Informatik für Schüler, Foliensatz 5
Wiederholung und Abweisschleife

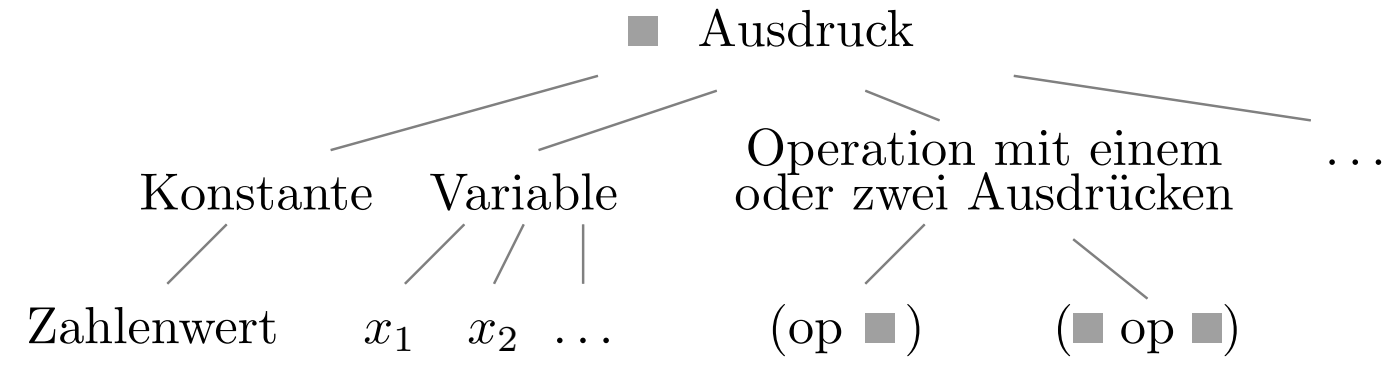
Prof. G. Kemnitz

Institut für Informatik, Technische Universität Clausthal

18. November 2009

Ausdruck

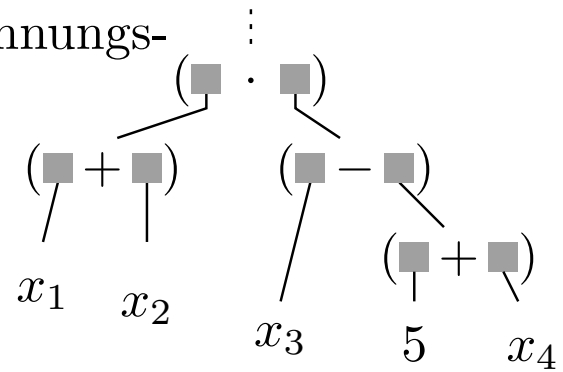
- ein Ausdruck kann sein:
  - eine Konstante, z.B. 0.1, 8, 'hallo ' ...
  - eine Variable
  - eine Operation mit einem oder zwei Ausdrücken z.B. (3 * a)



Beispielausdruck

$(x_1 + x_2) \cdot (x_3 - (5 + x_4))$

Berechnungs-baum



Variablen und Zuweisung

- eine Variable
  - ist ein Speicherplatz für ein Ergebnis oder ein Zwischenergebnis
  - hat einen Namen, einen Wert und einen Typ
- Zuweisung eines Wertes (in Python auch des Typs):

  Variable = Ausdruck

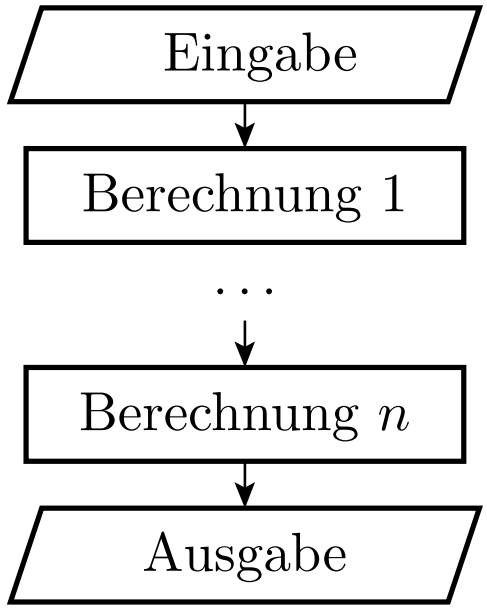
```
a = 3 + 5;
```

- Abfrage des Wertes (Ausdruck ohne Zuweisungsziel eingeben)
- Abfrage des Typs:

```
type(a)
```

- Mögliche Typen: 'int', 'float', 'bool', ...

Einfaches Progamm



Eingabe:

```
Variable = input('Ausgabetext')
```

Ausgabe:

```
print [Ausdruck {, Ausdruck}]
```

Programmbeispiel:

```
# Kommentar ...
a = input('a = ')
b = input('b = ')
c=a*b
print 'a*b = ', c
```

```
⇒ a = 5
⇒ b = 7
⇒ a*b = 35
```

(... Eingabe + <Enter>)

Funktionen aus externen Modulen

- Die meisten fertigen Funktionen stehen in Zusatzmodulen; viele mathematische Zusatzfunktionen z.B. im Zusatzmodul »math«
- Import des Moduls »math« und Nutzung der Funktion cos():

```
import math
y = math.cos(math.pi/2)
```

  - Objekten aus Modulen Modulnamen vorangestellen
- Abfrage aller im Modul »math« vereinbarten Funktionen und Variablen

```
help(math)
```

(alles Weitere siehe Übung 3)

Wiederholschleife

```
for sv in sequenz:
    Anweisungsfolge
```

Wiederhole für x in *sequenz*
$y = \cos(0.1 \cdot x)$
print 'x =', x, 'cos(0.1*x) =', y

Sequenzobjekte in Python:

- Tupel:

  ```
  (True, False)
  ```

  Beispiel:

  ```
  for x in (True, False):
      print 'x= ', x
  ```

- Liste (wie Tupel, nur eckige Klammern):

  ```
  [0, 1, 2, 'Text', 4]
  ```

- String (Wiederhole für alle Zeichen einer Zeichenkette)

- Funktion zur Erzeugung einer Sequenz, z.B.:

  `range([a,] e [,s])`

  Erzeugt eine Liste der ganzen Zahlen kleiner »e« beginnend mit »a« , Schrittweite »s«; ohne »a« und »s« eine Liste der natürlichen Zahlen kleiner »e«;

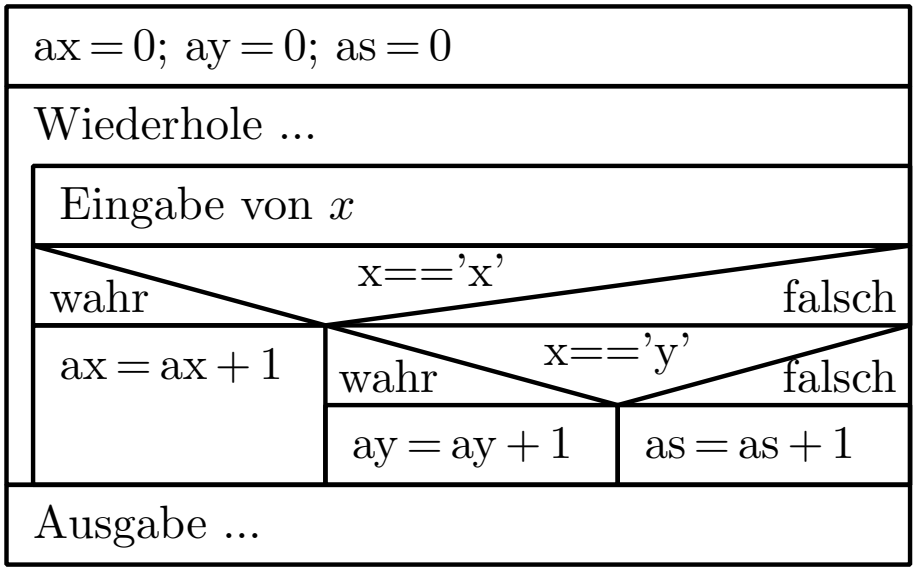
  `range(5)` ⇒ `[0, 1, 2, 3, 4]`

  `range(1, 5, 2)` ⇒ `[1, 3]`

```
import math
for idx in range(9):
  # Schleifenkörperanfang => Einrückung beginnen
  x = idx*math.pi/16
  y = math.cos(x)
  print 'x =', x, 'cos(x) = ', y
  # Schleifenkörperende   => Einrückung beendet
print 'Schleife abgearbeitet'
```

Fallunterscheidung

```
ax = 0; ay = 0;
as = 0;
for idx in range(10):
  x = input('x =');
  if x=='x':
    ax = ax + 1
  elif x=='y':
    ay = ay +1
  else:
    as = as + 1
print 'ax =', ax, 'ay =', ay, 'as =', as
```



Bildung logischer Ausdrück

- Vergleichswerte:

a<b	a<=b	a>b	a>=b	==	!=
kleiner	kleiner gleich	größer	größer gleich	gleich	ungleich

- logische Verknüpfungen:

a	b	a and b	a or b	not a
False	False	False	False	True
False	True	False	True	False
True	False	False	True	
True	True	True	True	

- Werte der Variablen und Ausdrücke?

```
(3 < 5) or ('x' > 'y')
a = (5==7)
b = (8 >= 3)
(a and b) or (not (a and b))
```

Neu: Abbruchschleife

```
while  Bedingung:
  Anweisungsfolge
[else:
 Anweisungsfolge]
```

- Beispiel: »Wiederhole, bis Eingabe korrekt«

```
z=0
while not(type(z)==type(1) and z>0 and z<10):
    z=input('Eingabe einer Zahl zwischen 0 und 10: ')
print 'akzeptiert wurde nur z = ', z
```

```
⇒ Eingabe einer Zahl zwischen 0 und 10: -1
⇒ Eingabe einer Zahl zwischen 0 und 10: True
⇒ Eingabe einer Zahl zwischen 0 und 10: 5
⇒ akzeptiert wurde nur z = 5
```

(... Eingabe)

Aufgaben 5.1: Welche Zahlenfolge berechnet der Algorithmus?

$x_1 = 1;\ \ x_2 = 2$
Wiederhole für i im Bereich 1 bis 10
$x_3 = x_1 + x_2$
$x_1 = x_2;\ \ x_2 = x_3$
print 'Zahl', i, 'hat den Wert', x_3

- Bestimmen Sie, bevor Sie programmieren, die erwarteten Ausgabewerte für $i = 1$ bis 5.
- Schreiben Sie ein Programm mit dem gegeben Algorithmus und vergleichen Sie die Programmausgabe mit ihren vorab berechneten Werten.

Aufgabe 5.2: Zeichkettenverarbeitung

Schreiben Sie ein Programm, das solange zur Eingabe von Zeichenketten auffordert, bis die eingegebe Zeichenfolge insgesamt

- mindestens viermal das Zeichen »a« enhält
- mindesten vier Zeichen »a« hintereinander enthält.

Die Eingaben sollen, sofern es sich um Zeichenketten handelt, aneienandergekettet und zum Schluss ausgegeben werden.

Aufgabe 5.3: Ausdruck einer Wertetabelle

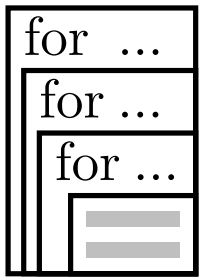
Schreiben Sie ein Programm, das für die logische Funktion

$$y = x_1x_2 \vee x_1x_3 \vee x_2x_3$$

die Wertetabelle wie folgt auf dem Bildschirm ausgibt:

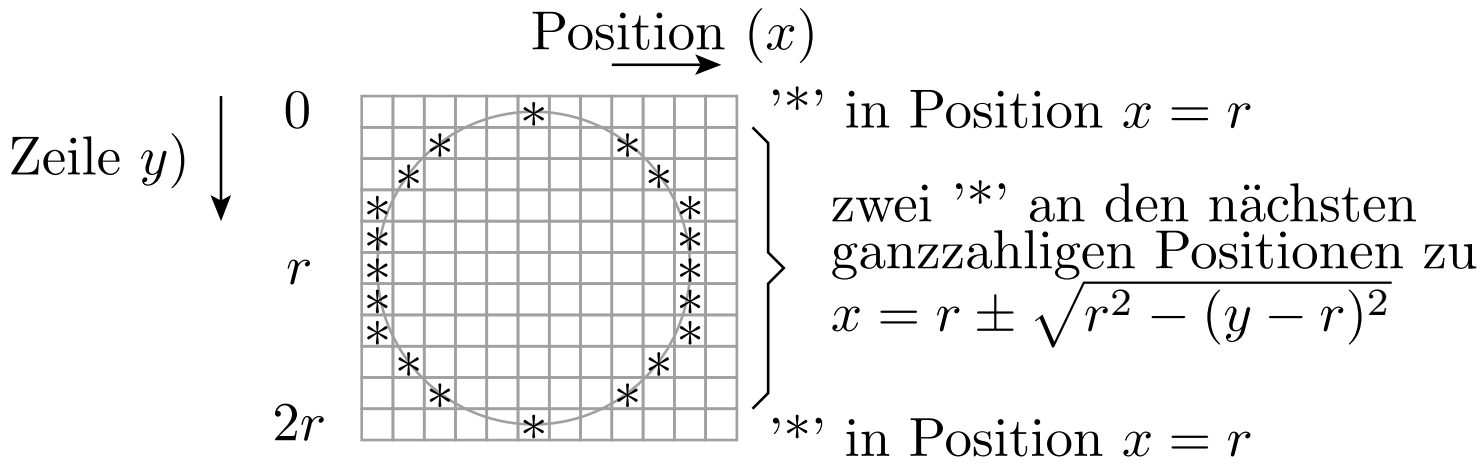
```
   x3     x2      x1        y
-----------------------------
 False  False  False    False
 False  False  True       ...
 False  True   False      ...
        ...
```

Hinweis: Benutzen Sie die Variablen x1 bis x2 als Schleifenzähler von drei ineinander verschachtetelten Schleifen, die über die Sequenz (True, False) iterieren. Verschachtelt heißt, dass eine Schleife den Schleifenkörper der andern bildet.



Aufgabe 5.4: Bildschirmkreis zeichnen

Entwickeln Sie ein Programm, dass eine ganze Zahl im Bereich von 4 bis 40 anfordert und gemäß der folgenden Skizze einen Kreis auf den Bildschirm zeichnet.



Benötigte Funktionen:

- math.sqrt(...): Funktion zur Berechnung der Wurzel (setzt »import math« voraus)
- round(...): Funktion zum Runden auf die nächste ganze Zahl
- int(...): Funktion zur Umwandlung in eine ganze Zahl

Wiederholung zur Arbeit mit Dateien und Verzeichnissen unter Linux

- Hilfe:

  `man [Befehl]`
- Python-Interpreter beenden: »Str-D«
- Verzeichnisinhalt anzeigen (list):

  `ls [-l][-a] [Pfad][Datei]`
- neues Verzeichnis erzeugen (make directory):

  `mkdir Verzeichnisname`
- Verzeichnis ändern (change directory):

  `cd Pfad`
- Start des Python-Programms:

  `python BBaum.py`