

**And now for something
completely different**

@DannyEngelbarts
github.com/engelbarts





And now for something completely different

@DannyEngelbarts
github.com/engelbarts

!!! WARNING !!!



!! WARNING !!

**This presentation
may contain traces of
Perl**

!! WARNING !!

This presentation

~~may~~ **contain** ~~s~~ **traces of**

Perl

Regular Expressions In Python

**Everything
you know about
Regular Expressions
is wrong**

A little history

50

50

Regular Events

50
Regular Events

60

50
Regular Events

60



50
Regular Events

60
QED text editor

50
Regular Events

60
QED text editor

70

50
Regular Events

60
QED text editor

70
UNIX

50
Regular Events

60
QED text editor

70
ed
UNIX

50
Regular Events

60
QED text editor

70
UNIX ed grep

50
Regular Events

60
QED text editor

70
ed
UNIX g/re/p

50

Regular Events

60

QED text editor

70

awk

ed

UNIX

g/re/p

50

Regular Events

60

QED text editor

sed
awk ed
UNIX g/re/p

50

Regular Events

60

QED text editor

70

sed
awk **ed**
UNIX **g/re/p**
emacs

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **g/re/p**

emacs

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **j/p**
emacs

V

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **vi** **e/p**
emacs

80

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **j/p**
emacs **vi**

80

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **vi** **l/p**
emacs

80

Larry Wall

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **j/p**
emacs

vi

80

That evil person

50

Regular Events

60

QED text editor

70

sed
awk **ex** **ed**
UNIX **/e/p**
emacs **vi**

80

PERL
That evil person

50
Regular Events

60
QED text editor

sed
awk
ex
ed
UNIX
emacs
vi
j/p

PCRE
80
That evil person

90

90

Python



Python

regex

regexp

API v1

regex

regex

API v2

regex

regex_syntax

regex

regex_syntax

awk

regex

regex_syntax

awk
grep

regex

regex_syntax

awk

grep

egrep

regex

regex_syntax

awk
grep
egrep
emacs

regex

regex_syntax

awk

grep

egrep

emacs (default!)

re

re

API v3

re → re1

re

Version 2

re

pre

re

pre

PCRE

re

pre

sre

re

pre

sre

re

sre


re

sre

re




**Everything
you know about
Regular Expressions
is wrong**



**Everything
you know about
Regular Expressions
is wrong**

Damian Conway



**Everything
you know about
Regular Expressions
is wrong**

Another Evil Person

Not your fault.

**“A regular expression is a
declarative specification describing
the textual structure to which a
matching string must conform”**

**A regular expression
Is not declarative**

**A regular expression
Is not descriptive**

**A regular expression
Does not specify structure**

**A regular expression
Does not conform**

**“A regular expression is a
<SOMETHING> <SOMETHING>
<SOMETHING-ing> the <SOMETHING>
to which a <SOMETHING> string must
<SOMETHING>”**

“A regular expression is NOT a declarative specification describing the textual structure to which a matching string must conform”

A regular expression is code

**“A regular expression is a
specification of a block-structured
instruction sequence, which is designed to
execute some task on a highly specialised
virtual machine”**

code

Commands
Loops
Assertions
Exception Handling

code

rotten code

code

**r"regex string" is a function
by itself**

**a function that is executed
by the regex engine**

**a function that returns
a result**

True or False

and might do extra's

regexes are code

**To understand regexes you
just need to learn the language**

**And learn the execution
model**

**Regexes run in their
own little world**

Regexes run in their
own little ~~world~~ “vm”

**Theoretically ... a
Finite State Machine**

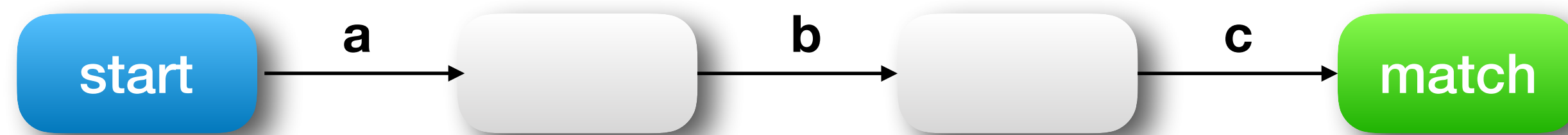
**In practice ... a
stack-based machine**

**In practice ... a
stack-based machine
like everything else**

**Theoretically ... a
Finite State Machine**

r"abc"

r"abc"



```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```



```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

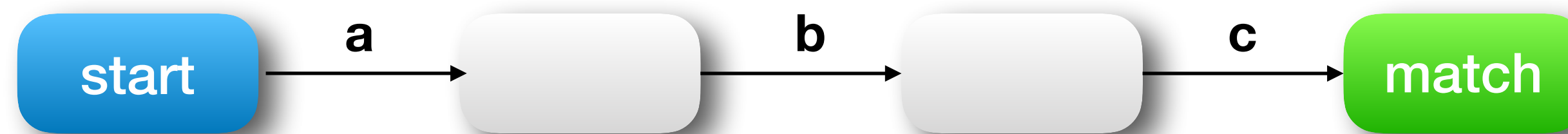
```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

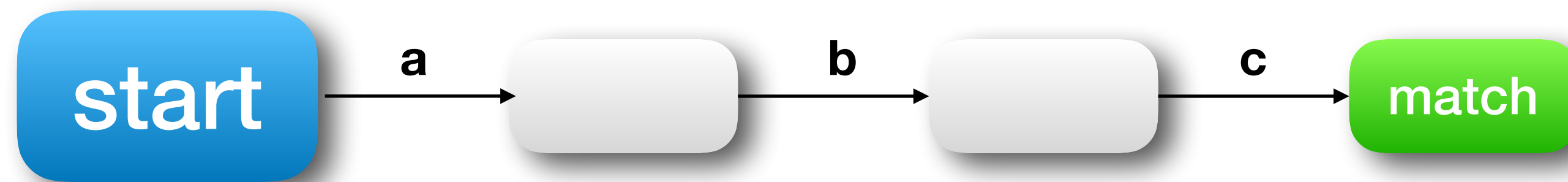
```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

```
def match(myStr):  
    for index in range(len(myStr)):  
        try:  
  
            if myStr[index] != 'a':  
                raise  
  
            if myStr[index+1] != 'b':  
                raise  
  
            if myStr[index+2] != 'c':  
                raise  
  
            return True  
  
        except:  
            continue # backtracking  
    return False
```

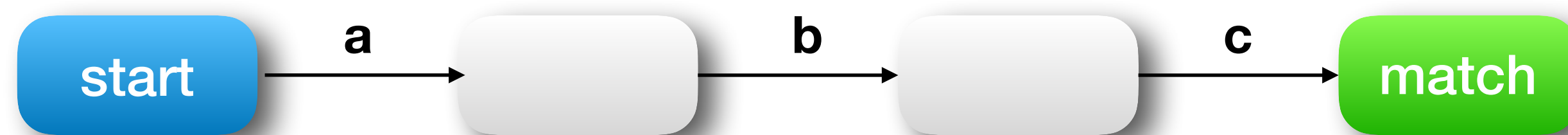
```
re.search(r"abc", "12ababc")
```



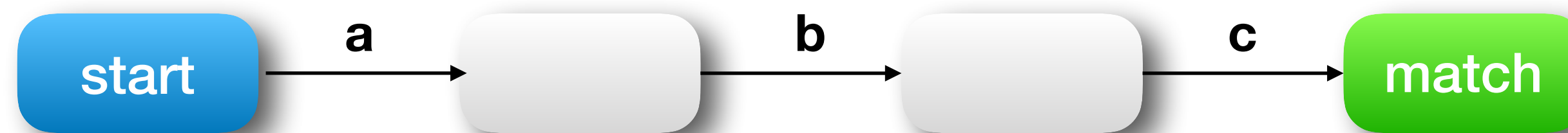
```
re.search(r"abc", "12ababc")
```



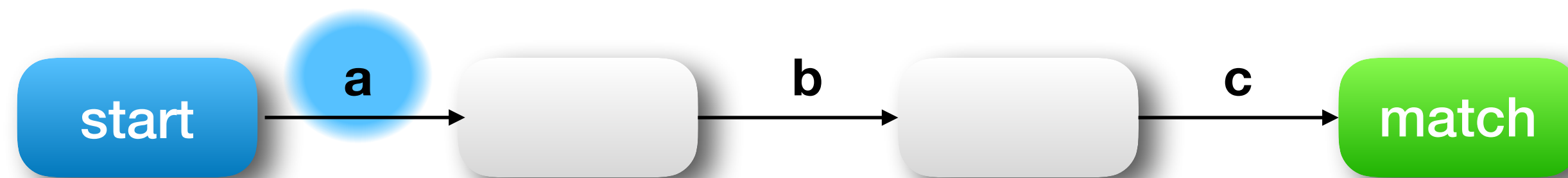

```
re.search(r"abc", "12ababc")
```



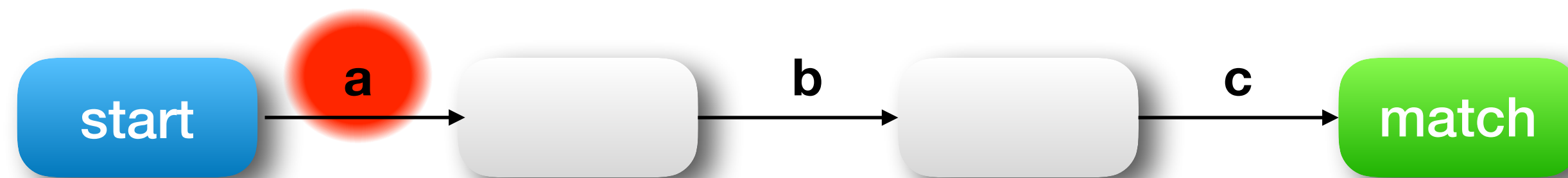
```
re.search(r"abc", "12ababc")
```



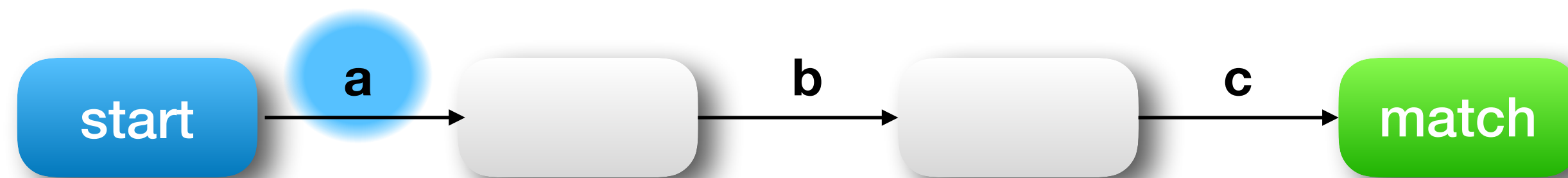
```
re.search(r"abc", "12ababc")
```



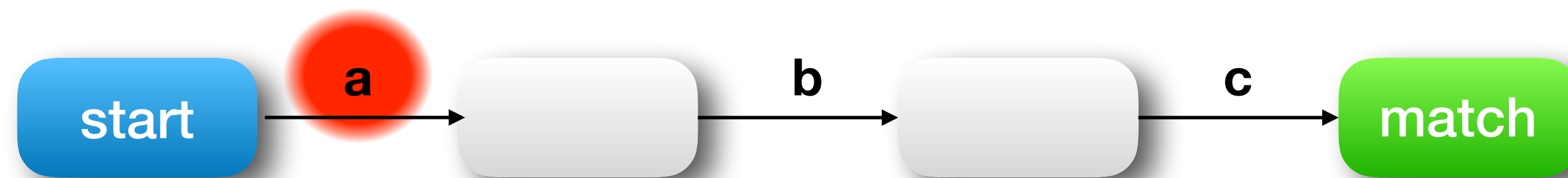
```
re.search(r"abc", "12ababc")
```



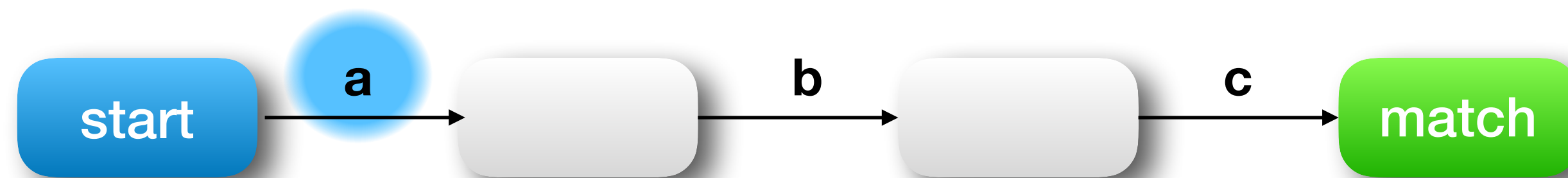
```
re.search(r"abc", "12ababc")
```



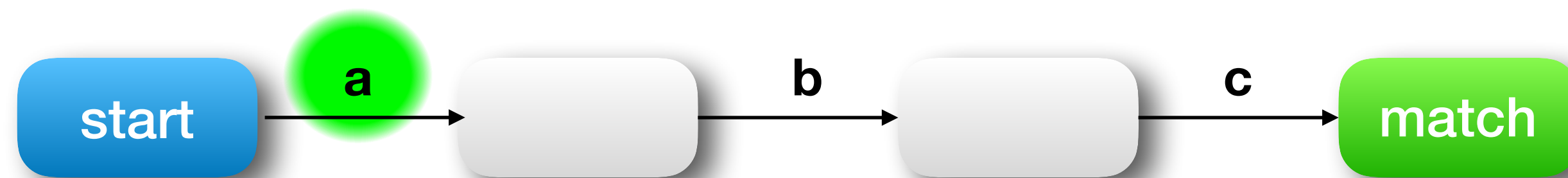
```
re.search(r"abc", "12ababc")
```



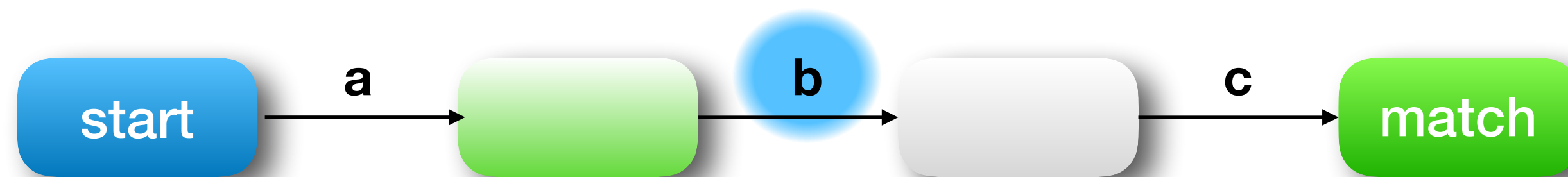
```
re.search(r"abc", "12ababc")
```



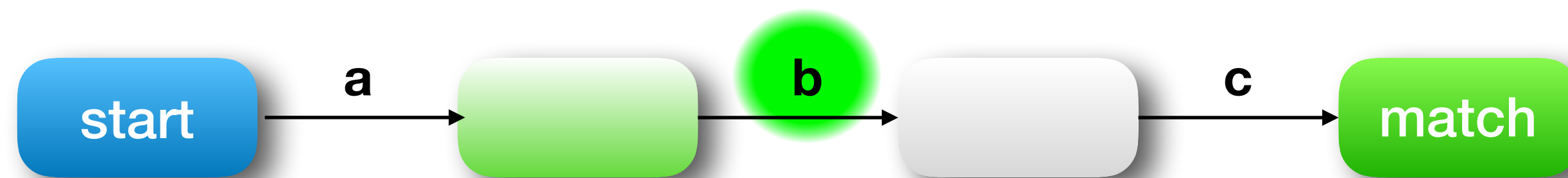
```
re.search(r"abc", "12ababc")
```



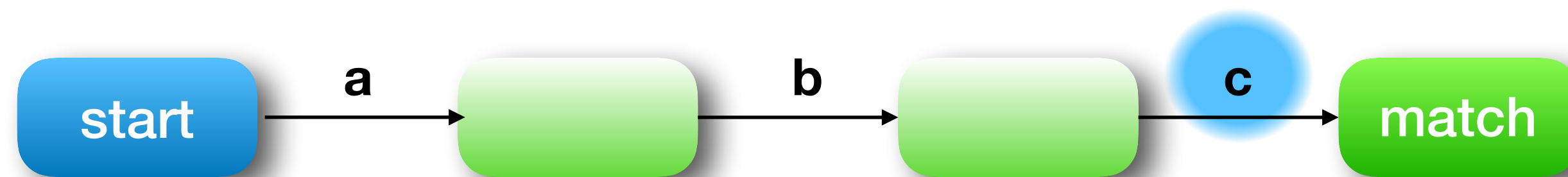

```
re.search(r"abc", "12ababc")
```



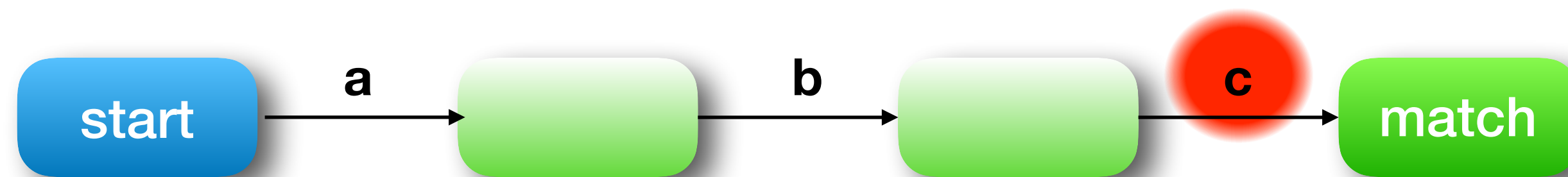
```
re.search(r"abc", "12abc")
```



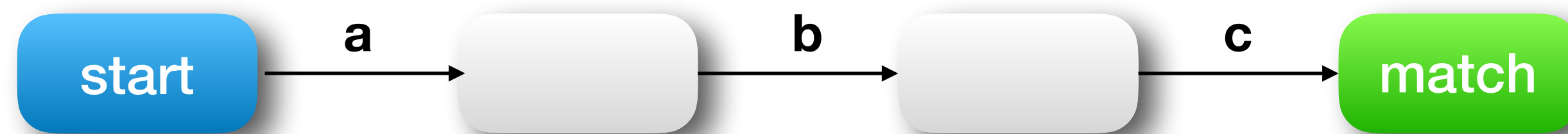
```
re.search(r"abc", "12ababc")
```



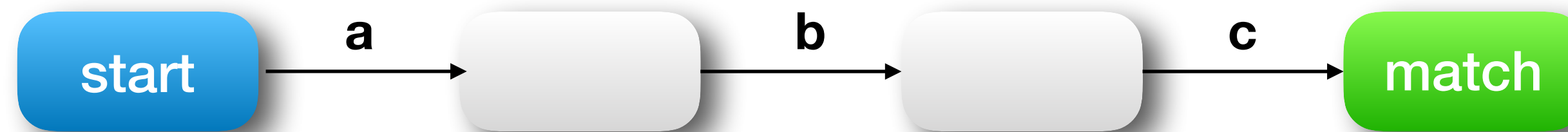
```
re.search(r"abc", "12ababc")
```



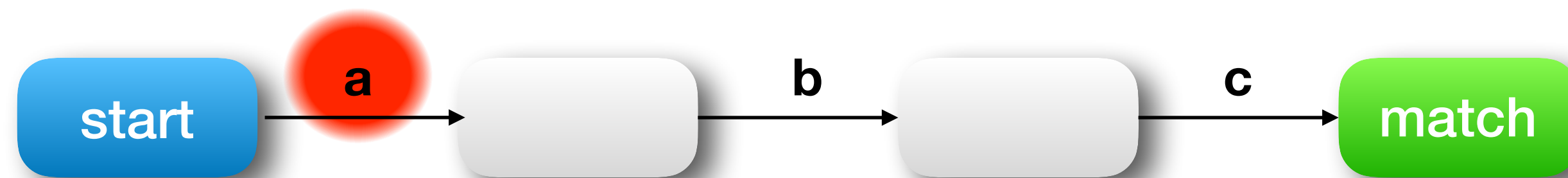
```
re.search(r"abc", "12ababc")
```



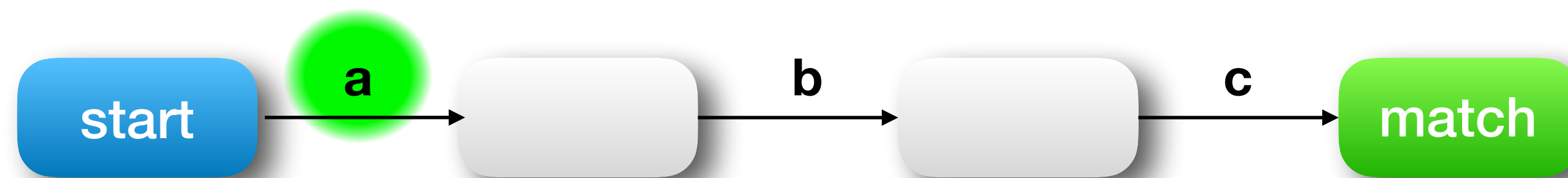
```
re.search(r"abc", "12ababc")
```



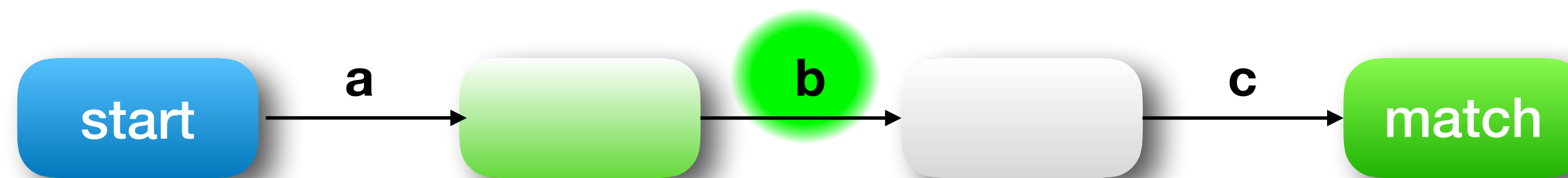
```
re.search(r"abc", "12abc")
```



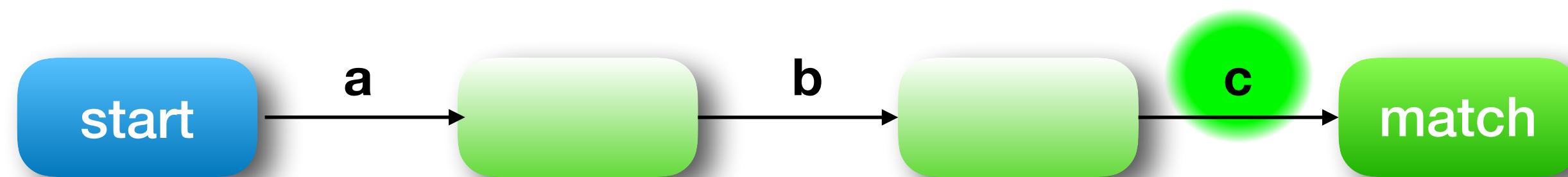
```
re.search(r"abc", "12ababc")
```



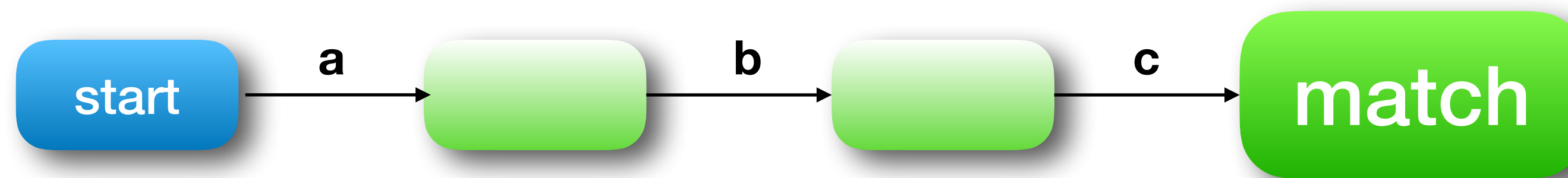

```
re.search(r"abc", "12ababc")
```



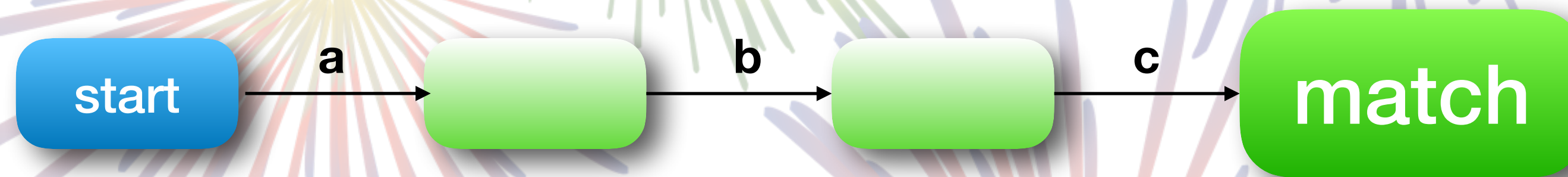
```
re.search(r"abc", "12ababc")
```



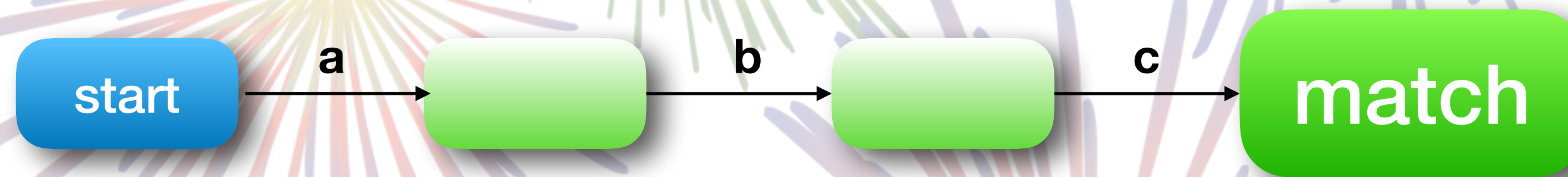
```
re.search(r"abc", "12ababc")
```



`re.search(r"abc", "12ababc")`



`re.search(r"abc", "12ababc")`



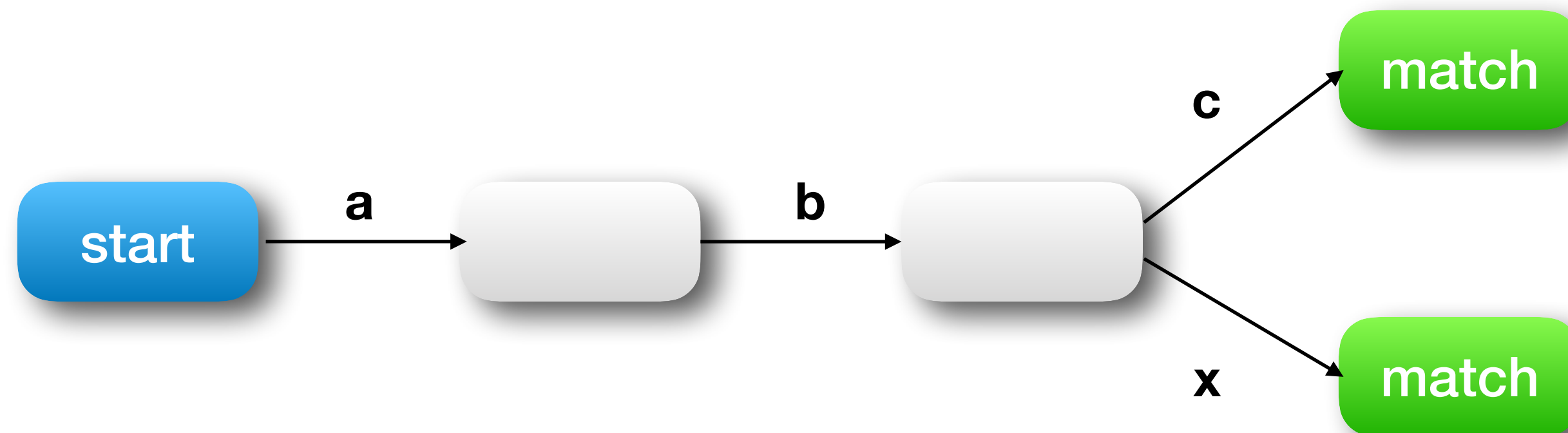
TADA.WAV

That's it.

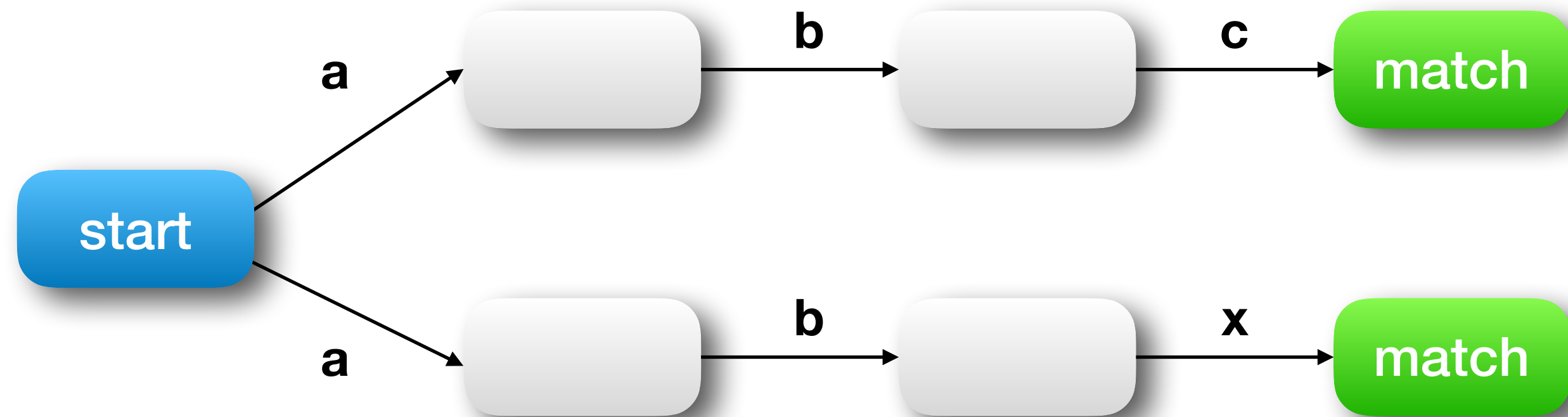
That's it.
More or less.

r"abc | abx"

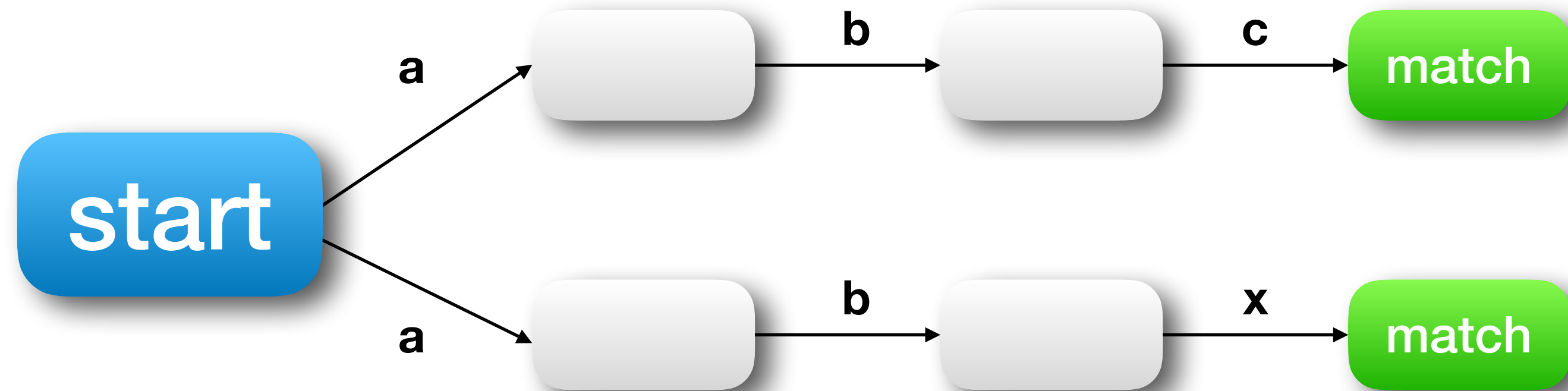
$r"abc \mid abx"$



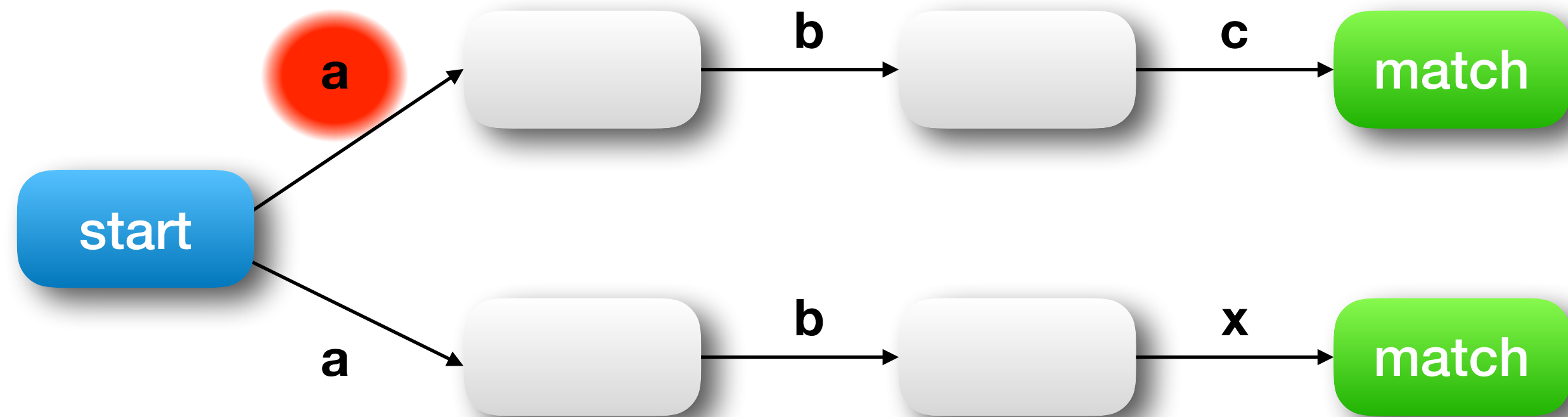
$r"abc \mid abx"$



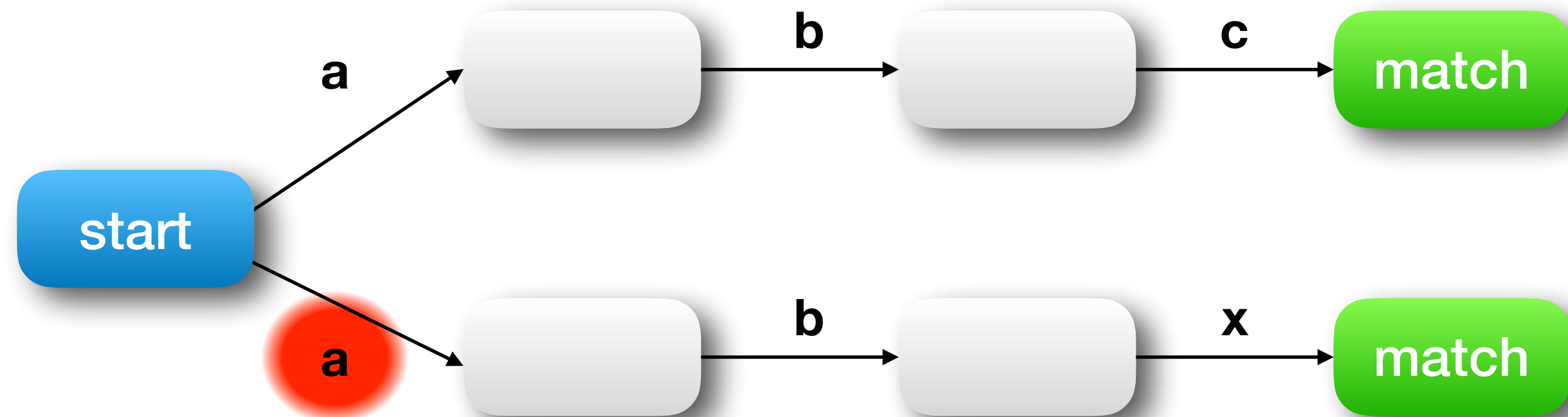
```
re.search(r"abc|abx", "pabx")
```



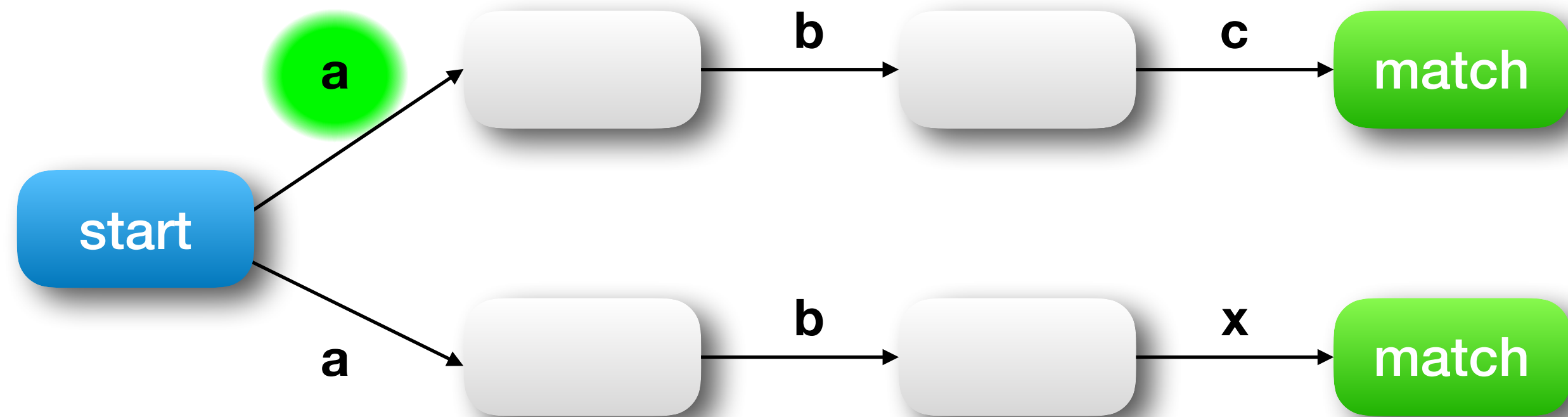
```
re.search(r"abc|abx", "pabx")
```



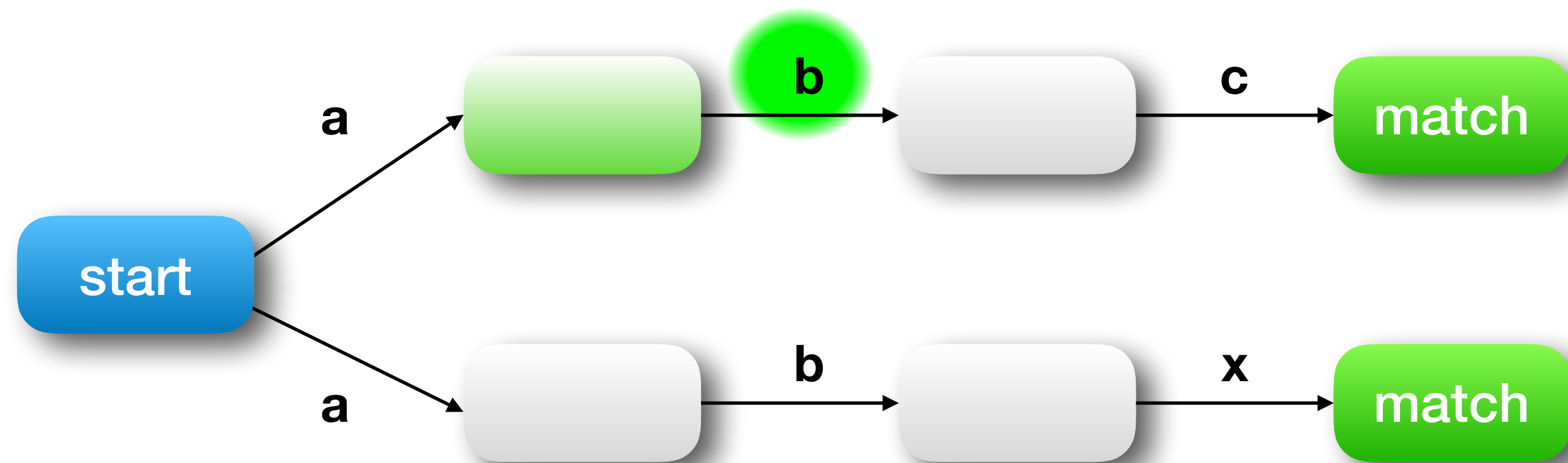
```
re.search(r"abc|abx", "pabx")
```



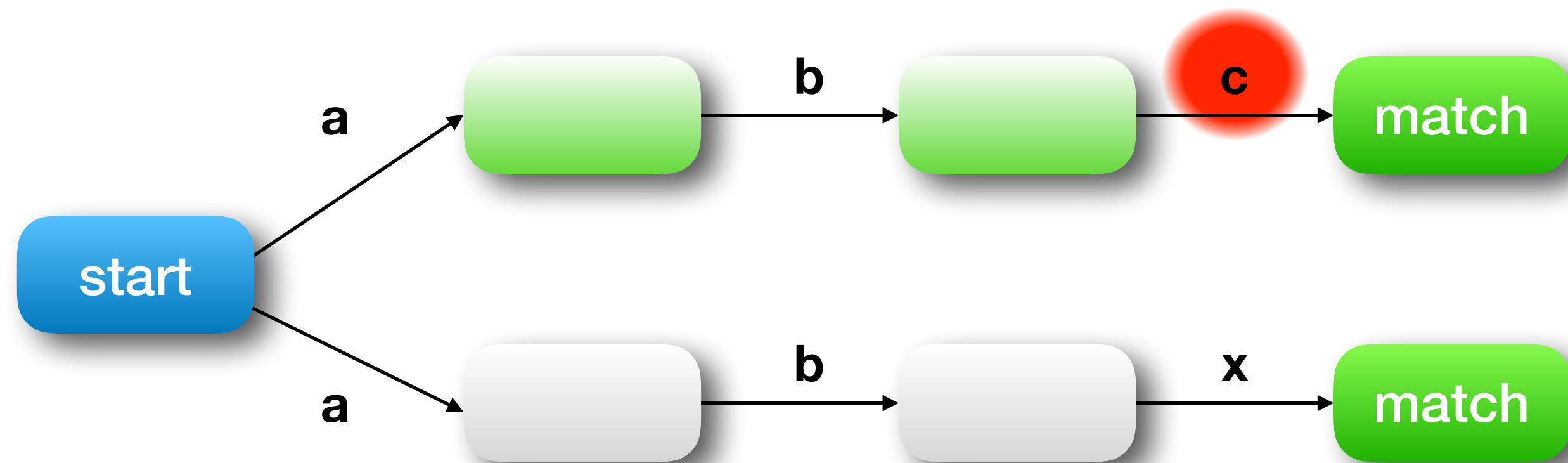
```
re.search(r"abc|abx", "pabx")
```



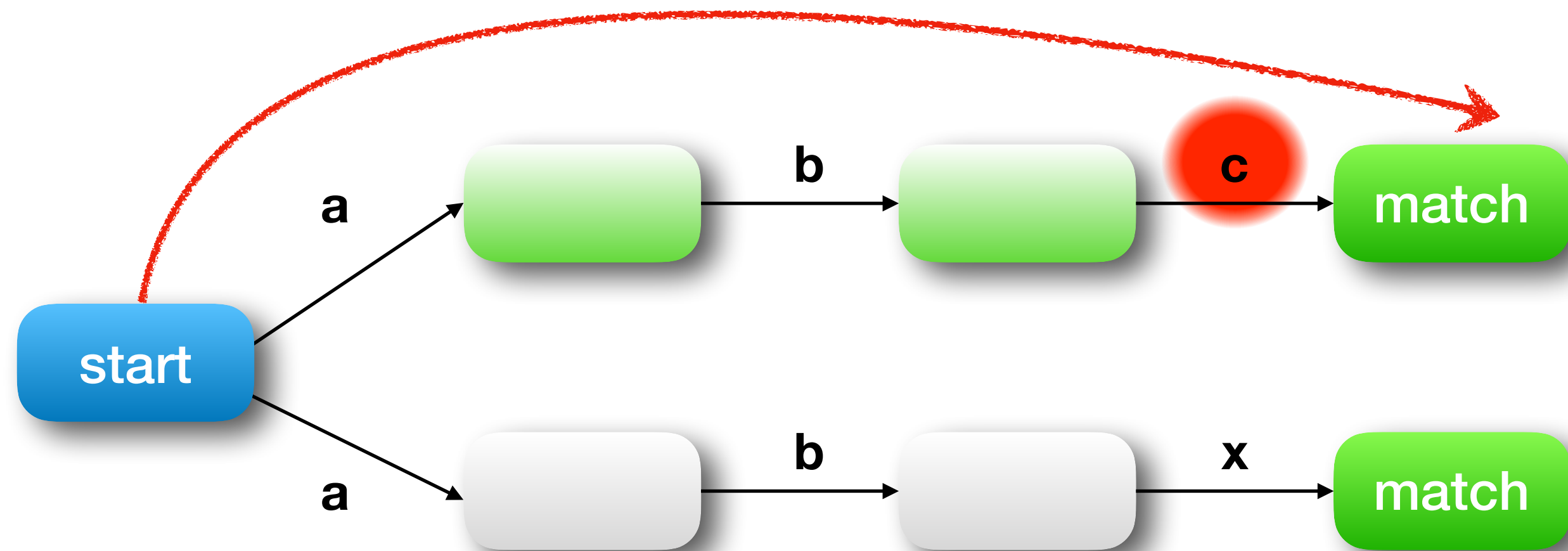
```
re.search(r"abc|abx", "pabx")
```



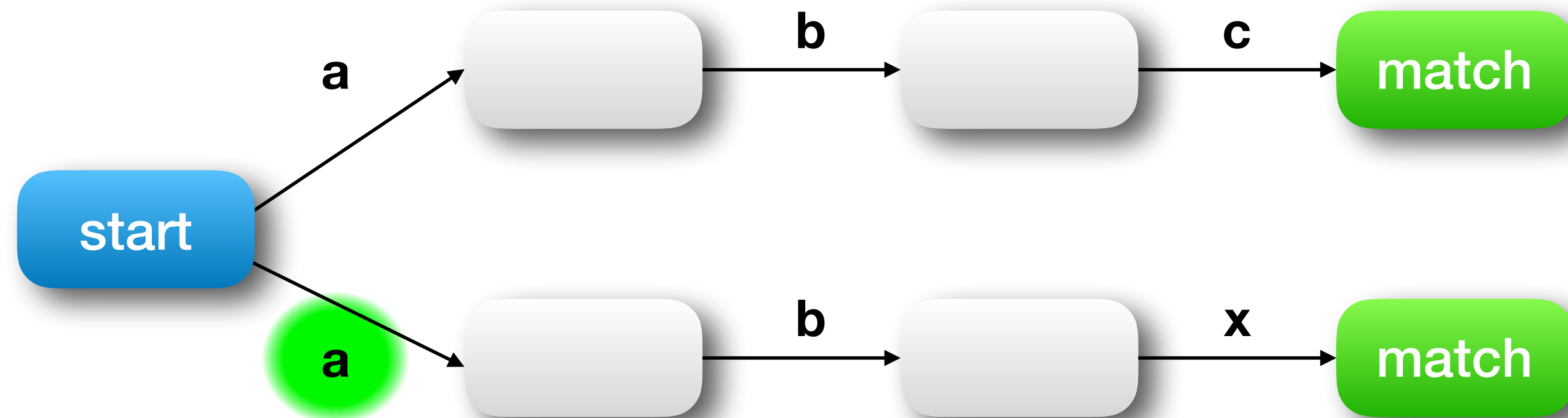
```
re.search(r"abc|abx", "pabx")
```



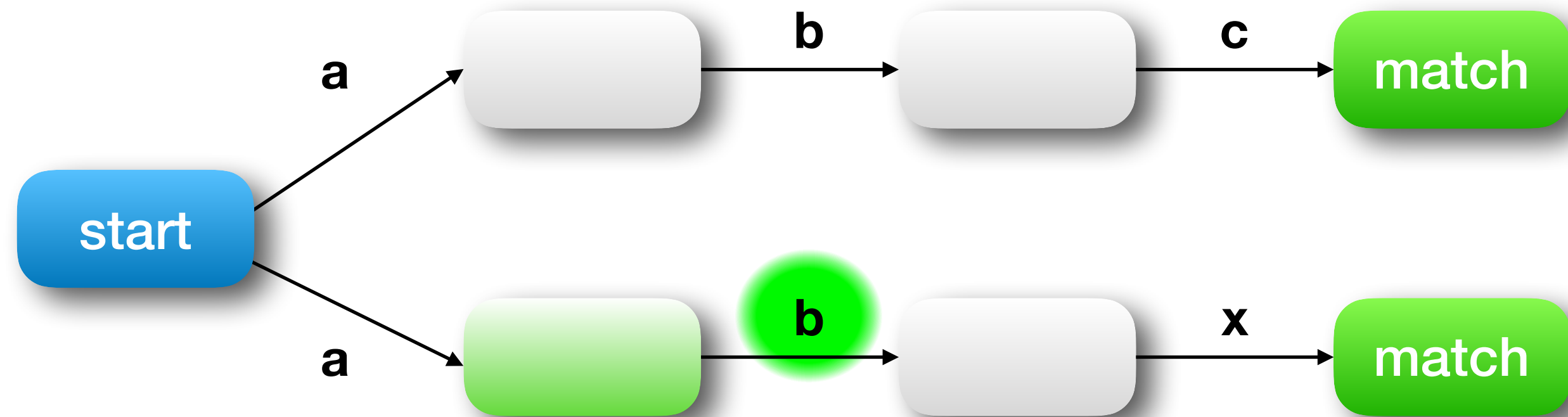

```
re.search(r"abc|abx", "pabx")
```



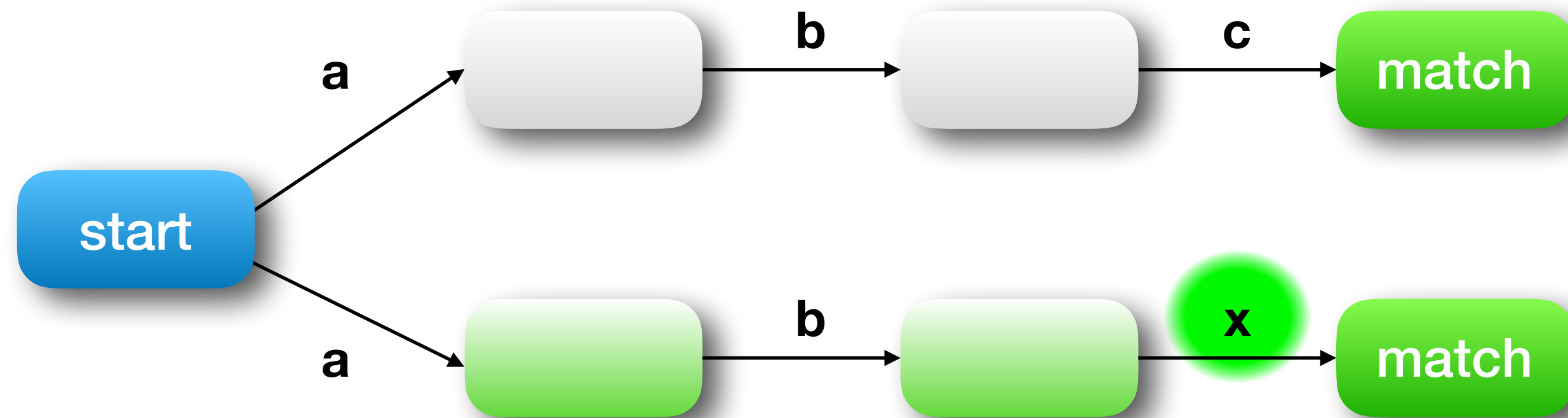
```
re.search(r"abc|abx", "pabx")
```



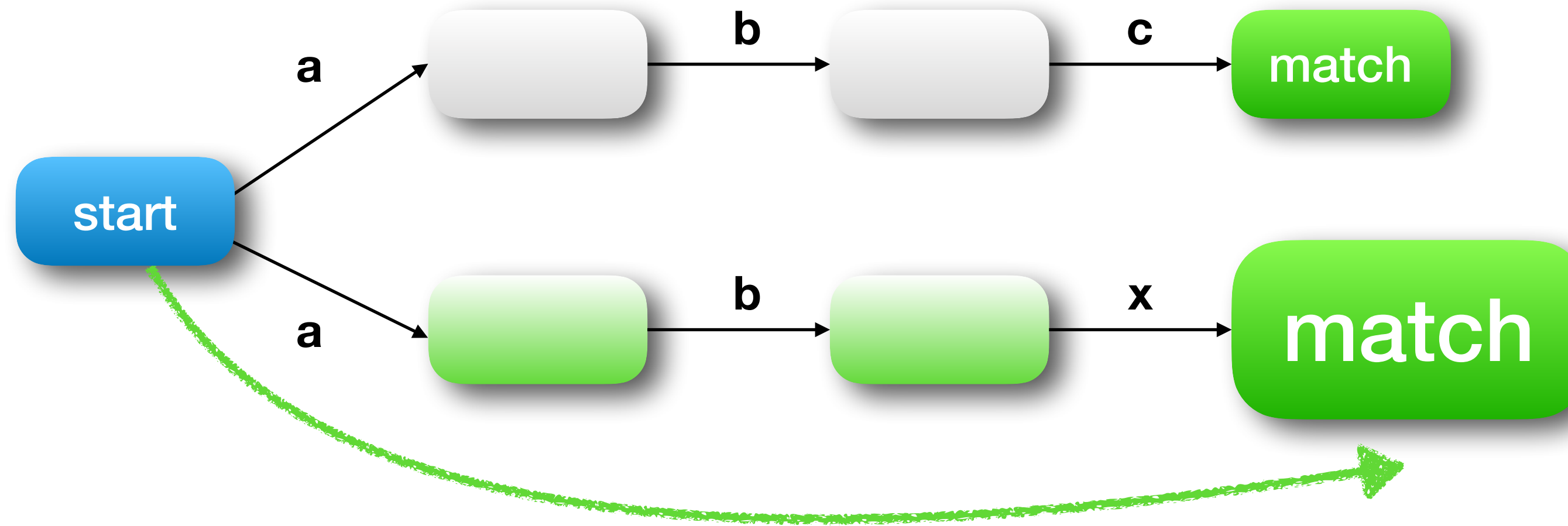
```
re.search(r"abc|abx", "pabx")
```



```
re.search(r"abc|abx", "pabx")
```



```
re.search(r"abc|abx", "pabx")
```



Regex Execution

Regex Execution

Try every path

Regex Execution

Try leftmost first

Regex Execution

Try leftmost first

The upper path in the graph

Regex Execution

Success on the first full match

Regex Execution

Fail when all paths fail

Regex Execution

Try all paths before moving along

```
r = re.compile(r"anteater|antelope|ant")  
r.search("An ant encountered an anteater")
```

```
r = re.compile(r"anteater|antelope|ant")  
r.search("An ant encountered an anteater")
```

```
r = re.compile(r"ant|anteater|antelope")  
r.search("An anteater encountered an ant")
```

```
r = re.compile(r"ant|anteater|antelope")  
r.search("An anteater encountered an ant")
```


Regex Execution

Looks for the first match

Does not look for the best match

Regex

Command language

All alphanumeric characters

All alphanumeric characters

Does the current character match me?

All alphanumeric characters

Does the current character match me?

 **Next**

All alphanumeric characters

Does the current character match me?



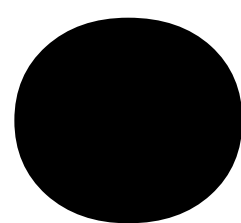
Next



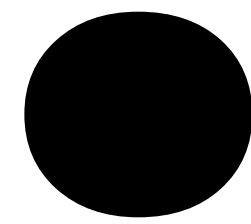
Backtrack

All alphanumeric characters

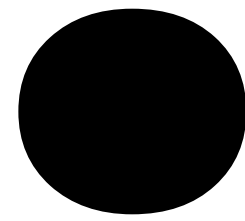
**All alphanumeric characters
and some punctuation**



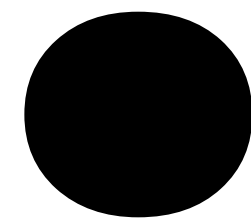
.Dot



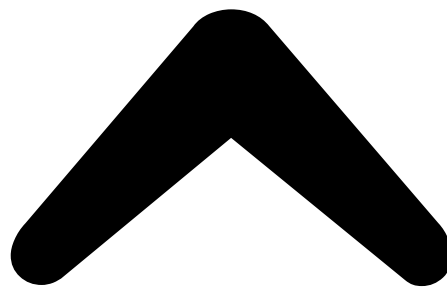
Match a “. ”

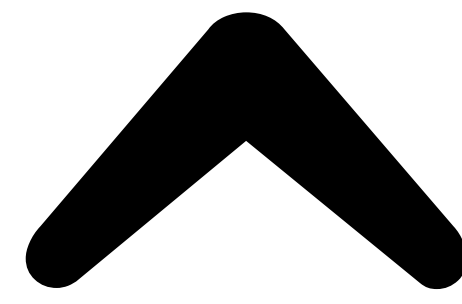


Match a “.” or ANY other char



**Match a “.” or ANY other char
Except Newline**





Matches the start

\$

\$

Matches the end

OR

[OR]

[OR]

Match an “O” or an “R”

[a - z]

[a - z]

Matches a to z

[a - z]

**Matches a to z
(lowercase)**

[-az]

[- a z]

Matches “_” or “a” or “z”

- [a z]

Matches “_” and “a” or “z”

[0 - 9 A - F a - f]

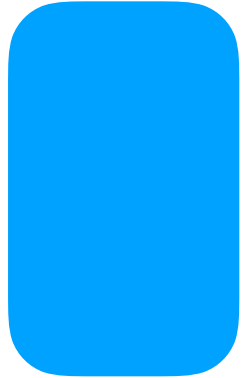
[0-9A-Fa-f]

[0-9A-Fa-f]

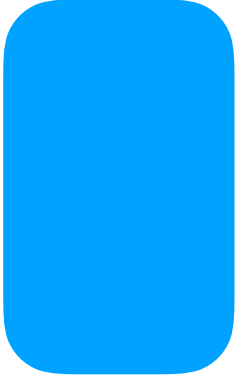
[0-9A-Fa-f]

[0-9 A-F a-f]

[0-9 A-F a-f]

[0-9  A-F a-f]

[0-9 A-F a-f]

[0-9 A-F  a-f]

[0-9 A-F a-f]

NOR

[^OR]

[^OR]

**Match anything but an
“O” or an “R”**

Loops

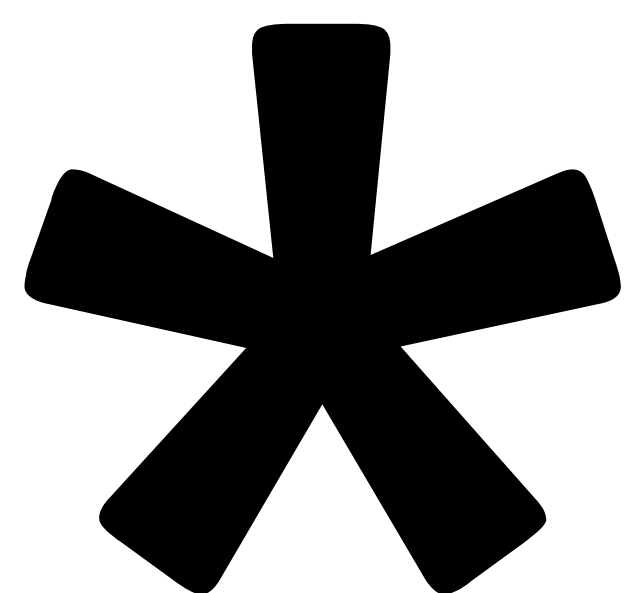
Loops??

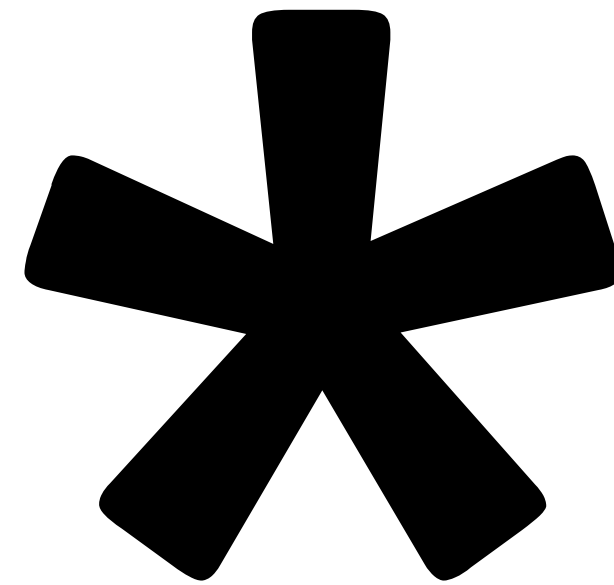
Loops

Like `while` and `for` in Python

**In regex all loops are
both `while` and `for`**

Loop **while** no exception,
but only **for** M to N iterations





**Match previous
zero or more times**

a*

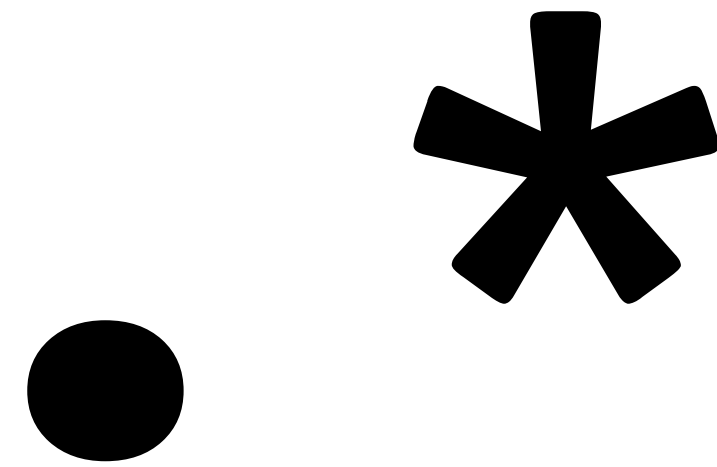
**Match an 'a'
zero or more times**

(ab)*

**Match an 'a' and 'b'
zero or more times**

[a b] *

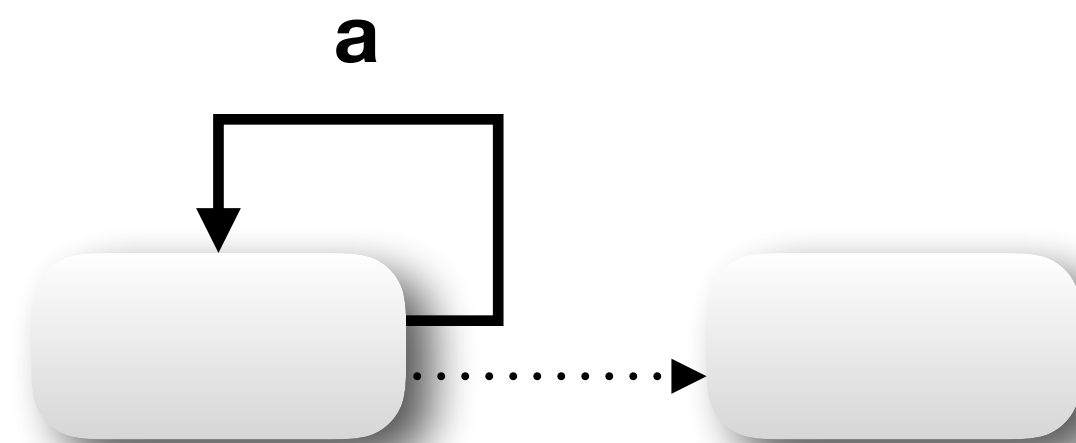
**Match either 'a' or 'b'
zero or more times**



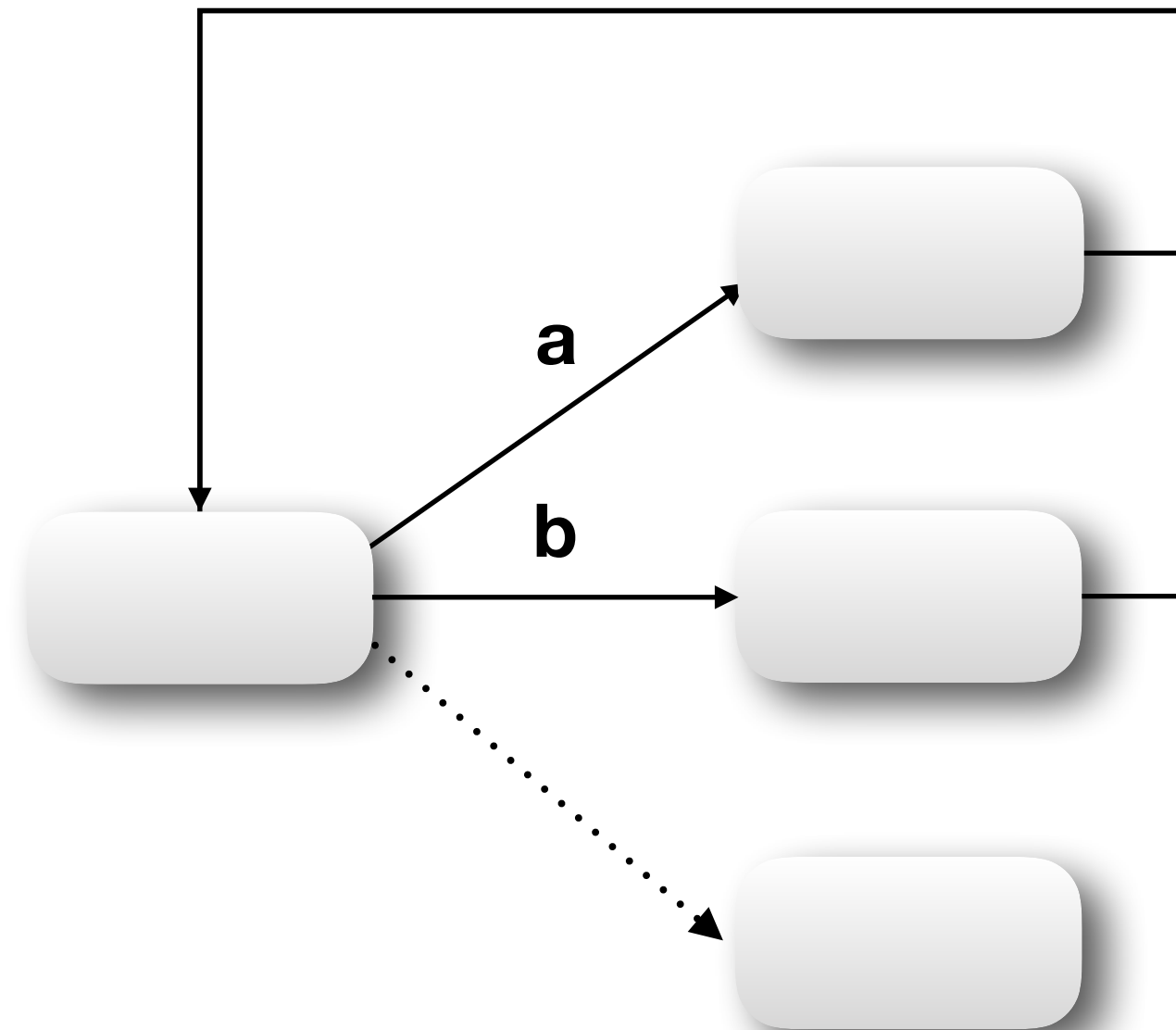
**Match anything
zero or more times**

$r''a^*$

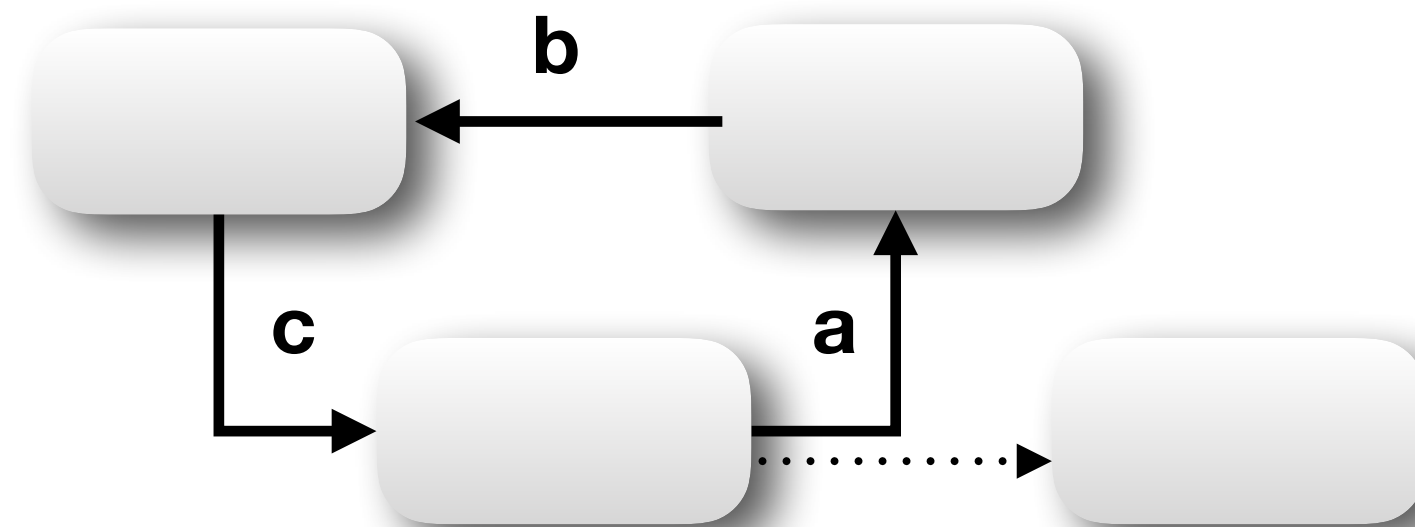
r^*a^*

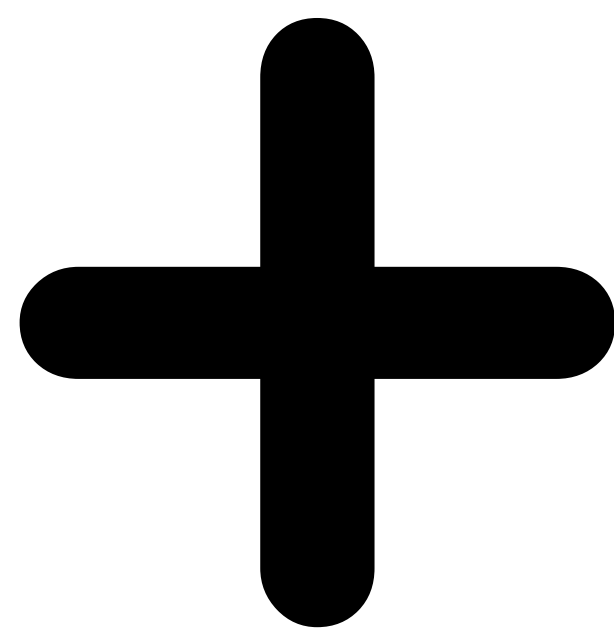


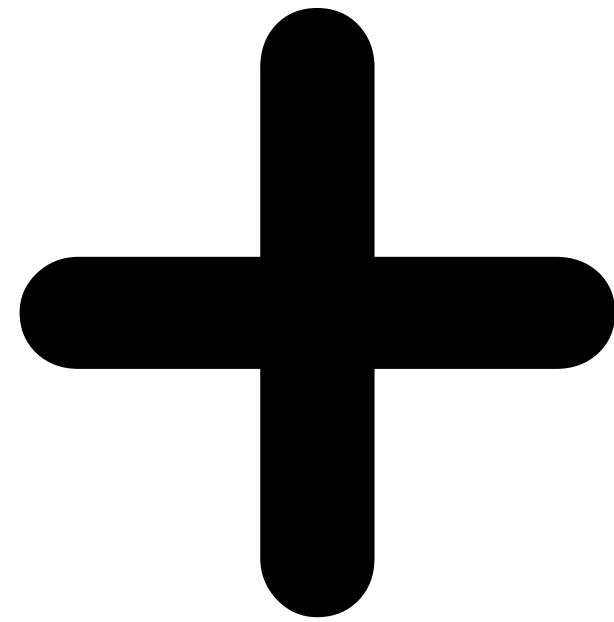
$r''[ab]^*$



$r''(abc)^*$

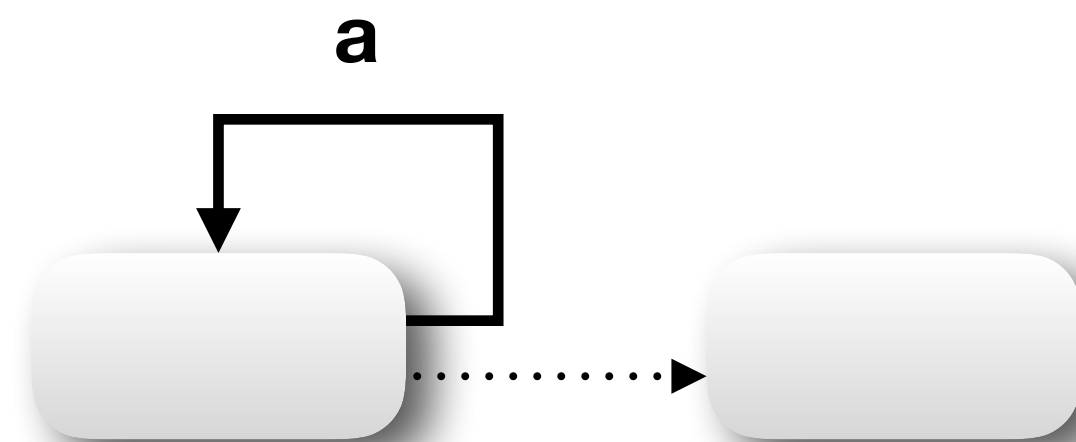




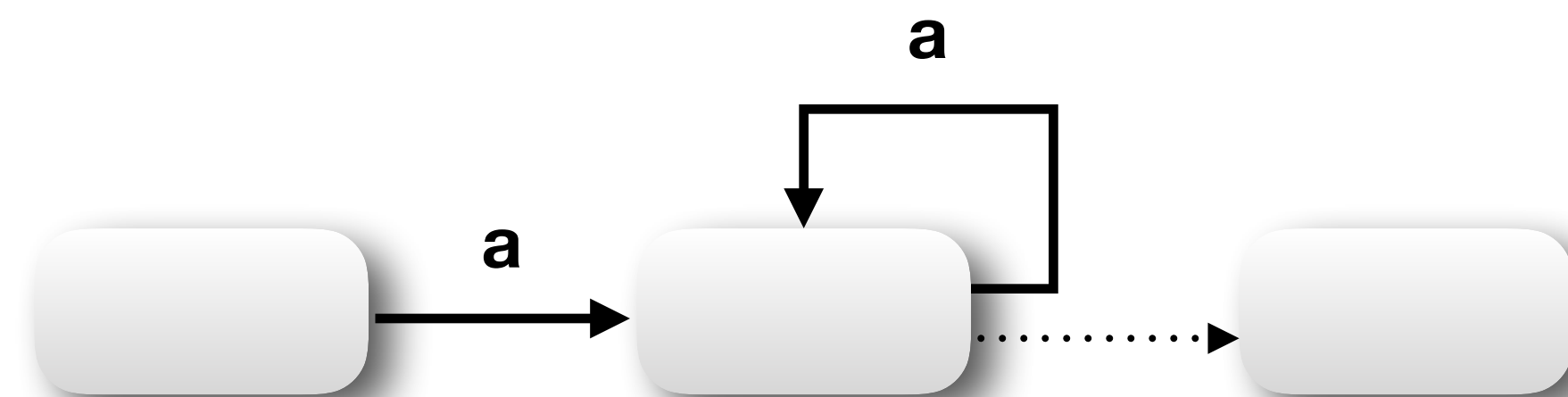


Match one or more

`r" a +"`



r^+a^+



?

?

Match zero or one

{m}

{ m }

Match m times

{m, n}

{ m , n }

Match m to n times

Regex loops can *unloop*

**Roll back previous
loop match**

Whut?

“For **example**.”

“For **example.**”

$r "< . * > "$

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r" < . * > "

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r" < . * > "

“For **example.**”

r" < . * > "

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example**.”

$r "< . * > "$

“For **example**.”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example**.”

r "<.*>"

“For **example.**”

r "<.*>"

“For **example.**”

$r "< . * > "$

Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Cras semper
auctor neque vitae tempus quam pellentesque
nec. Ornare suspendisse sed nisi lacus.
Cursus vitae congue mauris rhoncus. Egestas
egestas fringilla phasellus faucibus scelerisque
eleifend donec pretium. Senectus et netus et
malesuada fames ac. Malesuada pellentesque elit
eget gravida cum sociis natoque. Commodore nulla
facilisi nullam vehicula ipsum a. Nunc mattis enim
ut tellus. Sed vulputate odio ut enim.

Lorem ipsum dolor sit amet, consectetur
adipiscing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Cras semper
auctor neque vitae tempus quam pellentesque
nec. Ornare **suspendisse** sed nisi lacus.
Cursus vitae congue mauris rhoncus. Egestas
egestas fringilla phasellus faucibus scelerisque
eleifend donec pretium. Senectus et netus et
malesuada fames ac. Malesuada pellentesque elit
eget gravida cum sociis natoque. Commodore nulla
facilisi nullam vehicula ipsum a. Nunc mattis enim
ut tellus. Sed vulputate odio ut enim.

Minimal loops

Minimal loops

`r"<.*?>"`

“For **example**.”

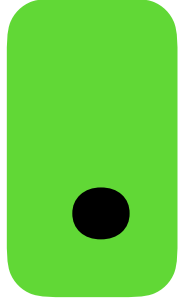
“For **example.**”

r“<. * ?>”


“For **example.**”

r“< . * ? >”

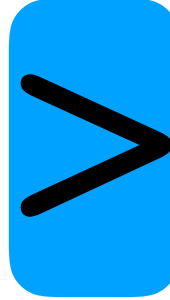
“For  **example.**”

r" < * ? > "


“For  **example.**”

r" < .  > "

“For   example.”

r" < . * ?  "

“For   example.”

r" < . * ?  "

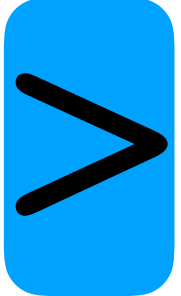
“For **example.**”

r" < . * ? > "

“For **example.**”

r" < . * ? > "

“For example.”

r" < . * ?  "

“For **example.**”

r" < . * ? > "

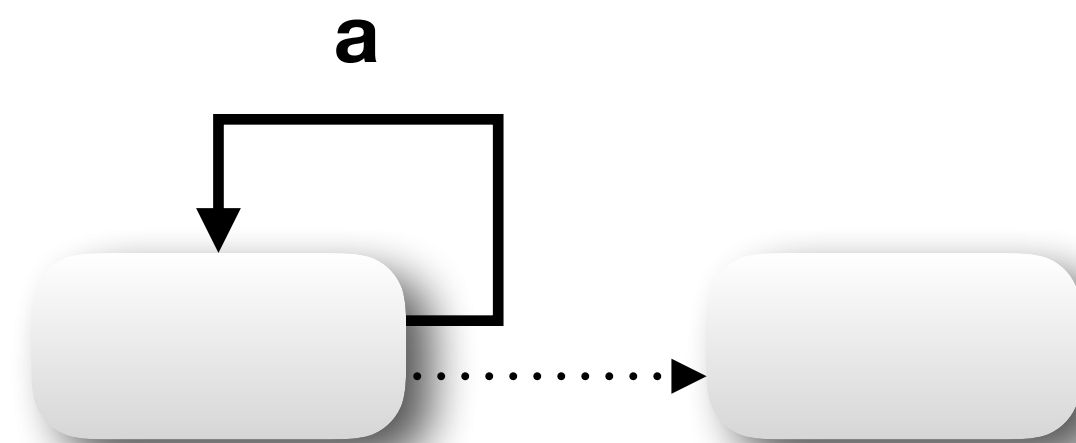
“For **example.**”

r"<.*?>"

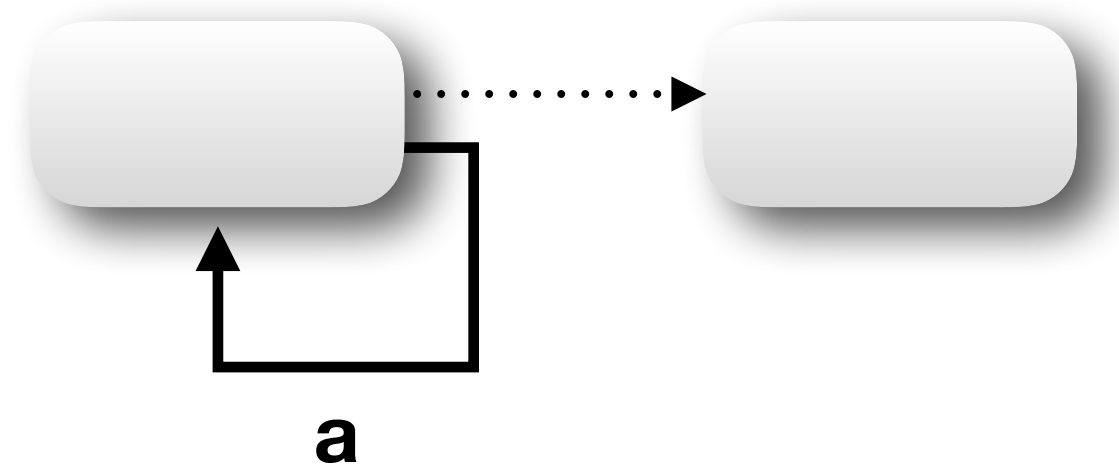
Minimal loops

**Execute the commands in the
loop in as few times as possible**

`r"a*?"`



`r"a*?"`



Minimal or Maximal?

Minimal or Maximal?

First: Use the one that produces
the expected result

Minimal or Maximal?

***Second:* Use the one that does
the least backtracking**

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. Cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. Cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium. # Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

1" # . * # "

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

r" # . * # "

633 ns ± 4.41 ns

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

r"#.*#"

633 ns ± 4.41 ns

r"#.*?#"

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

$r''\# \cdot * \#''$ 633 ns \pm 4.41 ns

$r''\# \cdot * ? \#''$ 1.79 μ s \pm 10.7 ns

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

`r"#.*#" 633 ns ± 4.41 ns`

`r"#.*?#" 1.79 μs ± 10.7 ns`

`r"#[^#]*#"`

Lorem #ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Cras semper auctor neque vitae tempus quam pellentesque nec. Ornare suspendisse sed nisi lacus. cursus vitae congue mauris rhoncus. Egestas egestas fringilla phasellus faucibus scelerisque eleifend donec pretium.# Senectus et netus et malesuada fames ac. Malesuada pellentesque elit eget gravida cum sociis natoque. Commodore nulla facilisi nullam vehicula ipsum a. Nunc mattis enim ut tellus. Sed vulputate odio ut enim.

<code>r"#.*#"</code>	633 ns ± 4.41 ns
----------------------	------------------

<code>r"#.*?#"</code>	1.79 μs ± 10.7 ns
-----------------------	-------------------

<code>r"#[^#]*#"</code>	429 ns ± 3.89 ns
-------------------------	------------------

$r''\# \cdot * \#''$

`r"#[^#]*#"`

The RE module

The RE module

```
myReg = re.compile(r"regex")
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.search("string")
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.search("string") # Will match anywhere in the string
```

The RE module

```
myReg = re.compile(r"^regex")
```

The RE module

```
myReg = re.compile(r"^regex")
```

```
myReg.search("string")
```


The RE module

```
myReg = re.compile(r"^regex")
```

```
myReg.search("string") # Will match from the start of the string
```

The RE module

```
myReg = re.compile(r"^regex")
```

```
myReg.search("string") # Will match from the start of the string
```

```
myReg = re.compile(r"regex")
```

```
myReg.match("string") # Will match from the start of the string
```

The RE module

```
myReg = re.compile(r"^regex$")
```

```
myReg.search("string") # Will only match the whole string
```

The RE module

```
myReg = re.compile(r"^regex$")
```

```
myReg.search("string") # Will only match the whole string
```

```
myReg = re.compile(r"regex")
```

```
myReg.fullmatch("string") # Will only match the whole string
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.split("string") # Like str.split() but split on regexes
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.split("string") # Like str.split() but split on regexes
```

```
myReg.findall("string") # return a list of matches
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.split("string") # Like str.split() but split on regexes
```

```
myReg.findall("string") # return a list of matches
```

```
myReg.finditer("string") # return an iterator of matches
```

The RE module

```
myReg = re.compile(r"regex")
```

```
myReg.split("string") # Like str.split() but split on regexes
```

```
myReg.findall("string") # return a list of matches
```

```
myReg.finditer("string") # return an iterator of matches
```

```
myReg.sub("repl", "string") # like str.sub() but with regexes
```


The RE module

Special Characters, unicode compatible!

`\d = [0-9]`

Digit

`\D = [^0-9]`

Not a digit

The RE module

Special Characters, unicode compatible!

<code>\d</code>	<code>= [0-9]</code>	Digit
<code>\D</code>	<code>= [^0-9]</code>	Not a digit
<code>\s</code>	<code>= [\t\n\r\f\v]</code>	Any whitespace char
<code>\S</code>	<code>= [^ \t\n\r\f\v]</code>	Not a whitespace char

The RE module

Special Characters, unicode compatible!

\d	= [0-9]	Digit
\D	= [^0-9]	Not a digit
\s	= [\t\n\r\f\v]	Any whitespace char
\S	= [^ \t\n\r\f\v]	Not a whitespace char
\w	= [a-zA-Z0-9_]+	A word
\W	= [^a-zA-Z0-9_]+	Not a word

The RE module

Special Characters, unicode compatible!

\d	= [0-9]	Digit
\D	= [^0-9]	Not a digit
\s	= [\t\n\r\f\v]	Any whitespace char
\S	= [^ \t\n\r\f\v]	Not a whitespace char
\w	= [a-zA-Z0-9_]*	A word
\W	= [^a-zA-Z0-9_]*	Not a word
\b		Word boundary
\B		Not a word boundary

The RE module

Special options or flags

re.A	re.ASCII	(?a)
re.I	re.IGNORECASE	(?i)
re.M	re.MULTILINE	(?m)
re.X	re.VERBOSE	(?x)

The RE module

Special options or flags

re.A re.ASCII (?a)

re.I re.IGNORECASE (?i)

re.M re.MULTILINE (?m)

re.X re.VERBOSE (?x)

re.search(r" (?xmia) regex", "some string")

The RE module

Special options or flags

re.A re.ASCII (?a)

re.I re.IGNORECASE (?i)

re.M re.MULTILINE (?m)

re.X re.VERBOSE (?x)

re.search(r" (?x)mia) regex", "some string")

**re.search(r"regex", "somestring",
 flags=re.A|re.I|re.M|re.X)**

That's it.

That's it.
More or less.

Thank You