Junioraufgabe 1: Reimerei

Team-ID: 00543

Team: pip install knowledge (Imanuel Fehse)

Bearbeiter dieser Aufgabe: Imanuel Fehse

15. November 2022

Inhaltsverzeichnis

Aufgabe

Schreibe ein Programm, das eine Liste von Wörtern einliest und daraus alle passenden Wortpaare berechnet. Ein Wortpaar ist passend, wenn es die folgenden Regeln erfüllt:

1) Die beiden Worte enden gleich: Sie haben dieselbe maßgebliche Vokalgruppe, und nach der maßgeblichen Vokalgruppe enthalten beide Wörter dieselben Buchstaben in derselben Reihenfolge. Dabei ist eine Vokalgruppe eine längstmögliche Folge von unmittelbar aufeinanderfolgenden Vokalen (z.B. hat das Wort Taifun die Vokalgruppen ‚ai' und ‚u'), und die maßgebliche Vokalgruppe eines Wortes ist seine vorletzte Vokalgruppe, wenn das Wort zwei oder mehr Vokalgruppen enthält. Enthält ein Wort nur eine Vokalgruppe, ist seine maßgebliche Vokalgruppe die eine vorhandene Vokalgruppe.
2) In jedem der beiden Wörter enthält die maßgebliche Vokalgruppe und was ihr folgt mindestens die Hälfte der Buchstaben.
3) Keines der beiden Wörter darf mit dem kompletten anderen Wort enden. Passende Wortpaare wären zum Beispiel Baum, Traum und singen, klingen; aber Tanne, Rinne verletzt Regel 1, Informatik, Akrobatik verletzt Regel 2, und kaufen, verkaufen verletzt Regel 3.

Lösungsidee

Zunächst werden die eingelesenen Daten getrennt, sodass jedes Wort einzeln steht. Danach werden vom Programm die Vokalgruppen, maßgeblichen Vokalgruppen und Wortendungen nach der Maßgeblichen Vokalgruppe ermittelt. Mit Hilfe dieser Daten werden anschließend Paare ermittelt, die die Regeln 1, 2 und 3 erfüllen und sich somit laut dem Programm reimen.

Umsetzung

Im ersten Schritt wird mithilfe eines Userinputs die Datei eingelesen. Falls die Datei existiert, fährt das Programm fort, falls nicht, frägt das Programm erneut nach einer Datei.
Ist das einlesen der Datei erfolgreich werden die Worte einzeln in einer Liste gespeichert und in Kleinschreibung formatiert (Bsp. Treppe → treppe), dass falls das Programm ein Wort einließt, dass mit der maßgeblichen Vokalgruppe beginnt und großgeschrieben wird, es trotzdem ohne Fehler verarbeitet werden kann. Nun wird Wort für Wort überprüft, wo sich die Vokalgruppen befinden und anschließend in einer weiteren Liste gespeichert. Aus dieser Liste werden im Anschluss die maßgeblichen Vokalgruppen ermittelt und in einer weiteren Liste gespeichert. Als letzter Schritt vor der Überprüfung auf die Regeln, werden die Wortendungen ermittelt und ebenfalls in einer weiteren Liste gespeichert.
Mit Hilfe dieser Listen kann das Programm nun prüfen, welche Wortpaare der Regel 1 entsprechen. Dazu wird geprüft, welche Worte dieselben maßgeblichen Vokalgruppen und Wortendungen nach den maßgeblichen Vokalgruppen haben. Durch die immer gleiche Indexierung der Listen ist immer klar zuordbar, welche Vokalgruppe oder Wortendung zu welchem Wort gehört. Dadurch können diese einfach auf Paare verglichen werden und bei passenden Paaren als Ergebnis dieser Regel gespeichert werden.
Diese Ergebnisse werden dann auf Regel 2 geprüft. Sind in beiden Wörtern, eines durch Regel 1 gefundenen Paares, die maßgebliche Vokalgruppe + Wortendung nach der maßgeblichen Vokalgruppe länger oder gleich die Hälfte der Länge des Wortes, so erfüllt das Paar auch Regel 2 und wird als Ergebnis von Regel 2 gespeichert. Erfüllt das Paar Regel 2 nicht, wird es nicht zu den Ergebnissen hinzugefügt und ist somit „ausgeschieden“
Dann durchlaufen die Ergebnisse der Regel 2 (welche alle Regel 1 erfüllen) die Regel 3. Hier wird geprüft, ob eines der Wörter mit dem jeweils anderen endet. Falls nicht, erfüllt das Paar auch Regel 3 und wird als finales Paar gespeichert, welches sich laut dem Programm reimt.

Als Letztes werden die finalen Ergebnisse der Regel 3 (welcher Regel 1 und 2 ebenfalls erfüllen, da nur ihre Ergebnisse weiterhin geprüft wurden) in der Konsole ausgegeben, sodass der Nutzer sie sich anschauen und nutzen kann.

Beispiele

Dies sind die Ausgaben, die das Programm in der Konsole macht, nachdem das Programm den Namen und Speicherort der Datei erhalten hat.

reimerei0.txt

Matching words: bemühen glühen
Matching words: biene hygiene
Matching words: biene schiene
Matching words: hygiene schiene
Matching words: knecht recht

reimerei1.txt

Matching words: bildnis wildnis
Matching words: brote note

reimerei2.txt

Matching words: epsilon ypsilon

reimerei3.txt

Matching words: absender kalender
Matching words: bahn zahn
Matching words: bank dank
Matching words: baum raum
Matching words: bein wein
Matching words: bier tier
Matching words: bitte mitte
Matching words: butter mutter
Matching words: dame name
Matching words: drucker zucker
Matching words: durst wurst
Matching words: fest test
Matching words: feuer steuer
Matching words: fisch tisch
Matching words: flasche tasche
Matching words: gas das
Matching words: glück stück
Matching words: gleis kreis
Matching words: gleis preis
Matching words: gleis reis
Matching words: gruppe suppe
Matching words: hand land
Matching words: hose rose

Matching words: hund mund
Matching words: kanne panne
Matching words: kasse klasse
Matching words: kasse tasse
Matching words: keller teller
Matching words: kind rind
Matching words: kind wind
Matching words: klasse tasse
Matching words: kopf topf
Matching words: kreis preis
Matching words: kunde stunde
Matching words: lohn sohn
Matching words: magen wagen
Matching words: platz satz
Matching words: rind wind
Matching words: rock stock
Matching words: sache sprache
Matching words: see tee
Matching words: sekunde stunde

Wichtige Teile des Codes

```
# check for pairs in definite vowel groups
gv1 = definite_vowel_group
gv2 = definite vowel group
results = []
for i in range(len(definite_vowel_group)):
    for j in range(len(definite_vowel_group)):

        # check if vowel pair is correct
        if gv1[i] == gv2[j] and i != j and [j, i] not in results:
            # check if the word endings are the same
            if word_endings[i] == word_endings[j]:
                results.append([i, j])
```

Prüft auf Paare nach Regel 1

```
results_rule2 = []
for item in results_rule1:
    # get length of definite vowel group + wordending
    length = len(definite_vowel_group[item[0]]) + len(word_endings[item[0]])

    # if length is smaller or half of length of word
    if not length <= len(words[item[0]])/2 and not length <= len(words[item[1]])/2:
        results_rule2.append(item)
    else:
        pass
```

Prüft auf Paare nach Regel 2

```
results_rule3 = []
for item in results_rule2:
    # check if word i ends with word j
    if words[item[0]].endswith(words[item[1]].lower()) or
words[item[1]].endswith(words[item[0]].lower()):
        pass
    else:
        results_rule3.append(item)
```

Prüft auf Paare nach Regel 3