Ingenieurinformatik I

Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

Prüfer: **Küpper, Krug, Hinz, Ressel, Tasin**

Bearbeitungszeit: **60 Minuten**

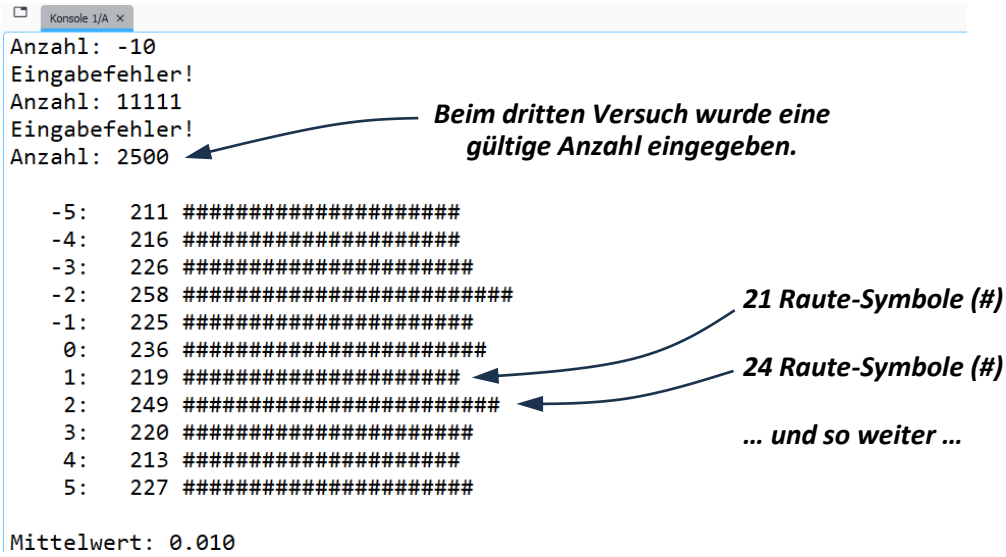
Hilfsmittel: **Taschenrechner nicht zugelassen,**
PC/Notebook/Tablet/Handy nicht zugelassen,
sonstige eigene Hilfsmittel sind erlaubt,
Bearbeitung mit Bleistift ist erlaubt.

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

Aufgabe 1: (ca. 20 Punkte)

Schreiben Sie ein Python-Skript, mit dem untersucht werden kann, ob die mit der Funktion randint() generierten Zufallszahlen gleichmäßig über einen vorgegebenen Zahlenbereich verteilt sind. Hinweis: Die Funktion randint() ist im Modul random enthalten.

1. Zunächst wird die Anzahl „n“ der Zufallszahlen per Tastatur eingegeben. Wenn eine Anzahl kleiner als 1000 oder größer als 5000 eingegeben wird, dann wird eine Fehlermeldung ausgegeben und die Eingabe wird wiederholt.
2. Nachdem die Anzahl „n“ eingegeben wurde, erzeugt Ihr Skript „n“ ganzzahlige Zufallszahlen im Bereich von einschließlich -5 bis einschließlich +5.
3. Die Häufigkeit, mit der die verschiedenen Zufallszahlen auftreten, soll von Ihrem Skript gezählt werden. Außer dem Modul random dürfen keine weiteren Module verwendet werden. Sie dürfen aber zur Speicherung der Häufigkeiten eine Python-Liste mit einem geeignet gewählten Index-Bereich verwenden.
4. Das Ergebnis soll übersichtlich auf dem Bildschirm ausgegeben werden, wie in dem abgebildeten Bildschirmfoto gezeigt:
    - Für jede der verschiedenen Zufallszahlen sollen die Anzahl und auch eine grafische Darstellung mit dem Raute-Symbol (#) ausgegeben werden.
    - Die Zufallszahl und die dazugehörende Anzahl sollen jeweils rechtsbündig mit einer Feldbreite von 5 Zeichen ausgegeben werden.
    - Ein einzelnes Raute-Symbol (#) entspricht jeweils einer Anzahl von 10. Tritt eine Zufallszahl beispielsweise 207 mal auf, werden 20 Raute Symbole angezeigt. Wären es 210 mal, dann würden 21 Raute Symbole angezeigt.
5. Schließlich wird noch der Mittelwert von allen Zufallszahlen mit drei Nachkommastellen auf dem Bildschirm ausgegeben.



Aufgabe 2: (ca. 20 Punkte)

Das folgende Python-Skript simuliert den Anlauf eines Elektromotors und zeigt den zeitlichen Verlauf der Motordrehzahl in einem Diagramm.

```
import math
import matplotlib.pyplot as plt

DT = 0.0001  # Simulations-Zeitschritt

u = w = t = 0.0
w7000 = 2 * pi * (7000 / 60)
t7000 = 0.0

tt = []
ww = []

for i in range(501)
    # Aktuelle Betriebsspannung
    u = 42.0 * t / 0.01
    if u > 42.0:
        u = 42.0

    # Zeit, Winkelgeschwindigkeit berechnen
    t += DT
    w += DT * 3567.2 * (u - w * 0.05236)

    # Auswertung
    tt.append(t)
    ww.append(w)
    if w < w7000:
        t7000 = t

if t7000 < t:
    msg = f"DC-Motor, 7000/min nach {t7000:.3f} s."
    else:
    msg = "DC-Motor, 7000/min nicht erreicht!"

# Die folgenden Zeilen dürfen im Struktogramm zu einem einzelnen Struktur-
# block "Grafische Ausgabe der Motordrehzahl“ zusammengefasst werden.
plt.plot(tt, ww, b-)
plt.plot([tt[0], tt[-1]], [w7000, w7000], "r--")         # ??
plt.xlabel("Zeit in s")
plt.ylabel("Winkelgeschwindigkeit in rad/s")
plt.title(msg)
plt.grid(True)
plt.show
```

2.1. Korrigieren Sie die **fünf Fehler**, die sich in den abgebildeten Quelltext eingeschlichen haben.

2.2. Wie oft wird die for-Schleife durchlaufen?

2.3. Welchen Wert hat die Variable u, **nachdem** die for-Schleife verlassen wurde?

2.4. Wozu dient die mit **# ??** markierte Programmzeile?

2.5. Zeichnen Sie ein Struktogramm, welches den genauen Ablauf des (korrigierten) Skripts zeigt. Alle im Quelltext vorhandenen Kontrollstrukturen müssen erkennbar sein.

Hinweis: Es ist zur Vereinfachung erlaubt, die **letzten sieben Programmzeilen in einem gemeinsamen Block** mit der Beschriftung „grafische Ausgabe der Motordrehzahl“ zusammenzufassen.

Aufgabe 3: (ca. 10 Punkte)

3.1. Wie lautet die Ausgabe des folgenden Python-Skripts?

```
sum = 0
for z in range(1, 6):
    for s in range(1, 10):
        sum += 1
        if s == 5:
            break
print(f"sum = {sum}")
```

Ausgabe:

3.2. Ändern Sie das Python-Skript aus Aufgabe 3.1 wie folgt:

- Ersetzen Sie die beiden for-Schleifen durch äquivalente while-Schleifen.
- Das Verhalten des Programms und die Ausgabe sollen exakt gleich bleiben. Nicht nur die Ausgabe, sondern auch der Kontrollfluss (Schleifenstruktur) soll beibehalten werden.
- Auch die if-Anweisung (inkl. break) soll in dem geänderten Skript weiterhin an der entsprechenden Position vorkommen.

Geändertes Skript:

Aufgabe 4: (ca. 12 Punkte)

Wie lauten die Ausgaben der folgenden print-Befehle?

`txt = "ABCDEFGHIJKLMNOP"` `print(f"{txt[0:5]}")`	*Ausgabe:*
`print(f"{txt[11:-1]}")`	*Ausgabe:*
`print(f"{txt[-3:-1]}")`	*Ausgabe:*
`print(f"{txt[-3:]}")`	*Ausgabe:*
`print(f"{txt[-3]}")`	*Ausgabe:*
`op = 99999999 % 5` `print(f"{op}")`	*Ausgabe:*

Aufgabe 5: (ca. 5 Punkte)

5.1. Wandeln Sie die folgende Dezimalzahl in eine Dualzahl (Binärsystem) um. Geben Sie nicht nur das Endergebnis, sondern auch **alle Zwischenschritte** Ihrer Berechnung an.

$123_{DEZ} = ??_{BIN}$

5.2. Wandeln Sie dieselbe Zahl 123_{DEZ} ins Hexadezimalsystem um.

$123_{DEZ} = ??_{HEX}$

(Platz für Notizen und Zwischenrechnungen)