Regular Expressions

Simple matching and searching

```
String: My name is Claus
 Regex: My name is
```

Simple matching and searching

```
String: My name is Claus
 Regex: My name is

 Match: My name is Claus
```

Simple matching and searching

```
String: My name is Claus
 Regex: My age is
```

Simple matching and searching

```
String: My name is Claus
 Regex: My age is

 Match: Does not match!
```

Commonly used special symbols in Python regular expressions

Symbol	Meaning
.	matches any character
+	1 or more
*	0 or more
()	capture group
\d	digit
\D	non-digit
\s	whitespace
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric
^	beginning of string
$	end of string

Commonly used special symbols in Python regular expressions

Symbol	Meaning
.	matches any character
+	1 or more
*	0 or more
()	capture group
\d	digit
\D	non-digit
\s	whitespace
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric
^	beginning of string
$	end of string

. matches any character,
+ means one or more

```
String: My name is Claus
 Regex: My .+ is
```

. matches any character,
+ means one or more

```
String: My name is Claus
 Regex: My .+ is

 Match: My name is Claus
```

. matches any character,
+ means one or more

```
String: My  is Claus
 Regex: My .+ is
```

. matches any character,
+ means one or more

```
String: My  is Claus
 Regex: My .+ is

 Match: Does not match!
```

* means zero or more

```
String: My  is Claus
 Regex: My .* is
```

* means zero or more

```
String: My  is Claus
 Regex: My .* is

 Match: My  is Claus
```

Commonly used special symbols in Python regular expressions

Symbol	Meaning
.	matches any character
+	1 or more
*	0 or more
()	capture group
\d	digit
\D	non-digit
\s	whitespace
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric
^	beginning of string
$	end of string

Capture groups

```
String: My name is Claus
 Regex: My name is (.+)
```

Capture groups

```
 String: My name is Claus
  Regex: My name is (.+)

  Match: My name is Claus
Group 1: Claus
```

Commonly used special symbols in Python regular expressions

Symbol	Meaning
.	matches any character
+	1 or more
*	0 or more
()	capture group
\d	digit
\D	non-digit
\s	whitespace
\S	non-whitespace
\w	alphanumeric
\W	non-alphanumeric
^	beginning of string
$	end of string

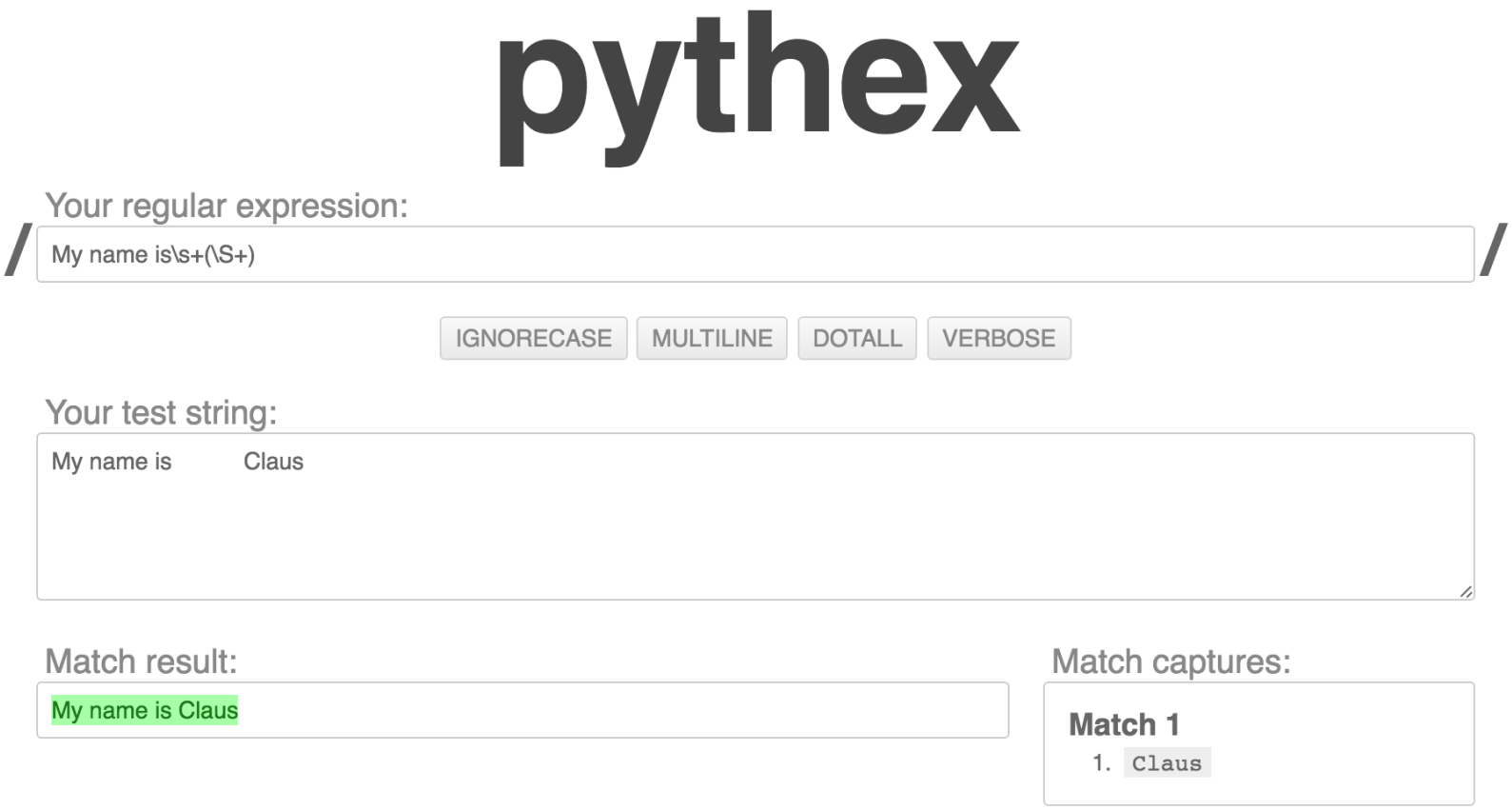
Capture groups

```
 String: My name is     Claus
  Regex: My name is\s+(\S+)

  Match: My name is     Claus
Group 1: Claus
```

Try it yourself

http://pythex.org/



Regular Expressions in Python

Digression: Raw strings

```
In [1]: print("line1\nline2")
Out[1]: line1
        line2
```

How can we print out "line1\nline2"?

```
In [2]: print("line1\\nline2") # escape the backslash
Out[2]: line1\nline2
```

Simpler alternative: use a raw string

```
In [3]: print(r"line1\nline2") # r stands for raw
Out[3]: line1\nline2
```

The key regex function we will use is `re.search()`

```
In [1]: import re  # load regular expression module

        test_string = "My name is Claus"
        match = re.search(r"name", test_string)
        if match: # did we find a match?
            print("Test string matches.")
            print("Match:", match.group())

Out[1]: Test string matches.
        Match: name
```

`match.group()` returns the part of the string that matched

```
In [1]: test_string = "My age is secret."
        match = re.search(r"My \S+ is", test_string)
        print(match.group())

Out[1]: My age is


In [2]: test_string = "My mood is good."
        match = re.search(r"My \S+ is", test_string)
        print(match.group())

Out[2]: My mood is
```

`match.group()` also recovers any captured groups

```
In [1]: test_string = "My age is secret."
        match = re.search(r"My (\S+) is (\S+)", \
                                              test_string)
        print("Match:", match.group(0))
        print("Captured group 1:" , match.group(1))
        print("Captured group 2:" , match.group(2))

Out[1]: Match: My age is secret.
        Captured group 1: age
        Captured group 2: secret.
```