

Aufgabe 1: Hopsitexte

Team-ID: 00578

Team-Name: Frederik Hamann

Bearbeiter/-innen dieser Aufgabe:
Frederik Hamann

17. 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 Endpositionen 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 „tkbootstrap“ ein GUI erstellt, welches ein Textfeld und eine Radialanzeige enthält. Das Textfeld dient der Eingabe des Hopsitextes, während die Radialanzeige zur Darstellung des Abstands zwischen den Endpositionen 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. Texthopper startet und somit in einem Fall, wo auf ein „ß“ 29 mal „ä“ folgen würde, der Abstand 29 betrüge.

```
function BERECHNE_ABSTAND_ENDPOSITIONEN(Wert1, Wert2)  
  if Wert1 > Wert2 then  
    Wert_diff ← Wert1 − Wert2  
  else  
    Wert_diff ← Wert2 − Wert1  
  end if  
  return Wert_diff  
end function  
function CHECK_HOPSI(Startposition)  
  if input ≤ 1 then  
    abstand_endpositionen ← 0  
  else  
    if sprungweite ≤ 2 then  
      i ← i + k  
    end if  
    return Stelle
```

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!

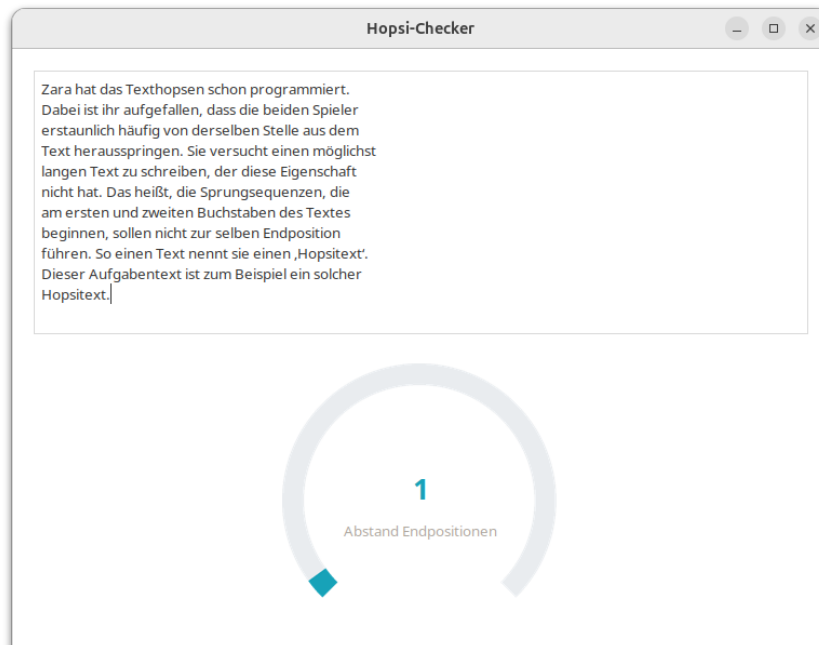


Abbildung 1:

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 können
    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) # hole den Text aus dem Textfeld und schreibe ihn in die
                                variable input
    re_input = re.sub('[^A-Za-zäöüÄÖÜß]', '', input) # Entfernt alle nicht-Buchstaben
    meter.configure(amountused = abstand_endpositionen) # Nutze Wert aus der Variable von
                                                        abstand_endpositionen
    if abstand_endpositionen <= 5: # setze Farbe der Radialanzeige auf Rot wenn
                                    abstand_endpositionen <= 5
        meter.configure(bootstyle="danger")
    elif abstand_endpositionen >= 15: # setze Farbe der Radialanzeige auf Grün wenn
                                    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 Variiabile 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()
```