Junioraufgabe 1: Reimerei

Team-ID: 00246

Team: http://localhost:80/

Bearbeiter/-innen dieser Aufgabe:
Alessio Caputo

20. November 2022

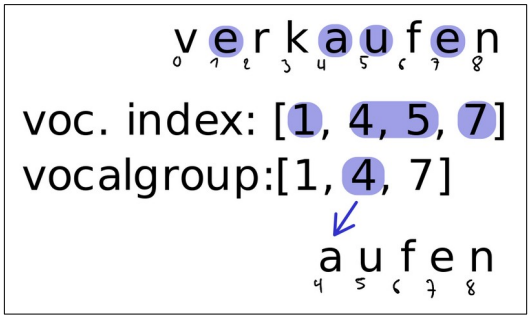
Inhaltsverzeichnis

Lösungsidee

Jedes Wort wird einmal mit jedem anderen Wort verglichen und dabei die drei Bedingungen aus der Aufgabenstellung abgearbeitet. Sollten alle Bedingungen erfüllt sein, so wird das Wortpaar ausgegeben. Um jedes Wort genau ein mal mit jedem anderen zu vergleichen wird das Prinzip einer Schachtabelle verwendet.

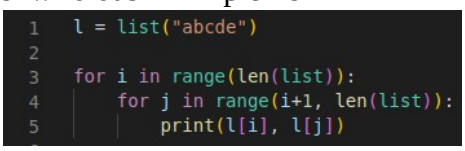
Umsetzung

Die Lösungsidee wird in Python implementiert. Es wird vorallem Python verwendet, da diese Sprache eine Unterstützüng für Umlaute in Strings bietet.Zunächst wird die Datei eingelesen und die Zeilen dieser in eine Liste mit dem Namen „words“ gespeichert. Für die Aufgabenstellung spielt die Ermittlung der maßgeblichen Vokalgruppe eine entscheidende Rolle, da diese in der ersten und zweiten Bedingung zu überprüfen ist. Um eine einfache ermittlung dieser zu ermöglichen wird eine globale Funktion definiert, die diese ermittelt. Um die maßgebliche Vokalgruppe und was ihr folgt bestimmen zu können müssen zuerst die Vokalgruppen ermittelt werden. Dazu müssen wirderrum zuerst die einzelnen Vokale ermittelt werden. Der Datentyp String hat die eigenschaft, dass er Iterabel ist, also können die einzelnen Buchstaben einfach mit var[position:int] ausgelsen werden. Mit einer for-schleife werden alle Vokale ermittelt und in eine Liste geschrieben. Diese Liste wird wiederrum rückwärts durch eine zweite for-schleife überarbeitet. Für jedes Element wird überprüft, ob die Differenz zu dem in der Liste vorher



auftretenden Wert 1 beträgt. Sollte dies der Fall sein, so wird das Element gelöscht um am ende eine Liste mit allen anfang-index-Werten der Vokalgruppen zu haben. Je nach dem, wie viele Vokalgruppen ein Wort hat, werden unterschiedliche Rückgabewerte definiert. Für den Fall, dass es zwei oder mehr Vokalgruppen gibt wird der abschnitt des Ausgangswortes von der vorlestzten Vokalgruppe bis zu dem Wortende zurückgegeben. Sollte nur eine Vokalgruppe vorliegen wird der Abschnitt von dieser bis zum Wortende zurückgegeben. Für den Fall, dass keine Vokalgruppen vorliegen wird -1 als Fehlercode zurückgegeben.

Da zentrale Teile der Berechnungen sich nicht doppeln sollten wird zuerst für jedes Wort die maßgebliche Vokalgruppe und was ihr folgt errechnet. Anschließend wird überprüft, ob diese die Hälfte des Wortes einnimmt. Sollte dies der Fall sein, so wird sie in eine Liste geschrieben. In dieser Liste wird nun jedes Objekt mit jedem anderen Objekt verglichen. Dabei wird das Prinzip einer Schachtabelle verwendet, um dopplungen zu vermeiden. (Siehe Bild) So kann jedes Objekt genau einmal mit jedem anderen Objekt verglichen werden. Innerhalb der zwei For-schleifen, wie sie im Bild abgebildet sind, werden nun die weiteren Bedingungen überprüft. Zunächst wird überprüft, ob eines der beiden Wörter mit dem gesammten anderen Wort endet. Danach wird überprüft, ob die beiden maßgeblichen Vokalgruppen identisch sind. Sollte dies der Fall sein, so wird das Wortpaar ausgegeben.



```
1  l = list("abcde")
2
3  for i in range(len(list)):
4      for j in range(i+1, len(list)):
5          print(l[i], l[j])
```

Beispiele

reimerei0.txt	reimerei1.txt	reimerei2.txt	reimerei3.txt
bemühen - glühen biene - hygiene biene - schiene hygiene - schiene knecht – recht	bildnis - wildnis brote - note	epsilon - ypsilon	absender - kalender ansage - frage ansage - garage bahn - zahn bank - dank baum - raum bein - wein bier - tier bild - schild bitte - mitte butter - großmutter butter - mutter dame - name dezember - november dezember - september drucker - zucker durst - wurst ermäßigung - kündigung ermäßigung - reinigung fest - test feuer - steuer fisch - tisch flasche - tasche frage - garage fuß - gruß gas - glas glück - stück gleis - kreis gleis - preis gleis - reis gruppe - suppe hand - land hand - strand hose - rose hund - mund kündigung - reinigung kanne - panne kasse - klasse kasse - tasse kassette - kette kassette - toilette keller - teller

			kette - toilette kind - rind kind - wind klasse - tasse kopf - topf kreis - preis kunde - stunde land - strand lohn - sohn magen - wagen nachmittag - vormittag november - september platz - satz rind - wind rock - stock s-bahn - zahn sache - sprache see - tee sekunde - stunde

Alle Programmausgaben ergeben sinn und sind fehlerfrei. Ein Aspekt, der hier nicht betrachtet werden kann ist die Geschwindigkeit, mit der die Wortpaare ausgegeben werden. Da alle Wörter mit allen anderen verglichen werden braucht das Programm vorallem bei reimerei3.txt eine etwas höhere Zeit, als notwendig. Zeit und Dateneingabe haben hier eine art Quadratischen zusammenhang. Jedoch müssen nicht alle Wörter mit jedem anderen Wort verglichen werden. An dem Punkt, an dem die Wörter nach der ermittlung der Maßgeblichen Vokalgruppe und was ihr folgt in die liste „words“ geschrieben werden könnten sie auch mit einem key-objekt nach ihren maßgeblichen Vokalgruppen geordnet werden. So müsste man lediglich die Beziehungen in diesen key-listen überprüfen und nicht ein wort mit jedem anderen. In dieser Key-überprüfung muss auch nicht zusätzlich überprüft werden, ob die maßgeblich Vokalgruppe identisch ist, da das Wort nur mit identischen verglichen wird. Es bleibt lediglich die Frage, ob diese Methode in Python performanter ist. Damit verbunden ist die Frage, die key-objekte in Python selbst implementiert sind. Sollten sie sehr langsam sein, so ist diese herangehensweise zwar interessant, jedoch nicht notwendig.

Quellcode

```
from sys import argv
vocals = list("aeiouäöü")
def getRhymepart(word:str):
	vocallist = []
	for i in range(len(word)):
		if word[i] in vocals:
			vocallist.append(i)
	for i in range(len(vocallist)-1, -1, -1):
		if vocallist[i]-1 in vocallist:
			vocallist.pop(i)
	if len(vocallist) == 1:
		return word[vocallist[0]:]
	elif len(vocallist) >= 2:
		return word[vocallist[len(vocallist)-2]:]
	else:
		return -1
def match(wordlist:list) -> None:
	count = 0
	for i in range(len(wordlist)):
		for j in range(i+1, len(wordlist)):
			wordA = wordlist[i].copy()
			wordB = wordlist[j].copy()
			if not ((wordA["word"].endswith(wordB["word"]))
			or (wordB["word"].endswith(wordA["word"]))):
				if (str(wordA["rhymepart"]) == str(wordB["rhymepart"])):
					print(wordA["word"] + " - "+wordB["word"])
					count+=1
	print("\nfound "+str(count)+" matches in "+ argv[1])
words = []
with open(argv[1], "r") as file:
	file = file.read().splitlines()
	for word in file:
		w = word.lower()
		rhymepart = getRhymepart(w)
		if rhymepart == -1: continue
		if len(rhymepart)<len(w)/2:continue
		words.append({
			"word": w,
			"rhymepart": rhymepart
		})
match(words)
```