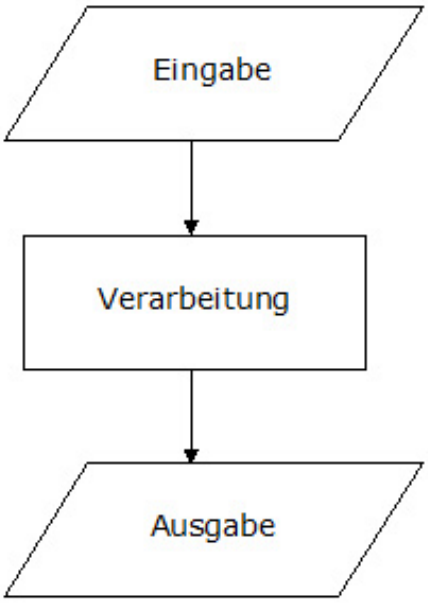
Informatik

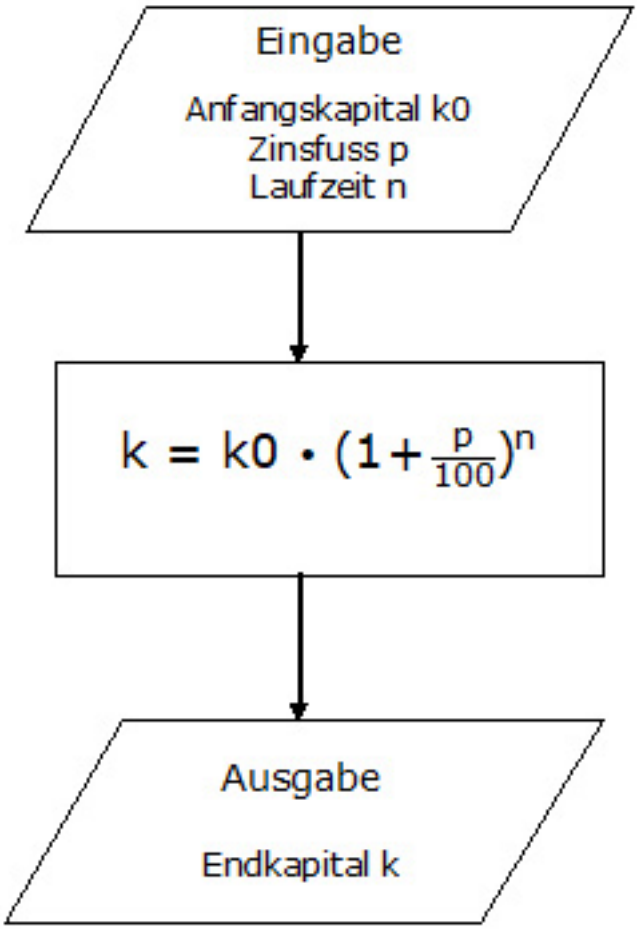
inf11 08.10.2020

Definition:
Unter einem **Algorithmus** verstehen wir ein aus endlich vielen Anweisungen bestehendes allgemeines Verfahren, welches eine Klasse von Problemen in endlich vielen Schritten löst.

Allgemeines Flußdiagramm eines Algorithmus:



Flußdiagramm des Algorithmus „Zinseszins“:



1. Lineare Algorithmen

Wenn bei der Abarbeitung eines Algorithmus die einzelnen Anweisungen sich längs eines einzigen Pfades aneinanderreihen, sprechen wir von einem linearen Algorithmus; insbesondere gibt es hier keine Verzweigungen. Beispiele: Quaderberechnung, Zinseszinsberechnung.

Aufgabe 1
Der Algorithmus „Quaderberechnung" ermittelt Volumen V und Oberfläche O eines Quaders, nachdem man die Länge a, Breite b und Höhe c eingegeben hat.

Aufgabe 2
Wird ein Anfangskapital **k0** zu einem Zinssatz von **p** % über eine Laufzeit von **n** Jahren mit Zinseszins angelegt, verfügt man am Ende der Laufzeit über das Kapital **k**. Der Algorithmus „Zinseszins" ermittelt das Endkapital **k** nach Eingabe des Anfangskapitals **k0**, des Zinsfußes **p** und der Laufzeit **n**. (Bemerkung: In entsprechender Weise lässt sich die Entwicklung des Preisindex nach n Jahren bei einer Inflationsrate von p % ermitteln.)

Lösung:
Den Ablauf entnehmen wir obenstehendem Flussdiagramm, welches von der gewählten Programmiersprache unabhängig ist. Wir codieren den Algorithmus in der Programmiersprache „Python".

zinseszins.py - F:\Informatik_2020\GK_inf_2020-21\MSS11\zinse...

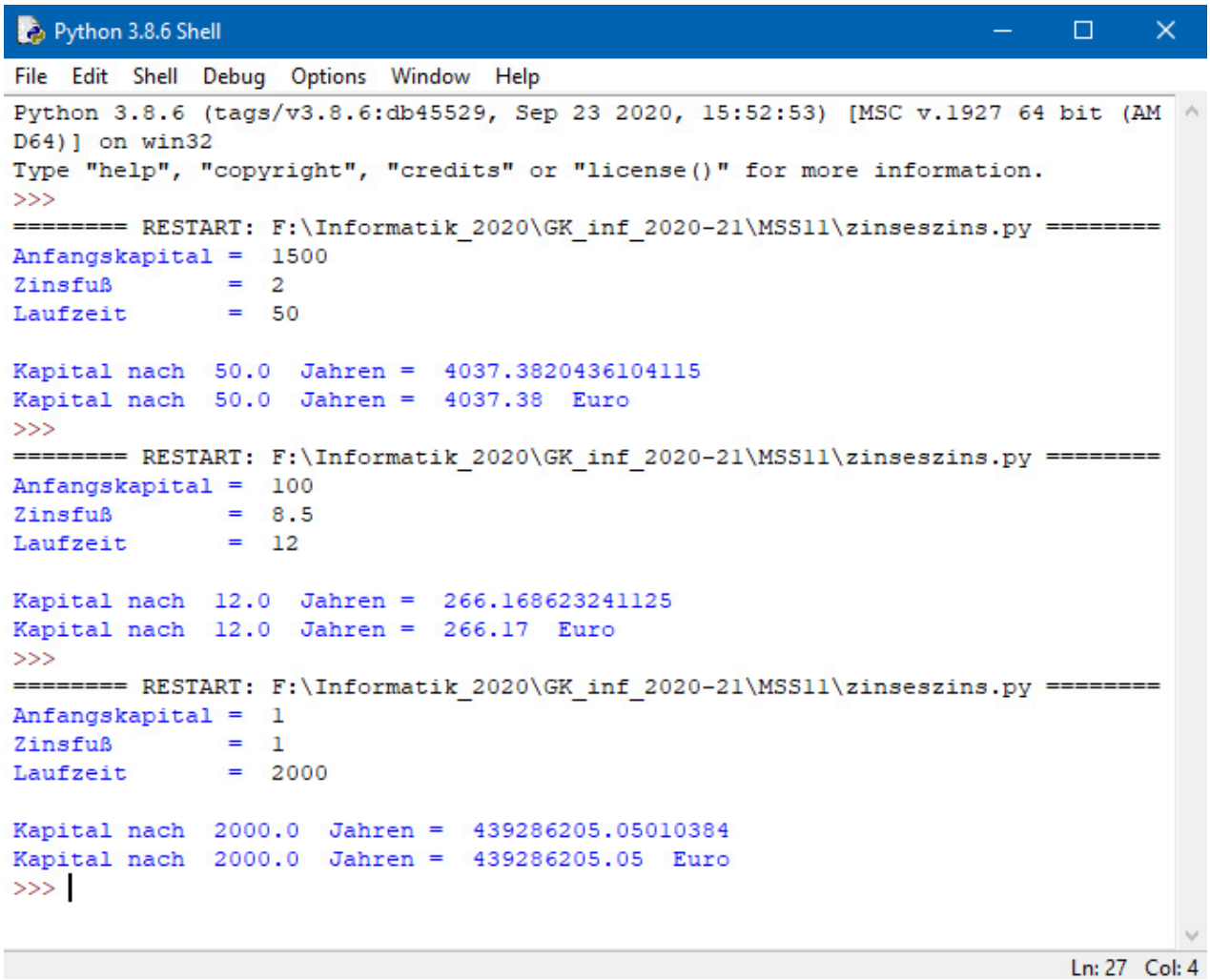
File Edit Format Run Options Window Help

```
# Zinseszins

# Mit '#' eingeleitete Zeilen bilden einen Kommentar,
# der auf den Programmablauf keinen Einfluß hat.

# Eingabe

k0=float(input('Anfangskapital =  '))
p=float(input('Zinsfuß         =  '))
n=float(input('Laufzeit        =  '))

# Verarbeitung

k=k0*(1+p/100)**n


# Ausgabe

# Die folgende Anweisung erzeugt eine Leerzeile,
# ist aber auch verzichtbar.
print ()
# Ausgabe des Ergebnisses auf viele Nachkommastellen
print ('Kapital nach ',n,' Jahren = ',k)
# Runden des Ergebnisses auf 2 Nachkommastellen
print ('Kapital nach ',n,' Jahren = ',round(k, 2),' Euro')
```

Ln: 26 Col: 0

Wenn wir nach Eingabe des Programmtextes im „IDLE"-Editor den Button „Run" anklicken, öffnet sich ein Kontextmenue, und wir starten das Programm durch Klick auf „Run Module".

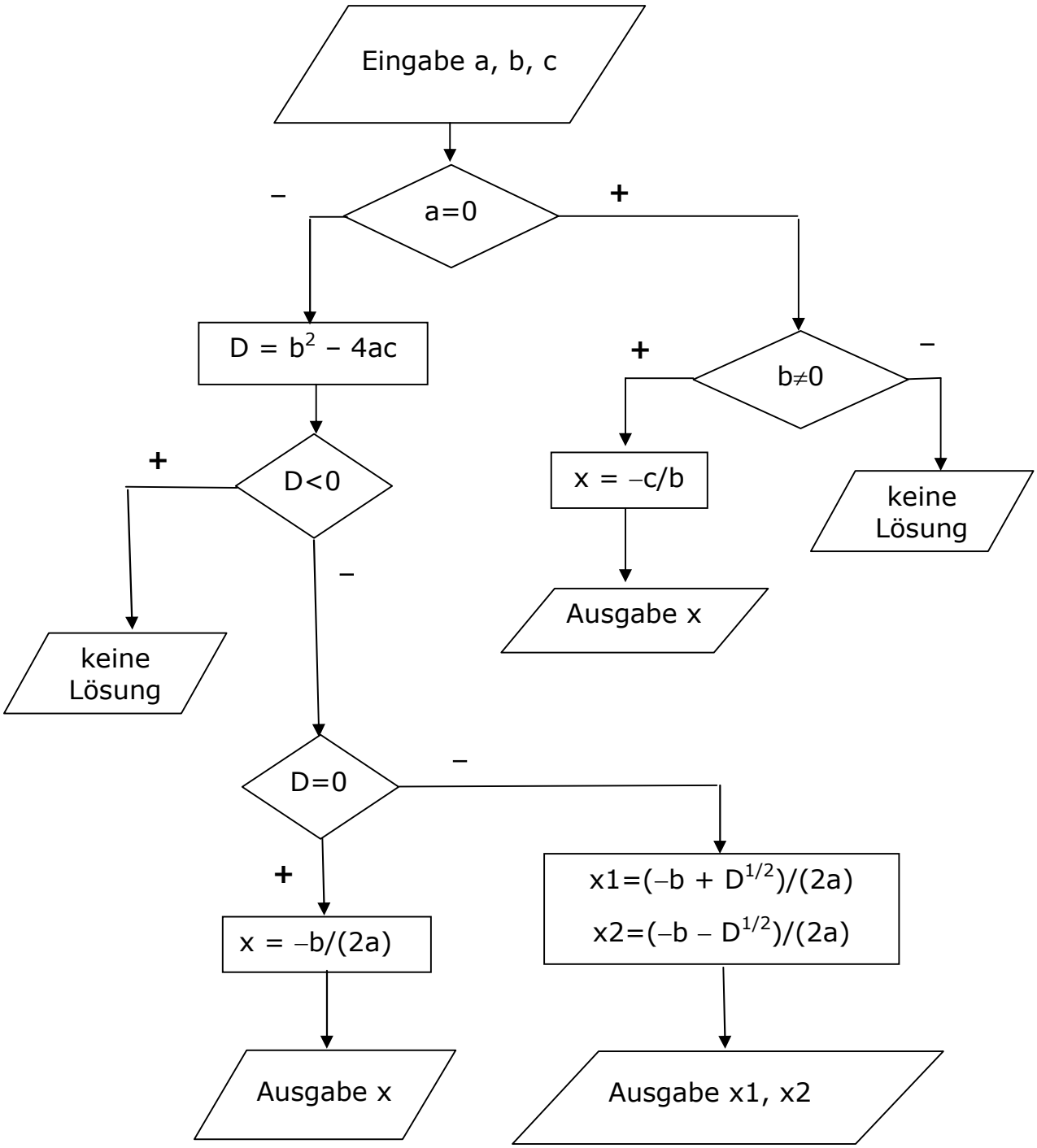
Nachdem man bestätigt hat, den eventuell geänderten Programmtext zu speichern, öffnet sich die „Python Shell", in der man die Eingaben macht und in der dann die Ausgabe des Ergebnisses oder der Ergebnisse erfolgt, falls man den Run-Befehl wiederholt erteilt.



Algorithmus **QUADRATISCHE GLEICHUNGEN**

Spezifikation:
Nach Eingabe der Koeffizienten a, b, c der allgemeinen quadratischen Gleichung **$ax^2 + bx + c = 0$** ermittelt der Algorithmus die Lösungsmenge und gibt diese aus.

Flußdiagramm:



05.10.2020

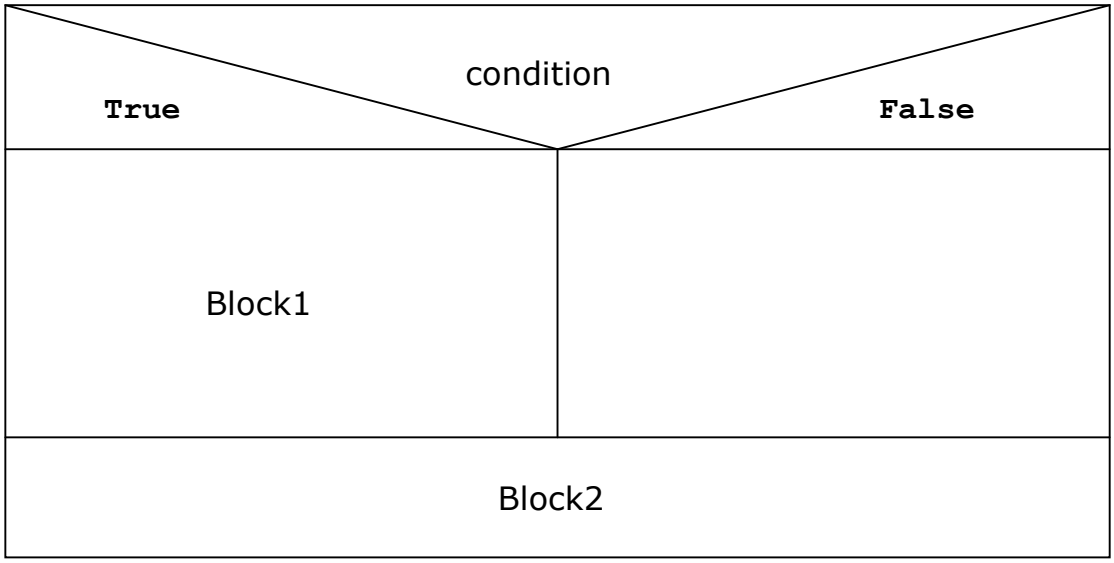
Verzweigte Algorithmen

Definition: Ein **Anweisungsblock** besteht aus einer Folge zusammengehörender Anweisungen, die nacheinander ausgeführt werden.
Ein Anweisungsblock, der innerhalb einer Schleife wiederholt wird, heißt **Schleifenrumpf**.
Den zu einer Funktion gehörenden Anweisungsblock nennen wir auch **Funktionsrumpf**.

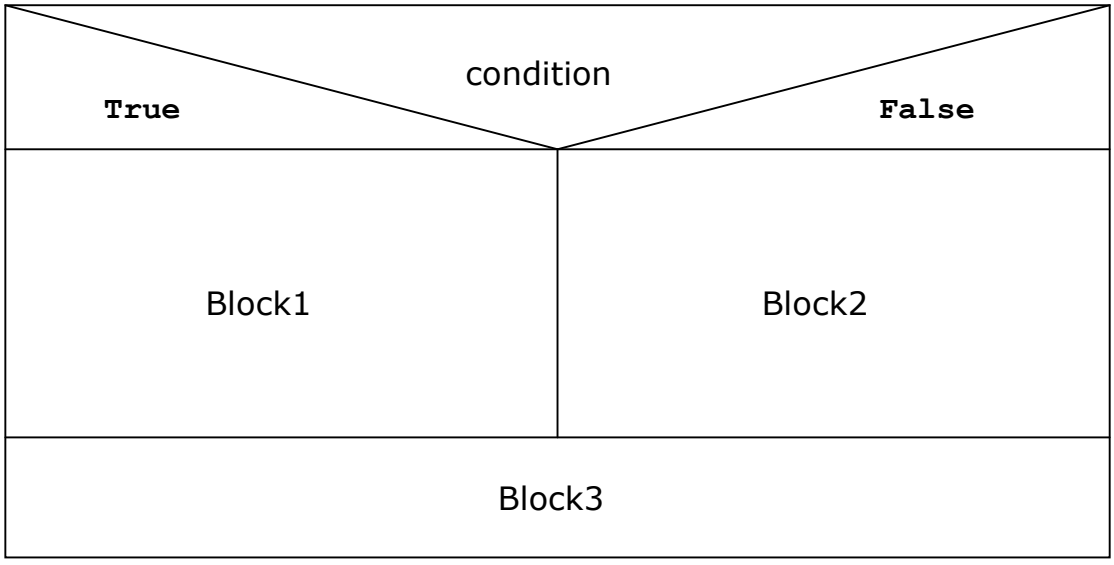
Bemerkungen:
- Anweisungsblöcke können auch ineinander verschachtelt sein.
- In Python wird ein Anweisungsblock durch Einrücken des Programmtextes gekennzeichnet.

Im folgenden verstehen wir unter ***condition*** einen Booleschen Term (der auch nur aus einer Booleschen Variablen bestehen kann), der die Werte **`True`** oder **`False`** annimmt. In Struktogrammen kennzeichnen wir **`True`** auch durch , **+** ', **`False`** durch , – '.

Einseitige Auswahl



Zweiseitige Auswahl

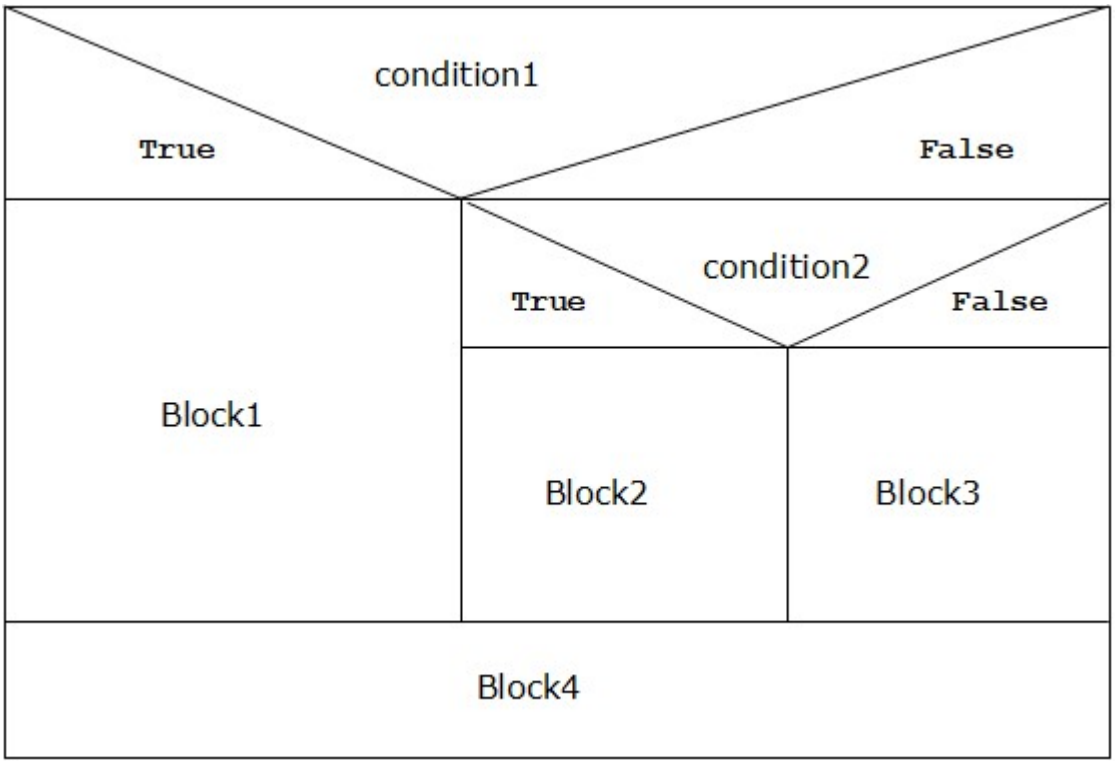


Formulierung in Python:

```
if condition:
    Block1
Block2
```

```
if condition:
    Block1
else:
    Block2
Block3
```

Mehrstufige Auswahl



Formulierung in Python:

```
if condition1:
    Block1
else:
   if condition2:
       Block2
   else:
       Block3
Block4
```

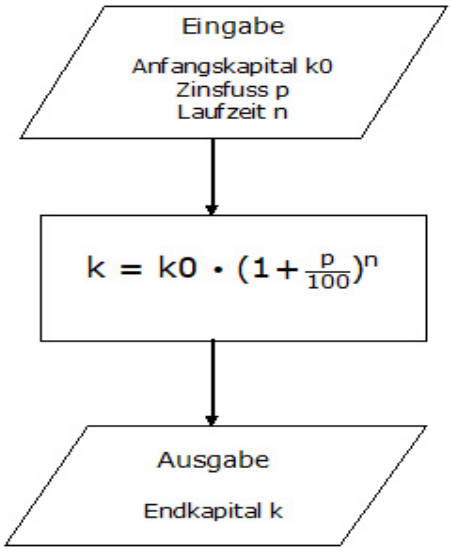
```
if condition1:
    Block1
elif condition2:
    Block2
else:
    Block3
Block4
```

Aufgabenblatt Nr. 1 inf11 05.11.2020

Information:
Unter einem **linearen Algorithmus** verstehen wir einen Algorithmus, bei dem die nacheinander auszuführenden Anweisungen sich längs eines einzigen Pfades aneinanderreihen; insbesondere enthält ein linearer Algorithmus keine Programmverzweigungen.

Beispiele:

a) **Zinseszins** Flußdiagramm:



Wenn ein Anfangskapital **k0** zu einem jährlichen Zinssatz **p** % über einen Zeitraum von **n** Jahren mit Zinseszins angelegt wird (der Zinsbetrag wird also am Ende jeden Jahres dem zu verzinsenden Kapital zugeschlagen), ermittelt der Algorithmus „Zinseszins" das Endkapital **k** nach **n** Jahren.

b) **Quaderberechnung** c) **Promillerechner** d) **Anhalteweg**

Verzweigte Algorithmen

Definition: Ein Algorithmus, bei dessen Abarbeitung unterschiedliche Anweisungsblöcke durchlaufen werden können, heißt **verzweigter Algorithmus**; dabei entscheidet die Abfrage einer Bedingung (in Form einer booleschen Variablen oder eines booleschen Ausdrucks) darüber, welcher Zweig durchlaufen wird.

1. **Mobilfunkrechung** (Verzweigter Algorithmus)

   Der Betreiber eines Mobilfunknetzes hat folgende Tarifgestaltung:

   Monatliche Grundgebühr (einschließlich 100 Gesprächsminuten): 15 €;
   für die nächsten, über 100 Minuten hinausgehenden Minuten sind 5 ct je Minute zu entrichten.

   a) Formuliere einen Algorithmus als Flußdiagramm, Struktogramm, Pythonprogramm, um nach Eingabe der Anzahl **x** der monatlichen Gesprächsminuten den Rechnungsbetrag **b** zu bestimmen.

   b) Bei einem anderen Tarif beträgt die Grundgebühr 25 € (einschließlich 100 Minuten), für die nächsten 200 Minuten werden 4 ct/min berechnet; darüberhinausgehende Minuten kosten 2 ct/min.
   Formuliere den Algorithmus als Struktogramm und Python-Programm.

2. Der Algorithmus **QuadEquation** bestimmt die Lösungsmenge einer quadratischen Gleichung (gemäß Flußdiagramm QuadEquation.pdf).
   Formuliere den Algorithmus QuadEquation als
   a) Struktogramm,
   b) Python-Programm.

3. Mit dem Body-Mass-Index (**BMI**) kann man abschätzen, ob jemand Normalgewicht hat. Der BMI ist eine dimensionslose Zahl (also ohne Maßeinheit) und berechnet sich wie folgt:
   BMI = gewicht / (groesse * groesse)
   mit
   gewicht = Maßzahl der Masse in kg
   groesse = Maßzahl der Körpergröße in m

   Beispiel:
   Mit Masse = 70 kg und Körpergröße = 1,80 m erhält man
   BMI = 70 / (1,80 * 1,80) = 70 / 3,24 ≈ 21,6

   Für BMI < 19 gilt man als untergewichtig, für BMI > 26 als übergewichtig; Normalgewicht verbindet man mit 19 ≤ BMI ≤ 26.

   Der Algorithmus BodyMassIndex soll folgendes leisten:
   Nach Eingabe des Gewichts (in kg) und der Größe (in m) wird BMI berechnet und ausgegeben, darüberhinaus erfolgt die Information, ob man als normal-, unter- oder übergewichtig gilt.

   Konzipiere ein

   a) Struktogramm,

   b) Python-Programm!

Algorithmen mit Wiederholungen

4. **Quadratzahltabelle**
   Der Algorithmus **quad** gibt nach Eingabe einer natürlichen Zahl **n** die Quadrate der Zahlen 1, 2, ... , n aus.

   Wir formulieren den Algorithmus als Struktogramm und Python-Programm.

5. Ein Algorithmus ist gesucht, der nach Eingabe einer natürlichen Zahl **n** die Summe **1 + 2 + ... + n** bestimmt.

6. Mit 4! (lies: 4-Fakultät) bezeichnen wir das Produkt 1 · 2 · 3 · 4 = 24; allgemein gilt für n ∈ {0, 1, 2, 3, 4, 5, ... }:

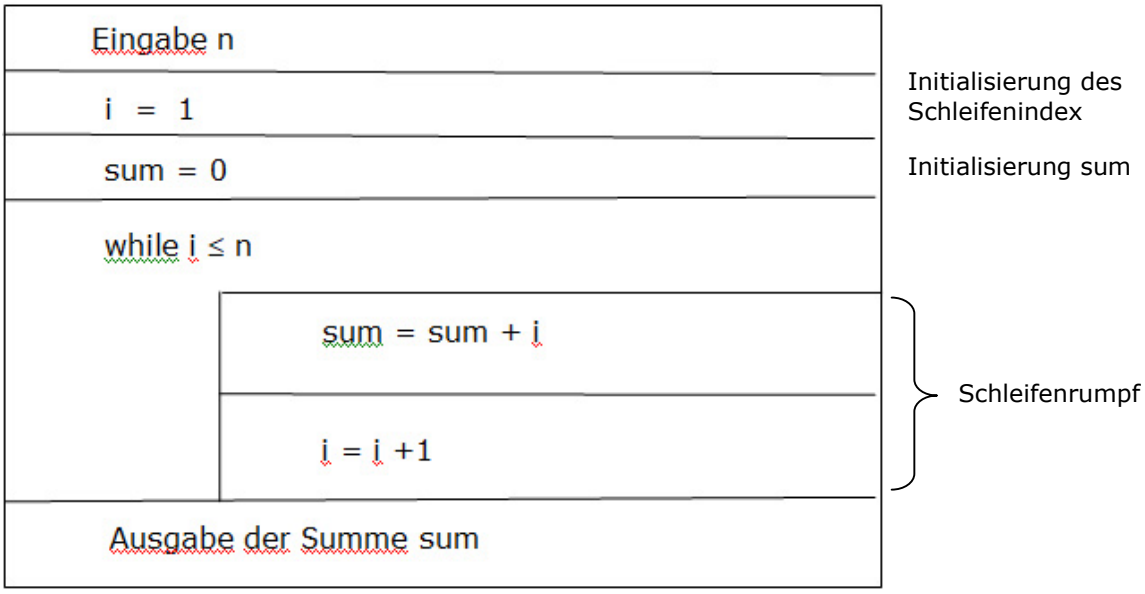
   0! = 1 (dies ist eine Festlegung!)

   n! = 1 · 2 · . . . . · n

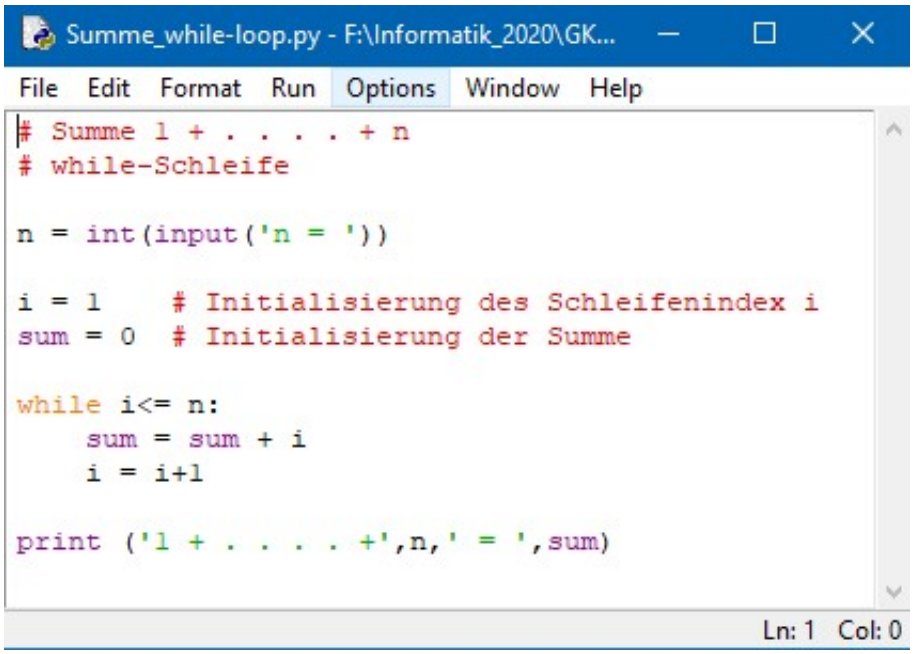
   Entwirf einen Algorithmus (Struktogramm und Python-Programm), der nach Eingabe einer nicht negativen ganzen Zahl **n** den Wert **n!** ermittelt.

Lösung zu Aufgabe 5 Blatt 2 vom 12.11.2020
Summe der ersten n natürlichen Zahlen

Struktogramm:



Python-Quelltext:




```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.6 (tags/v3.8.6:db45529, Sep 23 2020, 15:52:53) [MSC v.1927 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\Informatik_2020\GK_inf_2020-21\MSS11\Summe_while-loop.py =====
n = 100
1 + . . . . + 100  =  5050
>>>
===== RESTART: F:\Informatik_2020\GK_inf_2020-21\MSS11\Summe_while-loop.py =====
n = 999999
1 + . . . . + 999999  =  499999500000
>>>
```

20.11.2020