Ingenieurinformatik I

Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

Aufgabensteller: **Küpper, Krug, Hinz und KollegInnen**

Bearbeitungszeit: **60 Minuten**

Hilfsmittel: **Taschenrechner nicht zugelassen,**
PC/Notebook/Tablet/Handy nicht zugelassen,
sonstige eigene Hilfsmittel sind erlaubt,
Bearbeitung mit Bleistift ist erlaubt.

****** Viel Erfolg! ******

Aufgabe 1: (ca. 22 Punkte)

Ein Lithium-Ionen-Akku wird entladen. Die Akku-Spannung wird in regelmäßigen Zeitabständen gemessen und in die Textdatei „messung.txt“ geschrieben. In jeder Zeile der Textdatei steht zuerst der Zeitpunkt (in Sekunden) und danach die gemessene Akku-Spannung (in Volt).

Hier sehen Sie einen Ausschnitt aus der Textdatei. In der ersten Zeile erkennt man, dass zum Zeitpunkt t = 0,0 s eine Anfangsspannung von 4,19 Volt gemessen wurde:

```
0.0 4.19
250.0 4.095
500.0 4.015
750.0 3.97
1000.0 3.88
1250.0 3.835
1500.0 3.79

2750.0 3.601
3000.0 3.505
3250.0 3.305
3500.0 3.159
```

```
# Beispielskript: Zeitpunkte und Akku-Spannungen einlesen
with open("messung.txt", "r") as file:

    # Der Reihe nach alle Zeilen durchlaufen
    for line in file:

        values = line.split()
        t = float(values[0])
        u = float(values[1])

        # In t steht der Zeitpunkt, in u steht die
        # Akku-Spannung aus der aktuellen Zeile
        print(f"{t} {u}")
```

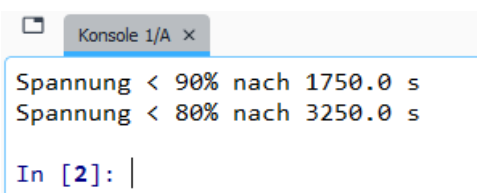
Hinweise:

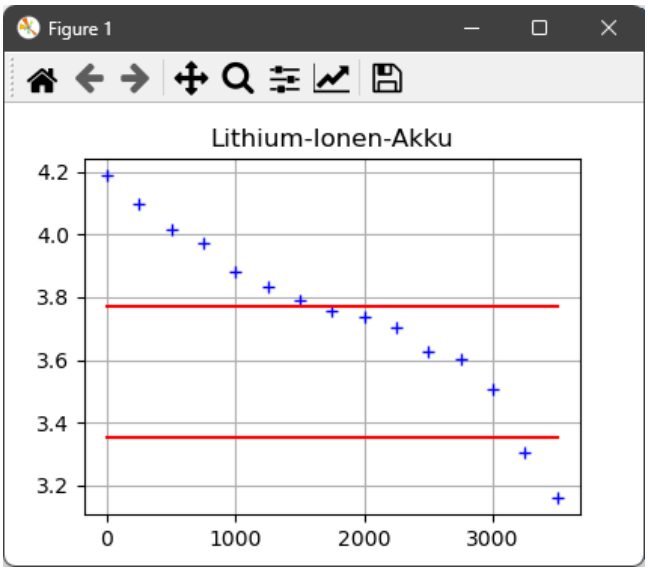
- Das abgebildete Beispielskript liest alle Zeitpunkte und Akku-Spannungen aus der Textdatei und gibt sie auf dem Bildschirm aus. Sie dürfen die Anweisungen aus dem Beispielskript in Ihrer Lösung verwenden.
- Die abgebildeten Werte sind nur Beispiele. Bei anderen Akkus ergeben sich andere Werte. Auch die Anfangsspannung (in der ersten Zeile) kann einen anderen Wert als 4,19 Volt haben.

Schreiben Sie ein Python-Skript mit den folgenden Funktionalitäten. Die Ausgabe Ihres Skripts soll so aussehen, wie es in den Bildschirmfotos gezeigt ist.

1. Alle Zeitpunkte und Akku-Spannungen werden aus der Textdatei eingelesen und in grafischer Form auf dem Bildschirm dargestellt (also nicht in Textform!):
   - Die einzelnen Messpunkte werden durch blaue Pluszeichen + angezeigt.
   - Zusätzlich werden zwei horizontale rote Linien gezeichnet (beide Linien im t-Intervall von 0 bis 3500 s): Die obere Linie bei 90% , die untere Linie bei 80% der Anfangsspannung.
   - Die Grafik soll die gezeigten Hilfslinien (Koordinatengitter) und den Titel „Lithium-Ionen-Akku“ enthalten.

2. Zusätzlich zur Grafik-Darstellung sollen die Zeitpunkte der folgenden Messungen ermittelt und mit einer Nachkommastelle in Textform ausgegeben werden:
   - Der Zeitpunkt, wann die Akku-Spannung zum ersten Mal unter 90% der Anfangsspannung sinkt.
   - Der Zeitpunkt, wann die Akku-Spannung zum ersten Mal unter 80% der Anfangsspannung sinkt.





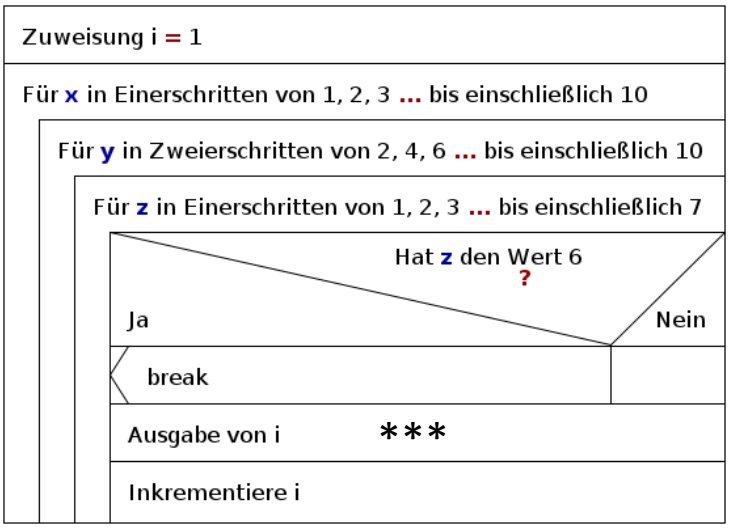
Aufgabe 2: (ca. 25 Punkte)

2.1. Gegen Ende des Struktogramms wird der Wert von i ausgegeben (Strukturblock mit *** markiert).

Welchen Wert hat die Variable i, wenn dieser Ausgabebefehl zum letzten Mal ausgeführt wird?

Antwort:

i =



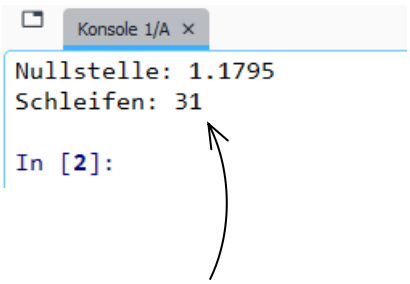
2.2. Schreiben Sie ein Python-Skript, welches dem abgebildeten Struktogramm genau entspricht. Achten Sie insbesondere auf die korrekte Einrückung der for-Schleifen und der Verzweigung.

2.3. Die Funktion bisektion() ermittelt die Nullstelle von $f(x) = x^3 + 2x - 4$ mit dem Bisektionsverfahren (Intervall-Halbierungsverfahren). Es werden zwei Rückgabewerte zurückgegeben: (1) die ermittelte Nullstelle und (2) die Anzahl der Schleifendurchläufe, die das Verfahren benötigt hat.

Ergänzen Sie das Python-Skript wie folgt: Rufen Sie die Funktion bisektion() auf und geben Sie die Nullstelle (mit vier Nachkommastellen) sowie die Anzahl der Schleifendurchläufe aus.

```
def bisektion():
    n = 0; x1 = 1; x2 = 2
    f1 = x1 ** 3 + 2 * x1 - 4
    while True:
        n += 1
        xn = (x1 + x2) / 2
        fn = xn ** 3 + 2 * xn - 4
        if f1 * fn > 0:
            x1 = xn
            f1 = fn
        else:
            x2 = xn
        if abs(fn) < 1e-10:
            break
    return xn, n

# AUFGABE: bisektion() aufrufen; Nullstelle und Schleifendurchläufe ausgeben
```



So soll die Ausgabe Ihres Skripts aussehen.

2.4. Zeichnen Sie ein Struktogramm, welches den genauen Ablauf der Funktion bisektion() wiedergibt. Alle im Quelltext vorhandenen Kontrollstrukturen sollen erkennbar sein.

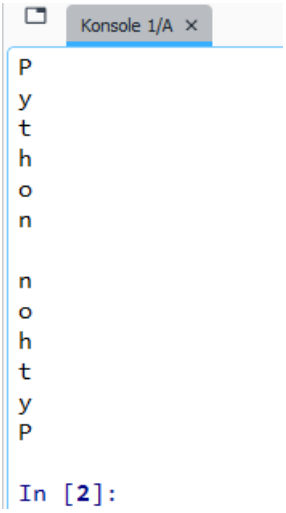
Aufgabe 3: (ca. 8 Punkte)

Die beiden Funktionen str_unter(txt) und str_rueck(txt) erwarten als Übergabeparameter jeweils eine Zeichenkette.

- Die Funktion str_unter(txt) gibt alle Zeichen der Zeichenkette von oben nach unten auf dem Bildschirm aus (siehe Bildschirmfoto).
- Die Funktion str_rueck(txt) gibt die Zeichenkette in umgekehrter Reihenfolge wieder aus, ebenfalls alle Zeichen untereinander.

Das Bildschirmfoto zeigt die Ausgabe, nachdem zuerst der Funktionsaufruf str_unter("Python") und danach der Funktionsaufruf str_rueck("Python") ausgeführt wurde.

Schreiben Sie den Python-Quelltext dieser beiden Funktionen.



Aufgabe 4: (ca. 12 Punkte)

4.1. Wie lauten die Ausgaben der folgenden Python-Befehle?

`print(99 / 4)`	Lösung:
`print(99 // 4)`	Lösung:
`print(99 % 4)`	Lösung:

4.2. Wandeln Sie die folgenden Dual- und Hexadezimalzahlen in Dezimalzahlen um.

$0{,}0101_2$	Dezimalzahl:
$1{,}0101_2$	Dezimalzahl:
AAA_{16}	Dezimalzahl:

4.3. Wandeln Sie die folgende Dezimalzahl in eine Dualzahl um.
Es genügt, wenn Sie das Ergebnis mit sechs Nachkommastellen aufschreiben.

$12{,}3_{10}$ = ??$_2$

(Platz für Nebenrechnungen und Notizen)