((?<=One,)Two)i
```

```
One,Two,Three
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: ?<=

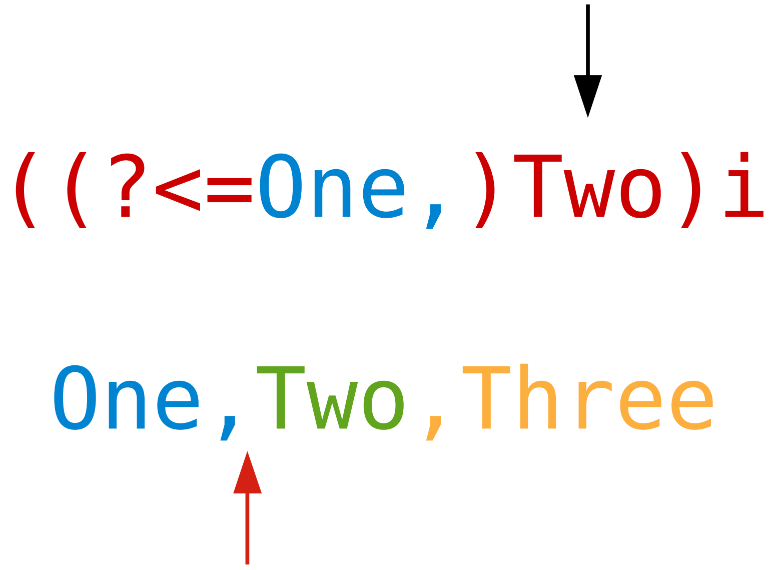


```
((?<=One,)Two)i
```

```
One,Two,Three
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: `?<=`

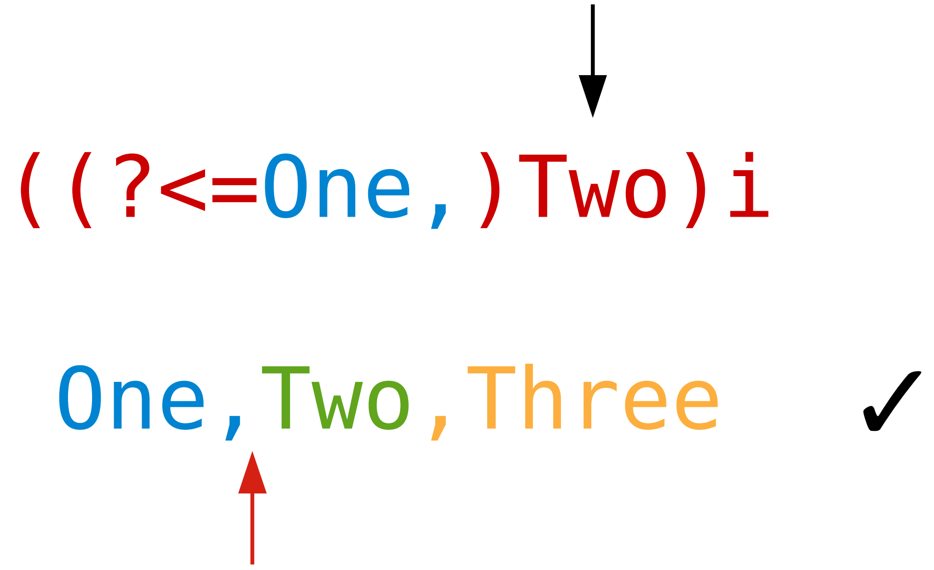


```
((?<=One,)Two)i
```

```
One,Two,Three
```

Other Assertions

- Look-Behind Assertions to the rescue
- Option: ?<=



```
((?<=One,)Two)i
```

```
One,Two,Three    ✓
```

Other Assertions

- Negative Look-Behind is possible
- Option: `?<!`

```
((?<!One,)Two)i
```

Other Assertions

- Negative Look-Behind is possible
- Option: `?<!`

↓

```
((?<!One,)Two)i
```

```
One,Two,Three
```
✗

Other Assertions

- Negative Look-Behind is possible
- Option: `?<!`

```
((?<!One,)Two)i
```

```
One,Two,Three      ✗
Three,Two,One      ✓
```

Unicode

Unicode

- UTF-8 Mode
  - Modifier u

```
(^abcdef$)u
```

- Valid UTF-8 needed in pattern and subject

Unicode

- UTF-8 Encoding
    - Bytes for all ASCII codes (0-127) identical
    - 2-4 Bytes used for further characters (Codepoints)
- Codepoints
    - Each Codepoint is considered to be <u>one</u> character

Unicode

Русский

Unicode

```
(\x{0420})u
```

Русский

Unicode

```
(\x{0420})u
```

Русский

↑

- `\x` Specify certain Unicode codepoints

Unicode

```
([\x{0400}-\x{A697}]+)u
```

Русский

- `\x` Specify certain Unicode codepoints
- Works within character classes

Unicode

```
(\p{Cyrillic}+)u
```

Русский

- \x Specify certain Unicode codepoints
- Works within character classes
- Predefined unicode character classes exist

Unicode

```
(\p{Cyrillic}+)u
```

Русский 한국어

- `\x` Specify certain Unicode codepoints
- Works within character classes
- Predefined unicode character classes exist

Unicode

```
(\p{L}+)u
```

Русский 한국어

- `\x` Specify certain Unicode codepoints
- Works within character classes
- Predefined unicode character classes exist

Performance

Performance

- The PCRE engines utilizes backtracking

Performance

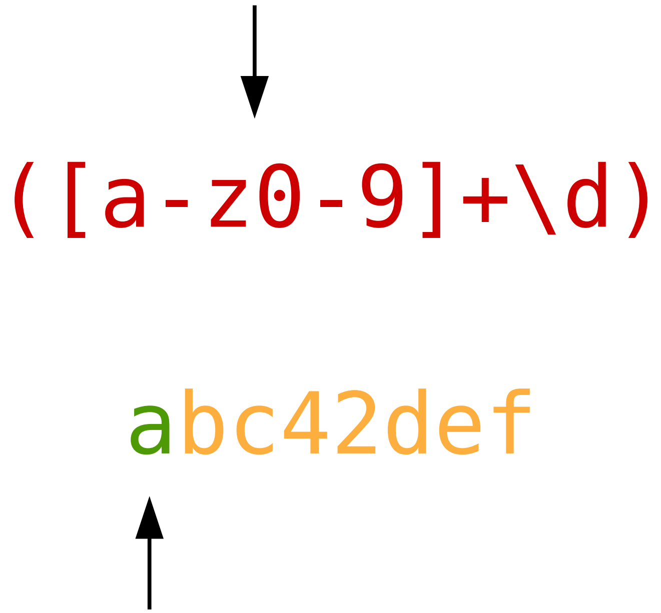
- The PCRE engines utilizes backtracking

```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

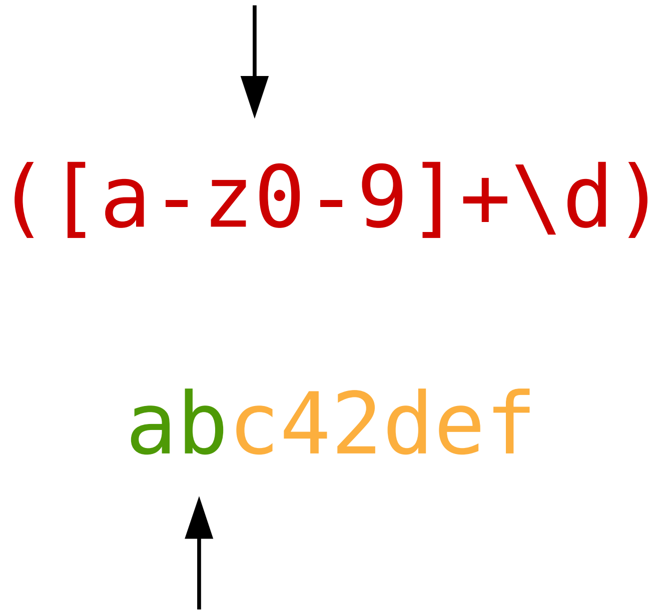


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

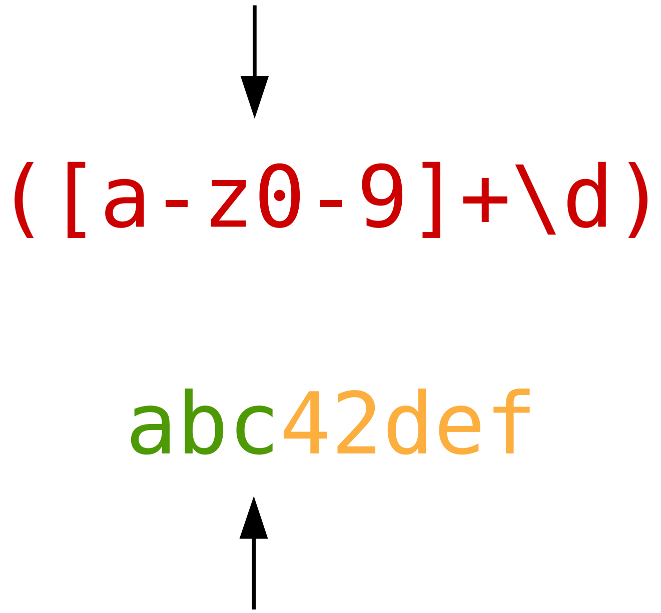


```
([a-z0-9]+\d)

abc42def
```

Performance

- The PCRE engines utilizes backtracking

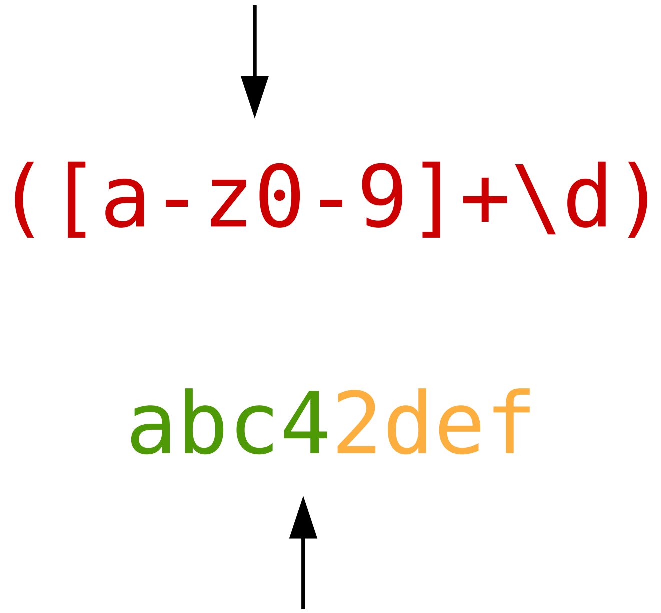


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

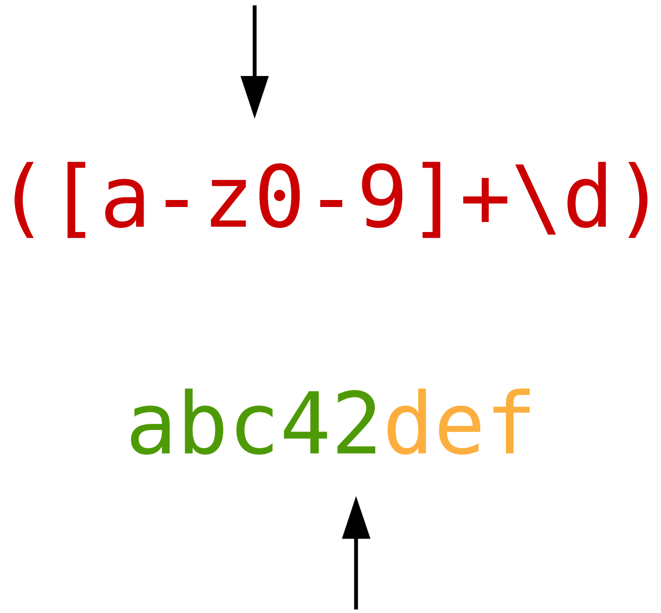


`([a-z0-9]+\d)`

`abc42def`

Performance

- The PCRE engines utilizes backtracking

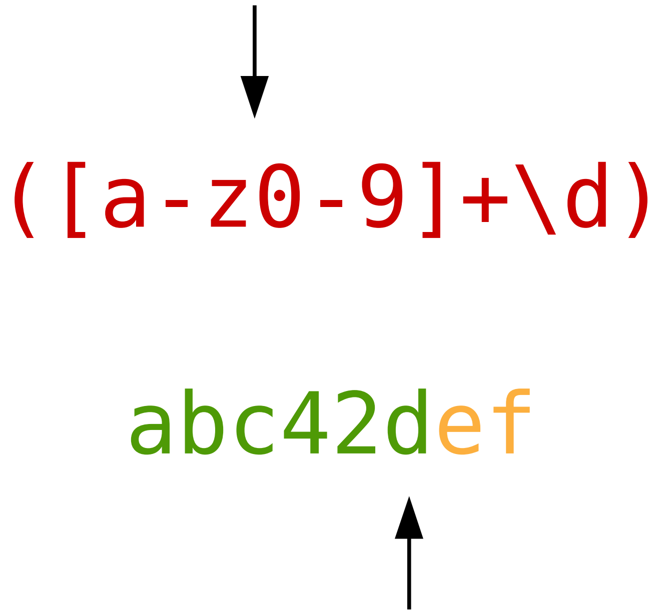


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

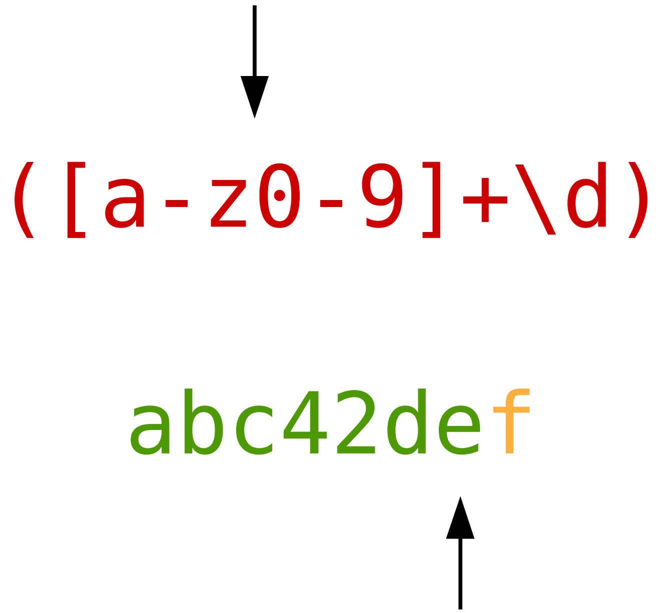


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

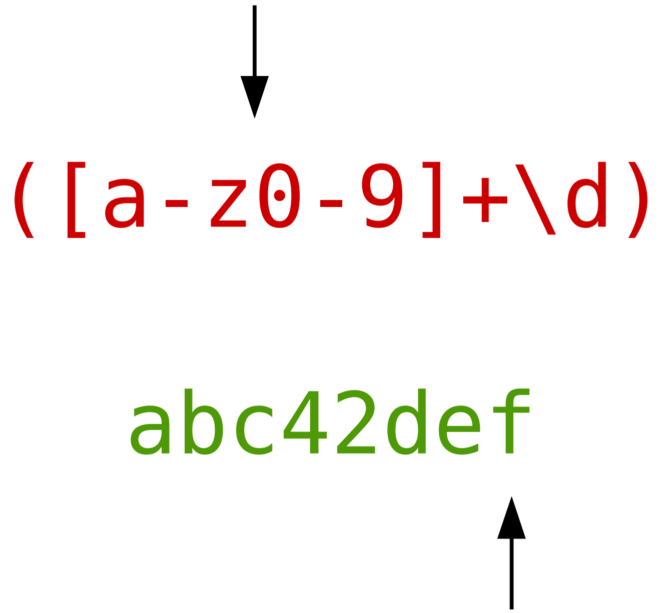


```
([a-z0-9]+\d)

abc42def
```

Performance

- The PCRE engines utilizes backtracking

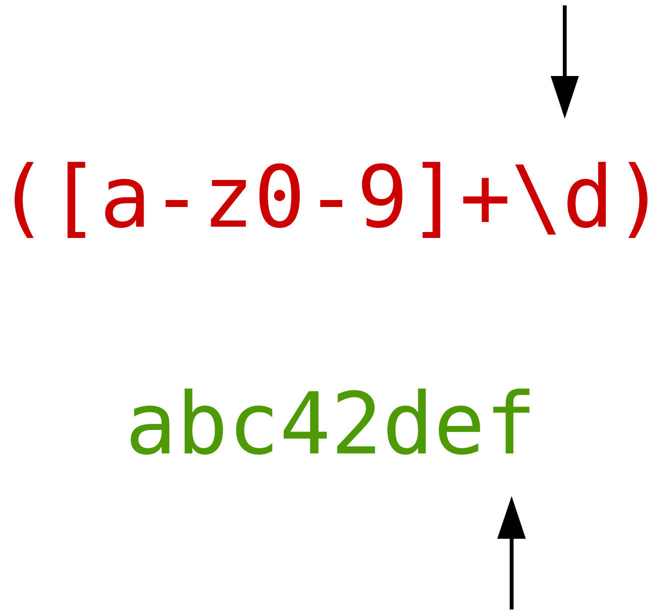


```
([a-z0-9]+\d)

abc42def
```

Performance

- The PCRE engines utilizes backtracking

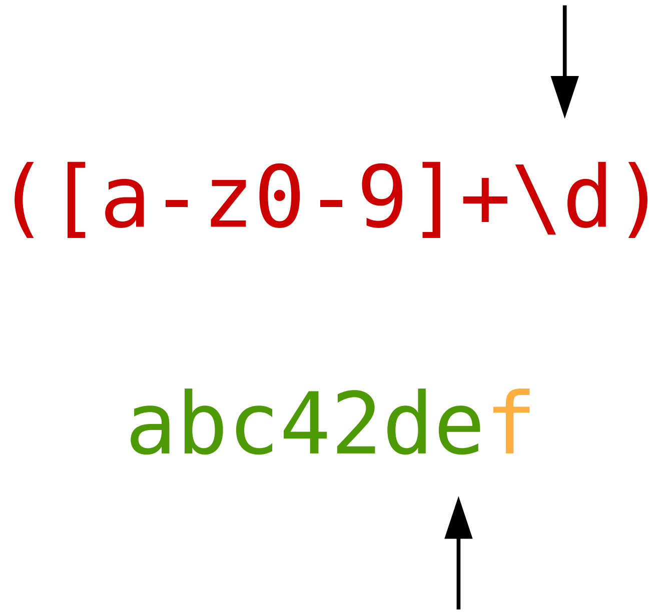


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

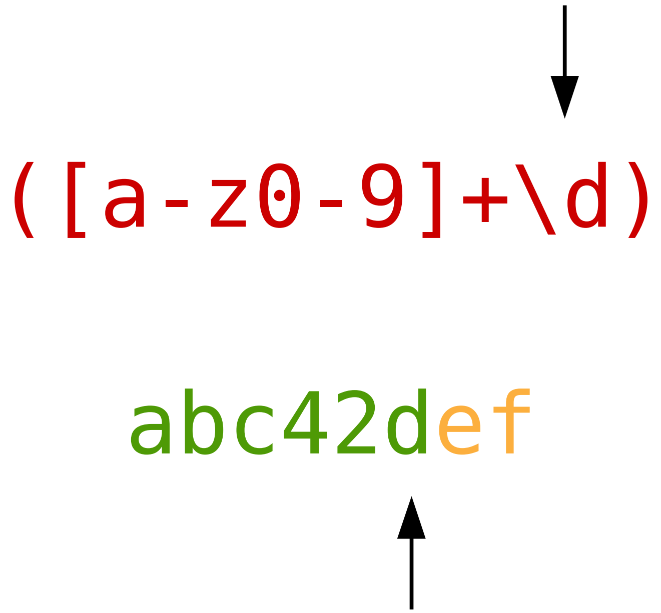


```
([a-z0-9]+\d)

abc42def
```

Performance

- The PCRE engines utilizes backtracking

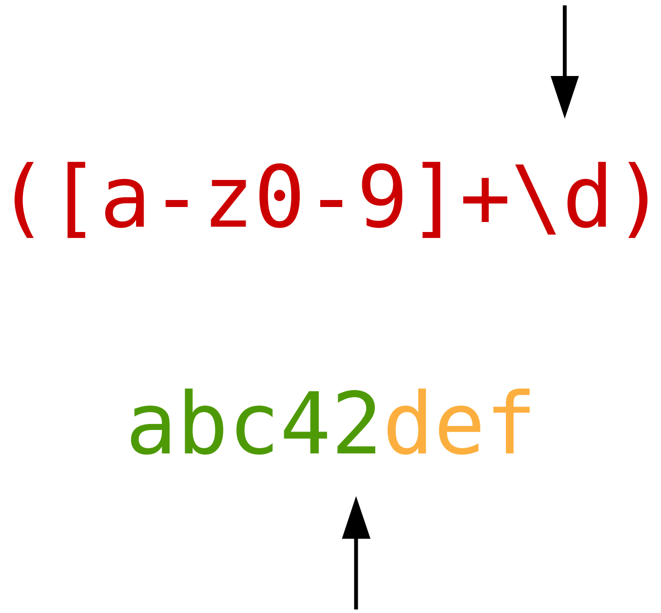


```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking



```
([a-z0-9]+\d)
```

```
abc42def
```

Performance

- The PCRE engines utilizes backtracking

```
([a-z0-9]+\d)
```

```
abc42def          ✓
```

Performance

```
([a-z0-9]+\d)
```

- Because of non disjunct character sets this match is quite slow

Performance

```
([a-z0-9]+\d)
```

- Because of non disjunct character sets this match is quite slow

Can it be optimized?

Greediness

- Usually the PCRE engine is greedy

- It tries to consume as much characters as possible to create a match

Greediness

- Advise the engine to be ungreedy
  - Modifier U

```
([a-z0-9]+\d)U
```

Greediness

- Advise the engine to be ungreedy
  - Modifier U

```
([a-z0-9]+\d)U
```

  - Question mark (?) after a quantifier

```
([a-z0-9]+?\d)
```

Greediness

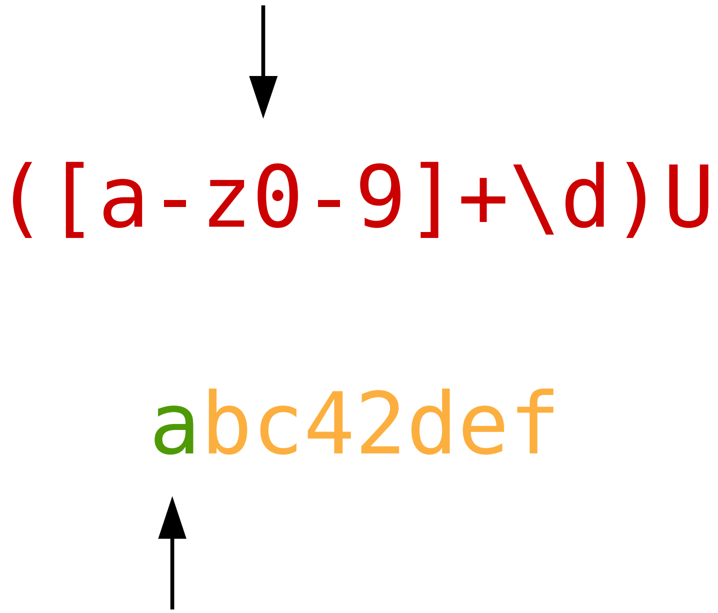
- Impact on the previous example

```
([a-z0-9]+\d)U
```

```
abc42def
```

Greediness

- Impact on the previous example

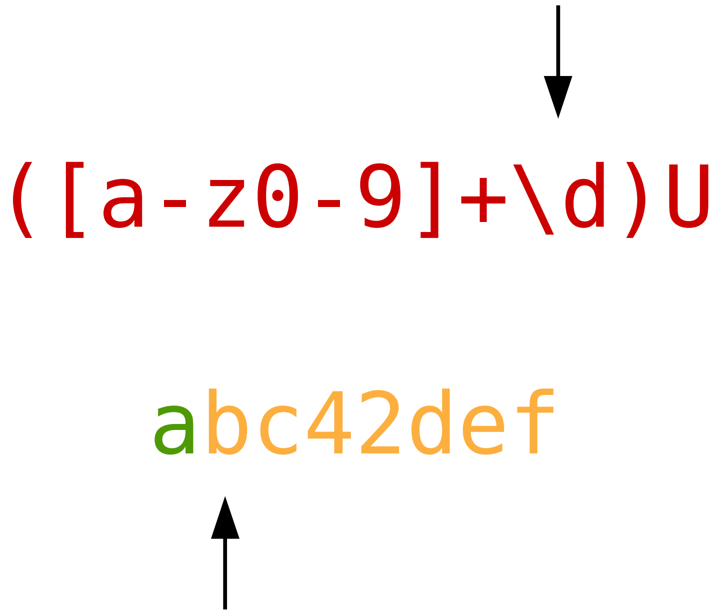


```
([a-z0-9]+\d)U

abc42def
```

Greediness

- Impact on the previous example

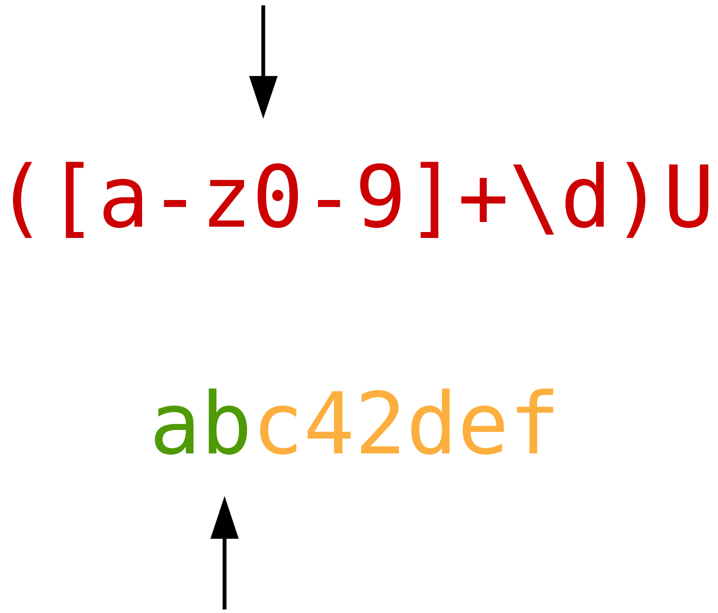


```
([a-z0-9]+\d)U

abc42def
```

Greediness

- Impact on the previous example

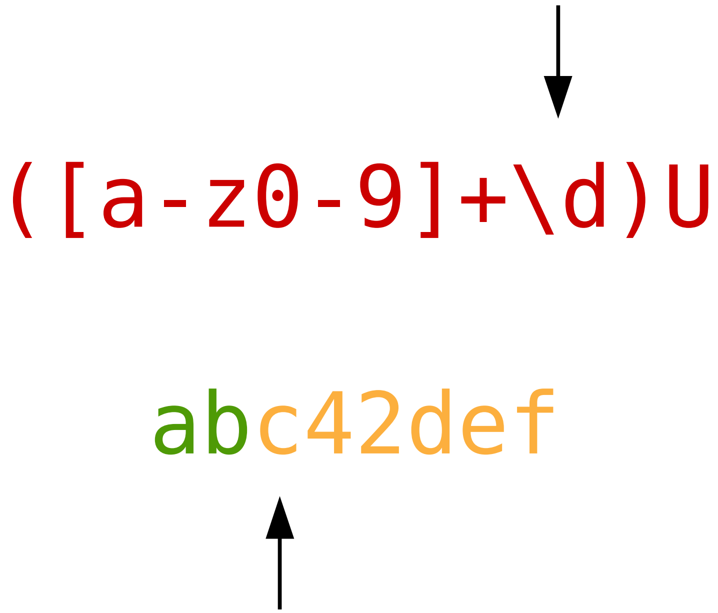


```
([a-z0-9]+\d)U

abc42def
```

Greediness

- Impact on the previous example

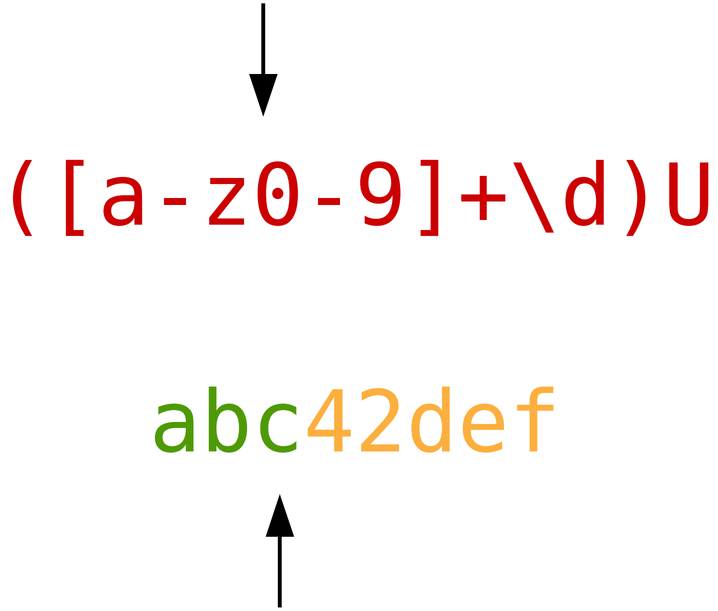


```
([a-z0-9]+\d)U
```

```
abc42def
```

Greediness

- Impact on the previous example

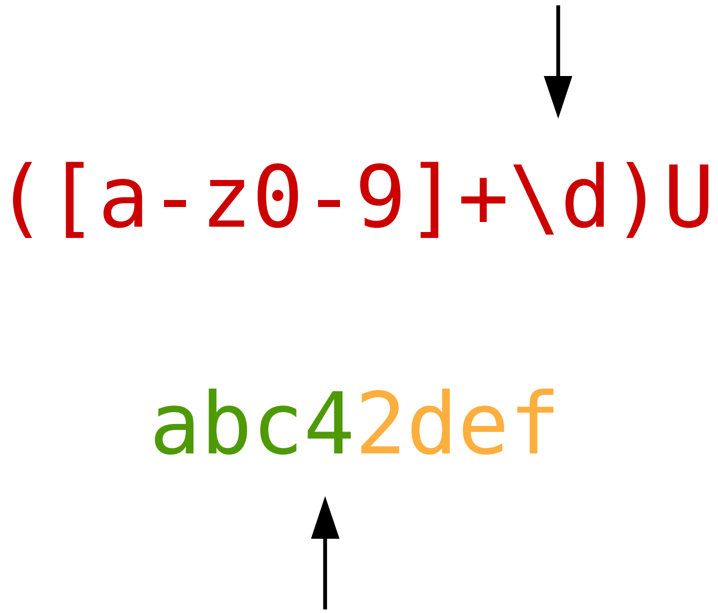


```
([a-z0-9]+\d)U
```

```
abc42def
```

Greediness

- Impact on the previous example



```
([a-z0-9]+\d)U

abc42def
```

Greediness

- Impact on the previous example

```
([a-z0-9]+\d)U

abc42def        ✓
```

Greediness

- Impact on the previous example

```
([a-z0-9]+\d)U
```

```
abc42def        ✓
```

- May produce different results than a greedy match

Greediness

- Ungreedy matching is not always faster than greedy matching
- In most situations it is even slower
- Can produce different matches than greedy matching

Atomic Groups

- Another possibility of controlling backtracking are Atomic Groups

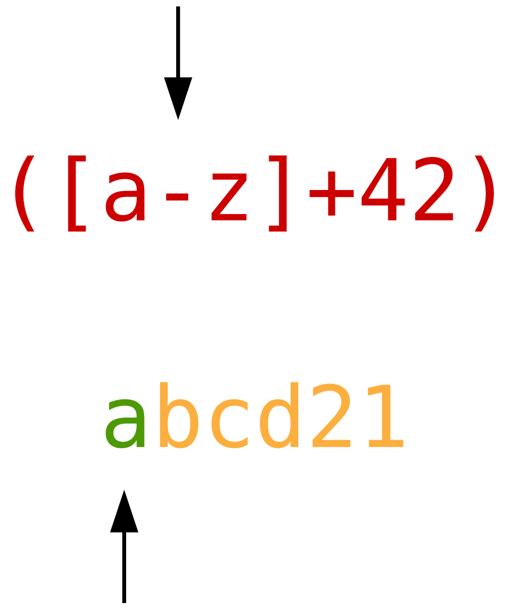
- Explicitly disable backtracking for a certain area of the Regular Expression

Atomic Groups

```
([a-z]+42)
```

```
abcd21
```

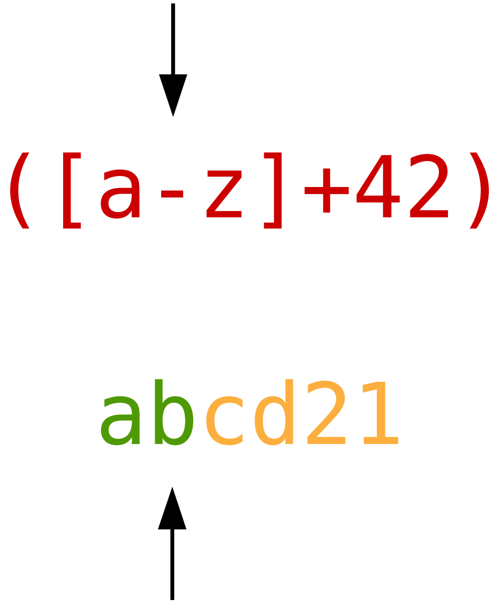
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

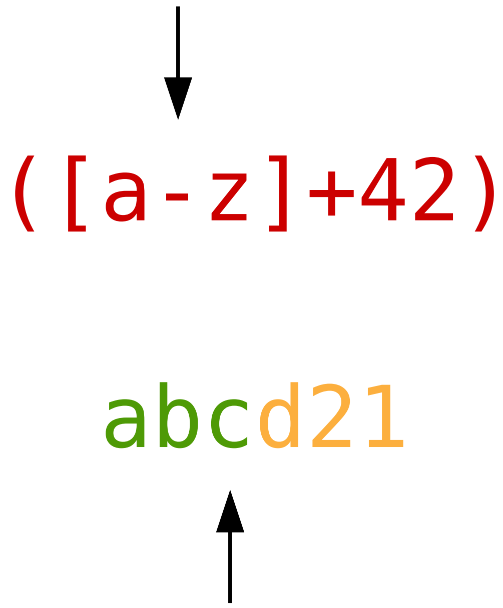
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

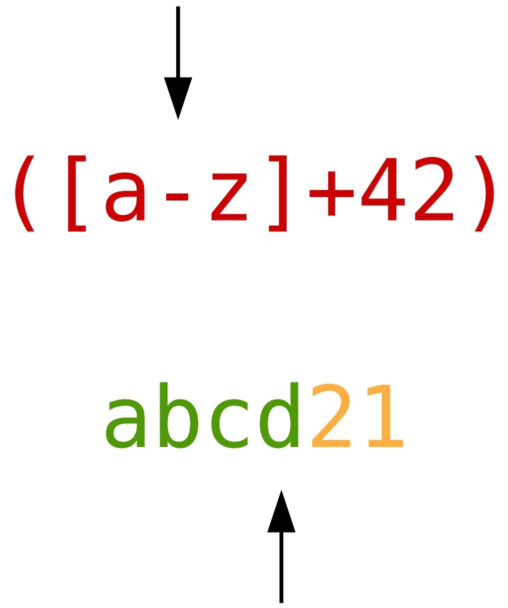
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

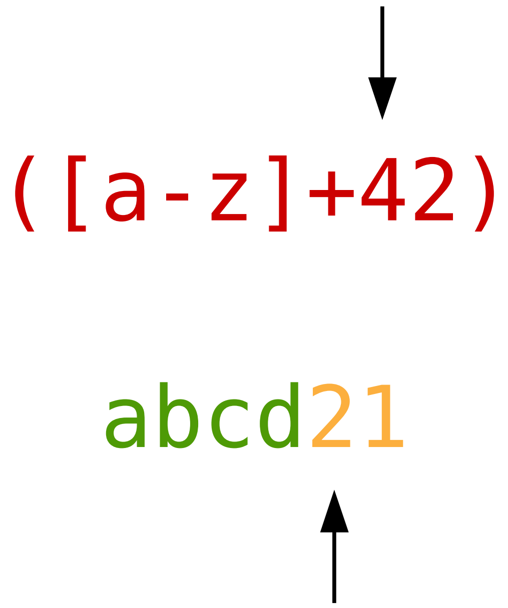
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

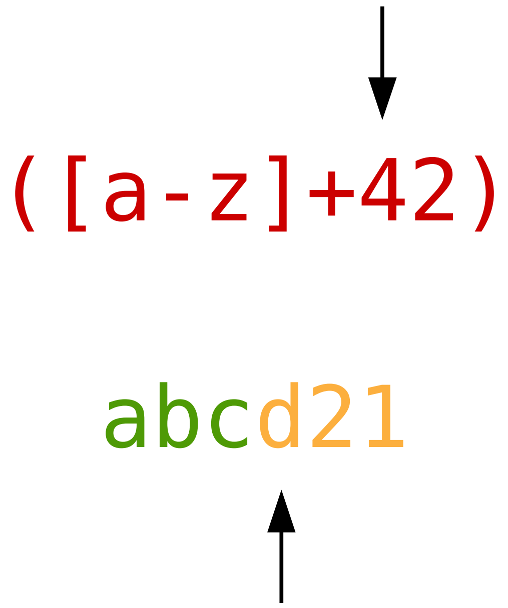
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

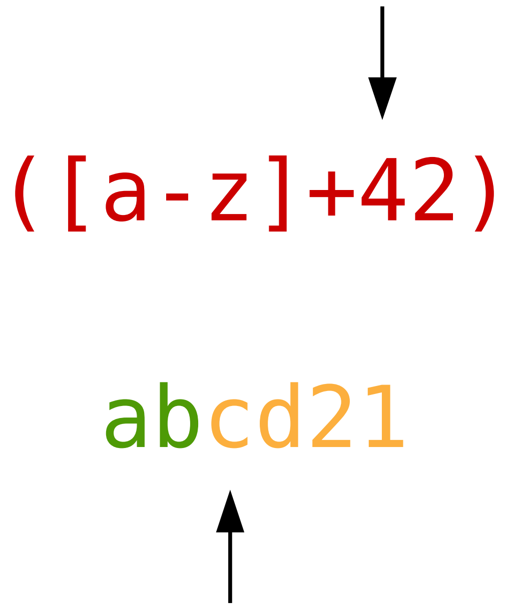
Atomic Groups



`([a-z]+42)`

`abcd21`

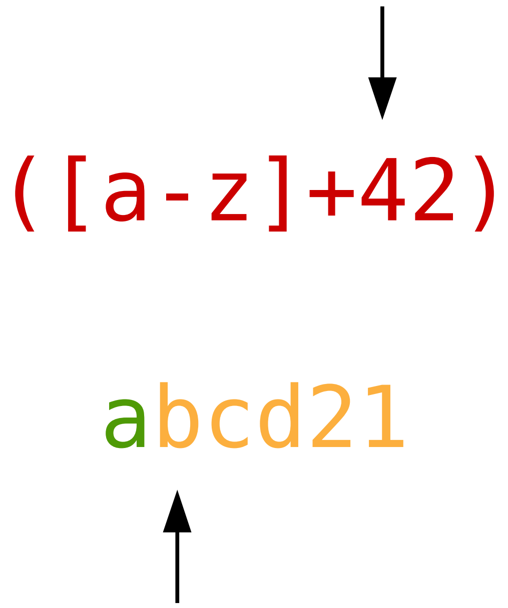
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

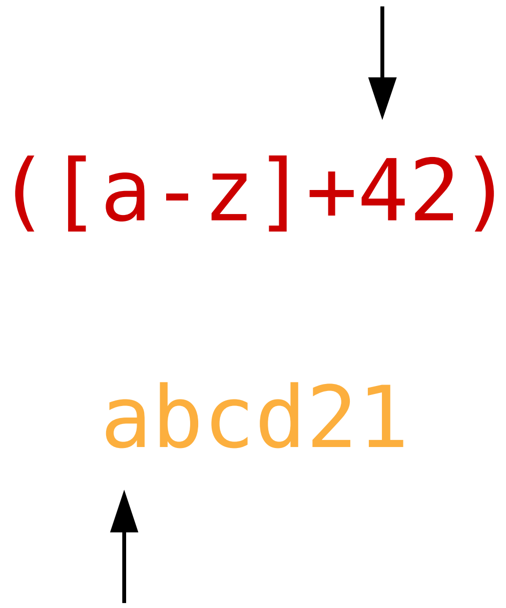
Atomic Groups



```
([a-z]+42)
```

```
abcd21
```

Atomic Groups



`([a-z]+42)`

`abcd21`

Atomic Groups

```
([a-z]+42)
```

```
abcd21
```
✗

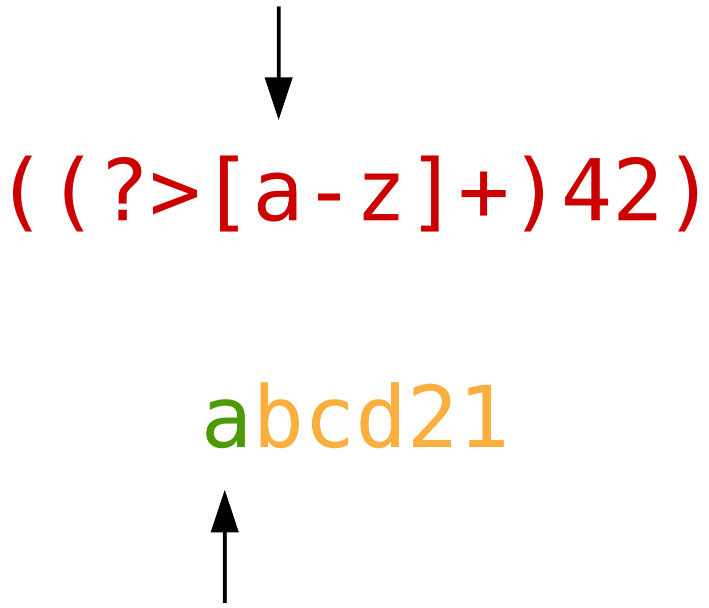
Atomic Groups

```
((?>[a-z]+)42)
```

```
abcd21
```

- Atomic Groups are enabled using a Subpattern option

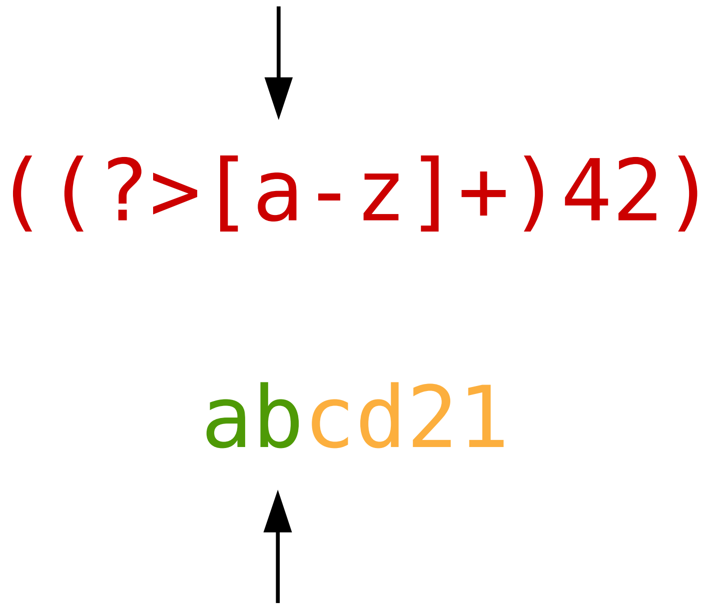
Atomic Groups



```
((?>[a-z]+)42)
```

abcd21

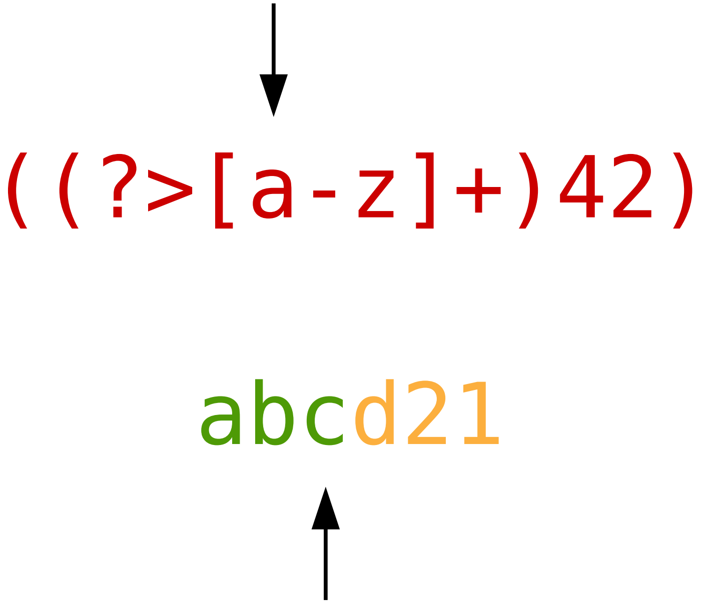
Atomic Groups



((?>[a-z]+)42)

abcd21

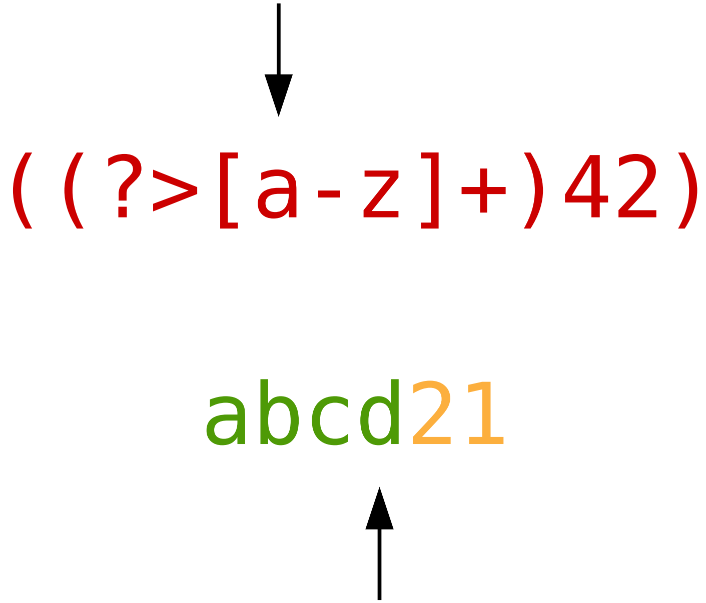
Atomic Groups



```
((?>[a-z]+)42)
```

```
abcd21
```

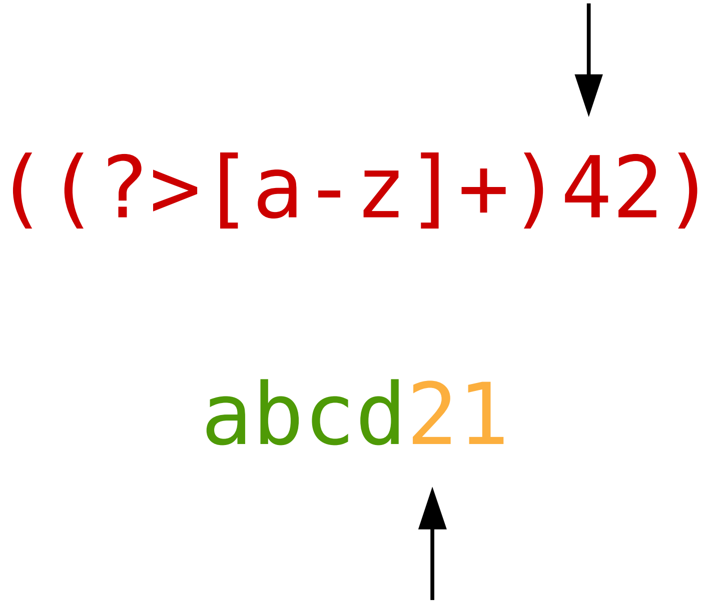
Atomic Groups



```
((?>[a-z]+)42)
```

```
abcd21
```

Atomic Groups



```
((?>[a-z]+)42)
```

```
abcd21
```

Atomic Groups

```
((?>[a-z]+)42)
```

```
abcd21    ✗
```

- No backtracking allowed for this subpattern, therefore immediate abort

Atomic Groups

- Backtracking may be prohibited on a per quantifier basis as well
  - + Possessive Quantifier

```
([a-z]++42)
```

Performance

- PCRE has a default limit of backtracking steps to use

- Can be configured while compiling the library (Default: 1,000,000)

- Can be configured in certain runtime environments

Thanks for your attention.

Jakob Westhoff
Mail: jakob@qafoo.com
Twitter: @jakobwesthoff

Qafoo
passion for software quality