Twist

Aufgabe 2

Teil 1 (Twisten)

Theorie

Um Elvis helfen zu können einen Text zu twisten, muss zunächst der Text aus einer Datei in das Programm / Skript eingelesen werden. Dies kann über die Standardeingabe (stdin) geschehen und ermöglicht es den Text direkt über eine Konsole einzugeben oder den Inhalt einer Datei mit einer Pipe in das Programm / Skript umzuleiten. Da zum Twisten eines Wortes nur dieses Wort und keine anderen benötigt werden, bietet sich diese Methode sehr an. Wörter mit einer Länge kleiner als drei können nicht getwistet werden und werden direkt wieder so wie sie eingelesen wurden ausgegeben. Damit ausschließlich Wörter mit alphabetischen Buchstaben getwistet werden, müssen Satz- und Sonderzeichen und Zahlen gefiltert und unverändert ausgegeben werden. Sollte eine von nicht-alphabetischen Zeichen abgegrenzte Zeichenfolge länger als drei Zeichen ist, wird diese Zeichenfolge getwistet. Hierfür wird das erste Zeichen des Worts (Zeichenkette) in einer Variablen (String) gespeichert, wobei die restlichen Zeichen in einer anderen Variablen (String) gespeichert werden. Zeichen an zufällig ermittelten Positionen der zweiten Variable werden aus dieser gelöscht und zur anderen Variablen hinzugefügt. Dies wird solange durchgeführt bis die Länge des zweiten Strings nur noch eins beträgt. Dabei wird beachtet, dass nie das letzte Zeichen der zweiten Variable in die erste verschoben wird und dieses als letztes in der zweiten Variable bleibt. Letztendlich wird das letzte Zeichen, der zweiten Variable zur ersten Variable hinzugefügt und das Wort sollte nach den gegebenen Vorgaben getwistet sein.

Implementierung

Das Programm bzw. Skript wurde in `Python3` implementiert und auf unter Ubuntu auf der Konsole ausgeführt. Zum Einlesen des Texts wird zunächst von `sys` `stdin` importiert und durch alle Zeilen in `stdin` und alle Zeichen (`ch`) in jeder Zeile iteriert. Wenn das Zeichen dem Alphabet angehört (Zeile 6), wird es einem Buffer (`buff`) hinzugefügt. Sobald ein Zeichen eingelesen wird, das nicht dem Alphabet angehört, wird ein die Länge des Buffers geprüft. Bei Längen größer als drei, wird das Wort aus dem Buffer mit der Funktion `twist(arg)` getwistet und anschließend ausgegeben (Zeile 10). Sollte das Wort allerdings kürzer als drei Zeichen lang sein, wird es komplett ausgegeben. Unabhängig von der Länge des Buffers wird dieser geleert und das nicht-alphabetische Zeichen ausgegeben.

```
from sys import stdin

def start():
    buff = ""
    for line in stdin:
        for ch in line:
            if ch.isalpha():
                buff += ch
            else:
                if len(buff) > 3:
                    print(twist(buff), end="")
                else:
                    print(buff, end="")
                buff = ""
                print(ch, end="")
```

Die hauptsächliche Aufgabe des Skripts bewältigt allerdings die Funktion namens `twist(arg)`. Sie liest nimmt einen String als Parameter ein und speichert den Inhalt dieses Parameters ab dem zweiten Zeichen in der Variable `word` ab. In der Variable `result` wird der erste Buchstabe und letztendlich das getwistete Wort gespeichert. Solange die Länge von `word` größer als `1`, ist wird `result` ein Zeichen einer zufälligen Position (außer der letzten) von `word` angehängt. Nachdem die Länge von `word` gleich `1` geworden ist, wird das letzte Zeichen zu `result` angehängt. Es kann zufällig dazu kommen, dass das Ergebnis (`result`) dem ursprünglichen Wort (`arg`) gleicht.

```
from random import randint

def twist(arg):
      result, word = cut_char_from_string(arg, 0)
      while len(word) > 1:
      ch, word = cut_char_from_string(word, randint(0, len(word)-2))
      result += ch
      result += word
    return result
```

Die oben verwendete Funktion namens `cut_char_from_string(str, idx)` gibt das Zeichen eines Strings vom angegebenen Index (`idx`) zurück und den restlichen String ohne dieses Zeichen.

Teil 2 (Detwisten)

Theorie

Die Vorgehensweise des Detwisters ist zunächst sehr ähnlich zum Twister. Es wird der Text über `stdin` eingelesen und ausschließlich Wörter mit einer Länge, die größer als drei Zeichen ist, detwistet. Um ein mögliches detwistetes Wort für ein getwistetes Wort zu finden, wird eine Wörterliste benötigt. Die verwendete Wörterliste ist in dem offiziellen Git-Repository vom BwInf zu finden. Ein mögliches detwistetes Wort wird aus der Wörterlisteversucht zu finden, indem der Anfangsbuchstabe, die Länge, der letzte Buschstabe und eine Check-Summe jedes Wortes der Wörterliste mit den des getwisteten Wortes verglichen werden. Sollten diese Dinge gleich sein wird das Wort von der Wörterliste zu einer weiteren Liste hinzugefügt werden. Die Check-Summe wird aus der Summe aller ASCII-Werte aller Buchstaben der Wörter als Kleinbuchstaben gebildet. Sollte die Länge dieser neuen Liste größer als eins sein, werden, durch einen genaueren Vergleich der Worte, *fehlerhafte* Worte ausgeschieden. Dieser genauere Vergleich, stellt fest, ob alle Buchstaben des getwisteten Wortes in dem Wort von der Wörterliste in derselben Anzahl vorkommen.

Implementierung

Dieser Teil wurde ebenso in Python implementiert und über die Konsole unter Ubuntu ausgeführt. Der Quellcode der Funktion, die nach möglichen detwisteten Wörtern sucht sieht wie folgt aus.

def detwist(arg):

```
	subset = []
	with open(dict_name, 'r', encoding='utf-8') as d:
		# iterate through d
		for word in d:
			# if:
			# * the first letters of the word of d and arg are equal
			# * and their lengths are equal
			# ...
			if word[0].lower() == arg[0].lower() and
len(word.strip()) == len(arg):
				# ... check if their last letters are equal ...
				if word[len(word.strip())-1] == arg[len(arg)-1]:
					# ... calc their checksums ...
					checksum_word = 0
					for ch in word.strip().lower():
						checksum_word += ord(ch)
					checksum_arg = 0
					for ch in arg.strip().lower():
						checksum_arg += ord(ch)
					# ... and check if the checksums are equal
					if checksum_word == checksum_arg:
						# if the checksums are equal append it
to the subset
						subset.append(word)
	# if you are left with only one word, just return it
	if len(subset) == 1:
		return arg[0]+subset[0].strip()[1:]
	# if you are left with more than one word
	# make sure that they have the same letters and the same amount of
them
	elif len(subset) > 1:
		subsubset =[]
		for word in subset:
			tmp=True
			for ch in word.strip().lower():
				if not arg.strip().lower().count(ch) ==
word.strip().lower().count(ch):
					tmp = False
			# if 'word' has the same letters and amount of them as
'arg'
			# append it to 'subsubset'
			if tmp:
				subsubset.append(arg[0]+word.strip()[1:])
		# finally return subsubset
		return subsubset
	else:
		# if you ended up with no result, just return the twisted word
		return arg.strip()
```

Beispiele

Es sind die Ergebnisse von den Beispielsdateien `twist1.txt` bis `twist4.txt` in diesem Dokument angeführt, während, aufgrund von Platzgründen, die Ergebnisse von `twist5.txt` in separaten Dateien unter `./twisted/twist5.txt` und `./detwisted/twist5.txt` zu finden sind.

Teil 1

```
~/BWINF_37/Twist# cat samples/twist1.txt | python3 twister.py
Der Tiwst
(Ecnlgsih twsit = Drnehug, Vheunredrg)
war ein Modneatz im 4/4-Tkat,
der in den ferühn 1960er Jrehan pulpoär
wurde und zu
Rcok'n'Rlol, Ryhhtm and Buels oder selezilper
Tiswt-Misuk geatnzt wird.

~/BWINF_37/Twist# cat samples/twist2.txt | python3 tHat der atle
Hesxnetmeeir scih dcoh enmail weegebbegn! Und nun seloln sinee Gteeisr auch
ncah mienem Wlieln lbeen. Senie Wort und Wrkee mrket ich und den Burcah,
und mit Geäskissettre tu ich Wuednr acuh.

~/BWINF_37/Twist# cat samples/twist3.txt | python3 twister.py
Ein Rratsanuet, wecelhs a la catre aeeritbt, beetit sien Aonebgt ohne enie
vhreor fgetsegetle Meeihnlgerfonüe an. Daucrdh heabn die Gstäe zawr mher
Sralepuim bei der Wahl iherr Seipsen, für das Rtsaeanurt ettsenehn jodech
zäzisehtulcr Aunfawd, da wneegir Pssechiaegrlhnunit vrehdanon ist.

~/BWINF_37/Twist# cat samples/twist4.txt | python3 twister.py
Aguutsa Ada Boyrn King, Cuntsoes of Lclvaoee, war enie btscirhie Algeide
und Memiiaahekttrn, die als die erste Pmmiigroererarn üpahuebrt glit.
Beteris 100 Jrahe vor dem Aemfukmon der etsren Phrepaaciemrsrogmrn enrasn
sie eine Rceehn-Meicnahk, der eginie Koeptnze meenrdor Prcahoiemmsregrrapn
voahrewngm.
```

Teil 2

```
~/BWINF_37/Twist# cat twisted/twist1.txt | python3 detwister.py
woerterliste.txt
Der Twsit
(Englisch twist = Drehung, Verdrehung)
war ein Mntoaedz im 4/4-Takt,
der in den ['frühen', 'führen'] 1960er Jahren populär
wurde und zu
Rock'n'Rlol, Rhythm and Blues oder spezieller
Twist-Musik getanzt wird.

~/BWINF_37/Twist# cat twisted/twist2.txt | python3 detwister.py
woerterliste.txt
Hat der alte Hexenmeister sich doch einmal wgegebeebn! Und nun sollen seine
Geister auch nach meinem Willen leben. Seine Wort und Werke merkt ich und
den Brauch, und mit Geistesstärke tu ich Wunder auch.
```

```
~/BWINF_37/Twist# cat twisted/twist3.txt | python3 detwister.py
woerterliste.txt
Ein Restaurant, welches a la ctare ['arbeitet', 'abrietet'], bietet sein
Angebot ohne eine vorher festgelegte Mlehgefeonnrüie an. Dadurch haben die
Gäste zwar mehr Spielraum bei der Wahl ihrer Speisen, für das Restaurant
entstehen jedoch zusätzlicher Aufwand, da weniger Pensionsberechtigt
vorhanden ist.

~/BWINF_37/Twist# cat twisted/twist4.txt | python3 detwister.py
woerterliste.txt
Agsuuta Ada Boyrn Knig, Cuetnoss of Laloevce, war eine britische Adelige
und Mathematikerin, die als die erste Programmiererin überhaupt gilt.
['Bereits', 'Bieters', 'Breites'] 100 Jahre vor dem Aufkommen der ersten
Programmiersprachen ersann sie eine Rechen-Mechanik, der einige Konzepte
moderner Programmiersprachen vorwegnahm.
```

Manche Wörter werden nicht in der Wörterliste gefunden. Um dies zu vermeiden, könnten während dem Twisten Wörter, die nicht in der Wörterliste zu finden sind, zu dieser hinzugefügt werden.