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**

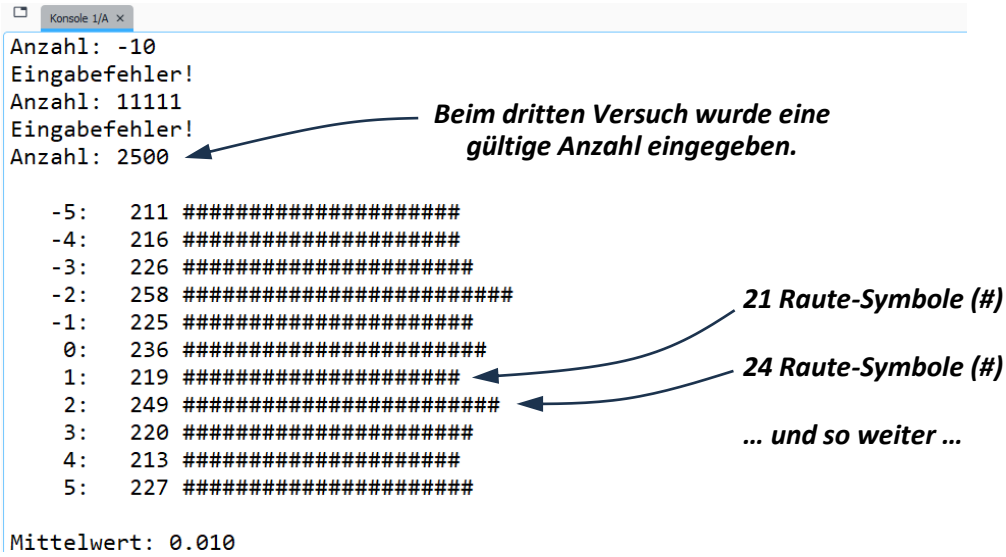
Hilfsmittel: **Taschenrechner nicht zugelassen,**
PC/Notebook/Tablet/Handy nicht zugelassen,
sonstige eigene Hilfsmittel sind erlaubt,
Bearbeitung mit Bleistift ist erlaubt.

****** 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)

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. 22 Punkte)

Schreiben Sie ein Python-Skript, welches die Nullstelle x_0 der Funktion $f(x)$ numerisch ermittelt. Hinweis: Sinus im Bogenmaß berechnen, nicht Gradmaß:

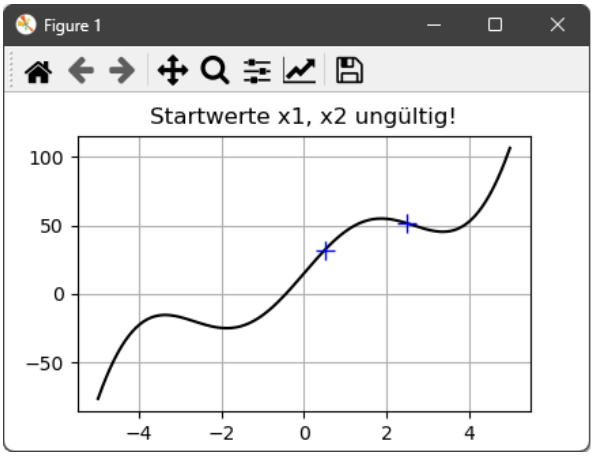
$$f(x) = x^3 + 35 \cdot \sin(x) + 15$$

Programmieren Sie den folgenden Ablauf, beachten Sie auch die beiden Bildschirmfotos:

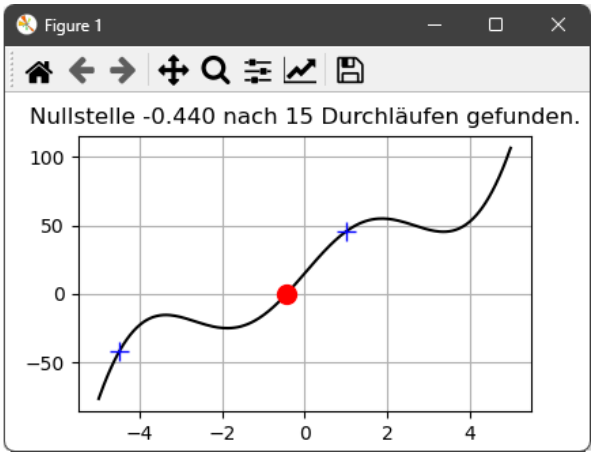
1. Zu Beginn werden zwei float-Werte x_1 und x_2 per Tastatur eingegeben.
2. Der Funktionsverlauf $f(x)$ wird im Bereich $-5 \leq x \leq 5$ in schwarzer Farbe grafisch ausgegeben. Die Positionen $f(x_1)$ und $f(x_2)$ werden durch blaue Plus-Zeichen (+) besonders markiert. Zeichnen Sie auch die in den Bildschirmfotos sichtbaren Hilfslinien (Koordinatengitter).
3. Falls $f(x_1) \cdot f(x_2) \geq 0$ ist, befindet sich die Nullstelle außerhalb des Intervalls (x_1, x_2) oder genau auf seinem Rand. In diesem Fall wird der Titel „Startwerte x1, x2 ungültig!“ am oberen Rand der Grafik ausgegeben. Es werden keine weiteren Aktionen ausgeführt.
4. Andernfalls – wenn also $f(x_1) \cdot f(x_2) < 0$ ist – wird die Nullstelle x_0 mit dem Bisektionsverfahren (Intervallhalbierungsverfahren, siehe weiter unten) numerisch ermittelt.
5. Die Nullstelle wird in der Grafik durch einen roten Punkt markiert.
6. Am oberen Rand der Grafik wird der folgende Titel ausgegeben: „Nullstelle (*) nach (**) Durchläufen gefunden.“ Dabei steht (*) für die Nullstelle x_0 mit drei Nachkommastellen und (**) für die Anzahl der Schleifen-Durchläufe, die vom Bisektionsverfahren benötigt wurden.
7. Definieren Sie die Funktion $f(x)$ nur einmal (!) als Python-Funktion und verwenden Sie diese für alle Berechnungen, um Code-Wiederholungen im Quelltext zu vermeiden.

Das Bisektionsverfahren (Intervallhalbierungsverfahren) wird folgendermaßen implementiert:

a. Es wird ein erster, grober Näherungswert für die Nullstelle x_0 berechnet: $x_0 = (x_1 + x_2)/2$
b. Solange der Betrag $|f(x_0)| > 10^{-3}$ ist, werden die folgenden Schritte in einer Schleife wiederholt:
   - Falls $f(x_0) \cdot f(x_1) \geq 0$ ist, wird die Zuweisung $x_1 = x_0$ durchgeführt, andernfalls $x_2 = x_0$.
   - Es wird ein neuer Wert x_0 berechnet: $x_0 = (x_1 + x_2)/2$
c. Nach dem Ende der Schleife befindet sich die gesuchte Nullstelle in der Variablen x_0.



Zwischen $x_1 = 0{,}5$ und $x_2 = 2{,}5$ befindet sich keine Nullstelle.



Zwischen $x_1 = -4{,}5$ und $x_2 = 1$ wurde die Nullstelle $x_0 = -0{,}440$ gefunden.

Aufgabe 2: (ca. 20 Punkte)

2.1. Wie lautet die Bildschirm-Ausgabe des folgenden Python-Skripts?
Hinweis: Mittels **end=""** wird die Ausgabe eines Zeilenumbruchs verhindert.

```
for i in range(5):
    k = 0
    while k < i:
        print("+", end="")
        k += 1
    while k < 5:
        print("o", end="")
        k += 1
    print("")
```

Ausgabe:

2.2. Zeichnen Sie ein Struktogramm, welches den genauen Ablauf des Skripts aus Aufgabe 2.1 wiedergibt. Alle im Quelltext vorhandenen Kontrollstrukturen müssen erkennbar sein.

2.3. Versuchen Sie, das folgende Skript zu verstehen. Wozu dienen die Befehle im „Abschnitt A“? Wozu wird die if-Abfrage in diesem Abschnitt benötigt?

```
import random

# Abschnitt A:
anz = 20
zz = []
while len(zz) < anz:
    z = random.randint(1, 100)
    if z not in zz:
        zz.append(z)

# Abschnitt B:
m1 = 0
i1 = 0
for idx in range(anz):
    if zz[idx] > m1:
        m1 = zz[idx]
        i1 = idx
print(f"Größtes Element: {m1}")

# Abschnitt C:
```

2.4. Programmieren Sie nun den „Abschnitt C“: Hier wird das zweitgrößte (!) Element in der Liste gesucht und auf dem Bildschirm ausgegeben.

Die Methode sort(), die Funktion sorted() sowie externe Bibliotheksfunktionen dürfen nicht verwendet werden.

Aufgabe 3: (ca. 9 Punkte)

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

```
from math import sin, cos, pi

def my_fun(a, b):
    a *= pi
    b *= pi
    return sin(a), cos(b)

a, b = my_fun(1, 2)
print(f"a = {a:.1f}, b = {b:.1f}")
```

Ausgabe:

3.2. Wie lautet die Ausgabe des folgenden Skripts? Achten Sie auch auf die korrekte Reihenfolge.

```
def f1():
    x = 20
    print("a:", x)

def f2():
    global x
    x = 30
    print("b:", x)

x = 10
print("c:", x)
f1()
print("d:", x)
f2()
print("e:", x)
```

Ausgabe:

Aufgabe 4: (ca. 4 Punkte)

Programmieren Sie die Funktion analyse(text), die einen Text auswertet:

- Sie gibt 1 zurück, wenn das Wort "computer" (in Kleinbuchstaben) irgendwo im Text vorkommt.
- Sie gibt 2 zurück, wenn das Wort "science" vorkommt.
- Sie gibt 3 zurück, wenn beide Wörter vorkommen.
- Sie gibt 0 zurück, wenn keines der beiden Wörter vorkommt.

```
def analyse(text):
```

```
# Hier beginnt das Hauptprogramm.
t = input("Text: ")
a = analyse(t)
print(f"Ergebnis: {a}")
```

Aufgabe 5: (ca. 12 Punkte)

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

`print(123 != 123)`	Lösung:
`print(9999999 % 4)`	Lösung:

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

$110{,}011_2$	Dezimalzahl:
AAA_{16}	Dezimalzahl:

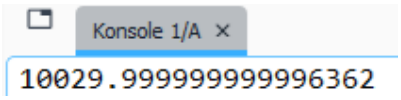
Ergebnis als Kommazahl schreiben, nicht mit Bruchstrich!

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

$100{,}3_{10} = ??_2$

5.4. Das folgende Skript gibt als Ergebnis die Zahl 10029.999999999996362 aus. Begründen Sie, wieso es zur Ausgabe dieser „krummen“ Zahl kommt.

```
x = 100.3
summe = 0
for i in range(100):
    summe += x
print(f"{summe:.15f}")
```



(Platz für Nebenrechnungen und Notizen)

Ingenieurinformatik I

Programmierung

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

Aufgabensteller: **Küpper, Tasin, Ressel und Koll.**

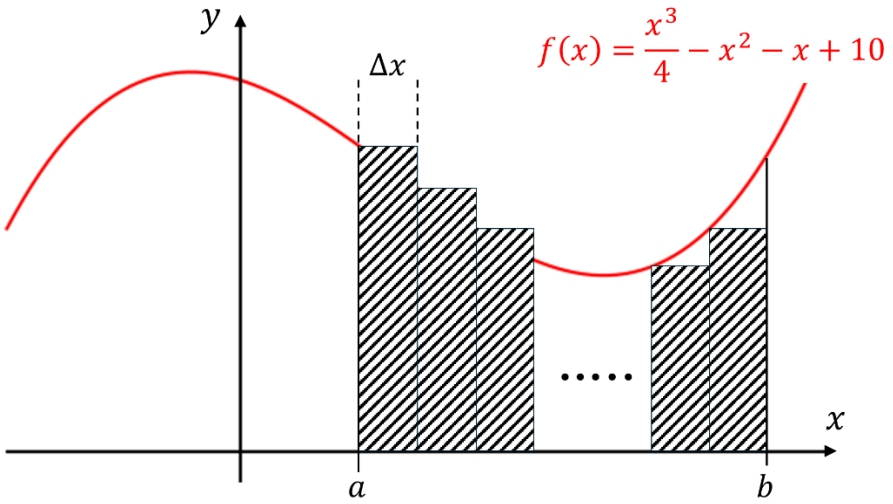
Bearbeitungszeit: **60 Minuten**

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. 21 Punkte)

Schreiben Sie ein Python-Skript, das die Fläche unter der Funktion $f(x)$ im Intervall $a \ldots b$ auf zwei unterschiedliche Arten ermittelt.



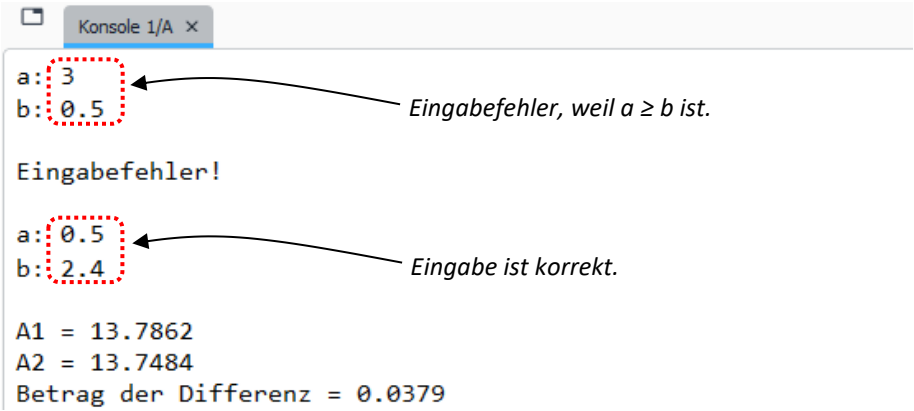
- Nach dem Start werden die Intervallgrenzen a und b über die Tastatur eingegeben.
- Falls $a \geq b$ ist, wird eine Fehlermeldung ausgegeben und die Eingabe wird wiederholt.
- Ihr Skript berechnet die gesuchte Fläche A_1 zunächst näherungsweise als Summe von 100 rechteckigen Teilflächen der Breite Δx:

$$A_1 \approx \sum_{i=0}^{99} \Delta x \cdot f(a + i \cdot \Delta x) \qquad \text{mit} \quad \Delta x = \frac{b-a}{100}$$

- Nun berechnet Ihr Skript über die Stammfunktion $F(x)$ den exakten Wert der Fläche A_2:

$$A_2 = \int_a^b f(x)\,\mathrm{d}x = F(b) - F(a)$$

- Beide Werte A_1 und A_2 werden mit vier Nachkommastellen auf dem Bildschirm ausgegeben.
- Auch der Betrag (!) der Differenz von A_1 und A_2 wird mit vier Nachkommastellen ausgegeben.
- Der Ablauf Ihres Python-Skripts soll so aussehen, wie im Bildschirmfoto gezeigt.



Hinweise und Tipps:

- Wie lautet die Stammfunktion $F(x)$ zur oben angegebenen Funktion $f(x)$?
- An einigen Stellen in Ihrem Skript müssen die Funktionswerte $f(x)$ bzw. die Werte der Stammfunktion $F(x)$ ausgerechnet werden. Dies geht besonders einfach, wenn Sie zu Beginn Ihres Skripts zwei Python-Funktionen zur Berechnung von $f(x)$ bzw. $F(x)$ definieren.

Aufgabe 2: (ca. 20 Punkte)

2.1 Welche Ergebnisse werden von dem folgenden Python-Skript auf dem Bildschirm ausgegeben?

```
a = 10
b = -10
c = -10

if a <= b:
    if a <= c:
        fall = 1
        wert = a
    else:
        fall = 2
        wert = c
else:
    fall = 4
    wert = c
    if b <= c:
        fall = 3
        wert = b

print(f"fall = {fall}")
print(f"wert = {wert}")
```

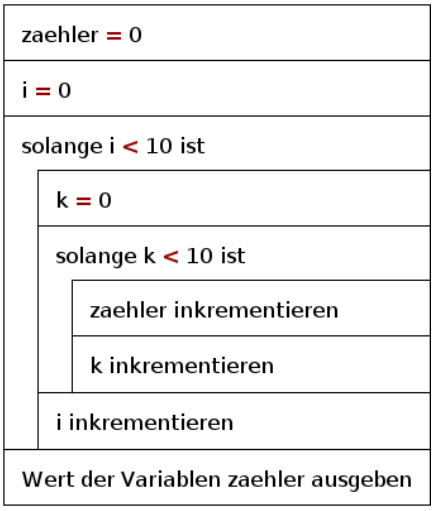
Ausgabe:

fall =

wert =

2.2. Zeichnen Sie ein Struktogramm, das dem Skript aus Aufgabe 2.1. genau entspricht.

2.3. Erstellen Sie ein Python-Skript, das dem abgebildeten Struktogramm genau entspricht.
Hinweis: Die Erhöhung eines Integer-Werts um 1 nennt man „inkrementieren“.



2.4. Welcher Wert wird am Ende des Skripts auf dem Bildschirm ausgegeben?

Ausgabe:

Aufgabe 3: (ca. 4 Punkte)

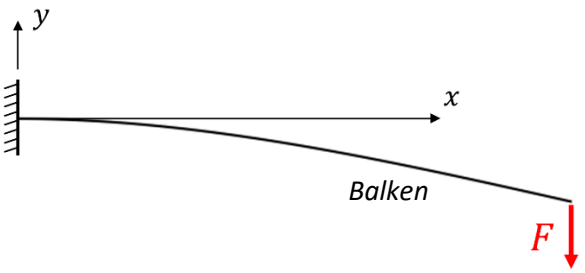
Welcher Text wird von dem folgenden Python-Skript auf dem Bildschirm ausgegeben?

```
txt = "MEaTsecchhbnaiuksstuuddii"

for i in range(1, 16, 2):
    print(f"{txt[i]}", end="")
```

Ausgabe:

Aufgabe 4: (ca. 12 Punkte)

Ein Balken der Länge $l = 0{,}3\ \mathrm{m}$ ist mit seinem linken Ende fest eingespannt. Am rechten Ende des Balkens wirkt die Kraft F nach unten.

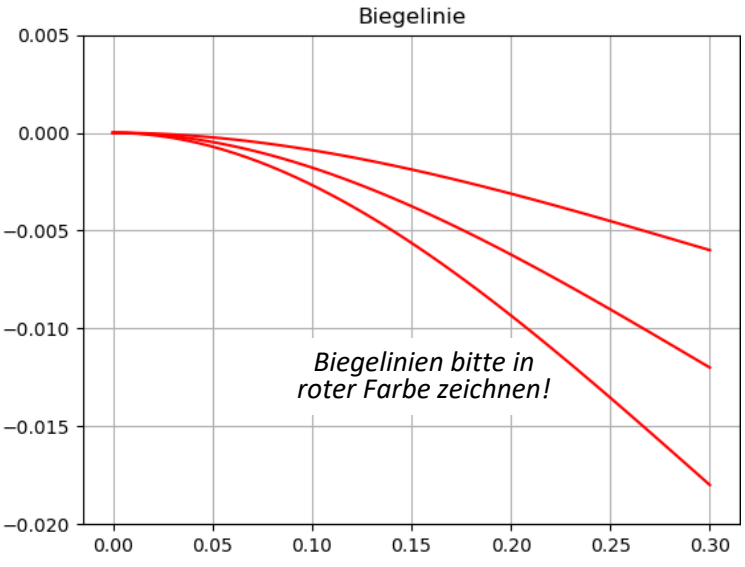


Die Durchbiegung $y(x)$ des Balkens [in m] verursacht durch die Kraft F [in N] an der Position x [in m] kann folgendermaßen berechnet werden:

$$y(x) = -\frac{F}{9} \cdot (0{,}9 \cdot x^2 - x^3)$$

Erstellen Sie ein Skript, welches die Durchbiegung $y(x)$ im Bereich $0\ \mathrm{m} \leq x \leq 0{,}3\ \mathrm{m}$ für drei verschiedene Kräfte $F_1 = 1\ \mathrm{N}$, $F_2 = 2\ \mathrm{N}$ und $F_3 = 3\ \mathrm{N}$ grafisch darstellt. Denken Sie auch an die Ausgabe des Koordinatensystems (Gitterlinien, siehe Bildschirmfoto!) und an den Titel „Biegelinie“.

Hinweis: Berechnen Sie mindestens 100 Punkte für jede Biegelinie.



Aufgabe 5: (ca. 10 Punkte)

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

`print(123 == 123)`	Lösung:
`print(25 % 2)`	Lösung:
`print(25 // 2)`	Lösung:

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

$00{,}011_2$	Dezimalzahl:
$11{,}011_2$	Dezimalzahl:
$A01_{16}$	Dezimalzahl:

5.3. Wandeln Sie die Dezimalzahl 4000 zunächst in eine Dualzahl und anschließend in eine Hexadezimalzahl um.

4000_{10} = ??$_2$ = ??$_{16}$

(Platz für Nebenrechnungen und Notizen)

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}")
```

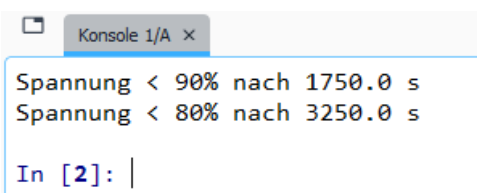
Hinweise:

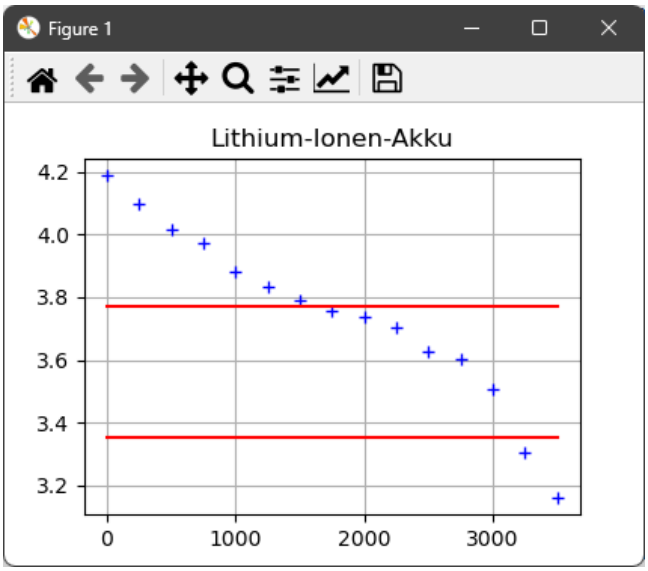
- Das abgebildete Beispielskript liest alle Zeitpunkte und Akku-Spannungen aus der Textdatei und gibt sie auf dem Bildschirm aus. Sie dürfen die Anweisungen aus dem Beispielskript in Ihrer Lösung verwenden.
- Die abgebildeten Werte sind nur Beispiele. Bei anderen Akkus ergeben sich andere Werte. Auch 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.





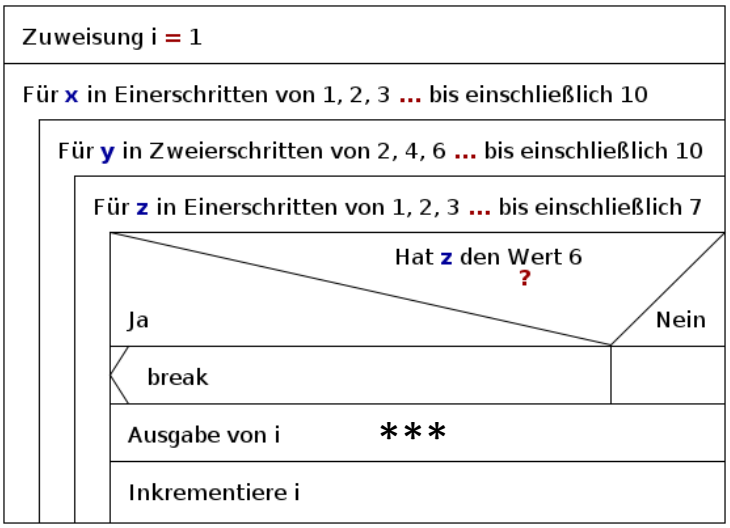
Aufgabe 2: (ca. 25 Punkte)

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 =

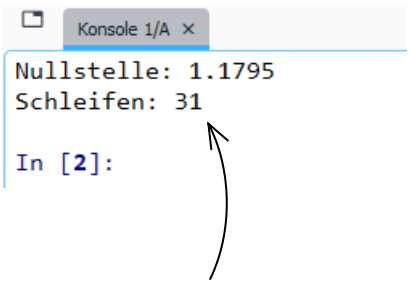


2.2. Schreiben Sie ein Python-Skript, welches dem abgebildeten Struktogramm genau entspricht. Achten Sie insbesondere auf die korrekte Einrückung der for-Schleifen und der Verzweigung.

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.

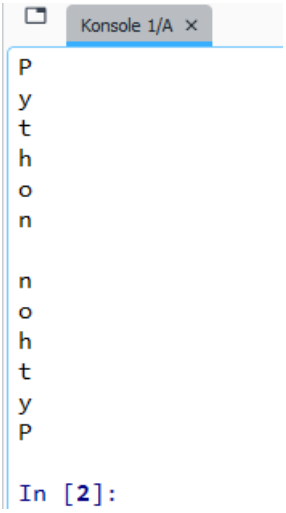
Aufgabe 3: (ca. 8 Punkte)

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)

Ingenieurinformatik

Teil 1, Programmierung in Python

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

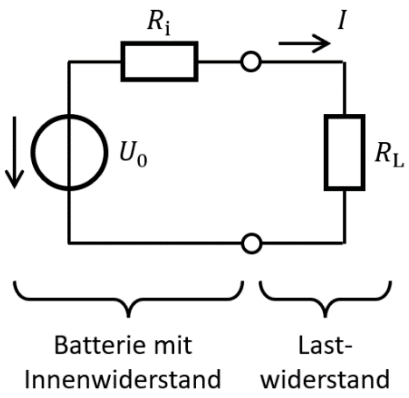
Aufgabensteller: **Reichl, Küpper und KollegInnen**

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. 20 Punkte)



Ein Lastwiderstand R_L ist an eine Batterie mit Leerlaufspannung $U_0 = 1{,}5$ V und Innenwiderstand R_i angeschlossen. Am Lastwiderstand R_L wird die Leistung P_L abgegeben. Am Innenwiderstand R_i geht die Verlustleistung P_i als Wärme verloren.

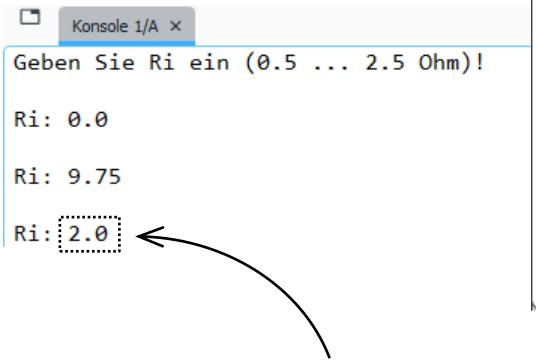
Schreiben Sie ein Python-Skript, welches die folgenden Aufgaben löst. Die Bildschirmfotos unten auf dieser Seite zeigen, wie Ihr Skript ablaufen soll.

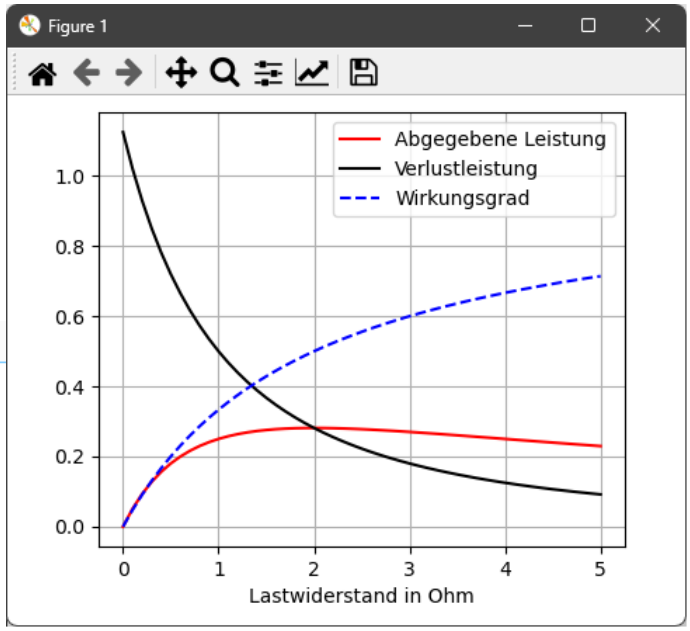
- Zu Beginn des Programms wird eine kurze Begrüßungsmeldung („Geben Sie Ri ein …“) angezeigt. Anschließend wird die Größe des Innenwiderstands R_i abgefragt.
- Wenn der eingegebene Wert nicht im Bereich von 0.5 bis 2.5 Ω liegt, wird die Eingabe wiederholt.
- Das Programm berechnet nun für alle Werte von R_L im Bereich 0 bis 5 Ω (mit einer Schrittweite von 0,1 Ω) die folgenden Größen:
  - Die abgegebene Leistung P_L (in Watt),
  - die Verlustleistung P_i (in Watt),
  - den Wirkungsgrad η (im Bereich 0 … 1).
- Die berechneten Werte von P_L, P_i und η werden grafisch auf dem Bildschirm ausgegeben:
  - x-Achse: Lastwiderstand R_L im Bereich von 0 bis 5 Ω,
  - y-Achse: P_L (rote Linie), P_i (schwarze Linie) und η (blaue gestrichelte Linie),
  - die Gitterlinien und auch die Beschriftung der x-Achse sollen angezeigt werden, die Ausgabe der Legende ist optional (also nicht erforderlich).

Alle Ergebnisse sollen in grafischer Form angezeigt werden.
Eine Ausgabe in Textform, zum Beispiel als Tabelle, ist nicht Teil dieser Aufgabe!

Tipps zur Berechnung:

$$I = U_0 \,/\, (R_i + R_L)$$
$$P_i = I^2 \cdot R_i$$
$$P_L = I^2 \cdot R_L$$
$$\eta = P_L \,/\, (P_i + P_L)$$





Bei der dritten Eingabe wird ein gültiger Wert für R_i eingegeben.
Anschließend werden die Berechnungsergebnisse grafisch dargestellt.

Aufgabe 2: (ca. 15 Punkte)

Der folgende Python-Quelltext ist gegeben:

```
from random import randint
zahl = randint(0, 100)
x = -1
anz = 0

while x != zahl:
    x = int(input("Geben Sie eine Zahl ein: "))
    anz += 1

    if x == zahl:
        print("Ende!")
    else:
        if x < zahl:
            print(f"{x} ist zu klein.")
        if x > zahl:
            print(f"{x} ist zu groß.")

print(f"Insgesamt {anz} Versuche.")
```

2.1. Zu Beginn des Skripts wird die Variable „zahl“ auf einen zufälligen Wert gesetzt.
In welchem Bereich liegt dieser Wert?

Kleinster möglicher Wert:	Größter möglicher Wert:

2.2. Warum wird die Variable „x“ zu Beginn des Skripts auf -1 gesetzt?

2.3. Wozu dient das abgebildete Python-Skript, was wurde hier programmiert?

2.4. Zeichnen Sie ein Struktogramm, welches den genauen Ablauf des abgebildeten Skripts wiedergibt. Alle im Quelltext vorhandenen Schleifen sowie die (verschachtelten) Verzweigungen sollen erkennbar sein.

Struktogramm:

Aufgabe 3: (ca. 7 Punkte)

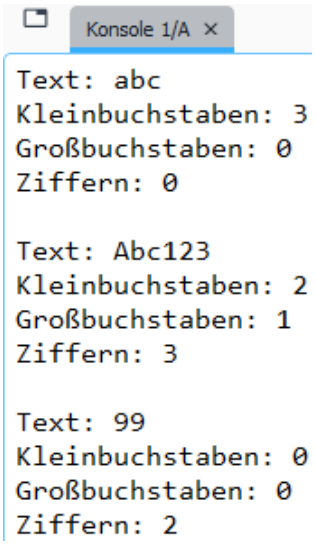
Welche Ausgabe erzeugt das folgende Skript auf dem Bildschirm?

Für jedes Leerzeichen, das ausgegeben wird, schreiben Sie bitte einen Strich _ in Ihre Lösung.

```
from math import sqrt
print("  Schleife")
for i in range(3):
    d = (i ** 2) ** 2
    f = sqrt(d)
    print(f"{d:4d};{f:5.1f}")
```

_	_	S	c	h	l	e	i	f	e				

Aufgabe 4: (ca. 15 Punkte)



4.1. Das folgende Python-Skript liest in einer Schleife immer weitere Texte (Strings) von der Tastatur ein. Nach jeder Eingabe werden die Klein- und Großbuchstaben sowie die Ziffern im Text (String) gezählt. Das Bildschirmfoto auf der rechten Seite zeigt, wie das Skript vom Rechner ausgeführt wird.

Wie lauten die fehlenden Teile des Quelltextes?

```
# Zu Beginn wird eine Funktion definiert.
[                                                    ]
    klein = gross = ziffer = 0
    for x in txt:
        if x in "abcdefghijklmnopqrstuvwxyz":
            klein += 1
        elif x in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
            gross += 1
        elif [                                       ]
            ziffer += 1
[                                                    ]

# Ab hier beginnt das "Hauptprogramm".
[                  ] = input("Text: ")
while len(my_text) > 0 and my_text != "exit":
    a, b, c = statistik( [                           ] )
    print(f"Kleinbuchstaben: {a}")
    print(f"Großbuchstaben: {b}")
    print(f"Ziffern: {c}")
[                                                    ]
```

4.2. Mit welcher Text-Eingabe kann das Skript aus Unterpunkt 4.1 beendet werden? Nennen Sie zwei unterschiedliche Möglichkeiten!

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

`print(999 % 4)`	Lösung:
`print(99 // 4)`	Lösung:
`print(1 == 2)`	Lösung:

Aufgabe 5: (ca. 10 Punkte)

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

$0,1010_2$	Dezimalzahl:
$1111\ 1111_2$	Dezimalzahl:
$11,11_2$	Dezimalzahl:
10_{16}	Dezimalzahl:
FF_{16}	Dezimalzahl:

Wandeln Sie die folgende Dezimalzahl in eine Dualzahl um.

$100_{10} = ??_2$

(Platz für Nebenrechnungen und Notizen)

Ingenieurinformatik

Teil 1, Programmierung in Python

Name	Vorname	Semestergruppe	Hörsaal

	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Summe

Aufgabensteller: **Reichl, Küpper und KollegInnen**

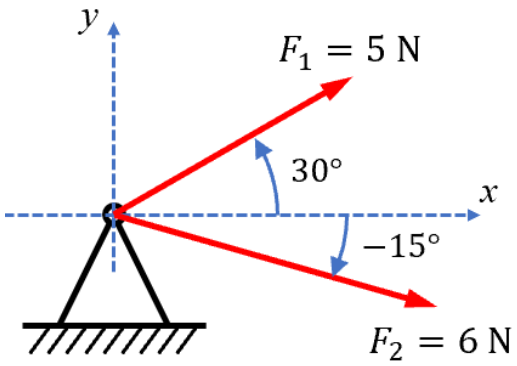
Bearbeitungszeit: **60 Minuten**

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. 25 Punkte)

Auf ein Festlager wirken beliebig viele einzelne Kräfte F_1, F_2, F_3 usw. wie im Bild rechts dargestellt. (Im Bild sind nur zwei Kräfte F_1 und F_2 abgebildet.)

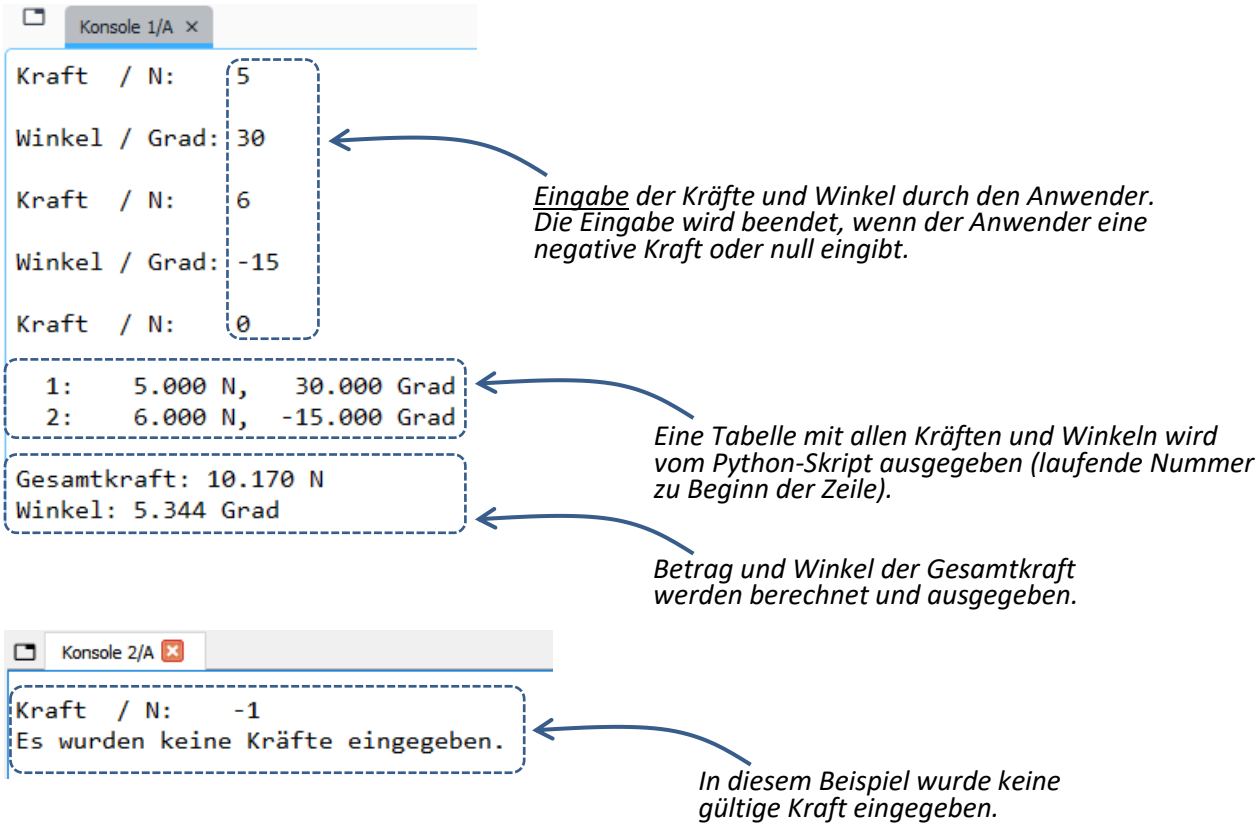


Schreiben Sie ein Python-Skript, welches die folgenden Aufgaben löst:

- Nach dem Start wird von allen Kräften der Betrag und der Winkel eingegeben. Sie dürfen davon ausgehen, dass alle Winkel im Bereich $-90° < \alpha < +90°$ liegen. Die Eingabe wird beendet, wenn als Kraft ein negativer Wert oder null eingegeben wird.
- Die eingegebenen Kräfte und Winkel werden in Python-Listen gespeichert.
- Nach dem Ende der Eingabe wird eine Tabelle mit allen Kräften und Winkeln ausgegeben. Die Kräfte und Winkel werden mit einer Feldbreite von 8 Zeichen und 3 Nachkommastellen ausgegeben. Zu Beginn jeder Zeile wird eine laufende Nummer ausgegeben, zum Beispiel „1:“.
- Schließlich wird die Gesamtkraft F_{ges} ausgerechnet: Der Betrag und der Winkel von F_{ges} werden mit 3 Nachkommastellen ausgegeben. Wenn keine Kräfte eingegeben wurden, dann lautet die Ausgabe: „Es wurden keine Kräfte eingegeben.“

Hinweise:

- Sie dürfen davon ausgehen, dass nur gültige Zahlenwerte eingegeben werden; Fehler bei der Eingabe müssen von Ihrem Skript also nicht erkannt werden.
- Die Ausgabe Ihres Skripts soll so aussehen, wie es in den Bildschirmfotos gezeigt ist!



Eingabe der Kräfte und Winkel durch den Anwender. Die Eingabe wird beendet, wenn der Anwender eine negative Kraft oder null eingibt.

Eine Tabelle mit allen Kräften und Winkeln wird vom Python-Skript ausgegeben (laufende Nummer zu Beginn der Zeile).

Betrag und Winkel der Gesamtkraft werden berechnet und ausgegeben.

In diesem Beispiel wurde keine gültige Kraft eingegeben.

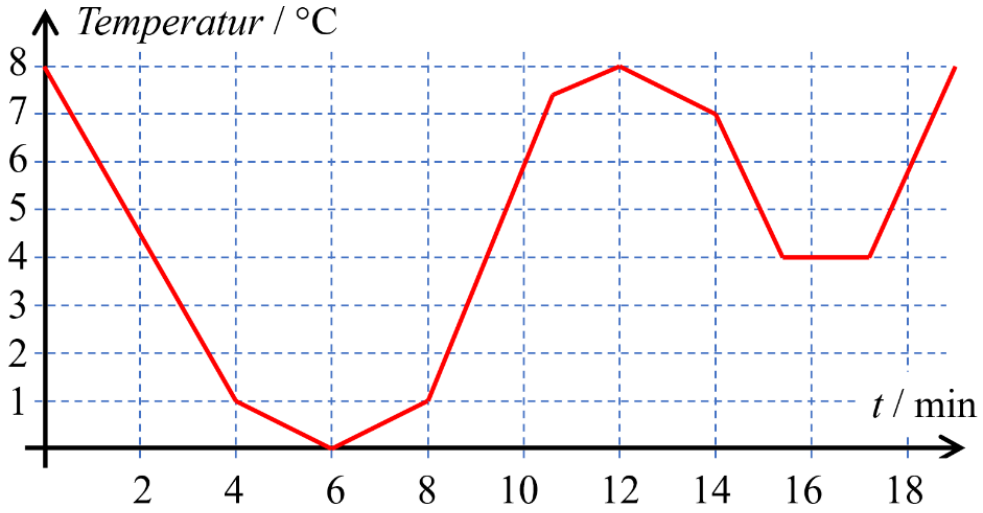
Aufgabe 2: (ca. 18 Punkte)

2.1. Das folgende Python-Skript läuft im Steuergerät eines Kühlschranks.

- Die Funktion get_temp() liefert die aktuelle Temperatur (in °C) im Kühlschrank.
- Die Funktion set_compr() dient zum Ein- bzw. Ausschalten des Kühlschrank-Kompressors.
- **Die genauen Definitionen dieser beiden Funktionen sind nicht abgedruckt, sie können im Modul kuehlschrank.py als gegeben vorausgesetzt werden..**

```
from kuehlschrank import get_temp, set_compr

temp1 = 6.0
temp2 = 2.0
state = 0

while True:
    temp = get_temp()

    if temp > temp1:
        if state == 0:
            set_compr(1)  # einschalten
            state = 1

    if temp < temp2:
        if state == 1:
            set_compr(0)  # ausschalten
            state = 0
```

Das Diagramm zeigt den zeitlichen Verlauf der Temperatur im Kühlschrank. **Markieren Sie im Diagramm die Zeitintervalle**, in denen der Kühlschrank-Kompressor eingeschaltet ist.



2.2. Das Python-Skript aus Unterpunkt 2.1 soll schon einmal getestet werden, obwohl der Kühlschrank noch gar nicht aufgebaut worden ist. Zu diesem Zweck wird die Funktion get_temp() durch die folgende Variante ersetzt: Dadurch werden simulierte (zufällige) Temperaturwerte erzeugt.

```
import random as rnd

def get_temp():
    t = rnd.randrange(15, 65) / 10
    return t
```

Welches ist der minimale bzw. maximale (simulierte) Temperaturwert, der von der abgebildeten Funktion get_temp() als Rückgabewert zurückgegeben werden kann?

Minimaler Temperaturwert: [°C] Maximaler Temperaturwert: [°C]

2.3. Zeichnen Sie ein Struktogramm, welches den genauen Ablauf des Python-Skripts aus dem Unterpunkt 2.1 zeigt. Alle Schleifen und Verzweigungen sollen im Struktogramm sichtbar sein.

Aufgabe 3: (ca. 7 Punkte)

Die Funktion find_first() ermittelt die Position (0, 1, 2 usw.), an der das Zeichen „ch“ zum ersten Mal in der Zeichenkette „s“ vorkommt. Falls das Zeichen „ch“ gar nicht in der Zeichenkette „s“ vorkommt, gibt find_first() den Rückgabewert -1 zurück. Vervollständigen Sie den Quelltext von find_first().

Hinweis: Die Python-String-Methode find() darf in dieser Aufgabe nicht verwendet werden!

```
def find_first(                ):
    anz =
    pos = -1
    for i in                   (0, anz):
        if                     :
                               
            break
    return
```

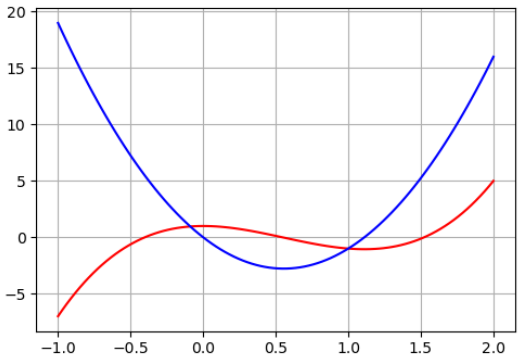
```
s = input("Zeichenkette eingeben: ")
ch = 'x'
pos = find_first(s, ch)

if pos == -1:
    print(f"Das Zeichen '{ch}' wurde nicht gefunden.")
else:
    print(f"Das Zeichen '{ch}' wurde an Position {pos} gefunden.")
```

Aufgabe 4: (ca. 10 Punkte)

Schreiben Sie ein Python-Skript, welches den Verlauf der Funktion $f(x) = 3x^3 - 5x^2 + 1$ im Bereich von $-1 \leq x \leq +2$ zusammen mit deren Ableitung $f'(x)$ grafisch auf dem Bildschirm ausgibt. In der Abbildung soll auch das Koordinatensystem sichtbar sein (siehe Bildschirmfoto).

Die Funktion $f(x)$ soll in roter Farbe, die Ableitung $f'(x)$ soll in blauer Farbe gezeichnet werden.

Tipp: Am einfachsten ist es, wenn Sie die Formel für die Ableitung $f'(x)$ selbst berechnen.



Aufgabe 5: (ca. 7 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.

149_{DEZ} = ??$_{BIN}$

5.2. Wandeln Sie dieselbe Zahl 149_{DEZ} ins Hexadezimalsystem um.

149_{DEZ} = ??$_{HEX}$

(Platz für Notizen und Nebenrechnungen)