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:

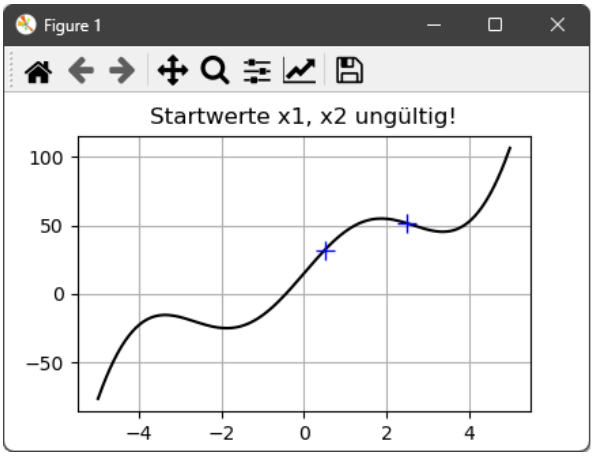
$$f(x) = x^3 + 35 \cdot \sin(x) + 15$$

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

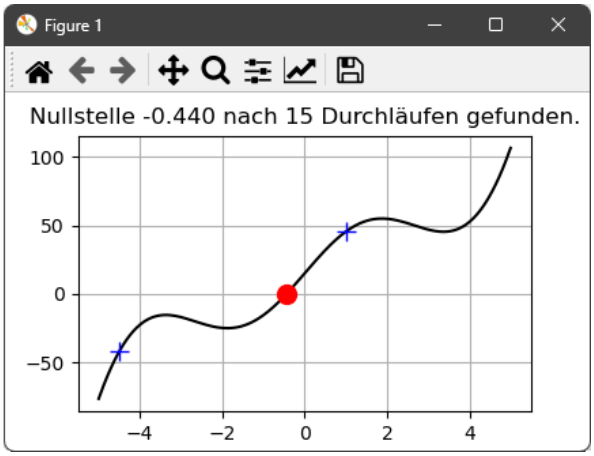
1. Zu Beginn werden zwei float-Werte x_1 und x_2 per Tastatur eingegeben.
2. Der Funktionsverlauf $f(x)$ wird im Bereich $-5 \leq x \leq 5$ in schwarzer Farbe grafisch ausgegeben. Die Positionen $f(x_1)$ und $f(x_2)$ werden durch blaue Plus-Zeichen (+) besonders markiert. Zeichnen Sie auch die in den 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:

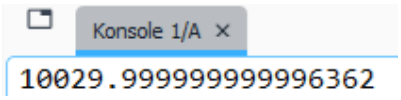
Ergebnis als Kommazahl schreiben, nicht mit Bruchstrich!

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)