Inhaltsverzeichnis

Lösungsidee....2
Umsetzung....3
Beispiel....8
Quellcode....8

Junioraufgabe1 :Reimerei

Von Richard Guo, 2022

Lösungsidee

Die Idee ist mit String Bearbeitungen(Python) zu checken ob sich Wörter reimen.
Dabei werden Teile von den jeweiligen Wörter genommen und mit andere Wörter verglichen.
Folgende Sachen müssen dabei beachtet werden:

1)Was ist ein Reim?Das heißt das Programm muss wissen was überhaupt ein Reim ist.

2)Jedes Wort mit jedes Wort zu vergleichen.Hier gibt es unterschiedliche Möglichkeiten.
Doch nach meiner Meinung nach ist strukturiert immer am besten.(Es ist nur meine Meinung).

Außerdem besteht die Regel 1 eigentlich aus mehreren Regeln.
Nämlich:
Sie haben dieselbe maßgebliche Vokalgruppe,nach der
maßgeblichen Vokalgruppe enthalten beide Wörter
dieselben Buchstaben in derselben Reihenfolge.

Das heißt der wichtigster Teil hier von einen Wort in Regel 1
ist nur von der maßgebliche Vokalgruppe(maßgebliche Vokalgruppe beihaltet)
bis zur Ende des Wortes wichtig(letzter Buchstabe auch beinhaltet).
Dann kontrolliert das Programm ob alle Buchstaben gleich sind.
Doch wegen eines Bugs die ich nicht finden konnte habe ich nochmal
hinzugefügt,dass die maßgebliche Vokalgruppe nochmal kontrolliert wird.

Umsetzung

Ich habe meinen Programm in der Programmiersprache Python Version 3.9 geschrieben.
Das Programm muss nicht in vorherige Versionen funktionieren.
Die Wörter die sich dann reimen wird dann ausgegeben.

Als erstes wurde die Beispieldatei/Beispieleingaben heruntergeladen und der Encoder
ist UTF-8.
Dieser Datei wurde dann mit eine for-schleife durchgegangen und zu eine Wörterliste hinzugefügt.
Jedes mal beim Hinzufügen musste dann noch mal die Leerzeichen vor und hinter den
Wort entfernt werden mit .strip().

```
with open("woerterListe1.txt",encoding="utf8") as file:
    for line in file:
        woerterListe.append(line.strip())
print(woerterListe)
for oneword in woerterListe:
    for anotherWord in woerterListe:
        if oneword == anotherWord:
            continue
        ifReim = checkifReim(oneword,anotherWord)
        if ifReim == True:
            print(oneword + " und " + anotherWord + " reimen sich.")
```

Danach wird eine Funktion geschrieben(checkifReim) und dieser hat zwei Eingaben.Wort1 und Wort2:

```
def checkifReim(wort1,wort2):
```

Danach werden ein paar Variable in der Funktion definiert.
IfReim dient dazu am Ende auszugegeben ob wort1 und wort2 sich reimen:

```
ifReim = True
```

Danach werden noch 2 Variablen definiert.Dieser dienen dazu um die Vokalen aus den jeweiligen Wörter zu speichern:

```
vokaleausWort1 = vokaelauseinenWort(wort1)
vokaleausWort2 = vokaelauseinenWort(wort2)
```

Wie man sieht wird hier schon wieder eine Funktion benutzt.
Diese Funktion sucht sich Vokalengruppen aus einen Wort heraus und dessen Position im Wort und tut das beide in einen String.Dieser kommt dann in eine Liste:vokaleListeFuerWort rein.
Das Programm schaut sich das Wort immer von hinten an und wenn ein Doppelvokal kommt dann tauscht die Buchstaben(damit der Doppelvokal wieder in der richtigen Reinfolge wieder ist).

Das war jetzt sehr viel Erlärung in der nächsten Folie wird diese Funktion etwas genauer erklärt.
Die Funktion hat eine Eingabe:wort

```
def vokaelauseinenWort(wort):
```

Zuerst muss man alle einzelne Vokalen in eine Liste reintun.
(Nachher wird erklärt warum es unnötig ist die Doppelvokalen in die Liste reinzutun),dabei darf man nicht Großbuchstaben vergessen.

```
vokale = ["a","e","i","o","u","A","E","I","O","U","ä","ü","ö","Ä","Ü","Ö"]
```

Danach gibt es eine for-range schleife,die ein Wort von hinten durchgeht:

```
for i in range(len(wort) - 1,-1,-1):
```

Dann wird mit einer for-schleife die Vokale Liste durchgegangen:

```
for oneVokal in vokale:
```

Darauf wird kontrolliert ob Wort an Position i ein Vokal hat:

```
if wort[i] == oneVokal:
```

Dann wird kontrolliert ob es Doppelvokale im Wort gibt.
Denn in diesen Programm gilt: Doppelvokale vor einzelne Vokalen

Der Code der Doppelvokale überprüft ist:

```
if wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] == "u":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))

elif wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] == "i":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))

elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] == "i":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))


elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] == "u":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))


elif wort[i - 1] == "ä" or wort[i - 1] == "Ä" and wort[i] == "u":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))

elif wort[i - 1] == "i" or wort[i - 1] == "I" and wort[i] == "e":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))

elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] == "e":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))
elif wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] == "a":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))
elif wort[i - 1] == "o" or wort[i - 1] == "O" and wort[i] == "o":
    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " + str(i))
```

Wie vorher gesagt wurde gilt Doppelvokale vor einzelne Vokal,aber dann gab es ein Problem.Der Programm musste wissen,ob hinter den einzelnen Vokal doch noch ein andere Vokal ist.
z.B arche-no-ah bei diesen Wort sind oa zusammen aber dennoch sind diese einzelne Vokalen keine Doppelvokalen, weil die nicht in eine Silbe sind.
Deswegen überprüft der Programm immer ob hinter den Vokal i,u oder e ist.
Falls ja dann überspringt das Programm den Vokal.

```
    else:

        if i + 1 < len(wort):
            if wort[i + 1] != "u" and wort[i + 1] != "i" and wort[i + 1] != "e":
                vokaleListeFuerWort.append(wort[i] + " " + str(i))
        if wort.endswith(oneVokal) and wort[-2] != oneVokal:
            vokaleListeFuerWort.append(wort[i] + " " + str(i))
```

Man sieht immer dass der Vokal hinzugefügt wird,aber es wird auch die Position des Vokals hinzugefügt.Dies ist wichtig,weil man wissen muss ab welchen Teil des Wortes(also eingabe in checkifReim) checken muss,ob es mit den jeweiligen anderen Wort gleich sind.
Kurz zusammen gefasst heißt es.Das Programm weiß jede Position von Vokalen im einen Wort.Ein Vokal mit seinen Position gespeichert in einer Variable sieht dann z.B. in der Liste so aus:e 5
Das Leerzeichen braucht man um danach mit der .split() Funktion die zwei Informationen auseinander zunehmen und dann in eine Liste zu tun.
Diese Liste kommt wieder in eine Liste was es dann zu einer Liste von Liste ergibt.

Wie die Liste von Liste erstellt wird:

```
splitetdVokaleAusWort1 = []
for oneitem in vokaleausWort1:
    splitetdVokaleAusWort1.append(oneitem.split(" "))
splitetdVokaleAusWort2 = []
for oneitem in vokaleausWort2:
    splitetdVokaleAusWort2.append(oneitem.split(" "))
```

----------Regel1----------

Was in Regel 1 wichtig ist,ist die maßgebliche Vokalgruppe zu finden.
Die Vokalen mit der Positionen hat das Programm schon.
Aber für Regel 1 muss man verschiede Szenarien betrachten:

1. Fall:Allerbeide Wörter die mehr als einen Vokal haben.
2. Fall:Beide Wörter haben nur einen Vokal.
3. Fall 3:Wort1 hat einen Vokal und Wort2 hat mehr als einen Vokal.
4. Fall 4:Wort1 hat mehr als einen Vokal und Wort2 hat einen Vokal.

Die 4 Fälle hab ähnlichen Aufbau.
Warum es so geschrieben wurde war wenn das Programm versucht hat den 2.Fall versucht auszuführen,war der Indexzugriff out of range.
Heißt man konnte nicht darauf zugreifen da es nur einen Vokal gab,aber das Programm war für

mehrere Vokale programmiert.
Deswegen wird hier nur der Code von Fall 1 und Fall 2 gezeigt,weil der Aufbau her eigentlich fast alles gleich ist.
1.Fall:
Zuerst überprüft das Programm ob die maßgeblichen Vokalgruppen jeweils gleich sind:

```
if splitetdVokaleAusWort1[1][0] != splitetdVokaleAusWort2[1][0]:
    ifReim = False
    return ifReim
```

Mit der .find() Funktion kann man in Python den Index von ein Zeichen in einen String finden.
Falls Python nichts findet gibt die .find() Funktion -1 zurück.
Mit dieser Funktion wird in wort 1 wort 2 gesucht ab der maßgebliche Vokalgruppe bis zur letzten Buchstabe und auch anderes herum.

```
if splitetdVokaleAusWort1[0][0] == splitetdVokaleAusWort2[0][0]:
indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[1][1]):len(wort2)])
indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[1][1]):len(wort1)])
```

Oben steht,dass .find() -1 zurück gibt falls Python nichts findet.Falls dies der Fall ist reimen sich die beiden Wörter nicht.
Außerdem muss man nach den Fall bedenken,falls die Wörter exakt gleich sind aber zum Beispiel bei den anderen Wort hinten noch -st drangehängt wird.
Der Code für diese Überprüfungen:

```
if indexfind1 == -1 or indexfind2 == -1 and len(wort2) - indexfind2 !=
len(wort1) - indexfind1:
    ifReim = False
    return ifReim
```

Der Code für Fall 2:

```
if len(vokaleausWort1) == 1 and len(vokaleausWort2) == 1:
    if splitetdVokaleAusWort1[0][0] != splitetdVokaleAusWort2[0][0]:
        ifReim = False
        return ifReim
    if splitetdVokaleAusWort1[0][0] == splitetdVokaleAusWort2[0][0]:
        indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[0]
[1]):len(wort2)])
        indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[0]
[1]):len(wort1)])
        if indexfind1 == -1 or indexfind2 == -1 and len(wort2) - indexfind2 !=
len(wort1) - indexfind1:
            ifReim = False
            return ifReim
```

----------Regel2----------
Nach meinen Verständnis habe ich verstanden,dass jedes Wort zu 50% gleiche Buchstaben haben muss wie andere,weil es steht zwar „In jedem der beiden Wörter enthält die maßgebliche Vokalgruppe und was ihr folgt mindestens die Hälfte der Buchstaben „ aber das würde bedeuten,dass Regel 1 keinen Sinn machen würde,weil es steht in Regel 1 „nach der maßgeblichen Vokalgruppe enthalten beide Wörter dieselben Buchstaben in derselben Reihenfolge.“ .
Das heißt nach meinen Verständnis,dass sich Regel 1 und Regel 2 auf die Buchstaben hinter die maßgebliche Vokalgruppe bezieht.
Deswegen nehme ich an,dass es eigentlich heißen soll, dass der kürzere Wort von den beiden Wörtern
mindestens 50 % oder weniger Buchstaben beinhaltet von den längeren Wort.
Ebenfalls muss das längere Wort mindestens 50 % oder mehr gleiche Buchstaben haben wie das

kürzere Wort.

Jetzt zu der wirklichen Umsetzung.
Zuerst muss man das längere Wort von den beiden Wörtern herausgefunden werden:

```
if len(wort1) > len(wort2):
    shortestWord = wort2
    longestWord = wort1
else:
    shortestWord = wort1
    longestWord = wort2
```

Dann gibt es einen Zähler,der zählt wie viele Buchstaben von den kürzeren Wort und den längeren Wort gleich ist.
Aber aus irgendeinen Grund(den ich nicht mehr weiß) habe ich statt von 0 bis ende des kürzesten Wort von hinten des kürzesten Wortes angefangen:

```
for i in range(-1,-1 * len(shortestWord),-1):
    if shortestWord[i] == longestWord[i]:
        counter += 1
```

Dann teilt das Programm die Länge des kürzesten Wortes durch zwei und kontrolliert,ob der Wert von den Zähler kleiner ist als die halbe Länge des kürzesten Wortes.

```
if counter < len(shortestWord) / 2:
    ifReim = False
    return ifReim
```

----------Regel3----------
In Python kann man mit .find() eine Zahl,String etc. in einen String finden.
Allerdings gibt Python nicht das Wort zurück sondern den Index von den ersten Zeichen,dass Python findet.
Falls .find() nichts findet gibt es den Wert -1 zurück.

Dann wird das kürzere Wort im längeren Wort gesucht.
Der Index wird dann mit einer Variable gespeichert.

```
ifwortinwort = longestWord.find(shortestWord)
```

In Python gibt auch die len() Funktion,die gibt die Länge eines Datenstrukturs oder Strings oder Intergers aus.

Wenn man dann die Länge des längsten Wortes minus die Länge des kürzesten Wortes und der Index
von .find() ist,dann heißt es,dass das längere Wort mit den kürzeren Wort endet.
Beispiel:
verkaufen-len(9)
kaufen – len(6)

Dann benutzt man die .find() Funktion und dann kommt 3 raus.
Wenn man dann die Länge von kaufen von verkaufen subtrahiert ergibt sich der Index von .find().
Der Code für diesen Schritt:

```
if len(longestWord) - len(shortestWord) == ifwortinwort:
    ifReim = False
    return ifReim
```

Falls dann alle diese Regeln erfüllt sind,sind diese beiden Wörtern ein Reim:

```
if ifReim == True:
    return ifReim
```

Beispiel

Die Wörterliste:

```
bemühen

Biene
breitschlagen
glühen
hersagen
Hygiene
Knecht
Recht
Schiene
schlank
Schwank
```

Notiz:Ich bin jetzt nicht sonderlich gut in Reimen also kann es sein,dass ich was falsch habe.

```
bemühen und glühen reimen sich.
Biene und Hygiene reimen sich.
Biene und Schiene reimen sich.
breitschlagen und hersagen reimen sich.
glühen und bemühen reimen sich.
hersagen und breitschlagen reimen sich.
Hygiene und Biene reimen sich.
Hygiene und Schiene reimen sich.
Knecht und Recht reimen sich.
Recht und Knecht reimen sich.
Schiene und Biene reimen sich.
Schiene und Hygiene reimen sich.
schlank und Schwank reimen sich.
Schwank und schlank reimen sich.
```

Quellcode

```
def checkifReim(wort1,wort2):
    ifReim = True
    vokaleausWort1 = vokaelauseinenWort(wort1)
    vokaleausWort2 = vokaelauseinenWort(wort2)
    #bei doppellauten ist die Zahl dahinter um 1 erhöht also auf die zweite
Buchstabe bezogen.
    splitetdVokaleAusWort1 = []
    for oneitem in vokaleausWort1:
        splitetdVokaleAusWort1.append(oneitem.split(" "))
    splitetdVokaleAusWort2 = []
    for oneitem in vokaleausWort2:
        splitetdVokaleAusWort2.append(oneitem.split(" "))
    if len(vokaleausWort1) == 1 and len(vokaleausWort2) == 1:
        if splitetdVokaleAusWort1[0][0] != splitetdVokaleAusWort2[0][0]:
            ifReim = False
            return ifReim
        if splitetdVokaleAusWort1[0][0] == splitetdVokaleAusWort2[0][0]:
            indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[0]
[1]):len(wort2)])
            indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[0]
```

```
[1]):len(wort1)])
            if indexfind1 == -1 or indexfind2 == -1 and len(wort2) -
indexfind2 != len(wort1) - indexfind1:
                ifReim = False
                return ifReim
    elif len(vokaleausWort1) > 1 and len(vokaleausWort2) > 1:
        if splitetdVokaleAusWort1[1][0] != splitetdVokaleAusWort2[1][0]:
            ifReim = False
            return ifReim
        if splitetdVokaleAusWort1[1][0] == splitetdVokaleAusWort2[1][0]:
            indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[1]
[1]):len(wort2)])
            indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[1]
[1]):len(wort1)])
            if indexfind1 == -1 or indexfind2 == -1 and len(wort2) -
indexfind2 != len(wort1) - indexfind1:
                ifReim = False
                return ifReim
    elif len(vokaleausWort1) > 1 and len(vokaleausWort2) == 1:
        if splitetdVokaleAusWort1[1][0] != splitetdVokaleAusWort2[0][1]:
            ifReim = False
            return ifReim
        if splitetdVokaleAusWort1[1][0] == splitetdVokaleAusWort2[0][1]:
            indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[0]
[1]):len(wort2)])
            indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[1]
[1]):len(wort1)])
            if indexfind1 == -1 or indexfind2 == -1 and len(wort2) -
indexfind2 != len(wort1) - indexfind1:
                ifReim = False
                return ifReim
    elif len(vokaleausWort1) == 1 and len(vokaleausWort2) > 1:
        if splitetdVokaleAusWort1[0][1] != splitetdVokaleAusWort2[1][0]:
            ifReim = False
            return ifReim
        if splitetdVokaleAusWort1[0][1] == splitetdVokaleAusWort2[1][0]:
            indexfind1 = wort1.find(wort2[int(splitetdVokaleAusWort2[1]
[1]):len(wort2)])
            indexfind2 = wort2.find(wort1[int(splitetdVokaleAusWort1[0]
[1]):len(wort1)])
            if indexfind1 == -1 or indexfind2 == -1 and len(wort2) -
indexfind2 != len(wort1) - indexfind1:
                ifReim = False
                return ifReim
#------------Regel 2---------------------
    if len(wort1) > len(wort2):
        shortestWord = wort2
        longestWord = wort1
    else:
        shortestWord = wort1
        longestWord = wort2
    counter = 0
    for i in range(-1,-1 * len(shortestWord),-1):
        if shortestWord[i] == longestWord[i]:
            counter += 1
    if counter < len(shortestWord) / 2:
        ifReim = False
        return ifReim
    #------------Regel3------------------------
    ifwortinwort = longestWord.find(shortestWord[0].lower() + shortestWord[1:])
    if len(longestWord) - len(shortestWord) == ifwortinwort:
        ifReim = False
        return ifReim
    if ifReim == True:
```

```
        return ifReim
#vokale aus einen Wort holen
def vokaelauseinenWort(wort):
    vokale = ["a","e","i","o","u","A","E","I","O","U","ä","ü","ö","Ä","Ü","Ö"]
    vokaleListeFuerWort = []
    for i in range(len(wort) - 1,-1,-1):
        for oneVokal in vokale:
            if wort[i] == oneVokal:
                if wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] == "u":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] ==
"i":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] ==
"i":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] ==
"u":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "ä" or wort[i - 1] == "Ä" and wort[i] ==
"u":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "i" or wort[i - 1] == "I" and wort[i] ==
"e":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "e" or wort[i - 1] == "E" and wort[i] ==
"e":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "a" or wort[i - 1] == "A" and wort[i] ==
"a":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                elif wort[i - 1] == "o" or wort[i - 1] == "O" and wort[i] ==
"o":
                    vokaleListeFuerWort.append(wort[i - 1] + wort[i] + " " +
str(i))
                else:
                    if i + 1 < len(wort):
                        if wort[i + 1] != "u" and wort[i + 1] != "i" and wort[i
+ 1] != "e":
                            vokaleListeFuerWort.append(wort[i] + " " + str(i))
                    if wort.endswith(oneVokal) and wort[-2] != oneVokal:
                        vokaleListeFuerWort.append(wort[i] + " " + str(i))
    return vokaleListeFuerWort
woerterListe = []
with open("woerterListe1.txt",encoding="utf8") as file:
    for line in file:
        woerterListe.append(line.strip())
for oneword in woerterListe:
    for anotherWord in woerterListe:
        if oneword == anotherWord:
            continue
        ifReim = checkifReim(oneword,anotherWord)
        if ifReim == True:
            print(oneword + " und " + anotherWord + " reimen sich.")
```