

OOP-Vorlesung

Konzepte imperativer und objektorientierter Programmierung

Vorkurs + Vorlesung + Übungen

SoSe 2020

Prof. Dr. Margarita Esponda

Die beliebteste Programmiersprachen?

TIOBE Index (April 2020)

<https://www.tiobe.com/tiobe-index/>

Apr 2020	Apr 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.73%	+1.69%
2	2		C	16.72%	+2.64%
3	4	▲	Python	9.31%	+1.15%
4	3	▼	C++	6.78%	-2.06%
5	6	▲	C#	4.74%	+1.23%
6	5	▼	Visual Basic	4.72%	-1.07%
7	7		JavaScript	2.38%	-0.12%
8	9	▲	PHP	2.37%	+0.13%
9	8	▼	SQL	2.17%	-0.10%
10	16	▲▲	R	1.54%	+0.35%
11	19	▲▲	Swift	1.52%	+0.54%
12	18	▲▲	Go	1.36%	+0.35%
13	13		Ruby	1.25%	-0.02%
14	10	▼▼	Assembly language	1.16%	-0.55%

Welche Sprache ist besser?

Es gibt noch keine Programmiersprache, die für alle Anwendungen die beste ist.

- Java ist sehr beliebt auf Grund der JVM
- C eignet sich besonders gut für die hardwarenahe Programmierung
- SQL *Structure Query Language* ist eine Datenbanksprache
- Python für die effiziente Entwicklung von Prototypen
- R für statistische Berechnungen und graphische Darstellungen
- Swift eine multiparadigmatische Programmiersprache von Apple entwickelt
- Go unterstützt die Nebenläufigkeit
- usw.

Eigenschaften von C

Gute Der Programmierer kann machen, was er will.

Schlechte Der Programmierer kann machen, was er will.

Der C-Compiler geht davon aus, dass der Programmierer genau weiß, was er will, und meldet **fast keine Warnungen beim Übersetzen des Programms.**

Die meisten Fehler bei C-Programmen treten erst zur Laufzeit auf.

Was ist Python?

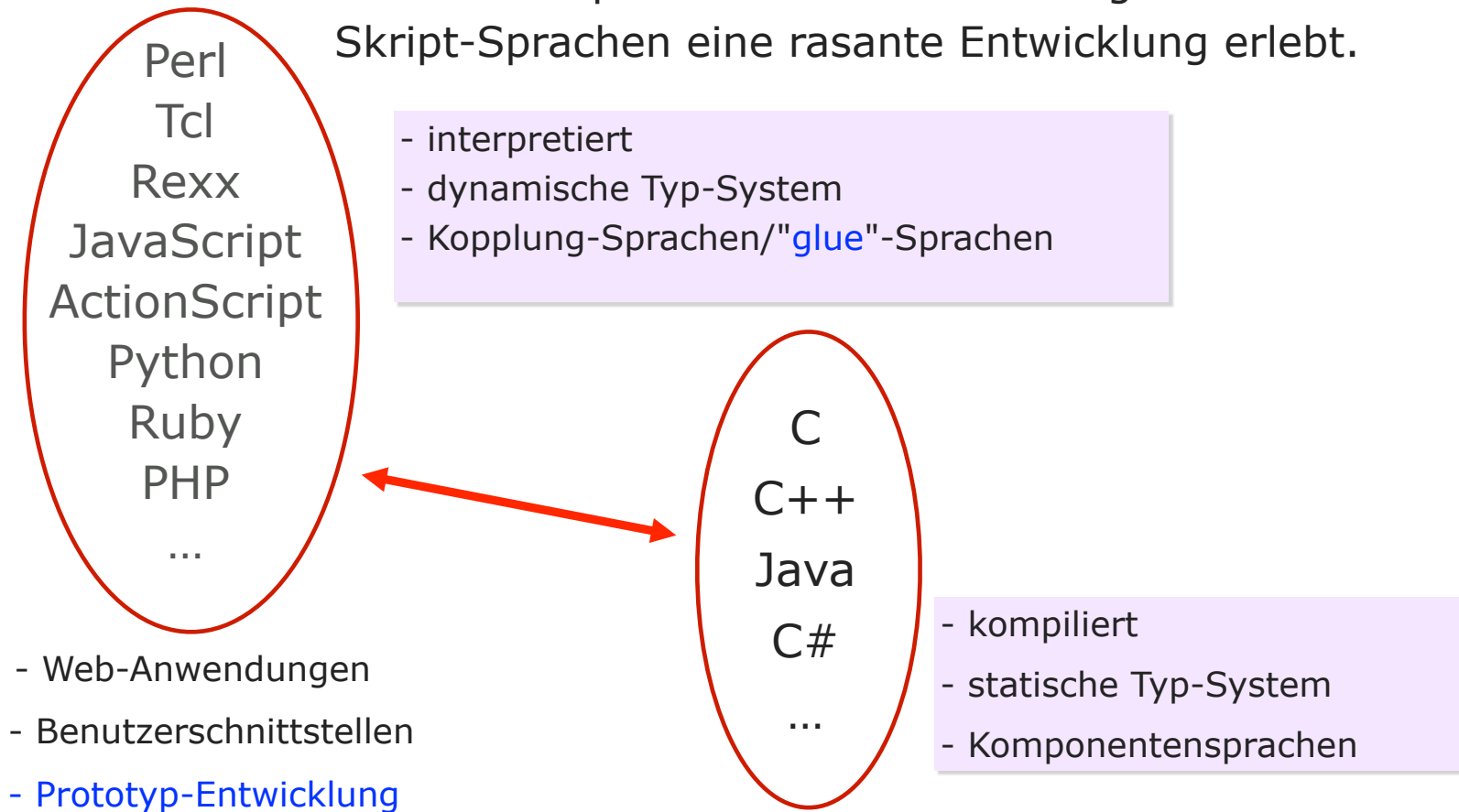
- eine **Script-Sprache**
- Anfang der **90er** Jahre entwickelt.
- Erfinder: Guido van Rossum an der **Universität von Amsterdam**
- Unterstützung des **strukturierten Programmierens** aus der **ABC**-Sprache übernommen.

ALGOL → ABC → Python

- **Philosophie:**
 - Simplizität, Lesbarkeit und Orthogonalität
 - **Schnelle Programmentwicklung** ist **wichtiger** als **schnelle Programme**

Scriptsprachen vs. konventionelle Sprachen

Durch die rapide Internet-Entwicklung haben Skript-Sprachen eine rasante Entwicklung erlebt.



Scriptsprachen

Vorteile

- Schnell erlernbar
- Schnelles Prototyping
- Kleiner Fehlersuche-Zyklus
- Kleinere Programme und schnellere Entwicklung
- Unterstützung vieler moderner Paradigmen
- Breite Verwendbarkeit in der Software-Industrie

Nachteile

- Speicherverbrauch ist größer
- Langsamer
- Keine statische Typkontrolle
- Für bestimmte Anwendungen nicht geeignet

Jedoch: große Unterschiede zwischen den einzelnen Skript-Sprachen!

Warum Python?

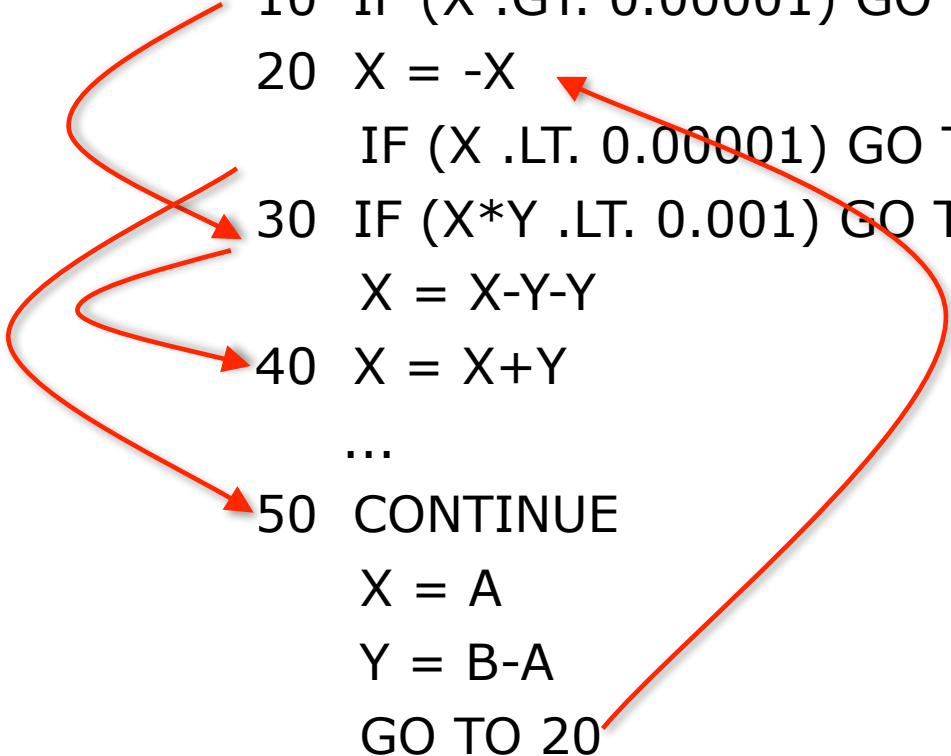
- speziell für die **Lehre** entwickelt
- als Ersatz für **Basic** und **Pascal** konzipiert
- die **Programme** sind sehr kompakt
- von Anfang an **strukturiertes Programmieren**
- einfache Syntax und Semantik
- Hybride Programmiersprache (**Multi-Paradigma**)
- Ermöglicht zuerst nur das rein imperative Programmieren zu lernen
- Höhere Datenstrukturen sind in der Sprache integriert
- Plattformunabhängig
- In der realen Welt verwendete Sprache
- Umfangreiche Standardbibliothek

GOTOs in Fortran

```
10 IF (X .GT. 0.00001) GO TO 30
20 X = -X
    IF (X .LT. 0.00001) GO TO 50
30 IF (X*Y .LT. 0.001) GO TO 40
    X = X-Y-Y
40 X = X+Y
    ...
50 CONTINUE
    X = A
    Y = B-A
    GO TO 20
    ...
```

GOTOs in Fortran

```
10 IF (X .GT. 0.00001) GO TO 30
20 X = -X
   IF (X .LT. 0.00001) GO TO 50
30 IF (X*Y .LT. 0.001) GO TO 40
   X = X-Y-Y
40 X = X+Y
   ...
50 CONTINUE
   X = A
   Y = B-A
   GO TO 20
   ...
```

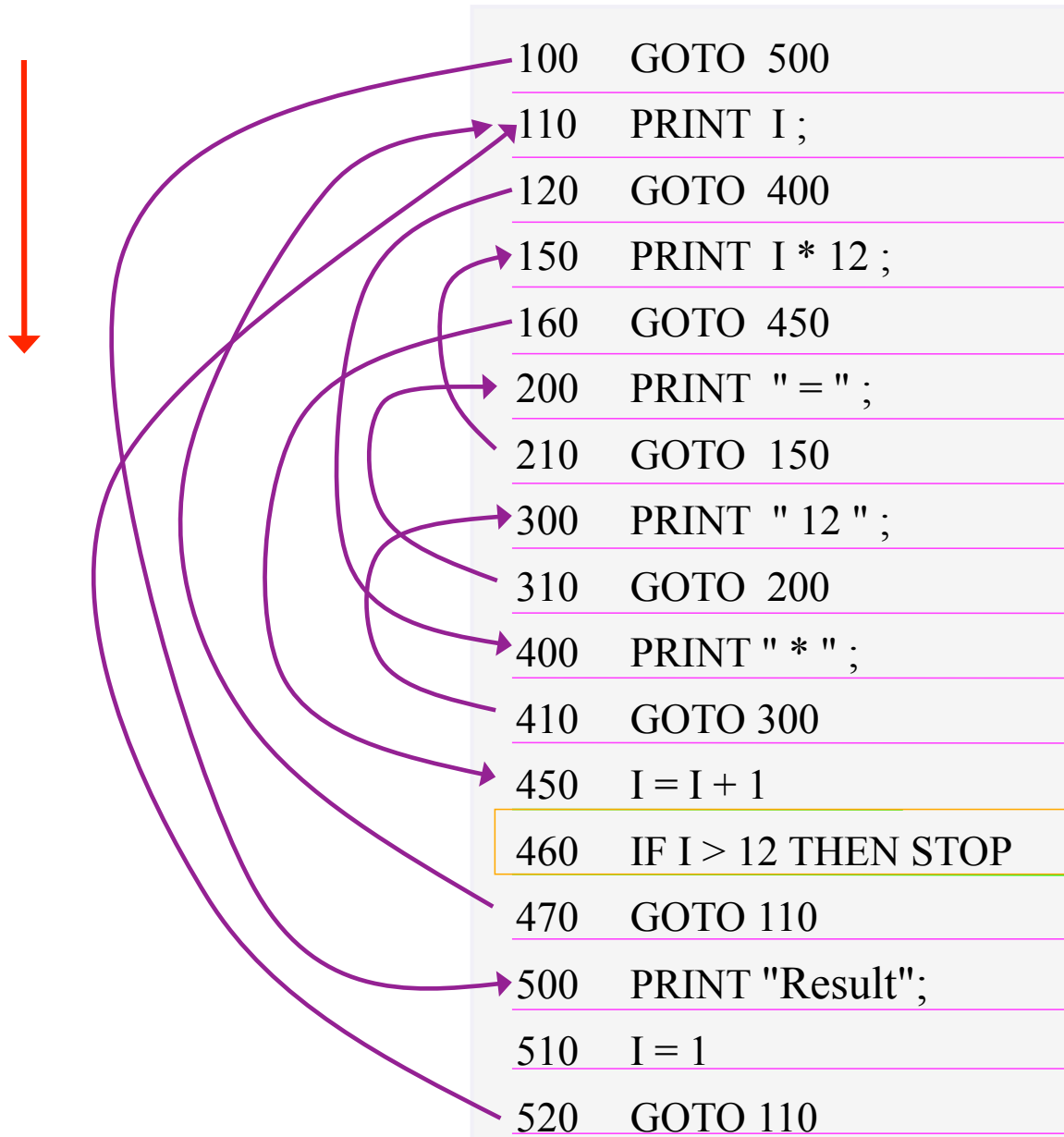




```
100  GOTO 500
110  PRINT I ;
120  GOTO 400
150  PRINT I * 12 ;
160  GOTO 450
200  PRINT " = " ;
210  GOTO 150
300  PRINT " 12 " ;
310  GOTO 200
400  PRINT " * " ;
410  GOTO 300
450  I = I + 1
460  IF I > 12 THEN STOP
470  GOTO 110
500  PRINT "Result";
510  I = 1
520  GOTO 110
```

Result**1 * 12 = 12****2 * 12 = 24****3 * 12 = 36****...**

„GOTOs und
Spaghetti code“

**Result****1 * 12 = 12****2 * 12 = 24****3 * 12 = 36****...**

„GOTOs und
Spaghetti code“

Die GOTO-Anweisung

- die GOTO-Anweisung ist die **meist umstrittene Kontroll-Anweisung** in der Welt der höheren Programmiersprachen.
- **theoretisch** kann jeder GOTO nach vorne mit einer **if-else**-Anweisung und jeder GOTO nach hinten mit einer **while**-Schleife ersetzt werden.
- **Assembler-Befehle** haben keine **if-else**- oder **while**-Anweisungen sondern nur bedingte oder unbedingte GOTOs.
- die Äquivalenz ist klar, sonst gäbe es keine Übersetzer.
- **C hat GOTOs, und Java hat beschränkte GOTOs.**
- **Python hat keine GOTO-Anweisung.**

Einrücken anstatt Klammern

- kein *begin* ... *end* wie in Pascal oder { ... } wie in C
- Die Anweisungen innerhalb eines Blocks beginnen immer an der gleichen Zeilenspalte

Beispiel:

```
x = int ( input() )  
if x <= 0:  
    x = 0  
    print ('zero')  
else:  
    x = 1  
    print ('one')
```

C ist eine formatfreie Sprache!

„The International
Obfuscated C Code Contest“

Quelle: <http://www.ioccc.org>

```
#include/*nui*/<stdio.h>70/*#}r[3]op;f(p;ok})i[;k-r*?(rc&(o)nr**s*2)!~mpi##
extern int n0;typedef int x;x//i/eu2->uuo0uo=nXfdx+1e8u0eh&k-x[e1(i)>{=ega,nii
n,u,k,o,_,i=1;static char//[X]/f/t]:n=t-rxt+0f[(-=+;t)*,aa!>1=dt0pzirpi(1)idtnn
d[1125][0x401];x main(){if(//)* nu]O[nc-(ac=;odxxlk})u)2ulr=(00+u2=ee&fos{n,*cc
i){for(n=0;1024>n;n++)//]Tkhng[0ur][u[h>u)hlor];>]-=Or):=1*Ou);+r4poa&(=ep(qnll
for(u=0x0;u<1025;)d[n][u++]=64//[n;o]ua0=a,<(=)X;no[n{8uo)=&{i]n)?f1!!g{u)=,uu
/2;for(u=n=0////////_]#p^#onui[/u+}?+r;d{r/X//////////c/(&if=-)p(l(xewt{ludd
/4;EOF!=(o=//////////]#ebdl#ah[0]n/1()//////////l f(k1)*ion)ths0; ,ee
getchar())//          //////////]u#oh[ ,;//////////          //////////; )2nc={ci(=ck<<
&&u<1024;//          //////////          //////////}{:t( ihl"Nh,ss
)u+=o-10//          //////////          ///////////2)nifaeYUaott
?n<1024//          //////////          //////////,n(r(uLr,dd
?d[n+//          @@@          //////////td(EuL*Xli
][u]=o//          @   @          //(+)Or;O,io
,k=k<//          @   @          //(Fii;xb.
n?n:k//          @   @          //(fi,.h
,0:0//          @@@          //(n/h>
:!(n//          //////////          //;0t/>/
=0);//          //////////          //!/|//
for(//          5          //=/*%
;k--//          12          =q tni
;)d[k//          //////////          //(u/
/01][u//          //////////          //lNi/
/1]/*n//          //////////          //(Ui;
>*/=!/*N//          //////////          //=-K~/
h*/1,/*UN//          //////////          //nq0~*
./puts/*n///          //////////          //[/? ,]
o*/(d[k])/*u////////          //////////          //l*q[u
i*//;}else{t////////          //////////          //,stup;
d[0][0]++;puts(*f////////          //]===[////////          //(N tni
ti&/N/////./s/It]]_bz8[//          //SHOUJO\vv////////          //;} "jvo"
su /U/////N///t]Ue]~J#phi[//          //SHUUMATSU|_]////////          //,"utf"
<ntt/fe=)|UI0{u;Nnu]^u#j[v//          //,^~\RYOKOU:)}]a#p[.//          //"c!tj!xb"
e/ n/ilI(|/(lep)/ *>->IOCCC//          //]^#dbi#`h#anuok^u#[//          //"S","/ttfm"
d/eit{i(rl/r-s-"/e/]o[^^^!/////]hfhu[Qj:FfT]uhp)~[//          //"iuspx!fsb!tho"
unn;n)h=a({aIl0onnt// "jiu!fmcjefoJ", "/zsbdt!fsb!tobnvI!"          //"", "-<l!osvufs<*2"
lriqi(w!hf)h=eltrin//".;2;1;1+*432&25\*3,o)_6)92x\sbiduvq@1=v@2:.o@4:.o),v,1/>"
cef ;n{Fci2cl}-iufi// "v*1?**)sbidufh>o))fmjix[*]ojbn!uoj<v-o!uoj31?i/pjeut=fe"
ntetli)Ot{3t; }lhte}// "vmdoj$", "svpU!utbM!(tmsjH">{=]041[ ]6[u,n*rahc;q tni nretxe
ixdn,aqEe)-u)}=Ced;//};0+nruter;)K(U;)+n*--;n*;K=n(rof{})(niam )O,K,U(N enifed#
#e#iIm((g)Ip ;I"r#0.[(c)2018][cffc189a]*/"Nuko");}return 0;}//>h.oidts<edulcni#
```

Python unterstützt mehrere Paradigmen

- flexibel in der Handhabung verschiedener Programmier-Paradigmen
 - * **Imperative Programmierkonzepte**
 - * Objektorientierte Programmierung
 - * Funktionale Programmierung
 - * Aspektorientierte Programmierung
- einfaches Einsetzen verschiedener Programmiertechniken
 - * Strukturierte Programmierung
 - * Entwurf gemäß Vertrag (DBC)

Java

```
public class HelloWorldProgram {  
    public static void main ( String[] args ) {  
        System.out.println( "Welcome to OOP-Lecture!" );  
    }  
}
```

C

```
#include <stdio.h>

int main(void){
    printf( "Welcome to OOP-Lecture!" );
    return 0;
}
```



Python

```
print ('Welcome to OOP-Lecture!')
```

Python 3.x

Python-Interpreter

Linux und Mac OS



Der Python-Interpreter
ist Teil der Standard-
Installation.



`python`

Windows und Mac OS



`IDLE`

Integrierte Development Environment
www.python.org/idle

`python 3.8`

Imperatives Programmieren

Ältestes und populärstes Programmierparadigma

Fortran, Cobol, Algol, C, Basic, Pascal, Modula, Java, Python, Pearl,
usw.

D, R, Dart, Ceylon, Go, Opa, Fantom, Haxe, Chapel, X10, Zimbu,
Swift, Julia, usw.

Widerspiegelt die dahinter stehende Hardware-Architektur

von Neumann Zustands-Maschine
gespeichertes Programm + Daten

Grundlegende Elemente von imperativen Programmen

- **Variablen**
- Datentypen
- Zuweisungen
- Ausdrücke
- Anweisungen für den Kontrollfluss innerhalb des Programms
- Gültigkeitsbereich von Variablen (*locality of reference*)
- Definition von Prozeduren und Funktionen

Variablen

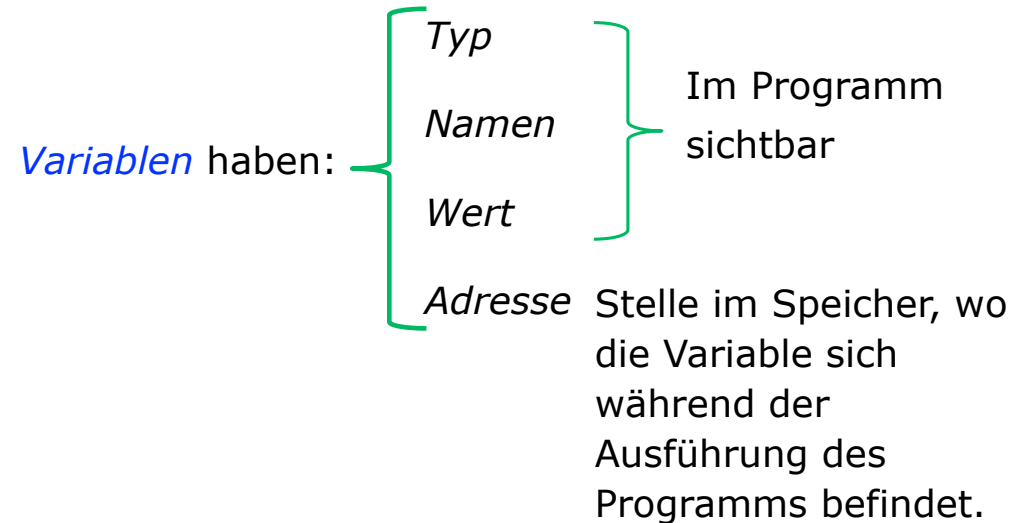
Funktionale Programmiersprachen

Eine Variable stellt nur den symbolischen Namen von einem Wert oder einem Ausdruck, der zu einem Wert ausgewertet werden kann, dar.

Der Wert einer Variablen kann nicht verändert werden.

Imperative Programmiersprachen

Variablen sind Stellen im Speicher, in denen Werte abgelegt werden können. Variablen speichern Zustände.



Variablen müssen normalerweise vor der erstmaligen Benutzung deklariert werden.

In Python nicht!

Konzepte imperativer Programmierung

Die eigentliche Geschichte der Programmiersprachen begann mit:

- dem Konzept der **von Neumann-Maschine**, die **1945** die Notwendigkeit eines **gespeicherten Programms** postulierte
- und mit dem Konzept des "*conditional control transfer*"
 - ***If-then***-Anweisung
 - ***looped***-Anweisungen (***for***- und ***while***-Schleifen)
 - ***Subroutines*** (**Funktionen**)

Datentyp einer Variablen

- * Python hat **dynamische Datentypen**, ist aber streng typisiert.
- * Der Datentyp einer Variable wird erst zur Laufzeit festgelegt.
- * Im Gegensatz zur statischen Typisierung, bei der der Datentyp einer Variable explizit deklariert werden muss, wird der Typ einer Variablen aus dem Typ des Werts **zur Laufzeit abgeleitet**.
- * **Quelle einiger schwierig zu findender Fehler.**
- * Streng typisiert.

Datentyp einer Variablen

- In **C** wird damit die **minimale Speichergröße**, die eine Variable braucht, festgelegt.
- In modernen Programmiersprachen wird damit der **Wert-Bereich**, den eine Variable annehmen kann, beschränkt.
- **Python** hat ein dynamisches Typsystem

Der Datentyp von Variablen wird in Python erst zur Laufzeit festgelegt und **kann** während der Ausführung des Programms **verändert werden**.

Kommentare in Python

```
""" Blockkommentare:
```

```
    können sich über mehrere Zeilen  
    erstrecken.
```

```
"""
```

```
# von hier aus bis Ende der Zeile wird dieser Text ignoriert
```

```
a = 4  # Kommentar
```

Eingabe und Ausgabe

```
input ( Prompt )  
...  
print ( Ausdrücke )
```

Die *input*-Funktion liest Zeichenketten aus der Standardeingabe ein und gibt diese als Rückgabewert der Funktion zurück.

Die *print*-Funktion kann Zeichenketten, Zahlenwerte oder zusammengesetzte Datenstrukturen ausgeben.

Dynamisches Typsystem

Python Virtuelle Maschine **PVM**

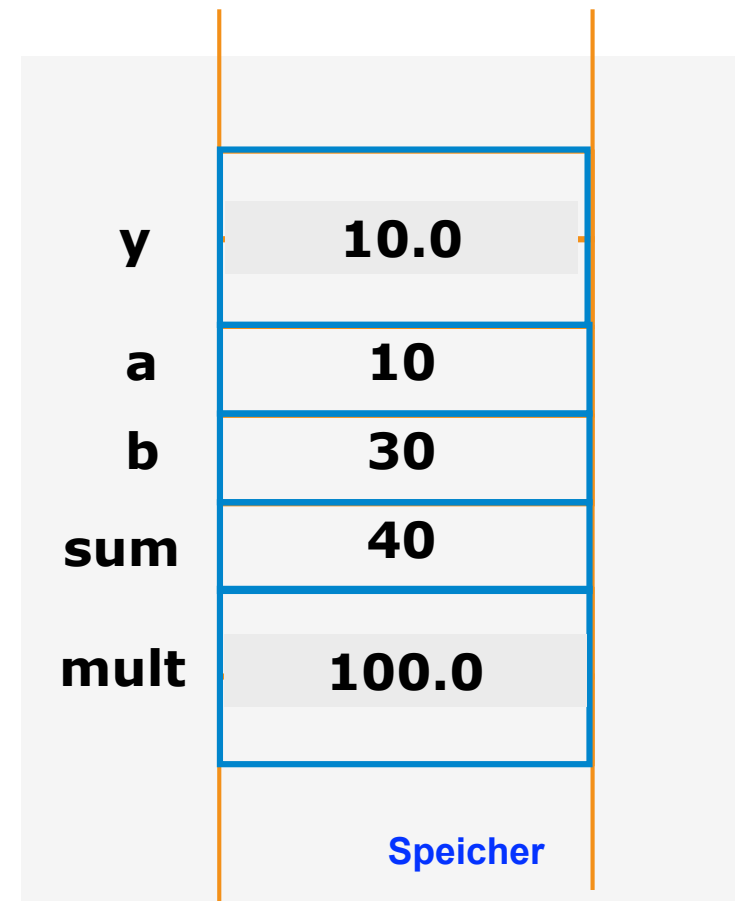
y = 10.0

a = 10

b = 30

sum = a+b

mult = a*y



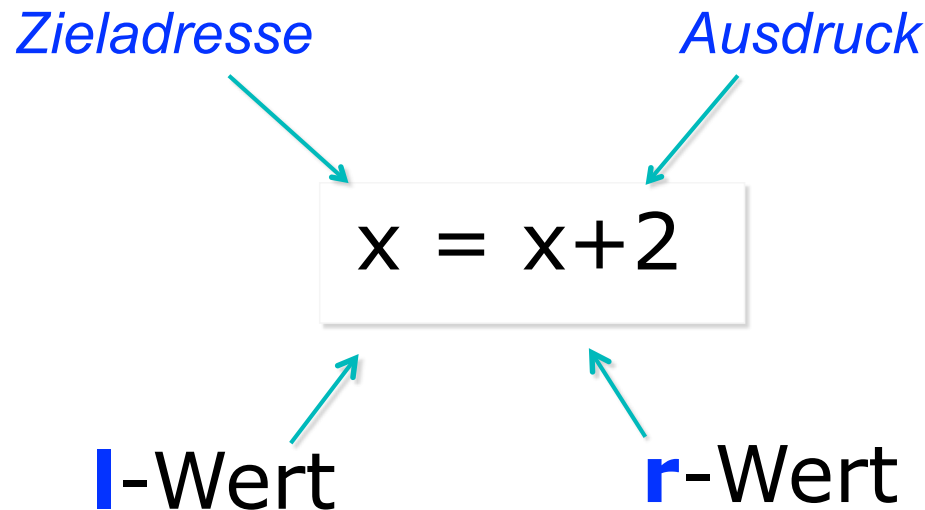
Zuweisungen



Zuweisungen sind destruktive Anweisungen.

Die Adresse mit dem symbolischen Namen **A** wird mit dem Wert des Ausdrucks **B** überschrieben.

Zuweisungen



Speicherbereich der Variable x

Aktueller Wert der Variable x

Zwei verschiedene semantische Bedeutungen des gleichen Symbols. ☹

Multiple-Zuweisungen

Python-Zucker 😊

```
a, b, c = 2, 3, 5  
print ( a, b, c )
```

```
>>>  
2 3 5
```

```
a = b = c = 2  
print ( a, b, c )
```

```
>>>  
2 2 2
```

```
x = [a, b] = [1, 2]  
print ( a, b, x )
```

```
>>>  
1 2 [1, 2]
```

Datentypen in Python

Datentyp	Beispiel	Bemerkungen
int	<code>a = 3</code>	
float	<code>x = 1.0</code>	
boolean	<code>w = True</code>	
Complex	<code>c = 2 + 3j</code>	
String	<code>t = "Text" oder t = 'Text'</code>	nicht veränderbar
Liste	<code>l = [5, 3, 1, 6, 8]</code>	veränderbar
Tuple	<code>p = (35, 0, "Name")</code>	nicht veränderbar
Dictionary	<code>d = { 1:'a', 2:'b', 3:'c'}</code>	veränderbar

Ausdrücke

Ein Ausdruck wird in der Informatik als eine Berechnungsvorschrift verstanden, die einen eindeutigen Wert darstellt.

Die einfachsten Ausdrücke in Python sind Konstanten beliebiger Datentypen

Beispiel: **1** **3.4** **"hello"** (**Literale**)

Ausdrücke haben keinen **|**-Wert und können nicht auf der linken Seite einer Anweisung stehen.

Beispiele:

a+b = c oder **3 = c**

Arithmetische Operatoren

Operator	Beschreibung	
+	unär	ändert nichts an der Zahl
-	unär	ändert das Vorzeichen der Zahl
**	binär	Power-Operator
*	binär	Multiplikation
+	binär	Addition
-	binär	Subtraktion
/	binär	Division
%	binär	Modulo (Restbildung)
//	binär	Division (immer ganzzahlig)

Ausdrücke

Ausdrücke haben einen **Wert** und einen **Datentyp**

Beispiele:

	Ausdruck	Wert	Type
type casting	<code>int("12345")</code>	12345	int
	<code>int(0xff)</code>	255	int
	<code>complex(3.14)</code>	(3.14+0j)	complex
	<code>(3*11//5)</code>	6	int
	<code>8%2</code>	0	int
	<code>8/3</code>	2.66666666..	float
	<code>float(3)</code>	3.0	float
	<code>2**(2+1)</code>	8	int
	<code>pow(2,3)</code>	8	int

weitere Operationen ...

```
>>> help ( int )
```

Methods defined here:

```
| __abs__(...)  
| x.__abs__() <==> abs(x)  
| __add__(...)  
| x.__add__(y) <==> x+y  
| __and__(...)  
| x.__and__(y) <==> x&y  
| __bool__(...)  
| x.__bool__() <==> x != 0  
| __ceil__(...)  
| Ceiling of an Integral returns itself.  
| __divmod__(...)  
| x.__divmod__(y) <==> divmod(x, y)  
| __eq__(...)  
| x.__eq__(y) <==> x==y  
| __float__(...)  
| x.__float__() <==> float(x)  
| __floor__(...)  
...
```

```
>>> import math  
>>> math.pi  
3.141592653589793  
>>> math.e  
2.718281828459045  
>>> math.gcd(56,14)  
14  
>>> math.cos(3.0)  
-0.9899924966004454  
>>> math.factorial(33)  
8683317618811886495518194401280000000  
>>> help(math)  
Help on module math:  
NAME      math  
MODULE REFERENCE  
           https://docs.python.org/3.6/library/math  
           The following documentation . . .  
DESCRIPTION  
           This module is always available. It provides . . .  
FUNCTIONS  
           acos(x) . . .  
           . . .
```

Import-Anweisung

Die **import**-Anweisung ermöglicht in Python-Skripten den Zugang auf vorprogrammierte Module

Einige interessante Module sind:

math:	exp, sin, sqrt, pow
string:	digits, whitespace
sys:	stdin, stderr, argv
os:	system, path
re:	split, match

Formel des gregorianischen Kalenders zur Berechnung des Wochentags

$$y_0 = year - \frac{14 - month}{12}$$

$$x = y_0 + \frac{y_0}{4} - \frac{y_0}{100} + \frac{y_0}{400}$$

$$m_0 = month + 12 \left(\frac{14 - month}{12} \right) - 2$$

$$Name = \text{mod} \left(\left(day + x + \frac{(31 \cdot m_0)}{12} \right), 7 \right)$$

Höhere Datenstrukturen

Python unterstützt vier *sequentielle höhere Datentypen*

Listen (dynamic arrays)

Tuples (immutable lists)

Dictionaries (hash tables)

Listen

Eine *Liste* ist eine **veränderbare** Sammlung von Objekten verschiedener Datentypen.

Beispiele:

`a = []` **# Leere Liste**

`b = [1, 3.5, "Zeichenkette", 10]`

`c = ['Hi', 4, 0.234]`

Zeichenketten-, Listen- und Tupel-Operatoren

Operator	Funktion	Beispiel	Wert
+	Verkettung	"Eins" + " Zwei"	'Eins Zwei'
*	Wiederholung	2 * "Eins"	'EinsEins'
in	enthalten in	1 in [1,2]	True
not in	nicht enthalten in	3 not in [1,2]	True
x[i]	Indizierung, liefert das Element aus x an der Position i	x = "hallo" x[1]	a
x[i:j]	liefert aus x die Elemente von i bis j-1 zurück	x = "hallo" x[0:3]	'hal'

Höhere Datenstrukturen in Python

Beispiele:

```

0 1 2 3
>>> a = [1, 2, 3, 4]

```

```

>>> a[2]

```

```

3

```

```

>>> a[2] = 10

```

```

>>> a

```

```

[1, 2, 10, 4]

```

```

>>> b = a

```

```

>>> b

```

```

[1, 2, 10, 4]

```

```

>>> a[1] = 2000

```

```

>>> b

```

```

[1, 2000, 10, 4]

```

```

>>> a = [7, 6, 5, 4]

```

```

>>> a

```

```

[7, 6, 5, 4]

```

```

>>> b

```

```

[1, 2000, 10, 4]


```

Hier wird nur eine
Speicheradresse
kopiert.

Hier wird eine neue
Liste erzeugt.

b zeigt aber auf die
alte Liste.

Beispiele:

```
          -5 -4 -3 -2 -1  
          0  1  2  3  4  
>>> a = [1, 2, 3, 4, 5]  
[1, 2, 3, 4, 5]  
>>> a[2:4]  
[3, 4]  
>>> b = a[:]   
>>> b  
[1, 2, 3, 4, 5]  
>>> a[1] = 2000  
>>> b  
[1, 2, 3, 4, 5]  
>>> a  
[1, 2000, 3, 4, 5]  
>>> a[-1]  
5  
>>> a[-2]  
4
```

Hier wird die Liste
vollständig kopiert.

Tupel

Ein *Tupel* ist eine Sammlung von Objekten verschiedener Datentypen zu einem Objekt.

Im Gegensatz zu einer Liste ist die Folge der Elemente dabei **nicht veränderbar**.

Beispiele:	<code>()</code>	# Leeres Tupel.
	<code>(1,)</code>	# Tupel mit einem Element.
	<code>(1,2,3)</code>	# Tupel mit drei Elementen.
	<code>(1, 'Eins')</code>	# Tupel mit zwei Elementen.
	<code>number = (1, 'eins')</code>	

Beispiele:

```
>>> a = (1,2,3)
```

```
>>> b = a
```

```
>>> b
```

```
(1, 2, 3)
```

```
>>> a[2]
```

```
3
```

```
>>> len(a)
```

```
3
```

```
>>> a == b
```

```
True
```

```
>>> a[0] = 5
```

Traceback (most recent call last):

File "<pyshell#76>", line 1, in <module>

a[0] = 5

TypeError: 'tuple' object does not support
item assignment

```
>>> a = (5,6,7)
```

```
>>> b
```

```
(1, 2, 3)
```

```
>>> a+b
```

```
(5, 6, 7, 1, 2, 3)
```

```
>>> a = a+a
```

```
>>> a
```

```
(5, 6, 7, 5, 6, 7)
```

```
>>> b
```

```
(1, 2, 3)
```

```
>>> a*3
```

```
(5, 6, 7, 5, 6, 7, 5, 6, 7, 5, 6, 7, 5, 6, 7, 5, 6, 7)
```


Vergleichsoperatoren

alle binär

$<$	Kleiner
$>$	Größer
$<=$	Kleiner oder gleich
$>=$	Größer oder gleich
$==$	Gleichheit
$!=$	Ungleichheit

Fallunterscheidung

if-else-Anweisung

```
a = int(input( "Zahl = " ))  
  
if a<0:  
    print ( "a ist negativ" )  
else:  
    print ( "a ist positiv" )
```

```
a = int(input( "Zahl = " ))  
  
if a<0:  
    print ( "a ist negativ" )  
elif a==0:  
    print ( "a ist gleich 0" )  
else:  
    print ( "a ist positiv" )
```

Einrücken anstatt Klammern

