Vorlesung zur Programmierung - Einführung

Ablauf einer Programmerstellung:

1. Was will ich berechnen? – **Gesucht?**
2. Wie kann ich das berechnen? – **Formel?**
3. Was für Werte benötige ich dazu?- **Gegeben?**
4. Woher bekomme ich diese Werte? – **Eingabe?**
5. Wie soll das Ergebnis bekannt gegeben werden? **- Ausgabe?**
6. Hilfsmittel zur Ordnung der Gedanken: **Programmablaufpläne (PAP)** - **auf Papier!** **(nicht nur im Kopf oder durch sofortige Programmierung überspringen)**
7. Umsetzung des PAP in ein **Programm**
8. **Test** des Programmes anhand eines Beispiels mit bekanntem Ergebnis
9. **Test** des Programmes auf Sonderfälle, Nulldivisionen, Endlosschleifen, …
10. **Verschönern:** Beseitigung unerwünschter Ausgaben (nicht im Beleg!), Einfügen von **Kommentare**n

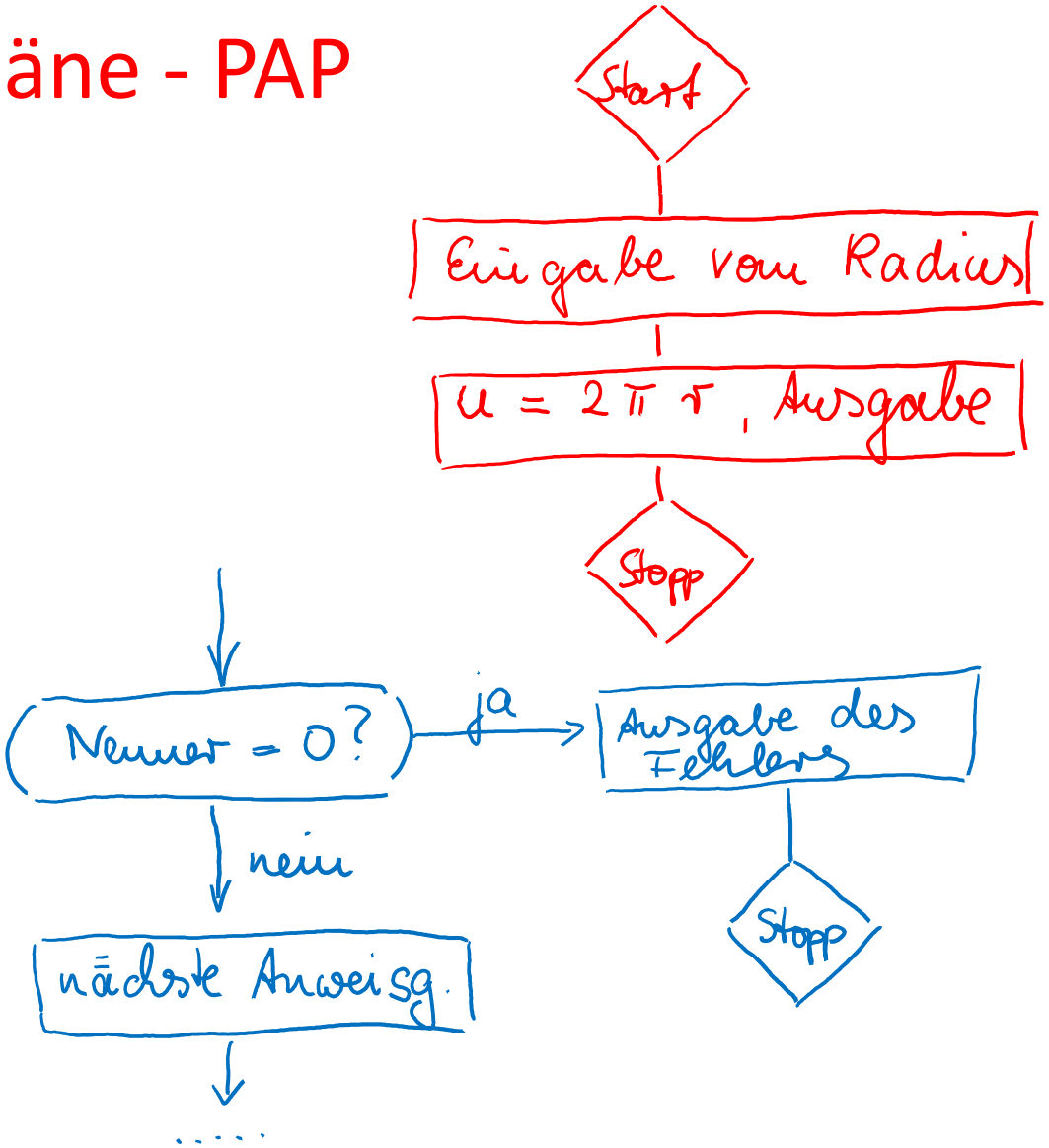
Beispiel: Programm zur Berechnung des Kreisumfanges

1. Kreisumfang
2. Umfang = 2*pi*Radius
3. pi= Konstante in Matlab
   Radius: veränderlich ➔ Variable r
4. Eingabe von r durch den Bediener
   ➔ Kommentar für den Bediener!
5. Ausgabe auf dem Bildschirm
   ➔ Kommentar für den Bediener
6. s. nächste Folie

Programmablaufpläne - PAP

- Anfang
- Ausführbare Anweisungen
- Ende

- Verzweigungen durch Fragen, auf die es unterschiedliche Antworten gibt,
  z.B. ist der Nenner = 0?
- Absicherung, dass jeder Programmzweig ein reguläres Ende hat



Beispiel Kreisumfang, Name des Skriptfiles: KU

7. Einfachste Version:

```
r = input('r: ')
u=2*pi*r
```

Aufruf: >> KU
>> r: 4
>> u=25.1327

8. Test mit bekanntem Ergebnis: bei r =1 gilt u = 2*pi

9. Keine Sonderfälle

10. „Verschönerung:“

```
% Skriptfile: Berechnung des Kreisumfangs u
r = input(' Eingabe des Radius r: ')
disp('Umfang u des Kreises mit dem Radius r: ')
u=2*pi*r
```

Aufruf: >> KU
>> Eingabe des Radius r: 4
>> Umfang u des Kreises mit dem Radius r:
>> u=25.1327

Anforderungen an neue Befehle

- ***Bisher: einfache ausführbare Anweisungen,***
  z.B. a = 3+7; b=a/5; c=a-b
- **Neu: Entscheidungen in Abhängigkeit von bestimmten Parametern** durch den Rechner,
  z.B. if-Anweisung oder case-Anweisung
- **Neu: Wiederholungen von Anweisungen** durch den Rechner,
  z.B. for-Schleife oder while-Schleife
- **Zusammenfassung von Befehlsgruppen**, die an verschiedenen Stellen der Berechnung wiederholt werden müssen,
- in Skriptfiles, die durch Nennung Ihres Namens aufgerufen werden, z.B.
- in Funktionsdateien, auch Funktionsunterprogrammen genannt, die Übergabeparameter benutzen und aufgerufen werden durch *Name(Übergabeparameter)*, z.B. sin(x) oder mit Zuweisung auf eine Ergebnisvariable: y=sin(x)

if-Anweisung

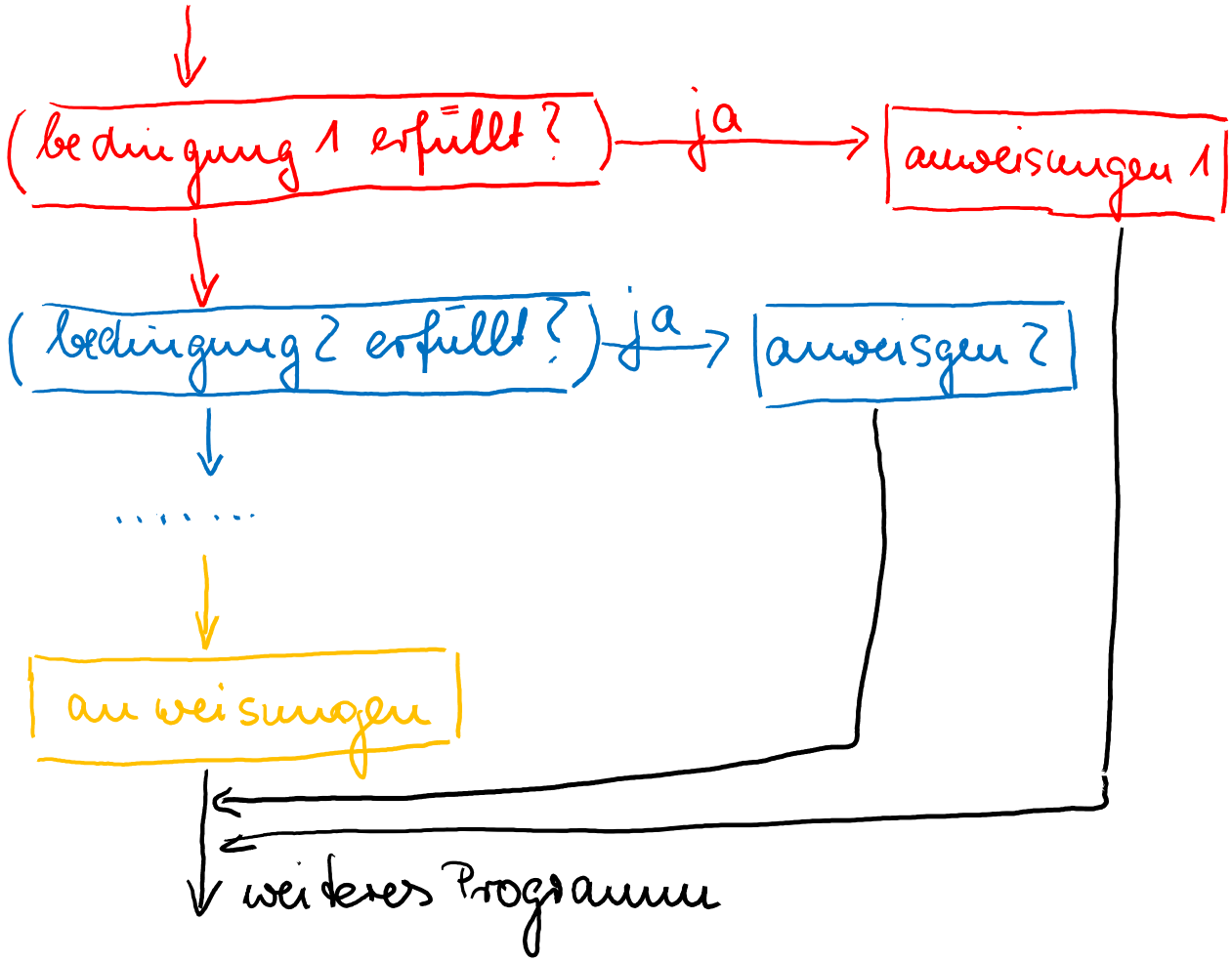
```
if  bedingung 1
        {anweisungen 1}
```

```
elseif bedingung 2
        {anweisungen 2}
```

.....

```
else
        {anweisungen}
```

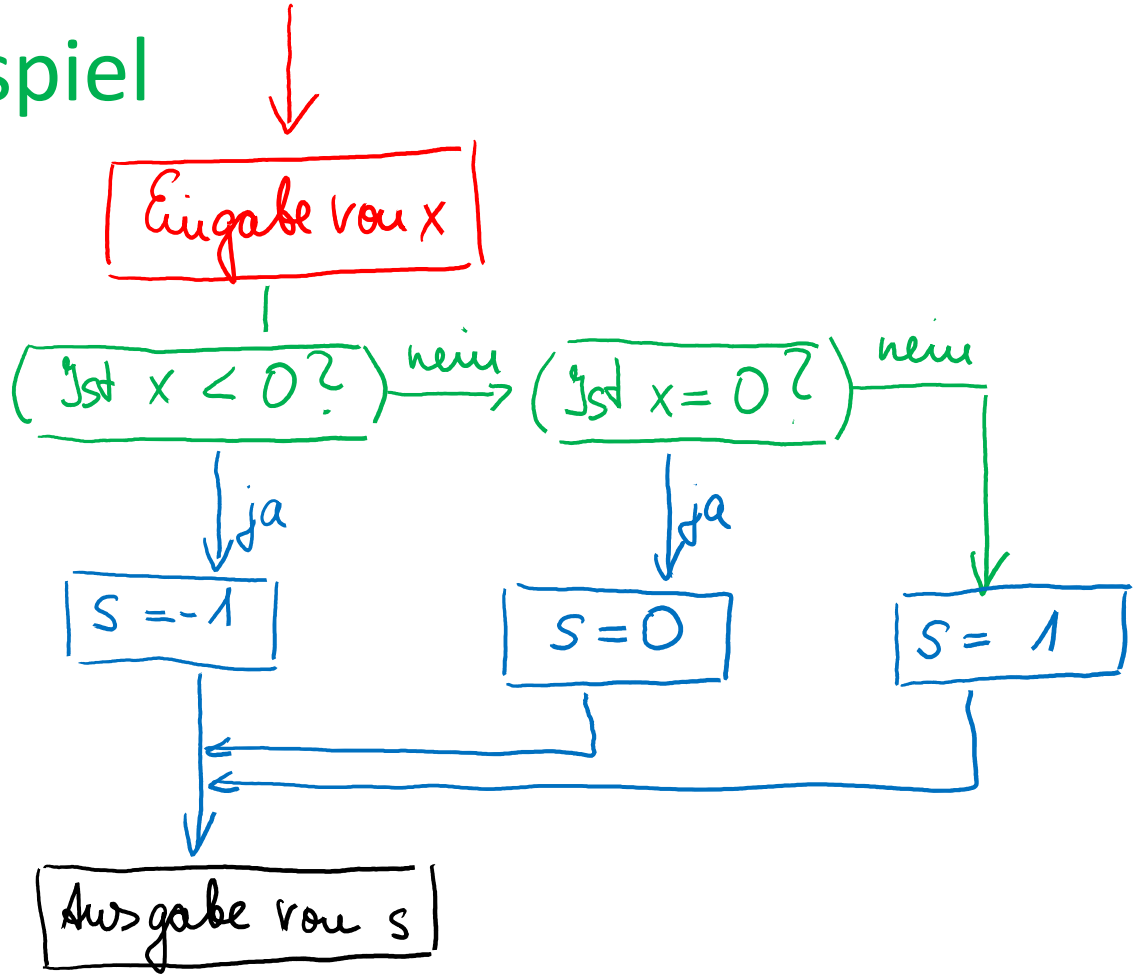
end



Vergleichsoperatoren in den „bedingungen“:
<, >, = =, <=, >=, ~ =, ||, &&, ~, …

if-Anweisung/ Beispiel

```
x=input(' x = ');
if x<0;
        s=-1;
  elseif x==0;
        s=0;
  else s=1;
end;
s
```



case-Anweisung

switch ausdruck

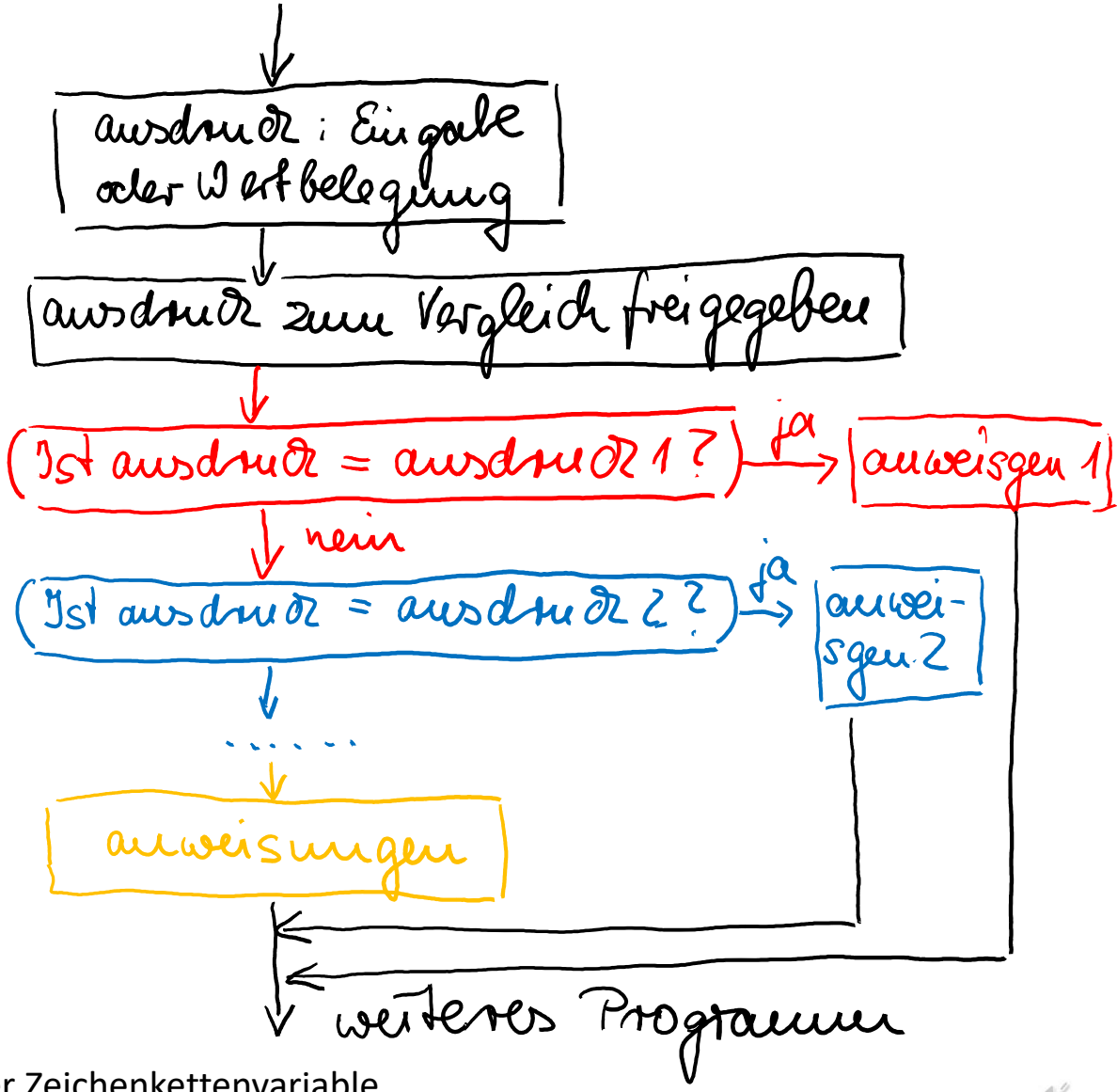
case ausdruck1
{anweisungen1}

case ausdruck2
{anweisungen2}

.....

otherwise {anweisungen}

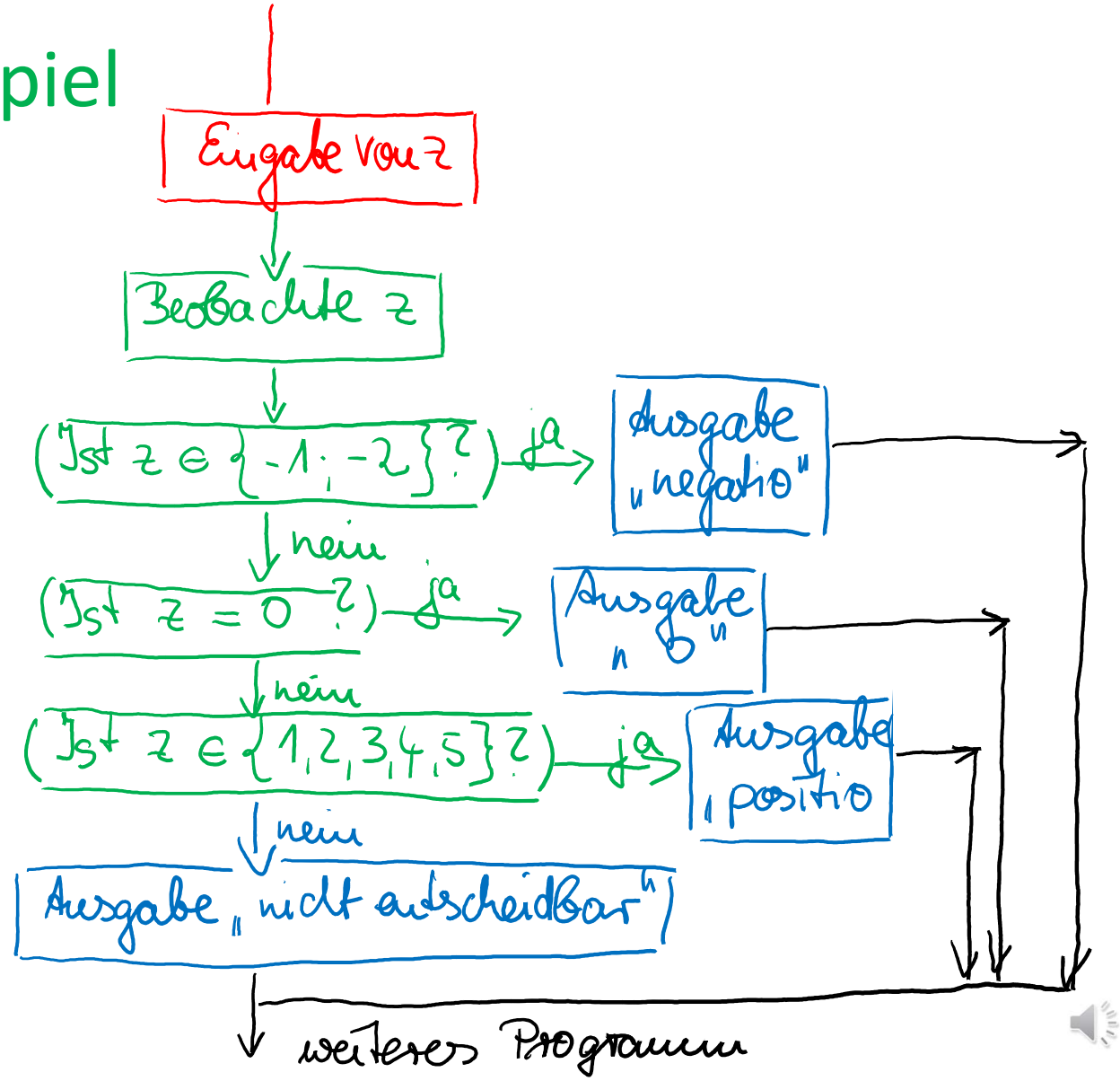
end



„ausdruck“ und „ausdruck1“, … sind Zeichenketten oder Zeichenkettenvariable

case-Anweisung/Beispiel

```
z = input('Gib eine Zahl ein! ');
switch z
case {-1,-2}
      disp('negativ')
case 0
      disp('null')
case {1,2,3,4,5}
      disp('positiv')
otherwise
      disp('nicht entscheidbar')
end
```

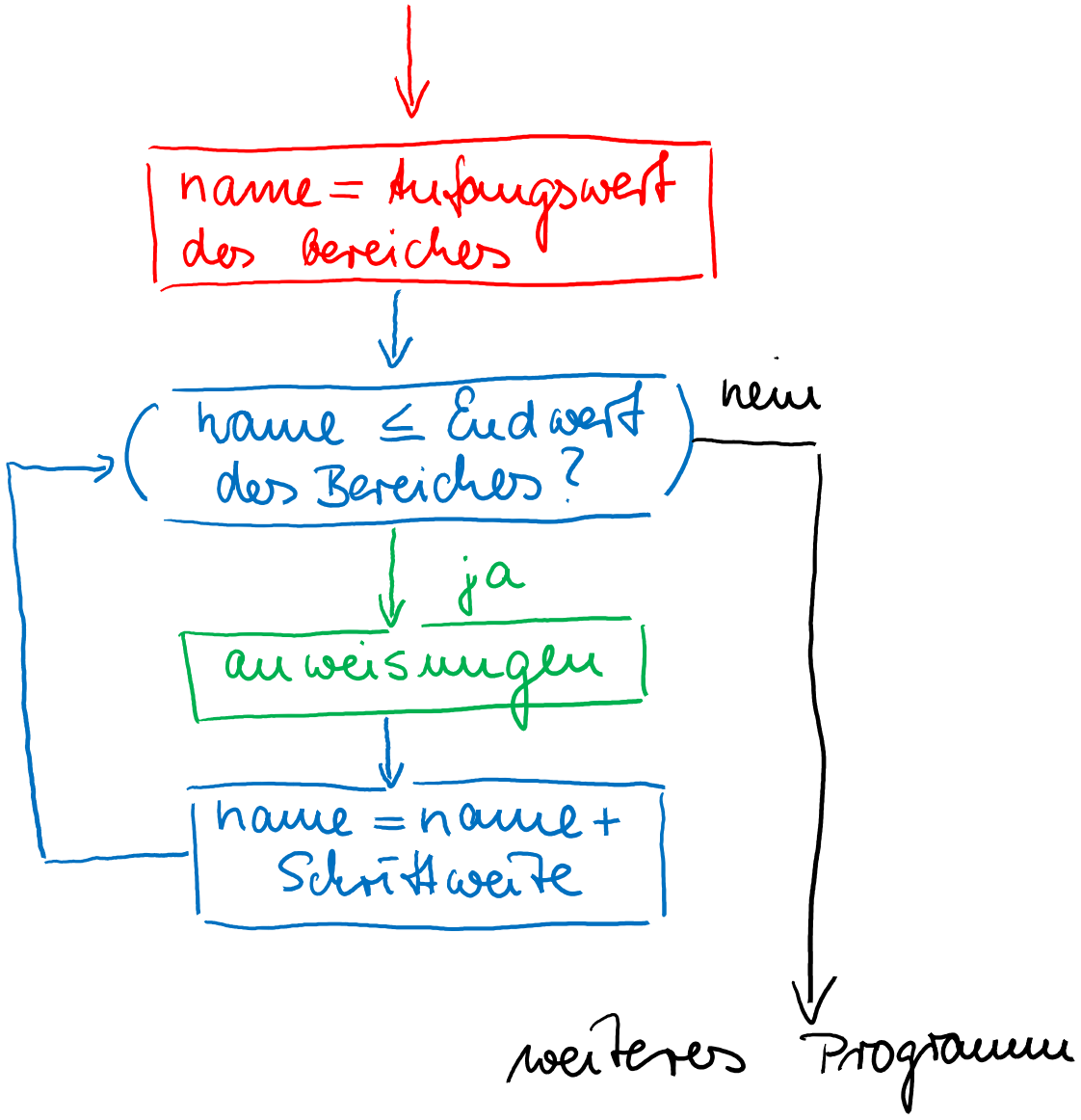


for-Schleife

```
for name=bereich
        {anweisungen}
end
```

- „name“ bezeichnet die Lauf-variable, die schrittweise den „bereich“ durchläuft
- „bereich“ gibt den Wertebereich für die Variable „name“ an
- Beispiel: Berechnung von $2^5 \cdot x$

```
for i =1:5;
      x=2*x
end
```



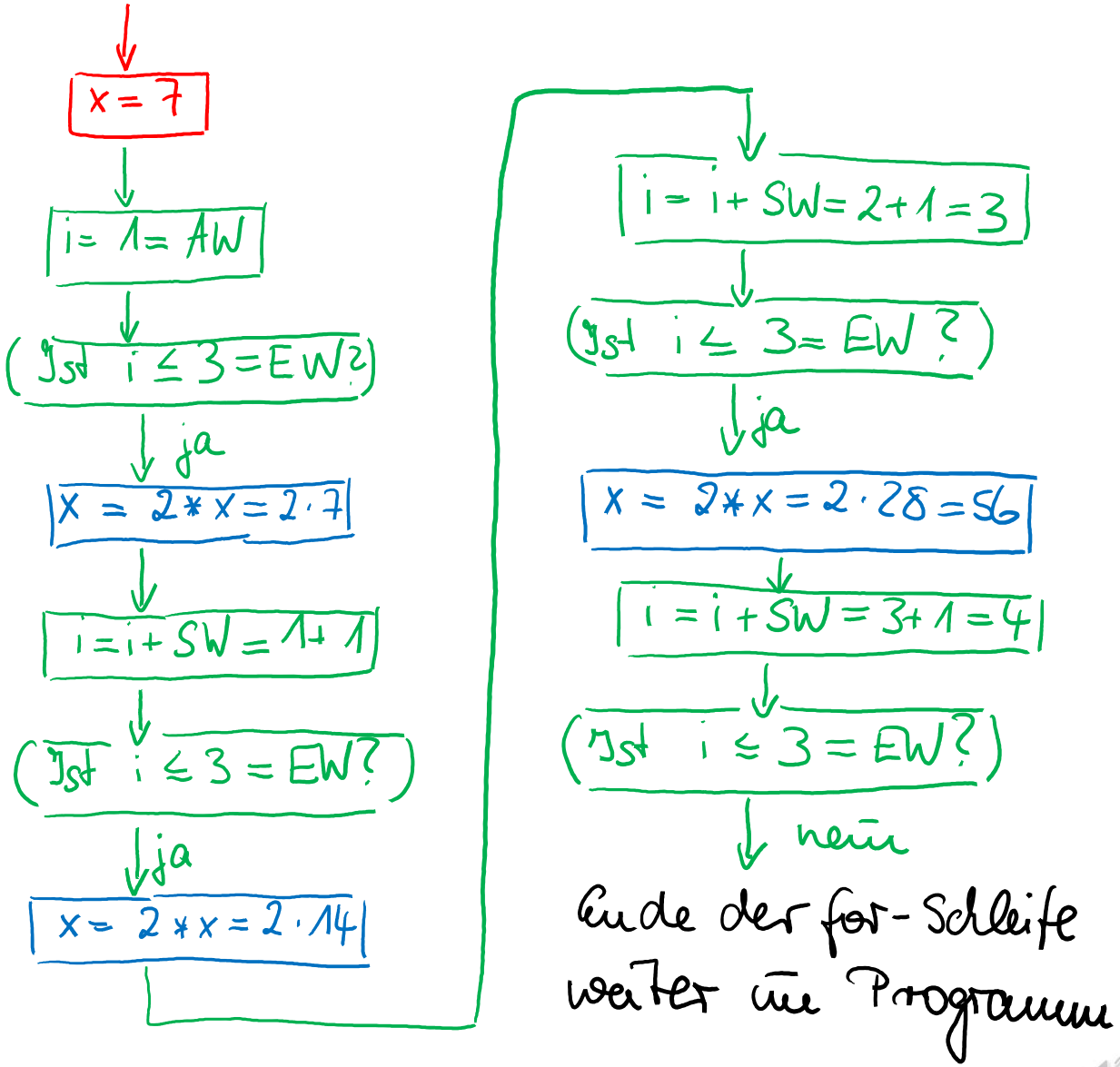
for-Schleife/Beispiel

```
x=7
for i =1:3; x=2*x,end
```

Antwort von Matlab:

```
x = 7
x = 14
x = 28
x = 56
>>
```

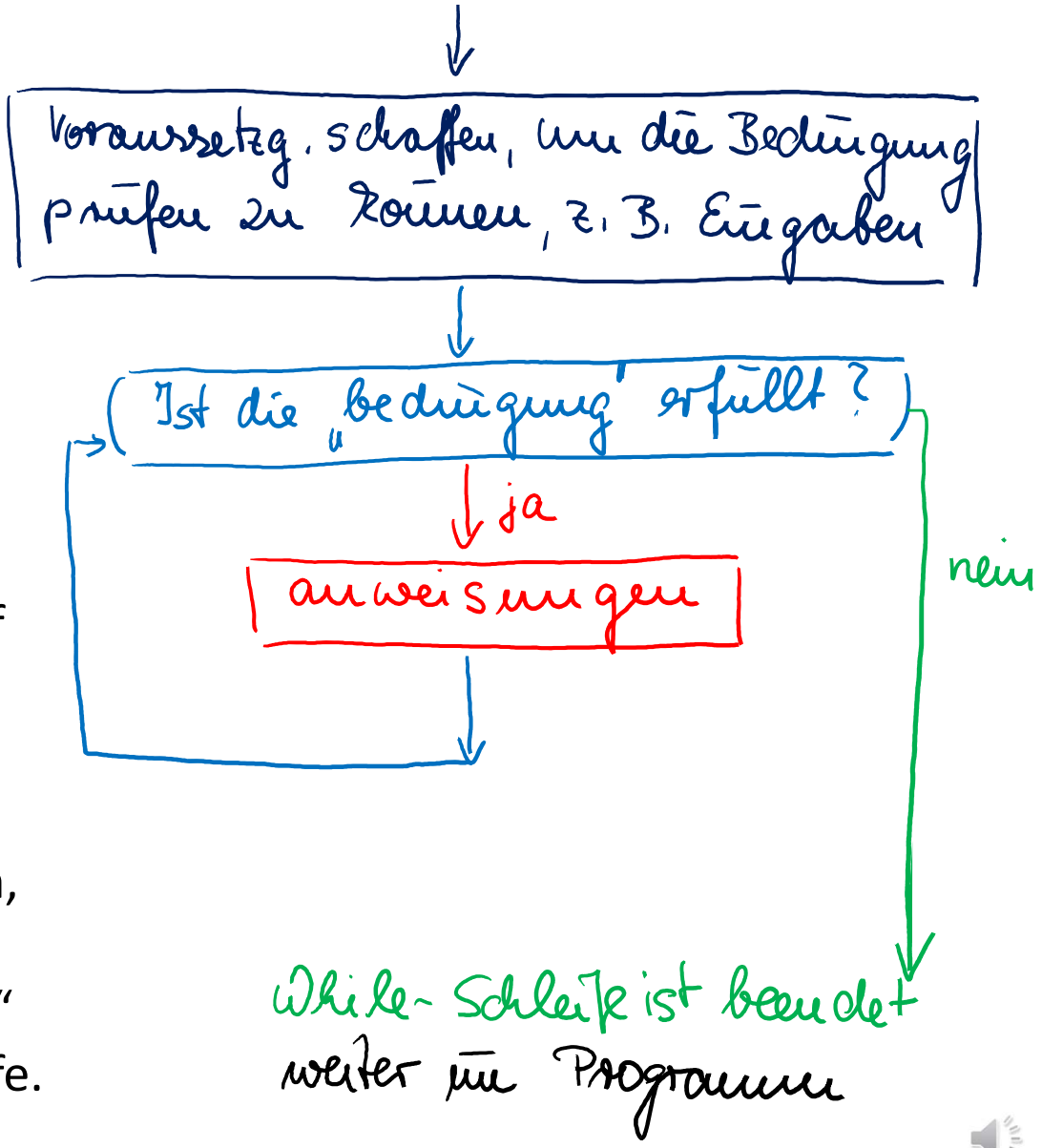
grün: Anweisungen, die der Rechner selbstständig ausführt



while-Schleife

```
while bedingung
        {anweisungen}
end
```

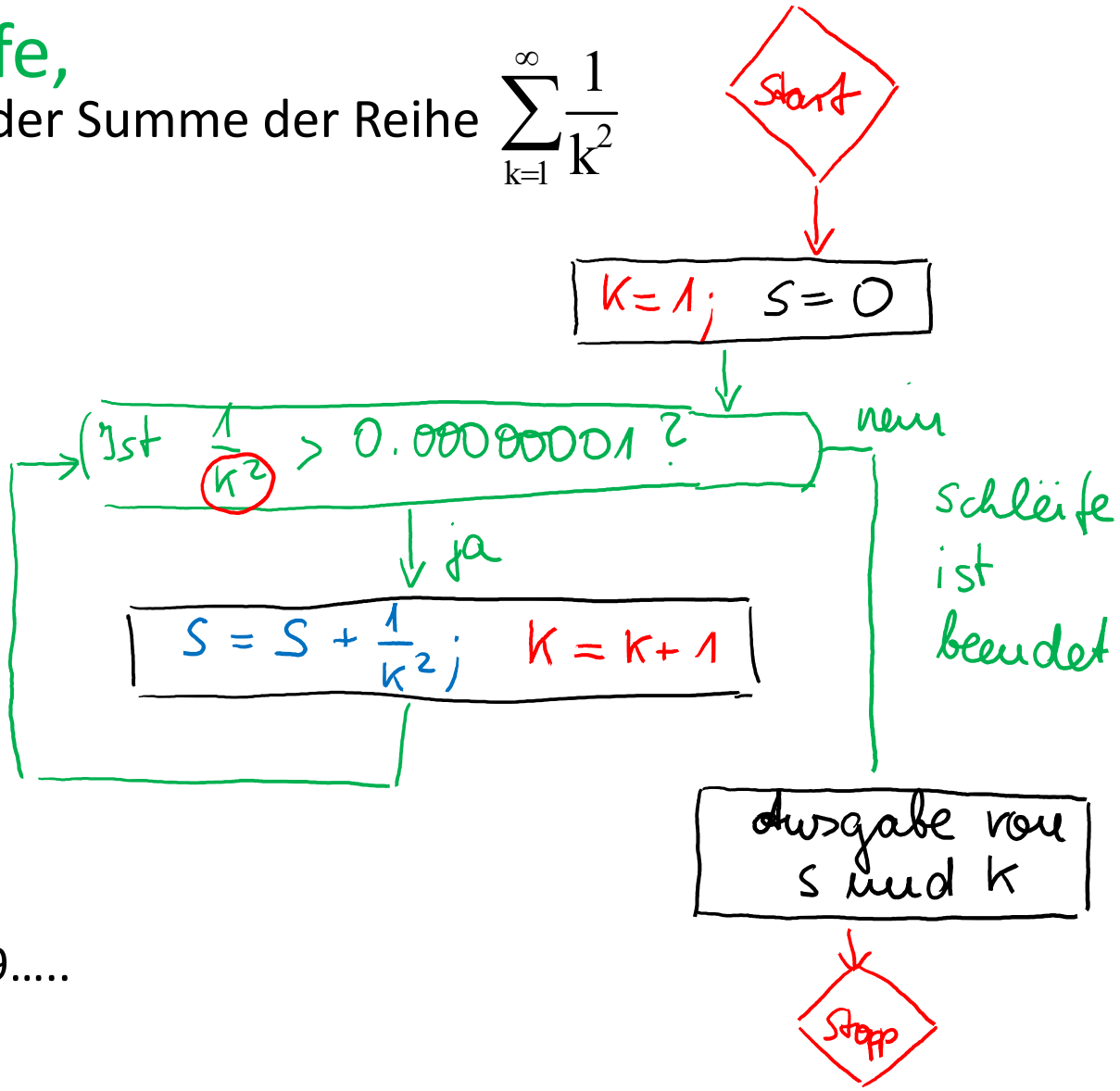
- Die „bedingung“ wird in jedem Durchlauf geprüft.
- Wenn sie erfüllt ist, werden die „anweisungen“ ausgeführt.
- Unter den „anweisungen“ muss eine sein, die Einfluss auf die Erfüllung/Nichterfüllung der „bedingung“ hat. Anderenfalls ist es eine Endlosschleife.



Beispiel/while-Schleife,

näherungsweise Berechnung der Summe der Reihe $\sum_{k=1}^{\infty} \frac{1}{k^2}$

```
k=1;
s=0;
while 1/k^2>0.00000001
        s=s+1/k^2;
        k=k+1;
end
s, k=k-1
```



Antwort von Matlab:

s = 1.6448

k = 9999

Exakter Wert: s =(pi)^2/6 = 1.6449.....

Skriptfiles

- als Hauptprogramm und Unterprogramm möglich
- Aufruf durch Angabe des Namens
- Zusammenstellung einzelner **Befehle** und **Kommentare**

  ➔ bessere Übersichtlichkeit,
  ➔ mehrfach verwendbar
  ➔ Reduktion von Programmtext
  ➔ modularer Aufbau des Gesamtprogrammes, wenn die Berechnung einzelner Bausteine in Skriptfiles zusammengefasst wird
- Variablen sind global verfügbar

  ➔ keine Parameterübergabe erforderlich
  ➔ Variablennamen nur einmal vergeben!
- Bsp.: Skriptfile KU zur Berechnung des Kreisumfanges

Funktionsdateien/Funktionsunterprogramme

- beginnen mit dem Schlüsselwort „function“,
- dienen zur Auslagerung wiederkehrender Berechnungen, die von (verschiedenen) Parametern abhängen,

➔ dienen den gleichen Zielen wie die Skriptfiles

➔ Unterschiede zu Skriptfiles:

- Eingabeparameter müssen im Aufruf angegeben werden.
- Variablen, die in der Funktionsdatei belegt werden, sind nur lokal definiert, d.h. nur in der Funktionsdatei (Ausnahme „ans“ und der Funktionsname selbst).
- An das rufende Programm werden nur die Größen zurückgegeben, die als Ausgabeparameter im Funktionskopf definiert sind.

Funktionsparameter/Funktionsaufbau

```
function [ output_args ] = Name( input_args )

    % zu beachten ist:
    % beim Abspeichern muss gelten: Filename = Name
    % [ output_args ] : Feld der Ausgabeparameter
    %  im einfachster Fall: Name der Ausgabevariable
    % Ausgabeparameter müssen in den Anweisungen belegt
      werden
    % input_args: Aufzählung der Eingabeparameter
Anweisungen
end
```


Beispiel für eine Funktion:
Berechnung des Kreisumfanges

```
function u = KUF( r )
            % Berechnung des Kreisumfanges u

if nargin < 1 % Notfallbehandlung bei
                  % fehlendem Parameter
   r=input('Eingabe des Radius r: ')
end

disp('Umfang u des Kreises mit dem Radius r: ')
u = 2*pi*r
end
```

Beispiel für den Aufruf: >>KUF(1) oder >>umfang=KUF(1)

Namenskonvention in den Belegen

- **Namen von selbsterzeugten Skriptfiles** werden nach den Hinweisen zum Beleg vergeben (s.o):
  Bsp.: **Mustermann_19w1_A1_B1_4.m**
- **Namen von Funktionen** enthalten an Stelle von Aufgaben- und Belegnummer eine frei wählbare Bezeichnung:
  Bsp.: **Mustermann_19w1_cramer.m**