Beispiellösung: Songwriter

Hinweis:
Der Aufgabentext wird hier nur der Vollständigkeit halber abgedruckt. Die Dokumentation zu einer Aufgabenbearbeitung muss und soll den Aufgabentext nicht enthalten.

Das Duo „Fake that" ist der aktuelle Stern am Pop-Himmel. Sie bringen einen Hit nach dem anderen heraus, aber nach einiger Zeit dämmert es selbst ihren größten Fans, dass die Texte ihrer Silben-Songs (so nennt das Duo seinen Stil) sich alle sehr ähneln. Das letzte Album hieß wohl nicht umsonst „It's Always The Same".

Nellie, ein mittlerweile etwas gelangweilter Fan der beiden, hat festgestellt, wie die Texte der Silben-Songs funktionieren:

> Eine Silbe wird aus einem Konsonanten und einem Vokal gebildet. Beispiele: 'do', 'nu', 'la'.
> Eine Zeile besteht aus einer ungeraden Anzahl von Wiederholungen einer Grundsilbe, wobei derjenigen in der Mitte ein 'p di' angehängt wird. Beispiele: 'sup di', 'da dap di da', 'ne ne nep di ne ne'.
> Eine Strophe besteht aus mindestens zwei Zeilen. Die Menge der Konsonanten und Vokale, aus denen die Grundsilben einer Strophe gebildet werden, ist immer sehr klein, z. B. {s, u, a}. Die Anzahl der Silbenwiederholungen ist für alle Zeilen einer Strophe gleich (Nellie spricht deshalb von der Silbenzahl einer Strophe). Am Ende einer Strophe steht gelegentlich ein markiger Call wie 'yeah!', 'yo man', 'fake that!' oder ähnlich pseudo-cooles Zeugs.
> Ein Song besteht aus mindestens zwei Strophen. Die Zeilenzahlen der Strophen eines Songs folgen einem Muster, z. B. „immer 3 Zeilen", „abwechselnd 4 und 6 Zeilen" usw. Auch die Silbenzahlen der Strophen folgen ähnlichen Mustern.

Nellie zweifelt nun an der Kreativität von „Fake that". Solche Songtexte kann man sich bestimmt vom Computer schreiben lassen!

Junioraufgabe

Schreibe einen „Songwriter", also ein Programm, das einen nach Nellies Regeln aufgebauten Text eines Silben-Songs erzeugen kann. Stelle dabei so weit wie möglich sicher, dass die von deinem Programm erzeugten Texte sich voneinander unterscheiden. Gib in der Dokumentation mindestens drei unterschiedliche Songtexte an, die dein Programm erzeugt hat.

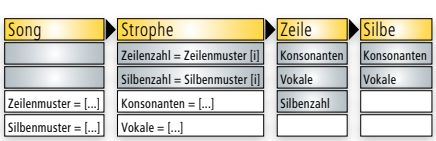
Lösungsidee

Die Lösung soll so funktionieren wie die Einheiten eines Silben-Songs beschrieben sind. Folgende, aufeinander aufbauende Einheiten gibt es: Silbe, Zeile, Strophe und Song. Für jede Song-Einheit soll ein eigener Programmteil entwickelt werden, der diese nach Vorschrift erzeugt. Dabei bauen auch die Programmteile aufeinander auf: 'Song' verwendet 'Strophe', 'Strophe' verwendet 'Zeile' und 'Zeile' verwendet 'Silbe'.

Die Programmteile sind nicht unabhängig voneinander:

> Die Anzahl der Zeilen einer Strophe und auch die Anzahl der Silben in den Zeilen der Strophe sollen je einem Muster folgen. Wenn ein Muster eine Folge von Zahlen ist (eine Zahl pro Strophe), kann 'Song' jeder 'Strophe' die passende Zahl übergeben.
> Für eine Strophe müssen die Mengen von Konsonanten und Vokalen bestimmt werden, aus denen die Silben der Zeilen bestehen müssen. 'Strophe' muss die beiden Buchstabenmengen an 'Zeile' übergeben (und 'Zeile' wiederum an 'Silbe').

Das folgende Bild stellt die Zusammenarbeit der verschiedenen Programmteile dar.



Für eine Strophe muss zusätzlich noch festgelegt werden, ob sie einen „Call" hat. Wenn ja, wird einer ausgewählt und als zusätzliche Zeile hinten an die Strophe angefügt.

Schließlich muss noch beachtet werden, dass „die von deinem Programm erzeugten Texte sich voneinander unterscheiden". Dazu kann bei der Bestimmung der Muster, der Konsonanten und Vokale einer Strophe, der Auswahl des Calls, der Auswahl von Konsonant und Vokal für eine Silbe usw. der Zufall eingesetzt werden.

Umsetzung

Die Lösungsidee wird in ein Programm in der Sprache Python umgesetzt. Für jede Song-Einheit wird eine Funktion geschrieben, jeweils mit den nötigen Parametern:

Song:	def *song*()
Strophe:	def *strophe*(anzahl_zeilen, anzahl_silben)
Zeile:	def *zeile*(anzahl_silben, konsonanten, vokale)
Silbe:	def *silbe*(konsonanten, vokale)

Außerdem wird eine Funktion benötigt, die die Muster (also die Zahlenfolgen) erzeugt. Die Funktion *muster*(laenge, grundmenge) liefert eine Folge von Elementen einer Grundmenge (z.B. der möglichen Zeilenanzahlen), wobei die Folge eine bestimmte Länge haben muss (z.B. die Anzahl der Strophen eines Songs). Die Folge ist zufällig von einer dieser drei Arten: (1) zwei sich abwechselnde Elemente, (2) alle Elemente gleich und (3) die Elemente der Grundmenge (nötigenfalls wiederholt, wenn die Grundmenge kürzer ist als die gewünschte Folgenlänge).

Die Zufallseffekte werden durch verschiedene Funktionen des Python-Moduls random erzielt:

random.choice(folge): wählt zufällig ein Element der Folge.
random.sample(folge, anzahl): wählt zufällig eine Anzahl von Elementen der Folge.
random.randint(int1, int2): wählt zufällig eine Integer-Zahl aus der Menge {int1, …, int2}.

Der Song wird in der Funktion strophe zeilenweise ausgegeben.

Beispiele

Durch einen Aufruf der Funktion *song* wird in der Python-Shell ein Songtext ausgegeben. Aus Platzgründen werden nur zwei Beispiele gezeigt.

```
>>> song()
da da dap di da da
bi bi bip di bi bi

bu bu bup di bu bu
do do dop di do do
BwInf rocks!
```

Dieser Song besteht aus zwei Strophen. Das Strophenmuster ist „immer zwei Zeilen", das Silbenmuster ist „immer fünf Silben". In Strophe 1 kommen die Konsonanten d und b sowie die Vokale a und i vor, in Strophe 2 b und d bzw. u und o. Strophe 2 hat einen Call.

```
>>> song()
li lip di li
li lip di li

le le lep di le le
sa sa sap di sa sa
se se sep di se se
sa sa sap di sa sa

gi gi gi gip di gi gi gi
bo bo bo bop di bo bo bo
yeah!
```

Dieser Song hat das Zeilenmuster [2,4,2] (zwei Zahlen abwechselnd) und das Silbenmuster [3,5,7] (die möglichen Silbenzahlen nacheinander). Nur die letzte Strophe hat einen Call.

Quelltext

```
import random

vokale = ['a','e','i','o','u']
# nur geeignete Konsonanten - Geschmackssache
konsonanten = ['b','d','g','l','n','s']
# mögliche Calls - auch 'leere' Calls
calls = ["yeah!", "fake that!", "BwInf rocks!", "", "", ""]
# mögliche Silbenanzahlen in den Zeilen - müssen ungerade sein
silbenzahlen = [3,5,7]
zeilenzahlen = [2,3,4,5] # mögliche Zeilenanzahlen in den Strophen
strophenzahlen = [2,3,4,5] # mögliche Strophenzahlen für einen Song

def silbe(konsonanten, vokale):
    """Bildet eine Silbe aus einem der Konsonanten und einem der Vokale."""
    return random.choice(konsonanten) + random.choice(vokale)

def zeile(konsonanten, vokale, anzahl_silben):
    """Bildet eine Zeile mit der gegebenen Anzahl von Silben."""
    # Die Silbe wird für die ganze Zeile einmal bestimmt.
    zeilensilbe = silbe(konsonanten, vokale)
    halbezeile = (anzahl_silben // 2) * (zeilensilbe + ' ')
    return halbezeile + zeilensilbe + 'p di ' + halbezeile

def call():
    """Sucht aus der Menge der Calls einen (eventuell leeren) aus."""
    auswahl = random.randint(0, len(calls)-1)
    return calls[auswahl]

def strophe(anzahl_zeilen, anzahl_silben):
    """Bildet eine Strophe mit den gegebenen Anzahlen für Zeilen und Silben."""
    anzahl_konsonanten = random.randint(2,3) # zwei oder drei Konsonanten
    auswahl_konsonanten = random.sample(konsonanten, anzahl_konsonanten)
    anzahl_vokale = random.randint(2,3) # zwei oder drei Vokale
    auswahl_vokale = random.sample(vokale, anzahl_vokale)
    for i in range(anzahl_zeilen):
        print(zeile(anzahl_silben, auswahl_konsonanten, auswahl_vokale))
    strophen_call = call()
    # ein nicht-leerer Call wird an die Strophe angehängt
    if strophen_call != "":
        print(strophen_call)
    print("") # eine leere Zeile am Ende der Strophe

def muster(laenge, grundfolge):
    """Waehlt eine Folge (der gegebenen Laenge) von Elementen der gegebenen Grundfolge auf drei verschiedene Arten aus."""
    folge = []
    auswahl = random.randint(1,3)
    if auswahl == 1:  # Art 1: abwechselnd zwei verschiedene Elemente
        elementpaar = random.sample(grundfolge, 2) # zwei Elemente auswählen
        for i in range(laenge):
            folge.append(elementpaar[i%2]) # i%1 ergibt abwechselnd 0 und 1
    elif auswahl == 2: # Art 2: immer das gleiche Element
        element = random.choice(grundfolge) # Element auswählen
        for i in range(laenge):
            folge.append(element)
    else: # auswahl == 3; Art 3: Grundmenge der Reihe nach
        for i in range(laenge):
            folge.append(grundfolge[i%len(grundfolge)])
    return folge

def song():
    """Bildet einen vollständigen Song."""
    strophenzahl = random.choice(strophenzahlen)
    zeilenzahlmuster = muster(strophenzahl, zeilenzahlen)
    silbenzahlmuster = muster(strophenzahl, silbenzahlen)
    for i in range(strophenzahl):
        strophe(zeilenzahlmuster[i], silbenzahlmuster[i])
```