Aufgabe 1: Hopsitexte

Team-ID: 00578

Team-Name: Frederik Hamann

Bearbeiter/-innen dieser Aufgabe:
Frederik Hamann

3. November 2024

Inhaltsverzeichnis

1 Lösungsidee

Die Idee war Zara beim Verfassen zu unterstützen, indem ich ein Programm entwickle, welches als Texteditor dient und in Echtzeit den Abstand zwischen den Endpositionen anzeigt. Dies ist ausreichend hilfreich, da Hopsitexte zwar durchaus in sich aufeinander aufbauen, allerdings reicht auch eine kleine Veränderung aus, um die Endpostionen drastisch zu verändern.
Dadurch ist eine Planung des Hopsitextes bereits beim schreiben nicht zwingend notwendig. Daher könnte Zara, sofern sie mit dem berechneten Abstand nicht zufrieden ist, beispielsweise Wörter wie „diese" zu „jene" oder „gut" zu „toll". Hierbei müsste bei den genannten Beispielen auch die Bedeutung nicht gravierend geändert werden.

2 Umsetzung

Es wird mithilfe der Python Bibliothek „ttkbootstrap" ein GUI erstellt, welche ein Textfeld und ein und eine Radialanzeige enthält. Das Textfeld dient der Eingabe des Hopsitextes, während die Radialanzeige zur Darstellung des Abstandes zwischen den Endpostionen genutzt wird. Es hat einen Anzeigebereich von 0 bis 29. Der Anzeigebereich wurde auf 29 berechnet, da der höchste Sprungwert bei 30 liegt und der Abstand somit nur bei max. 29 liegen kann, da der 2. Hopser einen Buchstaben nach dem 1. Texthopser startet und somit in einem Fall, wo auf ein „ß" 29 mal „ä" folgen würde, der Abstand 29 betrüge.

3 Beispiele

Genügend Beispiele einbinden! Die Beispiele von der BwInf-Webseite sollten hier diskutiert werden, aber auch eigene Beispiele sind sehr gut besonders wenn sie Spezialfälle abdecken. Aber bitte nicht 30 Seiten Programmausgabe hier einfügen!

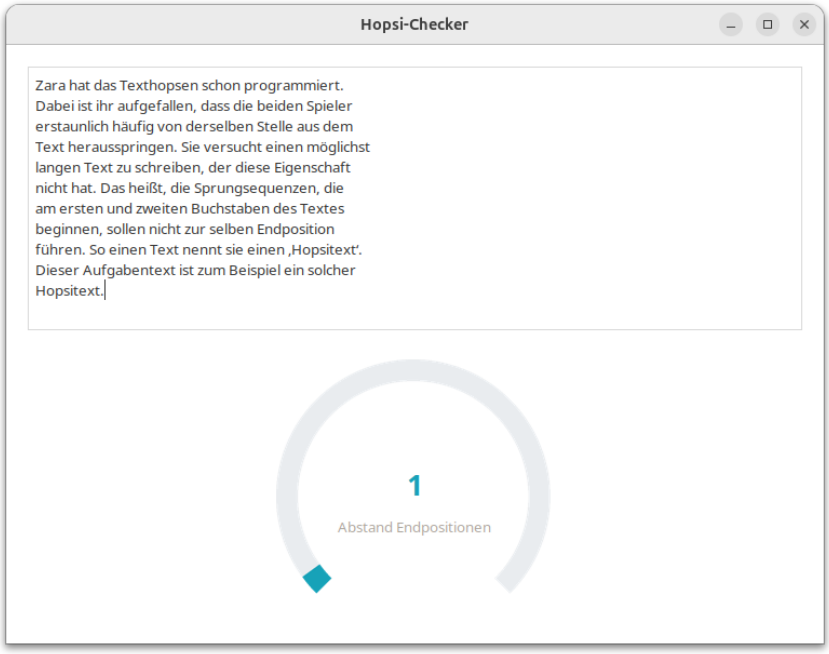

Abbildung 1:

4 Quellcode

```
from PIL import Image        # Image.CUBIC is deprecated (replaced by Image.BICUBIC)
Image.CUBIC = Image.BICUBIC # https://stackoverflow.com/a/76717474

import ttkbootstrap as ttk
from ttkbootstrap.constants import *
from ttkbootstrap.scrolled import ScrolledText
import threading
import time
import re


re_input = "" #erstelle Variable re_input
input = ""  #erstelle Variable input (notwendig da globale Variable)
abstand_endpositionen = 0 #erstelle Variable abstand_endpositionen


def sprungweite(buchstabe): # Nutze einen Index um die Sprungweite einen Buchstabens ueber die
                                                    Position im Index +1 bestimmen zu koennen
    alphabet = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p"
                                                    , "q", "r", "s", "t", "u", "v", "w", "x", "
                                                    y", "z", "ä", "ö", "ü", "ß"]
    return alphabet.index(buchstabe) + 1




def GUI():
    global input
    global re_input

    app = ttk.Window(title="Hopsi-Checker", themename="united")
    st = ScrolledText(app, padding=20, height=10, autohide=True) # erstelle Textfeld
    st.pack(fill=BOTH, expand=YES)
    meter = ttk.Meter( # erstelle Radialanzeige
        metersize=260,
        padding=5,
```

```
        amountused=25,
        amounttotal=29,
        meterthickness=20,
        metertype="semi",
        subtext="Abstand Endpositionen",
        interactive=False,
        bootstyle="info",
        )
    meter.pack()

    while True:
        input = st.get("1.0",END) # get the text from the text field
        re_input = re.sub('[^A-Za-zäöüÄÖÜß]', '', input) # remove all non-letter characters
        meter.configure(amountused = abstand_endpositionen) # Nutze Wert aus der Variable von
                                                            abstand_endpositionen
        if abstand_endpositionen <= 5:
            meter.configure(bootstyle="danger")
        if abstand_endpositionen > 15:
            meter.configure(bootstyle="success")
        else:
            meter.configure(bootstyle="info")
        app.update() # update the GUI



def check_hopsi(Startposition):
    global abstand_endpositionen
    not_finished = True #setze Variiable not_finished auf True
    Stelle = Startposition

    while not_finished == True:
        lt_re_input = list(re_input.lower()) # Wandelt ipnut in Liste um und wandelt alle
                                                            Buchstaben in Kleinbuchstaben um
        if len(lt_re_input) <= 1:
            abstand_endpositionen = 0
            time.sleep(0.5)
        else:
            if sprungweite(lt_re_input[Stelle]) + Stelle < len(lt_re_input):
                Stelle = Stelle + sprungweite(lt_re_input[Stelle])
            else:
                not_finished = False
    return Stelle


def berechne_differenz(Wert1, Wert2): #Funktion zur Berechnung der Differenz von zwei
                                                        positiven Werten
    if Wert1 > Wert2:
        Wert_diff = Wert1 - Wert2
    else:
        Wert_diff = Wert2 - Wert1
    return Wert_diff


def check_all():
    time.sleep(0.5)
    global abstand_endpositionen
    while True:
        time.sleep(0.1)
        check_hopsi(0)
        check_hopsi(1)
        abstand_endpositionen = berechne_differenz(check_hopsi(0), check_hopsi(1))



t1_GUI = threading.Thread(target=GUI)
t2_check_hopsi = threading.Thread(target=check_all)
t1_GUI.start()
t2_check_hopsi.start()
```