Sprachverarbeitung: Übung

SoSe 24

Janis Pagel
Department for Digital Humanities, University of Cologne

29 April 2025

Please submit your solution via Ilias, either as a Jupyter Notebook (`.ipynb`, you can export your Notebook in Jupyter by going to File > Download) or as a Python script (`.py`) if you are not working in Jupyter.

Exercise 1.

Write a single regular expression in Python that matches all the strings in the following lists, and if possible only these strings:

- `["pit", "spot", "spate", "slap two", "respite"]`
- `["rap them", "tapeth", "apth", "wrap/try", "sap tray", "87ap9th", "apothecary"]`
- `["affgfking", "rafgkahe", "bafghk", "baffgkit", "affgfking", "rafgkahe", "bafghk", "baffg kit"]`
- `["www.google.com", "http://www.google.pl/search?q=exercise+programming", "https://www.google.cz/search?q=exercise&var=variable"]`
- `["+424 161 727 363", "+000 000 000 000", "123 321 765", "654 789 123"]`

Try to make your regular expressions as compact as possible.

These exercises are taken from `https://regex.sketchengine.co.uk/`, you can check their website if you want more exercises to practice on.

Solution 1.

```
import re
def test_regex(regex, ex):
    for string in ex:
        if re.search(regex, string):
            print(string)
        else:
            print(f"{string} not matched")

test_regex("^(re)?s?[pl][ioa][tp]e?( two)?$", ["pit", "spot", "spate", "slap two", "
                                               respite"])
test_regex("^[8w]?[7tsr]?ap[ /eo9]?t[hr][a ]?y?(em)?(ecary)?$", ["rap them", "tapeth", "
                                               apth", "wrap/try", "sap tray", "87ap9th", "
                                               apothecary"])
test_regex("^[rb]?aff?g[f ]?h?k[ai]?[thn]?[eg]?$", ["affgfking", "rafgkahe", "bafghk", "
                                               baffgkit", "affgfking", "rafgkahe", "bafghk"
                                               , "baffg kit"])
test_regex("^(https?://)?www.google.(com|pl|cz)(/search\?q=exercise[+&](programming|var=
                                               variable))?",
           ["www.google.com", "http://www.google.pl/search?q=exercise+programming", "
                                               https://www.google.cz/search?q=exercise&var=
                                               variable"])
test_regex("^\+?\d+ \d+ \d+( \d+)?$", ["+424 161 727 363", "+000 000 000 000", "123 321
                                               765", "654 789 123"])
```

Exercise 2.

On `https://lehre.idh.uni-koeln.de/site/assets/files/5615/verwandlung.txt` you find the full text of Franz Kafka's *Die Verwandlung*. Read in the text with Python and write regular expressions that find the following:

- Find all chapter numbers (and only the chapter numbers)
- Find all words followed by either a question mark or an exclamation mark (including the question mark or exclamation mark)
- Find all occurrences of text in-between two commas within a sentence. For example, your regex should find ", wenn er den Kopf ein wenig hob,", but not ", fand er sich in seinem Bett zu einem ungeheueren Ungeziefer verwandelt. Er lag auf seinem panzerartig harten Rücken und sah,"
- Find all occurrences of direct speech
- Find all nominalizations that start with either "Be" or "Er" and end with "ung". Do you also find nouns that are not a nominalization and if yes, which ones? Would it be possible to exclude these "false" hits with regular expressions in a general way?

Solution 2.

```
with open("verwandlung.txt", "r") as fo:
    verwandlung = fo.read()

print(re.findall("I{1,3}\n", verwandlung))
print(re.findall("\w+?[?!]", verwandlung))
print(re.findall(",[\w\s]+?,", verwandlung))
print(re.findall("[›»].+?[‹«]", verwandlung))
print(re.findall("(?:Er|Be)\w+ung", verwandlung))
```