1 Freifach Programmieren

1.1 Numerische Algorithmen

Aufgabe A1 (Finanzmathematik) Für die Altersrente zahlt Frau Huber jedes Jahr am 1. Januar den Betrag 6000.- auf ein Sparkonto ein. Dem Konto wird am 31. Dezember jeweils ein Zins von 1% gutgeschrieben. Wie gross ist der Kontostand nach 35 Jahren?

Schreiben Sie ein Programm, das als erstes die drei Variablen `betrag=6000`, `zins=0.01` und `laufzeit=35` definiert und dann in einer `while`- oder `for`-Schleife für jedes Jahr den auf zwei Nachkommastellen gerundeten Kontostand am 31. Dezember (nach der Zinsgutschrift) ausgibt.

Zur Kontrolle: Die letzte Ausgabe sollte wie folgt lauten:

```
Kontostand am Ende des Jahres 35: 252461.27
```

Hinweis: Die Ausgabe von Variablen innerhalb von Strings geschieht besonders einfach mit f-strings (= formatted strings). Beispiel:

```
x = 2
print(f'Die Wurzel von {x} ist {x**0.5}.')
```

Das Zeichen `f` vor dem Anführungszeichen steht für formatted und verwandelt den normalen String in einen f-String.

Wenn man die Wurzel auf 5 Nachkommastellen runden möchte, geht das wie folgt.

```
print(f'Die Wurzel von {x} ist {x**0.5:.5f}.')
```

Das Zeichen `f` in der letzten geschweiften Klammer steht für float.

Aufgabe A2 (Wurzel einer Zahl berechnen, ohne die Wurzelfunktion von Python zu verwenden)

Die Wurzel $\sqrt{x}$ einer positiven Zahl x kann man näherungsweise berechnen, indem man zwei Variablen `links` und `rechts` so initialisiert, dass `links` $\leq \sqrt{x} \leq$ `rechts` gilt. Der gesuchte Wert $\sqrt{x}$ ist also «zwischen `links` und `rechts` eingesperrt». Zum Beispiel kann man `links` $= 0$ und `rechts` $= x$ setzen.

Nun berechnet man den Mittelwert `mittel` $= \frac{\texttt{links}+\texttt{rechts}}{2}$ und vergleicht sein Quadrat `mittel`^{2} mit x. Je nachdem, wie dieser Vergleich ausfällt, ersetzt man `links` oder `rechts` durch `mittel`, so dass nach der Ersetzung wieder `links` $\leq \sqrt{x} \leq$ `rechts` gilt.

Diesen Prozess wiederholt man solange, bis die Differenz `rechts` $-$ `links` kleiner als ein gewünschter Fehler ist.

Ergänzen Sie das folgende Programm so, dass der oben beschriebene Algorithmus realisiert wird!

```
x = 40
fehler = 0.000001
links = 0
rechts = x

while rechts - links > fehler:
    #
    # Hier ist Code zu schreiben.
    #
schaetzung = (links + rechts) / 2
print(f'{schaetzung} ist ungefähr die Wurzel aus {x}.')
print(f'{x**0.5} ist laut Python die Wurzel aus {x}.')
print(f'Der Fehler ist etwa {abs(mittel-x**0.5):.10f}.')
```

Bemerkung: Das beschriebene Verfahren heisst **Intervallschachtelung**, da $\sqrt{x}$ immer besser durch das kleiner werdende Intervall [`links`, `rechts`] eingeschachtelt wird.

Aufgabe A3 Die Zahlenfolge

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

heisst **Fibonacci-Folge**. Die beiden Folgenglieder am Anfang sind $f_0 = 0$ und $f_1 = 1$, jedes nachfolgende Folgenglied ist die Summe seiner beiden Vorgänger, d. h. $f_n = f_{n-2} + f_{n-1}$ für alle $n \geq 2$.

Schreiben Sie ein Programm, dass zeilenweise jeweils die Nummer n und dann das zugehörige Folgenglied f_n ausgibt. Eine am Anfang des Programms definierte Variable `maxn` gibt dabei an, bis zu welchem n die Folge ausgegeben werden soll. Für `maxn = 6` soll die Ausgabe wie folgt aussehen.

```
Nr Fibonaccizahl
0 0
1 1
2 1
3 2
4 3
5 5
6 8
```

Bemerkung: Die Zahlen in der Fibonacci-Folge heissen Fibonacci-Zahlen. Sie beschreiben zum Beispiel das Wachstum einer idealisierten Kaninchenpopulation, vgl. Wikipedia: Fibonacci-Folge, Antike und Mittelalter in Europa.

Aufgabe A4 Fibonacci-Folge als Liste: Schreibe ein Programm, das mit Hilfe einer `while`- oder `for`-Schleife die Liste aller Fibonacci-Zahlen bis zur Zahl mit dem Index `maxn` erzeugt und diese ausgibt.

Hinweis: Definiere anfangs die Liste der Fibonacci-Zahlen per

```
fibonaccifolge = [0, 1]
```

und hänge die weiteren Folgeglieder an diese Liste an. Der folgenden Befehl hängt die Zahl `42` ans Ende einer Liste `liste`:

```
liste.append(42)
```

Aufgabe A5 Fibonacci-Folge mit expliziter Formel: Die explizite Formel für das n-te Glied f_n der Fibonacci-Folge lautet (Formel von Moivre-Binet):

$$f_n = \frac{1}{\sqrt{5}}\left(\varphi^n - \psi^n\right) \qquad \text{wobei } \varphi = \frac{1+\sqrt{5}}{2} \text{ und } \psi = \frac{1-\sqrt{5}}{2}.$$

Schreibe ein Programm, das die beiden reellen Zahlen φ und ψ in Variablen `phi` und `psi` speichert und dann mit der Formel von Moivre-Binet und «list comprehension» (siehe unten) die Liste aller Fibonacci-Zahlen bis zur Zahl mit dem Index `maxn` erzeugt und diese ausgibt.

Hinweise: Mit «list comprehension» kann man elegant Listen, deren Elemente von einer Laufvariablen abhängen, erzeugen. Beispiel:

```
liste = [2**n for n in range(10)]
print(liste)
```

Bemerkung: Die reelle Zahl φ ist der goldene Schnitt; φ und ψ sind die beiden Lösungen der quadratischen Gleichung $x^2 - x - 1 = 0$.

⚒ Aufgabe A6 (Kreiszahl π durch ein Zufallsexperiment näherungsweise bestimmen)

Der folgende Python-Code erzeugt Zufallszahlen zwischen 0 und 1:

```
from random import random
n = 6
while n>0:
    z = random()
    print(z)
    n = n-1
```

```
0.15898359598028566
0.5755205940482977
0.26107854105841055
0.33108976504266363
0.5599435248050862
0.9949230825541513
```

In dieser Aufgabe sollen Sie die Kreiszahl π mit einem Zufallsexperiment näherungsweise bestimmen. Schreiben Sie dafür ein Python-Programm wie folgt:

- Bestimmen Sie zwei Zufallszahlen x und y zwischen 0 und 1.
- Dann ist (x, y) ein Punkt Einheitsquadrat (hier gemeint: das Quadrat der Seitenlänge 1 mit den Eckpunkten $(0,0)$, $(1,0)$, $(1,1)$, $0,1)$ im Koordinatensystem).
  Stellen Sie fest, ob dieser Punkt auch im Einheitskreis (= Kreis mit Radius 1 um den Ursprung des Koordinatensystems) liegt.
- Wiederholen Sie obiges Experiment 1000 oder 1'000'000 mal und zählen Sie die Anzahl der Punkte im Kreis.
- Berechnen Sie den Anteil der Punkte, die im Einheitskreis liegen.
- Wie gross müsste dieser Anteil theoretisch sein? Wie lässt sich somit π näherungsweise berechen? Wie genau ist das Resultat?

Hinweis: Die Kreiszahl π (oder genauer gesagt die Annäherung an π, mit der Python rechnet) kann man wie folgt ausgeben.

```
from math import pi
print(pi)
```

⚒ Aufgabe A7 Kleines Einmaleins:

Schreibe ein Programm, das die folgende Multiplikationstabelle ausgibt.

Hinweise:

- Verwende zwei ineinander verschachtelte Schleifen.
  In der inneren Schleife wird jeweils eine Zeile der Ausgabe aufgebaut.
- Der f-String (= formatted string) `f'{i:4}` erzeugt einen String der Länge 4, in dem die ganze Zahl `i` rechtsbündig steht und davor Leerzeichen (falls genügend Platz vorhanden ist).

```
   * |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   8 |   9 |  10 |
-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
   1 |   1 |   2 |   3 |   4 |   5 |   6 |   7 |   8 |   9 |  10 |
   2 |   2 |   4 |   6 |   8 |  10 |  12 |  14 |  16 |  18 |  20 |
   3 |   3 |   6 |   9 |  12 |  15 |  18 |  21 |  24 |  27 |  30 |
   4 |   4 |   8 |  12 |  16 |  20 |  24 |  28 |  32 |  36 |  40 |
   5 |   5 |  10 |  15 |  20 |  25 |  30 |  35 |  40 |  45 |  50 |
   6 |   6 |  12 |  18 |  24 |  30 |  36 |  42 |  48 |  54 |  60 |
   7 |   7 |  14 |  21 |  28 |  35 |  42 |  49 |  56 |  63 |  70 |
   8 |   8 |  16 |  24 |  32 |  40 |  48 |  56 |  64 |  72 |  80 |
   9 |   9 |  18 |  27 |  36 |  45 |  54 |  63 |  72 |  81 |  90 |
  10 |  10 |  20 |  30 |  40 |  50 |  60 |  70 |  80 |  90 | 100 |
```

Aufgabe A8 Das folgende Programm erzeugt zwei Zufallszahlen zwischen 2 und 10.

```
import random
x = random.randrange(2, 11)
y = random.randrange(2, 11)
print(x)
print(y)
```

Erweitere dieses Programm so, dass der Computer den Benutzer nach dem Produkt der Variablen `x` und `y` fragt und ihm mitteilt, ob er richtig gerechnet hat. Der Dialog mit dem Computer sollte sinngemäss so aussehen:

```
Was ist das Produkt von 5 und 7? 24
Dies ist leider falsch. Richtig wäre 35 gewesen.
```

Bonus: Es gibt viele Erweiterungsmöglichkeiten. Etwa könnten dem Benutzer 10 Multiplikationsaufgaben gestellt werden und es wird ihm danach mitgeteilt, wie viel Prozent der Aufgaben er richtig gelöst hat. Es könnten auch zufällig Divisionsaufgaben (ohne Rest) eingestreut werden.

1.2 Lösungen

Hinweise zu den Symbolen:

⚒ Diese Aufgaben könnten (mit kleinen Anpassungen) an einer Prüfung vorkommen. Für die Prüfungsvorbereitung gilt: “If you want to nail it, you’ll need it”.

Diese Aufgaben sind wichtig, um das Verständnis des Prüfungsstoffs zu vertiefen. Die Aufgaben sind in der Form aber eher nicht geeignet für eine Prüfung (zu grosser Umfang, nötige «Tricks», zu offene Aufgabenstellung, etc.). **Teile solcher Aufgaben können aber durchaus in einer Prüfung vorkommen!**.

Diese Aufgaben sind dazu da, über den Tellerrand hinaus zu schauen und/oder die Theorie in einen grösseren Kontext zu stellen.

⚒ Lösung zu A1 ex-renten-sparen

```
betrag = 6000
zins = 0.01
laufzeit = 35
n = 1
kontostand = 0
while n<=laufzeit:
    kontostand = kontostand+betrag
    kontostand = kontostand*(1+zins)
    print(f"Kontostand am Ende des Jahres {n}: {kontostand:.2f}")
    n = n+1
```

```
Kontostand am Ende des Jahres 1: 6060.00
Kontostand am Ende des Jahres 2: 12180.60
Kontostand am Ende des Jahres 3: 18362.41
Kontostand am Ende des Jahres 4: 24606.03
Kontostand am Ende des Jahres 5: 30912.09
Kontostand am Ende des Jahres 6: 37281.21
Kontostand am Ende des Jahres 7: 43714.02
Kontostand am Ende des Jahres 8: 50211.16
Kontostand am Ende des Jahres 9: 56773.28
Kontostand am Ende des Jahres 10: 63401.01
Kontostand am Ende des Jahres 11: 70095.02
Kontostand am Ende des Jahres 12: 76855.97
Kontostand am Ende des Jahres 13: 83684.53
Kontostand am Ende des Jahres 14: 90581.37
Kontostand am Ende des Jahres 15: 97547.19
Kontostand am Ende des Jahres 16: 104582.66
Kontostand am Ende des Jahres 17: 111688.49
Kontostand am Ende des Jahres 18: 118865.37
Kontostand am Ende des Jahres 19: 126114.02
Kontostand am Ende des Jahres 20: 133435.16
Kontostand am Ende des Jahres 21: 140829.52
Kontostand am Ende des Jahres 22: 148297.81
Kontostand am Ende des Jahres 23: 155840.79
Kontostand am Ende des Jahres 24: 163459.20
Kontostand am Ende des Jahres 25: 171153.79
Kontostand am Ende des Jahres 26: 178925.33
Kontostand am Ende des Jahres 27: 186774.58
Kontostand am Ende des Jahres 28: 194702.33
Kontostand am Ende des Jahres 29: 202709.35
Kontostand am Ende des Jahres 30: 210796.44
Kontostand am Ende des Jahres 31: 218964.41
Kontostand am Ende des Jahres 32: 227214.05
```

```
Kontostand am Ende des Jahres 33: 235546.19
Kontostand am Ende des Jahres 34: 243961.65
Kontostand am Ende des Jahres 35: 252461.27
```

⚒ Lösung zu A2 ex-wurzel-annaehern

```
x = 40
fehler = 0.000001
links = 0
rechts = x

while rechts - links > fehler:
    mittel = (links + rechts) / 2
    if mittel**2 < x:
        links = mittel
    else:
        rechts = mittel

schaetzung = (links + rechts) / 2
print(f'{schaetzung} ist ungefähr die Wurzel aus {x}.')
print(f'{x**0.5} ist laut Python die Wurzel aus {x}.')
print(f'Der Fehler ist etwa {abs(mittel-x**0.5):.10f}.')
```

```
6.324555575847626 ist ungefähr die Wurzel aus 40.
6.324555320336759 ist laut Python die Wurzel aus 40.
Der Fehler ist etwa 0.0000000425.
```

⚒ Lösung zu A3 ex-fibonacci-zahlen

```
maxn = 6
print('Nr Fibonaccizahl')
vorletztes_glied = 0
letztes_glied = 1
nummer = 0
print(nummer, vorletztes_glied)
nummer = nummer + 1
print(nummer, letztes_glied)
while nummer < maxn:
    summe = vorletztes_glied + letztes_glied
    vorletztes_glied = letztes_glied
    letztes_glied = summe
    nummer = nummer + 1
    print(nummer, letztes_glied)
```

Für Nerds, etwas kürzer und rekursiv (was man bei Fibonacci eigentlich nicht machen sollte):

```
maxn = 6
fib = lambda n: n if n < 2 else fib(n-1) + fib(n-2)
print([fib(x) for x in range(maxn + 1)])
```

⚒ Lösung zu A4 ex-fibonacci-zahlen-als-liste

```
maxn = 6
fibonaccifolge = [0, 1]
n = 2
while n <= maxn:
    fibonaccifolge.append(fibonaccifolge[n-1]+fibonaccifolge[n-2])
    n = n+1
print(fibonaccifolge)
```

Mit einer `for`-Schleife statt einer `while`-Schleife und mit «negativen» Indizes `-1` und `-2` (sie beziehen sich auf das letzte und vorletzte Element der Liste). Unwichtige Laufvariablen werden in Python bisweilen «Unterstrich» genannt.

```
maxn = 6
fibonaccifolge = [0, 1]
for _ in range(2, maxn+1):
    fibonaccifolge.append(fibonaccifolge[-1]+fibonaccifolge[-2])
print(fibonaccifolge)
```

⚒ Lösung zu A5 ex-fibonacci-zahlen-list-comprehension-explizite-formel

```
maxn = 6
phi = (1+5**0.5)/2
psi = (1-5**0.5)/2
fibonaccifolge = [1/5**0.5*(phi**n-psi**n) for n in range(maxn+1)]
print(fibonaccifolge)

# Alternative: Wir runden alle Ergebisse mit round(...)
# zu ganzen Zahlen (integers), was Rundungsfehler beseitigt.

fibonaccifolge = [round(1/5**0.5*(phi**n-psi**n)) for n in range(maxn+1)]
print(fibonaccifolge)
```

⚒ Lösung zu A6 ex-pi-wuerfeln

```
from random import random

n = 1000       # Anzahl Punkte = Experimente
i = 0          # Laufvariable für Wiederholung
drinnen = 0    # Zählvariable für Anzahl der Punkte im Kreis

while i<n:
    i += 1          # Kurzform für i = i + 1
    x = random()
    y = random()
    # Pythagoras lässt grüssen
    r = (x*x+y*y)**0.5
    if (r<1):
        drinnen +=1  # Kurzform für drinnen = drinnen + 1

print("Drinnen", drinnen, "von", n)
anteil = drinnen/n;
print("Anteil", anteil)
# Viertelkreisfläche ist pi/4, Quadratfläche ist 1
print("Pi ist ungefähr", anteil*4)
```

```
Drinnen 788 von 1000
Anteil 0.788
Pi ist ungefähr 3.152
```

⚒ Lösung zu A7 ex-multiplikationstabelle

⚒ Lösung zu A8 ex-basics-if-kopfrechentrainer