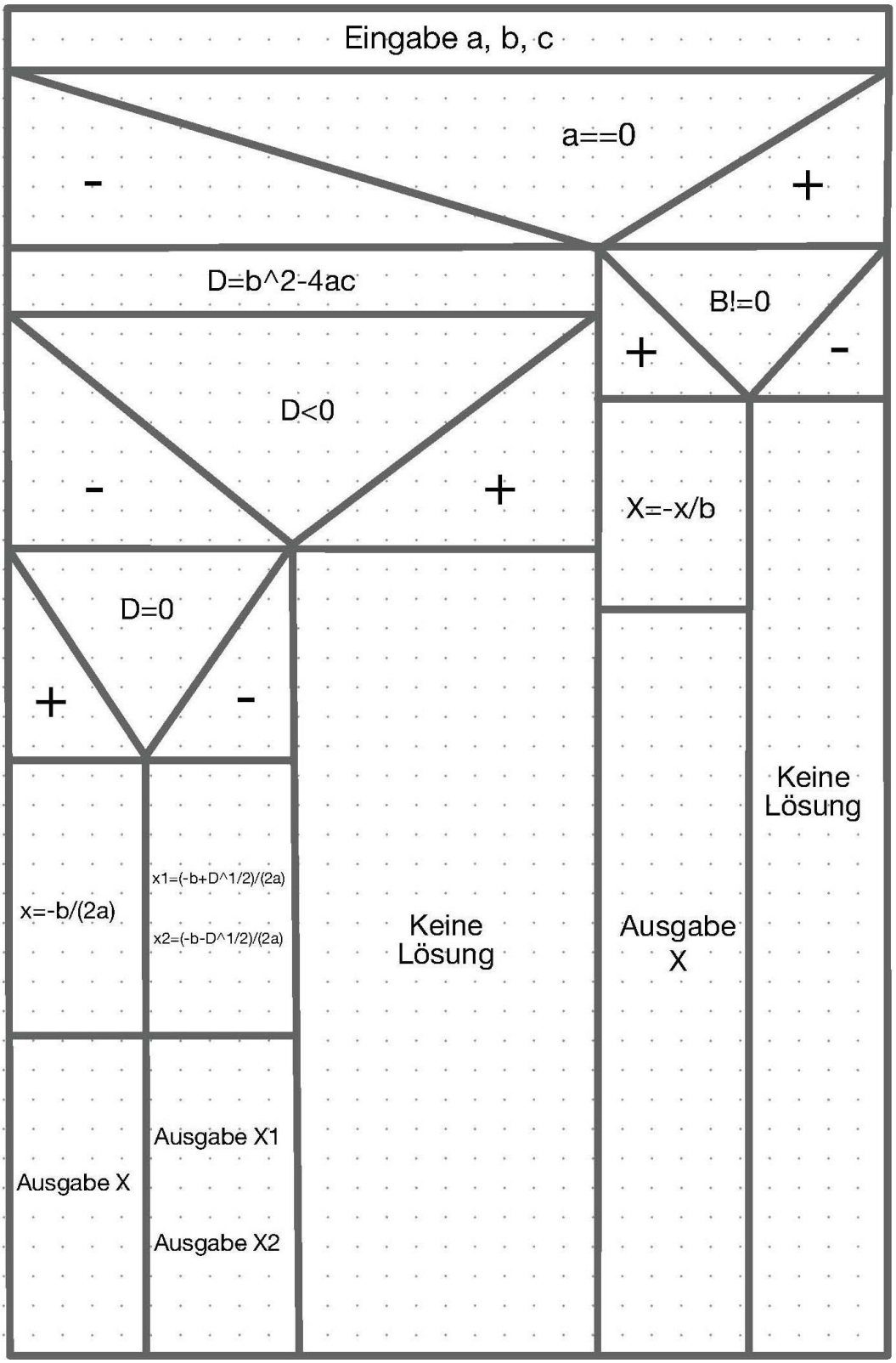
Informatik

inf11 10.03.2023

Lösung zu **Aufgabe 7**:

a) Struktogramm (Jakob)



Aufgabe:
Man überzeuge sich, daß das folgende Python-Programm gemäß obenstehendem Struktrogramm aufgebaut ist.

b) Python-Quelltext:

```
# Quadratische Gleichungen

'Eingabe der Koeffizienten'

print ("Quadratische Gleichung:  a*x^2 + b*x + c = 0")
a=float(input("a = "))
b=float(input("b = "))
c=float(input("c = "))
print()
print ("Loesungsmenge:")

'Verarbeitung der Daten und Ausgabe der Loesungen'

if a == 0:
   print ("Gleichung nicht quadratisch")
   if b!=0:
     print ("x =",-c/b)
   else:
     print ("keine Loesung!")

else:
   D = b**2 - 4*a*c

   if D < 0:
      print ("keine Loesung!")

   else:
      if D == 0:
         x = -b/(2*a)
         print ("x =",x)

      else:
         x1 = (-b + D**(1/2))/(2*a)
         x2 = (-b - D**(1/2))/(2*a)
         print ("x1 =",x1)
         print ("x2 =",x2)
```

3. Algorithmen mit Wiederholungen

Wenn ein Anweisungsblock innerhalb eines Algorithmus wiederholt auszuführen ist, verwenden wir eine Schleife (engl.: loop) als Kontrollstruktur; der zu wiederholende Anweisungsblock heißt auch Schleifenrumpf.

Die Programmiersprache Python kennt die (kopfgesteuerte) while-Schleife und die for-Schleife; in anderen Sprachen (z. B. Java, Pascal, C++) sind auch auch fußgesteuerte Schleifen (repeat-Schleife) implementiert.

while-Schleife

Syntax einer **while**-Schleife in Python:

```
while condition:                while condition:
        Anweisung1                      A
        Anweisung2
        Anweisung3
```

Dabei ist `condition` ein Boolescher Term; der aus einer Anweisung oder mehreren Anweisungen bestehende Schleifenrumpf **A** wird nur dann ausgeführt, falls `condition` den Wert `True` hat.

Beachte: Der Schleifenrumpf ist durch Einrücken des Programmtextes kenntlich zu machen!

Aufgabe 8 (Quadratzahltabelle)

Formuliere einen Algorithmus, welcher nach Eingabe einer natürlichen Zahl n die Quadrate der Zahlen 1, ... , n berechnet und ausgibt.

```
n = int(input('n = '))
```
Eingabe n

```
i = 1
```
Zuweisung eines Anfangswerts an die Zählvariable i (oder: Schleifenindex i)

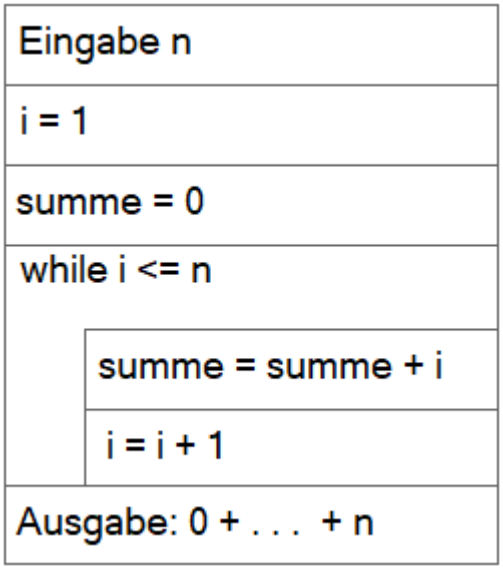
```
while i <= n:

    q = i * i
    print(i,'^2 =',q)
    i = i + 1
```
Schleifenrumpf

Aufgabe 9

Formuliere einen Algorithmus a) als Struktogramm, b) als Python-Programm, welcher nach Eingabe einer natürlichen Zahl n, $n \geq 0$, die Summe der Zahlen 0, ... , n berechnet und ausgibt.

Lösung:

a) Struktogramm:



Initialisierung des Schleifenindex i

Initialisierung der Variablen summe

Abfrage des Booleschen Terms i <= n

Schleifenrumpf

b) Quellcode in Python:

```
# Nach Eingabe einer natürlichen Zahl n wird die
# Summe der Zahlen 0, . . . , n berechnet und ausgegeben

n = int(input('n = '))

'Initialisierung des Schleifenindex i'
i = 1

'Initialisierung der Variablen summe'
summe = 0

while i <= n:
    summe = summe + i
    i = i + 1

print('Summe der Zahlen 0 , . . . ,',n,' =',summe)
```

Bemerkungen:

- Der Schleifenindex i heißt auch Zählindex oder Zähler.
- Da die als Boolescher Term formulierte Bedingung (hier: i <= n) vor Eintritt in den Schleifenrumpf abgefragt wird, handelt es sich bei der while-Schleife um eine kopfgesteuerte Schleife.

Aufgaben:

- Schreibe obenstehendes Struktogramm als Flußdiagramm.
- Verfolge gedanklich die Arbeitsschritte, die der Algorithmus nach der Eingabe von 0, 1, 2, 3 jeweils ausführt.

Trace

Anhand einer Trace-Tabelle können wir für bestimmter Eingabewerte überprüfen, ob der Algorithmus das Verlangte leistet; ein Trace liefert somit eine erste Information darüber, ob der Algorithmus korrekt ist.
Beachte: Ein Trace ersetzt nicht einen allgemeinen Korrektheitsbeweis.

In einer Tabelle notieren wir die Werte aller Variablen, Konstanten und Booleschen Terme (wir verwenden dieselben Variablennamen wie in obenstehendem Struktogramm). Solange der Term **`i <= n`** den Wert **`True`** hat, erfolgt ein weiterer Schleifendurchlauf; der Algorithmus bricht ab, sobald der Term **`i <= n`** den Wert **`False`** annimmt, denn es gibt dann keinen weiteren Schleifendurchlauf. Nach Abbruch wird der Wert der Variablen **`summe`** (hier: 15) ausgegeben.

Trace für den Eingabewert **n = 5**
Abkürzung: **SD** = Schleifendurchlauf

	n	**i**	**summe**	**i <= n**
vor dem 1. SD	5	1	0	True
vor dem 2. SD	5	2	1	True
vor dem 3. SD	5	3	3	True
vor dem 4. SD	5	4	6	True
vor dem 5. SD	5	5	10	True
nach dem 5. SD	5	6	15	False

Aufgabe 10

Formuliere den Algorithmus **FAKULTÄT** unter Verwendung einer while-Schleife
a) als Struktogramm,
b) als Python-Programm,
welcher nach Eingabe einer natürlichen Zahl n, $n \geq 1$, das Produkt der Zahlen 1, ... , n berechnet und ausgibt.

Anmerkung:
Man schreibt auch $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$ (Lies: n-Fakultät).

Beispiele: $6! = 720$ $13! = 6\,227\,020\,800$

Ergänzung:
Definitionsgemäß gilt: 0! = 1. Erweitere den Algorithmus so, daß für die Eingabe n = 0 der Wert 1 ausgegeben wird.

c) Erstelle eine Trace-Tabelle für n = 5.

Informatik 11

20.03.2023

range-Anweisung

Die **range**-Anweisung definiert einen Bereich ganzer Zahlen.

range(10)	definiert den Bereich 0, 1, . . . , 9
range(4,21)	definiert den Bereich 4, 5, . . . , 20
range(4,21,3)	definiert den Bereich 4, 7, 10, . . . , 16, 19
range(-4,3)	definiert den Bereich -4, -3, -2, -1, 0, 1, 2

Allgemein gilt:

range(start, stop)
definiert den Bereich **start**, ... , **stop-1** ganzer Zahlen,

range(start, stop, step)
definiert den Bereich **start**, . . . , **stop-1** mit der Schrittweite **step**.

Erstellen einer Liste ganzer Zahlen

a = list(range(4,13)) erzeugt die Liste
[4, 5, 6, 7, 8, 9, 10, 11, 12];
die (in diesem Fall 9) Elemente dieser Liste nennen wir auch Komponenten, auf die man mit **a[0], a[1], . . . , a[8]** zugreifen kann.

*Bemerkung: Unter einem **Feld** oder einem **array** verstehen wir eine Folge von Variablen gleichen Typs; mit vorstehendem Beispiel haben wir also ein array **a** ganzer Zahlen erzeugt mit den Komponenten a[0], a[1], . . . , a[8].*

Beispiele (ausgeführt in der Python-shell):

```
>>> list(range(1,9))
[1, 2, 3, 4, 5, 6, 7, 8]
>>> a = list(range(-5,25,3))
>>> print(a)
[-5, -2, 1, 4, 7, 10, 13, 16, 19, 22]
>>> print(a[0])
-5
>>> print(a[3])
4
>>> print(a[9])
22
```

For-Schleife

Das Python-Programm

```
n = int(input('n = '))
for i in range(1,n):
    print(i)
    print(i*i)
```

liefert nach Eingabe der natürlichen Zahl n die Zahlen 1, 2, . . . , n-1 und deren Quadrate; probiert es aus!

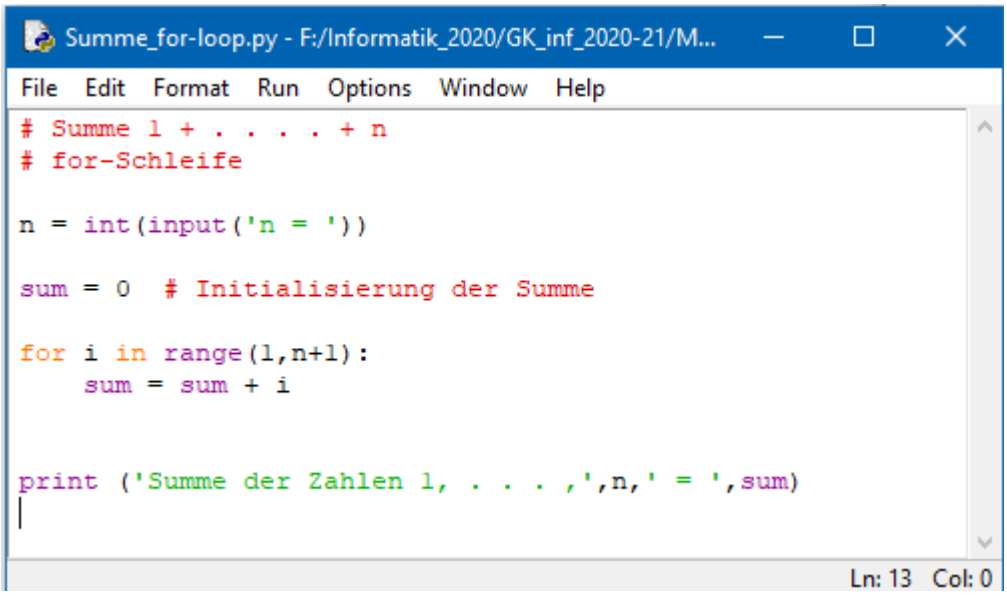
Syntax einer **for**-Schleife in Python:

```
for i in range(start, stop):
          Anweisung1
          Anweisung2
          Anweisung3
```

```
for i in range(start, stop):
                    A
```

Arbeitsaufträge:

1. Schreibe und teste ein Python-Programm, um die Siebener-Reihe auszugeben (also die Zahlen 7, 14, 21, . . .).
2. Erstelle (siehe screenshot) den Python-Programmtext zur Berechnung der Summe `1 + . . . + n` und teste das Programm mit unterschiedlichen Eingaben.



Aufgabe 11

Formuliere den Algorithmus **FAKULTÄT**, der nach Eingabe einer natürlichen Zahl **n**, $n \geq 0$, den Wert **n!** liefert, unter Verwendung einer **for**-Schleife als
a) Struktogramm,
b) Python-Programm!
c) Erstelle eine Trace-Tabelle für n = 4.

Aufgabe 12

Formuliere und teste ein Python-Programm, welches nach Eingabe der natürlichen Zahl k, $k \geq 1$, die Summe der ersten k ungeraden natürlichen Zahlen bestimmt, und zwar unter Verwendung einer for-Schleife.