Aufgabenblatt 9

Zur spielerischen Wiederholung der regulären Ausdrücke besuchen Sie
http://regexcrossword.com/challenges/tutorial/puzzles/1 und spielen Sie ein regex-Kreuzworträtsel.

Aufgabe 29: RegEx Klasse

Oft möchte man die Ergebnisse nicht nur ausgeben, sondern die Eingaben, und Ausgaben später wiederverwenden. Die Lösung ist das Erzeugen eines Objektes. Schreiben Sie daher eine Klasse 'RegExMatch'. RegExMatch soll mit folgenden Attributen initialisiert werden können:

- *search* - die Regex
- *sub* - die Substituierung
- *search_string* - was soll durchsucht werden?

In einem weiteren Attribut *result* soll das Ergebnis nach der Substituierung festgehalten werden. Ihr 'RegExMatch' Objekt soll mit *str()* in den Ergebnisstring wandelbar sein. Testen Sie Ihre Klasse, mit den Eingaben aus Ausgabe 2.

```
>>> myregclass = RegExMatch("a","b","hallo")
>>> print(myregclass)
hbllo
```

Aufgabe 30: n-gramme

Schreiben Sie ein Modul `ngrams`, das folgende Funktion enthält:

`makeNGrams(filename, n) -> list`

Diese Funktion soll den Inhalt einer Textdatei in n-gramme zerlegen und als Liste von Listen von Strings zurückgeben.

Beispiel:

```
>>> import ngrams
>>> ngrams.makeNGrams("ngrams.txt", 2)
[['Das', 'ist'], ['ist', 'eine'], ['eine', 'kleine'], ['kleine', 'Datei'],
['Datei', 'zum'], ['zum', 'Testen']]
```

Die Argumente dieser Funktion sind der Dateiname und ein numerischer Wert, der n angibt. Im Beispiel oben war $n = 2$, wir haben also Bigramme erzeugt. Analog für Trigramme:


```
>>> ngrams.makeNGrams("ngrams.txt", 3)
[['Das', 'ist', 'eine'], ['ist', 'eine', 'kleine'], ['eine', 'kleine', 'Datei'],
['kleine', 'Datei', 'zum'], ['Datei', 'zum', 'Testen']]
```

Hinweise:

- Machen Sie sich nochmals mit *Slices* vertraut. Sie sind für diese Aufgabe äußerst hilfreich.
- Um den Dateiinhalt zu tokenisieren, können Sie weiterhin die `split`-Methode in ihrer simpelsten Form verwenden.

Aufgabe 31: n-gramme Klasse

Implementieren Sie ein Modul `ngrams2`, mit einer Klasse NGrams, das bei Instanziierung einen Dateinamen erhält und den Text in einen String einliest:

```
>>> from ngrams2 import NGrams
>>> ng = NGrams("ngrams.txt")
```

Die Klasse muss also eine Variable enthalten, die den Text hält und bei der Instanziierung erzeugt wird.

Weiterhin soll die Klasse eine Funktion getNGrams enthalten, die die Funktionalität von `makeNGrams` aus der vorigen Aufgabe erfüllt. Natürlich muss nun der Dateiname nicht mehr übergeben werden.

```
>>> ng.getNGrams(2)
[['Das', 'ist'], ['ist', 'eine'], ['eine', 'kleine'], ['kleine', 'Datei'],
['Datei', 'zum'], ['zum', 'Testen']]
```

Vergessen Sie nicht, das Modul zu dokumentieren und Doctests zu schreiben.

Aufgabe 32: n-gramme Vergleich

In den vorigen beiden Aufgaben haben Sie das selbe Problem auf zwei verschiedene Arten gelöst. Einmal mit einer einfachen Funktion, einmal mit einer Klasse.

Vergleichen Sie das Ergebnis dieser Aufgaben und erörtern Sie die Vor- und Nachteile der beiden Implementierungen.

Stichworte: Planungsaufwand, Programmieraufwand, Dokumentationsaufwand, Laufzeitvorteil oder -nachteil, Erweiterbarkeit, Wiederverwendbarkeit