

UCLL
HOGESCHOOL

Regular Expressions
Programming 2

Janne Gilis – Lies Bollens – Ruben Naudts – Bart Thumas – Rob Coekaerts

PBA Toegepaste Informatica – PBA Applied Computer Science
2025 – 2026

Agenda

- Regex
  - Introduction
  - Definition
  - Re-module in Python
  - Patterns & Examples
  - Tips & Tricks

Introduction



- This is not how we will Regex
- *Though at the end of the course, you will know what the cat was looking for*

Definition

- Regex (regular expressions) refers to a powerful and concise language for describing patterns in strings
- The re module in Python
- It's a versatile tool for tasks such as searching for specific patterns, extracting information from strings, or replacing text based on a given pattern

RE Module Python Functions

re.match(pattern, string, flags=0)

Return match for pattern with **beginning** of the string

re.fullmatch(pattern, string, flags=0)

Return match for pattern with the **whole string**

re.search(pattern, string, flags=0)

Return match for pattern **anywhere** in the string

re.findall(pattern, string, flags=0)

Return all **non-overlapping matches** of the pattern in the string

re.sub(pattern, repl, string, count = 0, flags=0)

Replace occurrences of the pattern in the string with the repl

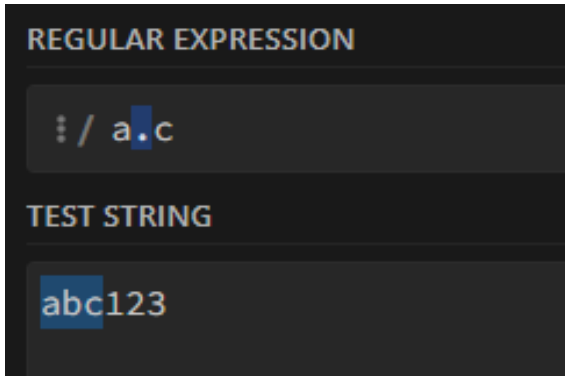
Patterns: “.”

```python
import re

text = "abc123"

pattern_dot = r'a.c'
match_dot = re.search(pattern_dot, text)

if match_dot:
    print(f"Dot Match: {match_dot.group()}")
# Result: Dot Match: abc
```



- ‘.’ Any character except a newline

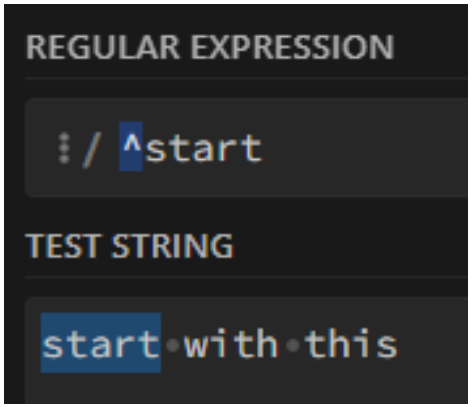
Patterns: “^”

```python
import re

text = "start with this"

pattern_caret = r'^start'
match_caret = re.search(pattern_caret, text)

if match_caret:
    print(f"Caret Match: String starts with '{match_caret.group()}'")
# Result: String starts with 'start'
```



- ‘^’ Matches the start of the string

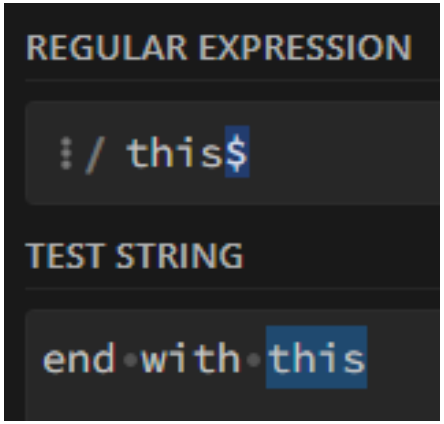
Patterns: “$”

```python
import re

text = "end with this"

pattern_caret = r'this$'
match_caret = re.search(pattern_caret, text)

if match_caret:
    print(f"Caret Match: String ends with '{match_caret.group()}'")
# Result: String ends with 'this'
```



- ‘$’ Matches the end of the string

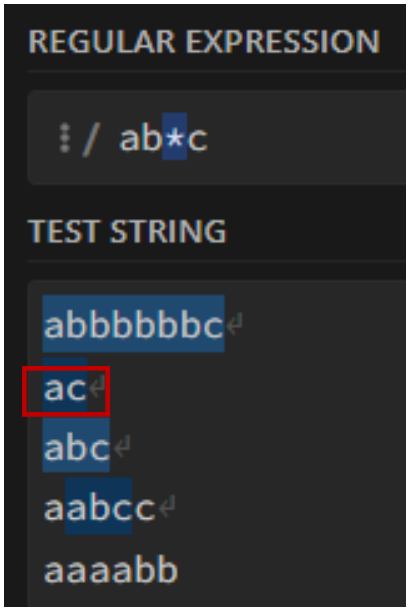
Patterns: “*”

```
import re

text = "abbbbbbc"

pattern_asterisk = r'ab*c'
match_asterisk = re.search(pattern_asterisk, text)

if match_asterisk:
    print(f"Asterisk Match: {match_asterisk.group()}")
# Result: Asterisk Match: abbbbbbc
```



- ‘*’ Matches zero or more occurrences, repetitions

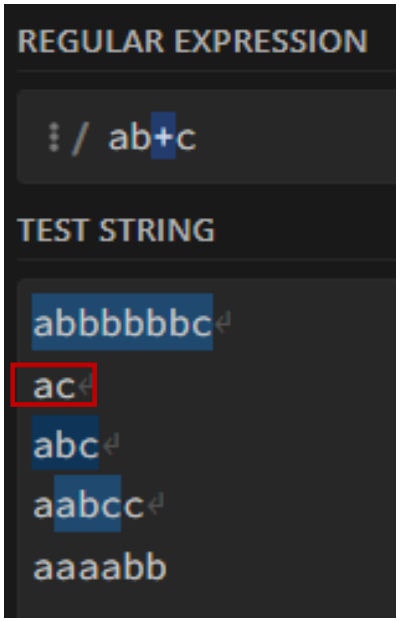
Patterns: “+”

```
import re

text = "abbbbbbc"

pattern_plus = r'ab+c'
match_plus = re.search(pattern_plus, text)

if match_plus:
    print(f"Plus Match: {match_plus.group()}")
# Result: Plus Match: abbbbbbc
```



- ‘+’ Matches one or more occurrences, repetitions

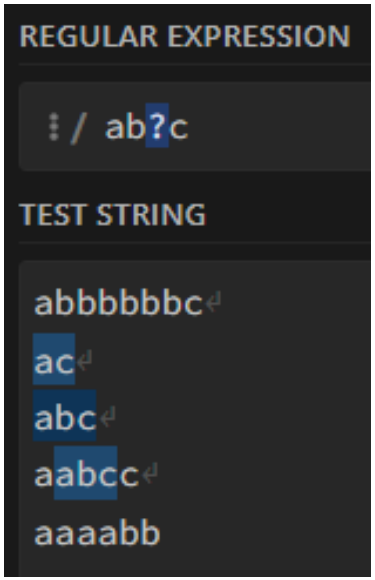
Patterns: “?”

```python
import re

text = "abbbc"

pattern_question = r'ab?c'
match_question = re.search(pattern_question, text)

if match_question:
    print(f"Question Mark Match: {match_question.group()}")
# Result: Nothing gets printed
```



- ‘?’ Matches zero or one occurrence, repetition

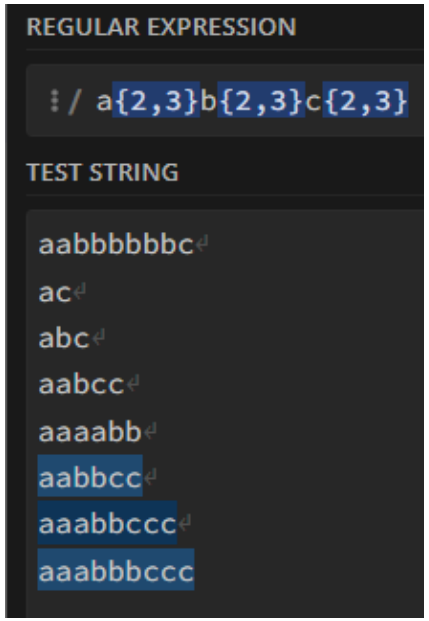
Patterns: “{m}”

```
import re

text = "aaabbbccc"

pattern_curly = r'a{2,3}b{2,3}c{2,3}'
match_curly = re.search(pattern_curly, text)

if match_curly:
    print(f"Curly Braces Match: {match_curly.group()}")
# Result: Curly Braces Match: aaabbbccc
```



- ‘{m}’ Specifies that exactly m copies of the previous RE should be matched

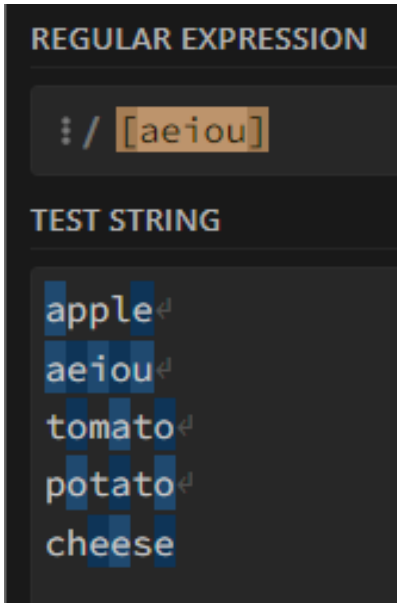
Patterns: “[]”

```python
import re

text = "apple"

pattern_square = r'[aeiou]'
match_square = re.search(pattern_square, text)

if match_square:
    print(f"Square Brackets Match: {match_square.group()}")
# Result: Square Brackets Match: a
```



- ‘[]’ Matches any one of the characters inside the brackets

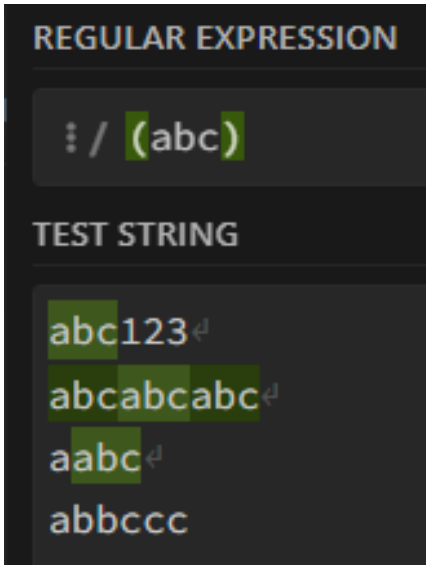
Patterns: “()”

```python
import re

text = "abc123"

pattern_parentheses = r'(abc)'
match_parentheses = re.search(pattern_parentheses, text)

if match_parentheses:
    print(f"Parentheses Match: {match_parentheses.group(1)}")
# Result: Parentheses Match: abc
```



- ‘()’ Matches whatever regular expression is inside the parentheses

Patterns: “|”

```
import re

text = "cat or dog"

pattern_pipe = r'cat|dog'
match_pipe = re.search(pattern_pipe, text)

if match_pipe:
    print(f"Pipe Match: {match_pipe.group()}")
# Result: Pipe Match: cat
```

- ‘|’ Matches either the pattern on its left or right

Examples

```
import re

text = "The cat and the hat"

pattern_search = r'cat'
match_search = re.search(pattern_search, text)

if match_search:
    print(f"Search Result: {match_search.group()}")
# Result: Search Result: cat
```

- Scan through string looking for the first location where the regular expression pattern produces a match
- Return None if no position in the string matches the pattern

Examples

```python
import re

text = "cat and dog"

pattern_match = r'cat'
match_match = re.match(pattern_match, text)

if match_match:
    print(f"Match Result: {match_match.group()}")
else:
    print("No match at the beginning of the string.")
# Result: Match Result: cat
```

```python
import re

text = "dog and cat"

pattern_match = r'cat'
match_match = re.match(pattern_match, text)

if match_match:
    print(f"Match Result: {match_match.group()}")
else:
    print("No match at the beginning of the string.")
# Result: No match at the beginning of the string.
```

- If zero or more characters at the beginning of string match the regular expression pattern
- Return None if the string does not match the pattern

Examples

```python
import re

text = "cat"

pattern_fullmatch = r'cat'
match_fullmatch = re.fullmatch(pattern_fullmatch, text)

if match_fullmatch:
    print(f"Fullmatch Result: {match_fullmatch.group()}")
else:
    print("The entire string does not match the pattern.")
# Result: Fullmatch Result: cat
```

```python
import re

text = "cats"

pattern_fullmatch = r'cat'
match_fullmatch = re.fullmatch(pattern_fullmatch, text)

if match_fullmatch:
    print(f"Fullmatch Result: {match_fullmatch.group()}")
else:
    print("The entire string does not match the pattern.")
# Result: The entire string does not match the pattern.
```

- If the whole string matches the regular expression pattern
- Return None if the string does not match the pattern

Examples

```python
import re

text = "apple banana cherry"

pattern_findall = r'\b\w{5}\b'
# \b matches empty string
# \w matches unicode word characters
matches_findall = re.findall(pattern_findall, text)

print(f"Findall Result: {matches_findall}")
# Result: Findall Result: ['apple']
```

```python
import re

text = "apple banana cherry"

pattern_findall = r'\b\w{6}\b'
# \b matches empty string
# \w matches unicode word characters
matches_findall = re.findall(pattern_findall, text)

print(f"Findall Result: {matches_findall}")
# Result: Findall Result: ['banana', 'cherry']
```

- If the whole string matches the regular expression pattern
- Return None if the string does not match the pattern

Examples

```python
import re

text = "Hello, my name is John. John is 20 years old."

pattern_sub = r'John'
replacement = 'Alice'
modified_text = re.sub(pattern_sub, replacement, text)

print(f"Original Text: {text}")
print(f"Modified Text: {modified_text}")
# Result: Original Text: Hello, my name is John. John is 20 years old.
#         Modified Text: Hello, my name is Alice. Alice is 20 years old.
```

```python
import re

text = "aaa"

pattern_sub = r'aa'
replacement = 'b'
modified_text = re.sub(pattern_sub,
replacement, text)

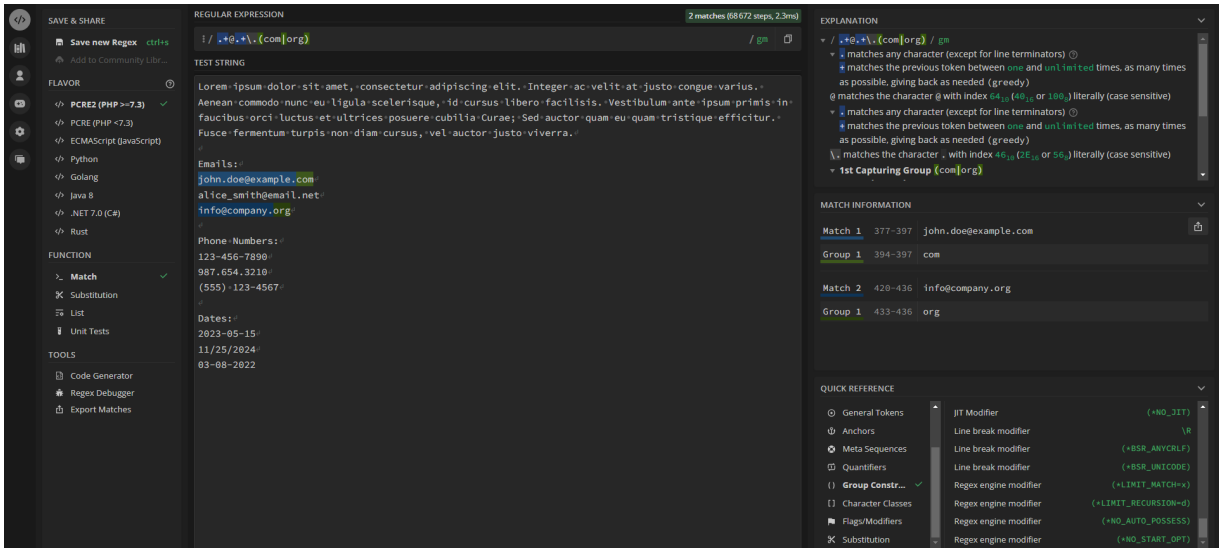
print(f"Original Text: {text}")
print(f"Modified Text: {modified_text}")
# Result: Original Text: aaa
#         Modified Text: ba
```



- Return the string obtained by replacing the leftmost non-overlapping occurrences of pattern in string by the replacement
- If the pattern isn't found, string is returned unchanged

Tips & Tricks

- [Python Docs (Manual)]
- [Regex101]



Exercise

Try the following exercises

04-regex

