ETH zürich

Departement Informatik

Programmieren und Problemlösen

Einführung in die Vorlesung

Dennis Komm

Frühling 2021 – 25. Februar 2021

Willkommen zur Vorlesung

Material

Vorlesungswebseite

`https://lec.inf.ethz.ch/ppl`

Moodle-Kurs

`https://moodle-app2.let.ethz.ch/course/view.php?id=14883`

Das Team

Dozent	Dennis Komm
Assistierende	Manuela Fischer
	Jonas Hein
	David Sommer
	Cathirn Elich
	Lea Fritschi
	Sarah Kamp
	Safira Piasko
	Sara Steiner

Termine

Vorlesung	Donnerstag, 16:15 – 18:00
Übungen	Montag, 13:15 – 15:00 Donnerstag, 10:15 – 12:00
Klausur	Ende des Semesters

Ziel der heutigen Vorlesung

- Allgemeine Informationen zur Vorlesung
- Übungsbetrieb mit **[code]expert**
- Einführung in Algorithmen
- Das erste Python-Programm

Einführung in die Vorlesung

Computer und Algorithmen

Computer – Konzept

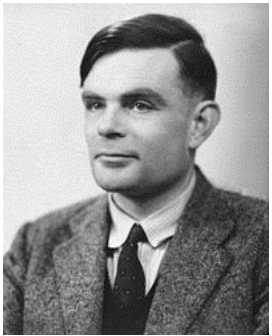
- Was muss ein Computer können, um zu rechnen?
- Muss er z. B. die Multiplikation beherrschen?
- Reicht nicht die Addition?

Computer – Konzept

- Was muss ein Computer können, um zu rechnen?
- Muss er z. B. die Multiplikation beherrschen?
- Reicht nicht die Addition?

Turingmaschine [Alan Turing, 1936]

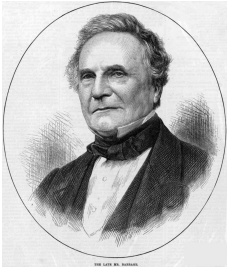
- Endliche Anzahl von Zuständen
- Speicher, der aus beliebig vielen Zellen besteht
- Pointer auf aktuelle Zelle
- Pointer kann Inhalt der Zelle ändern und sich nach links oder rechts bewegen



Alan Turing [Wikimedia]

Computer – Umsetzung

- **Analytical Engine** – Charles Babbage (1837)
- **Z1** – Konrad Zuse (1938)
- **ENIAC** – John von Neumann (1945)



Charles Babbage [Wikimedia]

Konrad Zuse [Wikimedia]

John von Neumann [Wikimedia]

Algorithmus: Kernbegriff der Informatik

Algorithmus

- Handlungsanweisung zur schrittweisen Lösung eines Problems

Algorithmus: Kernbegriff der Informatik

Algorithmus

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keinen Intellekt, nur Genauigkeit

Algorithmus: Kernbegriff der Informatik

Algorithmus

- Handlungsanweisung zur schrittweisen Lösung eines Problems
- Ausführung erfordert keinen Intellekt, nur Genauigkeit
- nach **Muhammad al-Chwarizmi**;
  Autor eines arabischen
  Mathematik-Lehrbuchs (um 825)

“Dixit algorizmi. . . ” (lateinische Übersetzung) [Wikimedia]

„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

Eingabe: a und b
Solange $b \neq 0$
    Wenn $a > b$ dann
        $a = a - b$
    Sonst
        $b = b - a$
Ausgabe: a

„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

Eingabe: a und b
Solange $b \neq 0$
    Wenn $a > b$ dann
        $a = a - b$
    Sonst
        $b = b - a$
Ausgabe: a

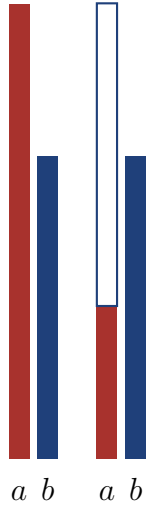
„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

Eingabe: a und b
Solange $b \neq 0$
    Wenn $a > b$ dann
        $a = a - b$
    Sonst
        $b = b - a$
Ausgabe: a



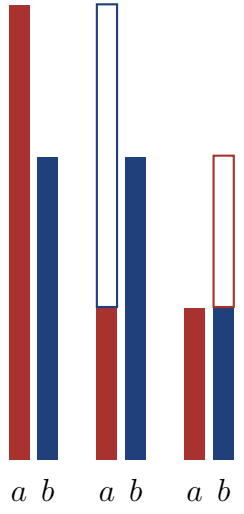
„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

Eingabe: a und b
Solange $b \neq 0$
    Wenn $a > b$ dann
        $a = a - b$
    Sonst
        $b = b - a$
Ausgabe: a



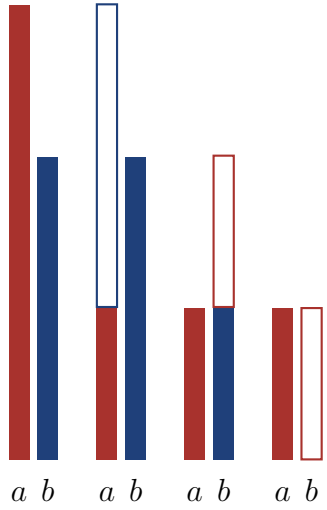
„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

Eingabe: a und b
Solange $b \neq 0$
    Wenn $a > b$ dann
        $a = a - b$
    Sonst
        $b = b - a$
Ausgabe: a



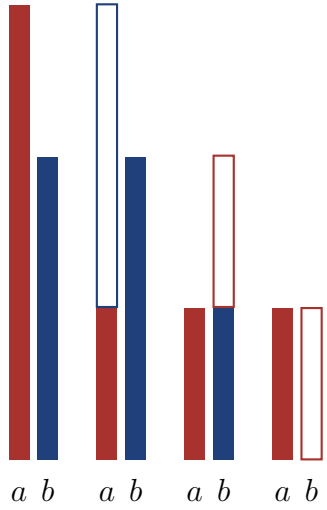
„Der älteste (bekannte) nichttriviale Algorithmus“

Euklidischer Algorithmus

aus Euklids *Elementen*, 300 v. Chr.

- Eingabe: ganze Zahlen $a > 0$, $b > 0$
- Ausgabe: ggT von a und b

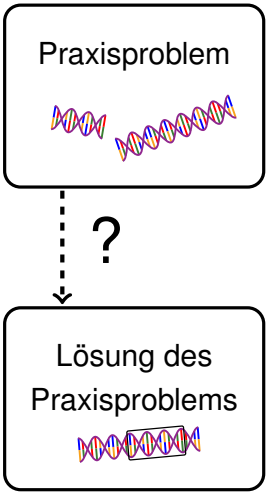
Eingabe: a und b
while b **!= 0:**
    if a **>** b**:**
        $a = a - b$
    else:
        $b = b - a$
Ausgabe: a



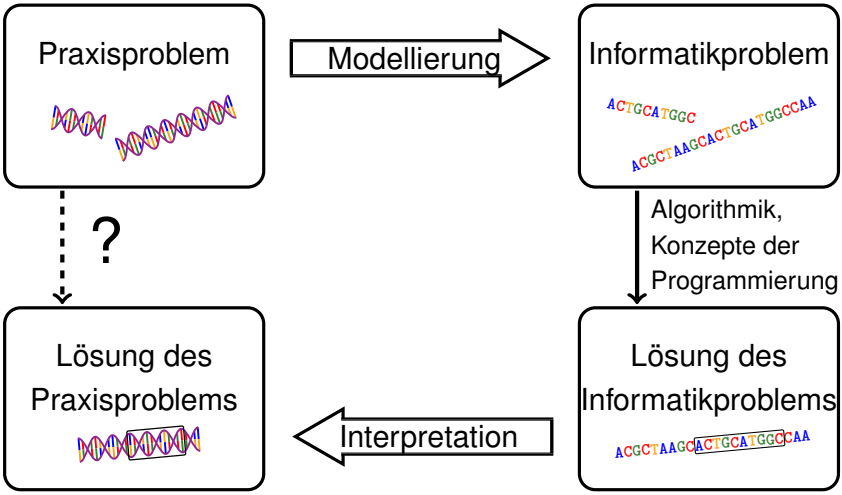
Einführung in die Vorlesung

Ziele

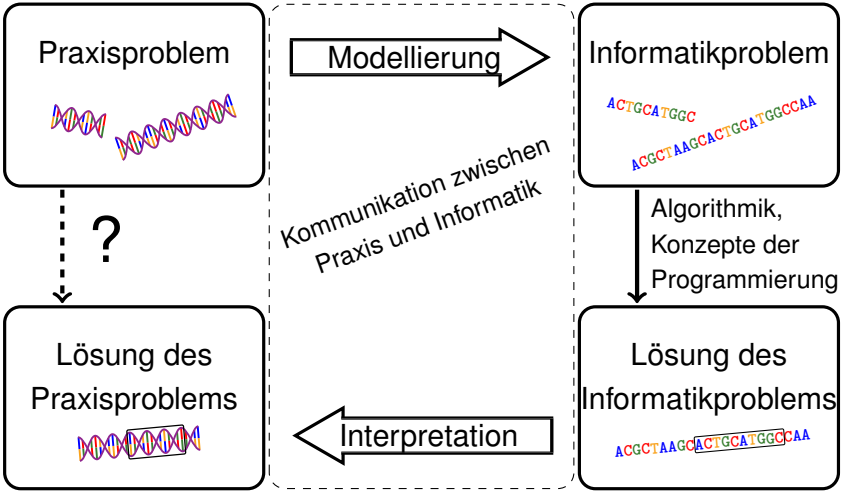
1. Informatik in der Naturwissenschaft



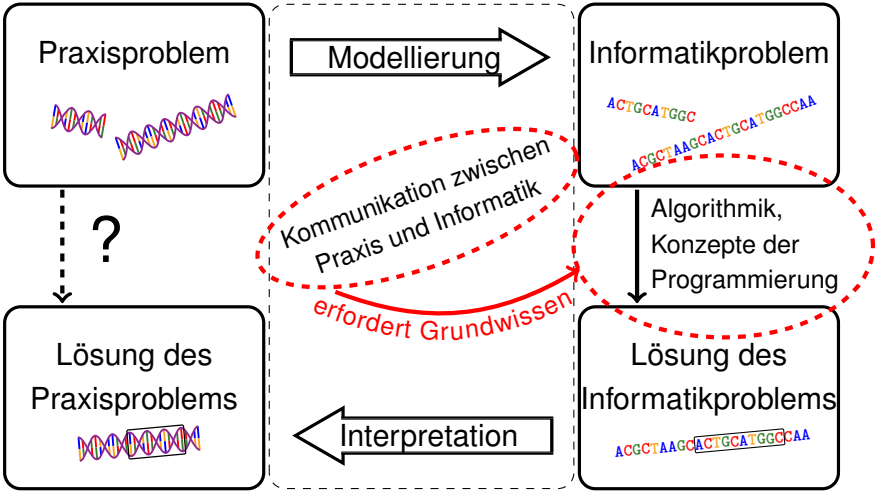
1. Informatik in der Naturwissenschaft



1. Informatik in der Naturwissenschaft



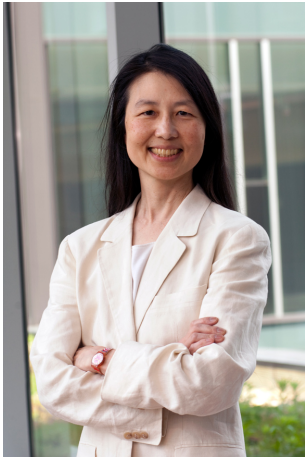
1. Informatik in der Naturwissenschaft



2. Algorithmisches Denken

- Systematisches Lösen gegebenener Probleme
- Dies beinhaltet Kreativität, Abstraktionsvermögen etc.
- Formulierung der Lösung als Algorithmus
- Lösung kann von einem Computer „verstanden“ werden

2. Algorithmisches Denken



Jeannette Wing

„Computational thinking is a way humans solve problems; it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting.“

3. Entwurfstechniken für Algorithmen

Die meisten Probleme in der Praxis haben einfache Lösungen

3. Entwurfstechniken für Algorithmen

Die meisten Probleme in der Praxis haben einfache Lösungen

- Oft einfach umzusetzen
- Basieren darauf, viele Möglichkeiten („Lösungskandidaten“) auszuprobieren
- Das bedeutet oft einen unpraktikablen Zeitaufwand

3. Entwurfstechniken für Algorithmen

Die meisten Probleme in der Praxis haben einfache Lösungen

- Oft einfach umzusetzen
- Basieren darauf, viele Möglichkeiten („Lösungskandidaten“) auszuprobieren
- Das bedeutet oft einen unpraktikablen Zeitaufwand

Viele Probleme haben „schnellere“ Lösungen

3. Entwurfstechniken für Algorithmen

Die meisten Probleme in der Praxis haben einfache Lösungen

- Oft einfach umzusetzen
- Basieren darauf, viele Möglichkeiten („Lösungskandidaten“) auszuprobieren
- Das bedeutet oft einen unpraktikablen Zeitaufwand

Viele Probleme haben „schnellere“ Lösungen

- Erfordern oft ein bisschen Geschick
- Verschiedene Techniken: **Greedy-Algorithmen**, **Divide-and-Conquer**, **Dynamische Programmierung** etc.

Einführung in die Vorlesung

Projekte

Projekte

Über das Semester bearbeiten Sie kleinere Projekte

Projekte

Über das Semester bearbeiten Sie kleinere Projekte

- Die Aufgabenstellungen werden bearbeitet via [code]expert

`https://expert.ethz.ch`

Projekte

Über das Semester bearbeiten Sie kleinere Projekte

- Die Aufgabenstellungen werden bearbeitet via [code]expert

  `https://expert.ethz.ch`

- Die Bearbeitung erfolgt selbstständig
- Die Übungsstunden dienen Ihnen, um Fragen zu stellen
- Präsentation der Lösungen via Zoom

Projekte

Die Projekte werden in den Übungen präsentiert

Projekte

Die Projekte werden in den Übungen präsentiert

- Abgabegespräch mit Assistierenden
- Teams von 2 Studierenden
- Bewertung durch Assistierende, Feedback durch Studierende

Projekte

Die Projekte werden in den Übungen präsentiert

- Abgabegespräch mit Assistierenden
- Teams von 2 Studierenden
- Bewertung durch Assistierende, Feedback durch Studierende
- **Abgabe obligatorisch**
- **Aber nicht benotet**

Projekte

Die Projekte werden in den Übungen präsentiert

- Abgabegespräch mit Assistierenden
- Teams von 2 Studierenden
- Bewertung durch Assistierende, Feedback durch Studierende
- **Abgabe obligatorisch**
- **Aber nicht benotet**
- [code]expert erlaubt, Lösungen vor der Abgabe selbst zu testen

Einführung in Python

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern vom Python-Programmtext

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern vom Python-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinensprache (bzw. Zwischencode)

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern vom Python-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinensprache (bzw. Zwischencode)
- **Computer:** Gerät zum Ausführen von Programmen in Maschinensprache

Was braucht es zum Programmieren?

- **Editor:** Programm zum Ändern, Erfassen und Speichern vom Python-Programmtext
- **Compiler:** Programm zum Übersetzen des Programmtexts in Maschinensprache (bzw. Zwischencode)
- **Computer:** Gerät zum Ausführen von Programmen in Maschinensprache
- **Betriebssystem:** Programm zur Organisation all dieser Abläufe (Dateiverwaltung, Editor-, Compiler- und Programmaufruf)

Deutsch vs. Programmiersprache

Deutsch

„Wissenschaft ist, was wir gut genug verstehen,
um es einem Computer zu erklären,
Kunst ist alles andere.“

DONALD KNUTH

Deutsch vs. Programmiersprache

Deutsch

„Wissenschaft ist, was wir gut genug verstehen,
um es einem Computer zu erklären,
Kunst ist alles andere.“

DONALD KNUTH

Python

```
# computation
b = a * a   # b = a**2
b = b * b   # b = a**4
```

Syntax und Semantik

- Programme müssen, wie unsere Sprache, nach gewissen Regeln geformt werden
  - **Syntax:** Zusammenfügungsregeln für elementare Zeichen (Buchstaben)
  - **Semantik:** Interpretationsregeln für zusammengefügte Zeichen

Syntax und Semantik

- Programme müssen, wie unsere Sprache, nach gewissen Regeln geformt werden
  - **Syntax:** Zusammenfügungsregeln für elementare Zeichen (Buchstaben)
  - **Semantik:** Interpretationsregeln für zusammengefügte Zeichen
- Entsprechende Regeln für ein Computerprogramm sind einfacher, aber auch strenger, denn Computer sind vergleichsweise dumm

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.

Syntaktisch und semantisch korrekt

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.
- DasAuto fuh r zu sxhnell.

Syntaktisch und semantisch korrekt

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.
  Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell.
  Syntaxfehler: Wortbildung

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.
- DasAuto fuh r zu sxhnell.
- Rot das Auto ist.

Syntaktisch und semantisch korrekt

Syntaxfehler: Wortbildung

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell. — Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell. — Syntaxfehler: Wortbildung
- Rot das Auto ist. — Syntaxfehler: Satzstellung

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.
- DasAuto fuh r zu sxhnell.
- Rot das Auto ist.
- Man empfiehlt dem Dozenten nicht zu widersprechen

Syntaktisch und semantisch korrekt

Syntaxfehler: Wortbildung

Syntaxfehler: Satzstellung

Fehlerarten illustriert an deutscher Sprache

Das Auto fuhr zu schnell.	Syntaktisch und semantisch korrekt
DasAuto fuh r zu sxhnell.	Syntaxfehler: Wortbildung
Rot das Auto ist.	Syntaxfehler: Satzstellung
Man empfiehlt dem Dozenten, nicht zu widersprechen.	Syntaxfehler: Satzzeichen fehlen

Fehlerarten illustriert an deutscher Sprache

Das Auto fuhr zu schnell.	Syntaktisch und semantisch korrekt
DasAuto fuh r zu sxhnell.	Syntaxfehler: Wortbildung
Rot das Auto ist.	Syntaxfehler: Satzstellung
Man empfiehlt dem Dozenten nicht zu widersprechen	Syntaxfehler: Satzzeichen fehlen
Sie ist nicht gross und rothaarig.	

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell. — Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell. — Syntaxfehler: Wortbildung
- Rot das Auto ist. — Syntaxfehler: Satzstellung
- Man empfiehlt dem Dozenten nicht zu widersprechen — Syntaxfehler: Satzzeichen fehlen
- Sie ist nicht gross und rothaarig. — Syntaktisch korrekt aber mehrdeutig [kein Analogon]

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell. — Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell. — Syntaxfehler: Wortbildung
- Rot das Auto ist. — Syntaxfehler: Satzstellung
- Man empfiehlt dem Dozenten nicht zu widersprechen — Syntaxfehler: Satzzeichen fehlen
- Sie ist nicht gross und rothaarig. — Syntaktisch korrekt aber mehrdeutig [kein Analogon]
- Die Auto ist rot.

Fehlerarten illustriert an deutscher Sprache

Das Auto fuhr zu schnell.	Syntaktisch und semantisch korrekt
DasAuto fuh r zu sxhnell.	Syntaxfehler: Wortbildung
Rot das Auto ist.	Syntaxfehler: Satzstellung
Man empfiehlt dem Dozenten nicht zu widersprechen	Syntaxfehler: Satzzeichen fehlen
Sie ist nicht gross und rothaarig.	Syntaktisch korrekt aber mehrdeutig [kein Analogon]
Die Auto ist rot.	Syntaktisch korrekt, doch grammatikalisch und semantisch fehlerhaft: Falscher Artikel [Typfehler]

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell.
  Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell.
  Syntaxfehler: Wortbildung
- Rot das Auto ist.
  Syntaxfehler: Satzstellung
- Man empfiehlt dem Dozenten nicht zu widersprechen
  Syntaxfehler: Satzzeichen fehlen
- Sie ist nicht gross und rothaarig.
  Syntaktisch korrekt aber mehrdeutig [kein Analogon]
- Die Auto ist rot.
  Syntaktisch korrekt, doch grammatikalisch und semantisch fehlerhaft: Falscher Artikel [Typfehler]
- Das Fahrrad galoppiert schnell.

Fehlerarten illustriert an deutscher Sprache

Das Auto fuhr zu schnell.	Syntaktisch und semantisch korrekt
DasAuto fuh r zu sxhnell.	Syntaxfehler: Wortbildung
Rot das Auto ist.	Syntaxfehler: Satzstellung
Man empfiehlt dem Dozenten nicht zu widersprechen	Syntaxfehler: Satzzeichen fehlen
Sie ist nicht gross und rothaarig.	Syntaktisch korrekt aber mehrdeutig [kein Analogon]
Die Auto ist rot.	Syntaktisch korrekt, doch grammatikalisch und semantisch fehlerhaft: Falscher Artikel [Typfehler]
Das Fahrrad galoppiert schnell.	Syntaktisch und grammatikalisch korrekt. Semantisch fehlerhaft [Laufzeitfehler]

Fehlerarten illustriert an deutscher Sprache

- Das Auto fuhr zu schnell. — Syntaktisch und semantisch korrekt
- DasAuto fuh r zu sxhnell. — Syntaxfehler: Wortbildung
- Rot das Auto ist. — Syntaxfehler: Satzstellung
- Man empfiehlt dem Dozenten nicht zu widersprechen — Syntaxfehler: Satzzeichen fehlen
- Sie ist nicht gross und rothaarig. — Syntaktisch korrekt aber mehrdeutig [kein Analogon]
- Die Auto ist rot. — Syntaktisch korrekt, doch grammatikalisch und semantisch fehlerhaft: Falscher Artikel [Typfehler]
- Das Fahrrad galoppiert schnell. — Syntaktisch und grammatikalisch korrekt. Semantisch fehlerhaft [Laufzeitfehler]
- Manche Tiere riechen gut.

Fehlerarten illustriert an deutscher Sprache

Das Auto fuhr zu schnell.	Syntaktisch und semantisch korrekt
DasAuto fuh r zu sxhnell.	Syntaxfehler: Wortbildung
Rot das Auto ist.	Syntaxfehler: Satzstellung
Man empfiehlt dem Dozenten nicht zu widersprechen	Syntaxfehler: Satzzeichen fehlen
Sie ist nicht gross und rothaarig.	Syntaktisch korrekt aber mehrdeutig [kein Analogon]
Die Auto ist rot.	Syntaktisch korrekt, doch grammatikalisch und semantisch fehlerhaft: Falscher Artikel [Typfehler]
Das Fahrrad galoppiert schnell.	Syntaktisch und grammatikalisch korrekt. Semantisch fehlerhaft [Laufzeitfehler]
Manche Tiere riechen gut.	Syntaktisch und semantisch korrekt. Semantisch mehrdeutig [kein Analogon]

Einführung in Python

Verwendete Software

Verwendete Software

- Es gibt unzählige Python-Entwicklungsumgebungen (IDEs)
- Diese beinhalten einen Editor und diverse Tools

Verwendete Software

- Es gibt unzählige Python-Entwicklungsumgebungen (IDEs)
- Diese beinhalten einen Editor und diverse Tools
- Wir verwenden **[code]expert**

```
https://expert.ethz.ch/enroll/SS21/ppl
```

Verwendete Software

- Es gibt unzählige Python-Entwicklungsumgebungen (IDEs)
- Diese beinhalten einen Editor und diverse Tools
- Wir verwenden **[code]expert**

```
https://expert.ethz.ch/enroll/SS21/ppl
```

- Ausserdem empfohlen (offline): **PyCharm Education**

```
https://www.jetbrains.com/pycharm-educational/download/
```

- Laden Sie die Community-Edition herunter

Einführung in Python

Ein erstes Python-Programm

Ein erstes Python-Programm

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x    # y ist das Quadrat von x
print("Der Wert von y ist", y)
z = y * y    # z ist das Quadrat von y
print("Der Wert von z ist", x * x * x * x)
```

Verhalten eines Programms

Zur Kompilationszeit

- Vom Compiler akzeptiertes Programm (syntaktisch korrektes Python)
- Compiler-Fehler

Verhalten eines Programms

Zur Kompilationszeit

- Vom Compiler akzeptiertes Programm (syntaktisch korrektes Python)
- Compiler-Fehler

Zur Laufzeit

- korrektes Resultat
- inkorrektes Resultat
- Programmabsturz
- Programm **terminiert** nicht (Endlosschleife)

Kommentare

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x    # y ist das Quadrat von x
print("Der Wert von y ist", y)
z = y * y    # z ist das Quadrat von y
print("Der Wert von z ist", x * x * x * x)
```

Kommentare

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x    # y ist das Quadrat von x
print("Der Wert von y ist", y)
z = y * y    # z ist das Quadrat von y
print("Der Wert von z ist", x * x * x * x)
```

Kommentare und Layout

Kommentare

- hat jedes gute Programm
- dokumentieren, **was** das Programm **wie** macht und wie man es verwendet
- werden vom Compiler ignoriert
- Syntax: # bis Zeilenende

Kommentare und Layout

Kommentare

- hat jedes gute Programm
- dokumentieren, **was** das Programm **wie** macht und wie man es verwendet
- werden vom Compiler ignoriert
- Syntax: # bis Zeilenende

Bitte beachten

- Leerzeilen werden ignoriert
- Python gibt Einrückungen vor, die die Programmlogik widerspiegeln

Einführung in Python

Anweisungen

Anweisungen (Statements)

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```

Anweisungen (Statements)

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```

Anweisungen (Statements)

Anweisungen

- sind Bausteine eines Python-Programms
- werden (sequenziell) **ausgeführt**
- stehen in einer Zeile

Anweisungen (Statements)

Anweisungen

- sind Bausteine eines Python-Programms
- werden (sequenziell) **ausgeführt**
- stehen in einer Zeile

Jede Anweisung hat (potenziell) einen **Effekt**

Anweisungen – Werte und Effekte

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```

Anweisungen – Werte und Effekte

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```

Effekt: Ausgabe des Strings `Dies ist...`

Effekt: Variable `x` wird erzeugt und erhält Wert 20

Einführung in Python

Variablen

Fundamentaltypen

Variablen repräsentieren (wechselnde) Werte

- Ganze Zahlen (Integers)
- Reelle Zahlen (Floats)
- Zeichenketten (Strings)
- ...

Fundamentaltypen

Variablen repräsentieren (wechselnde) Werte

- Ganze Zahlen (Integers)
- Reelle Zahlen (Floats)
- Zeichenketten (Strings)
- ...

Anders als beispielsweise in **Java** oder **C** wird der Typ nicht explizit angegeben, wenn die Variable deklariert (zum ersten Mal verwendet) wird

Einführung in Python

Ausdrücke

Ausdrücke (Expressions)

Ausdrücke

- repräsentieren **Berechnungen**

Ausdrücke (Expressions)

Ausdrücke

- repräsentieren **Berechnungen**
- sind entweder **primär** (`x`)
- oder **zusammengesetzt** (`x * x`)

Ausdrücke (Expressions)

Ausdrücke

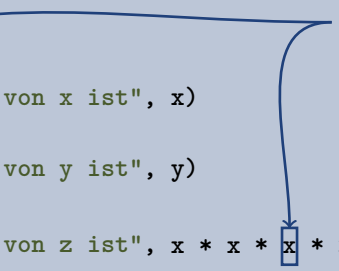
- repräsentieren **Berechnungen**
- sind entweder **primär** (`x`)
- oder **zusammengesetzt** (`x * x`)
- . . . aus anderen Ausdrücken, mit Hilfe von **Operatoren**
- . . . und Klammern

Ausdrücke (Expressions)

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x )
```

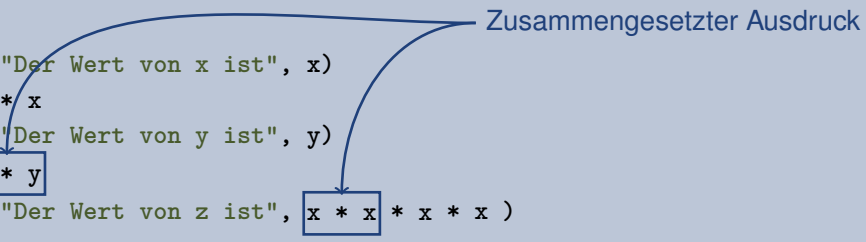
Ausdrücke (Expressions)

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x )
```



Ausdrücke (Expressions)

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x )
```



Ausdrücke (Expressions)

- repräsentieren **Berechnungen**
- sind **primär** oder **zusammengesetzt**
  (aus anderen Ausdrücken und Operationen)

Beispiel

`a * a` ist zusammengesetzt aus
Variablenname, Operatorsymbol, Variablenname
Variablenname: primärer Ausdruck

- können geklammert werden

`a * a` kann geschrieben werden als `(a * a)`

Einführung in Python

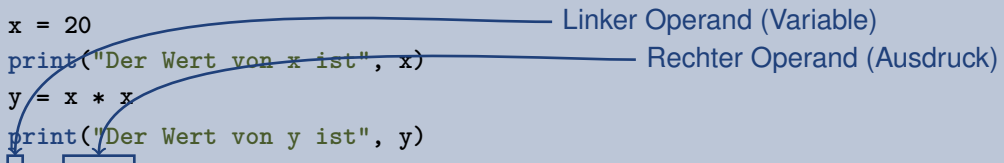
Operatoren und Operanden

Operatoren und Operanden

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```

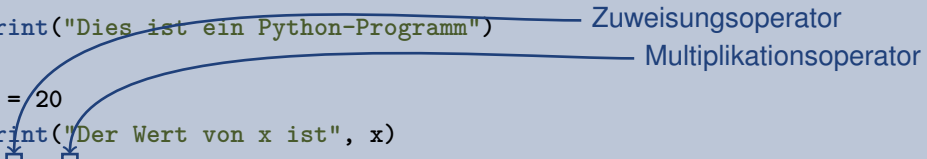
Operatoren und Operanden

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```



Operatoren und Operanden

```
print("Dies ist ein Python-Programm")

x = 20
print("Der Wert von x ist", x)
y = x * x
print("Der Wert von y ist", y)
z = y * y
print("Der Wert von z ist", x * x * x * x)
```



Operatoren

Operatoren

- machen aus Ausdrücken (**Operanden**) neue zusammengesetzte Ausdrücke
- haben eine Stelligkeit

Beispiel (Multiplikation) `a * a`

Operand `a`, Operator `*` , Operand `a`

Multiplikationsoperator $*$

Multiplikationsoperator

- erwartet zwei Werte vom gleichen Typ als Operanden (Stelligkeit 2)
- „gibt Produkt als Wert des gleichen Typs zurück“, das heisst formal:

> Der zusammengesetzte Ausdruck repräsentiert den Wert des Produktes der Werte der beiden Operanden

Beispiele

- `a * a`
- `b * b`

Zuweisungsoperator =

- Weist linkem Operanden den Wert des rechten Operanden zu und gibt den linken Operanden zurück

Beispiele

- `b = b * b`
- `a = b`

Zuweisungsoperator =

- Weist linkem Operanden den Wert des rechten Operanden zu und gibt den linken Operanden zurück

Beispiele

- `b = b * b`
- `a = b`

Achtung

Der Operator „=“ entspricht dem Zuweisungsoperator in der Mathematik ($:=$), nicht dem Vergleichsoperator ($=$)

Übung – Celsius-zu-Fahrenheit-Rechner

Schreiben Sie ein Programm, das

- eine ganze Zahl (wie z. B. 31) als Temperatur in Grad Celsius interpretiert
- dieselbe Temperatur in Grad Fahrenheit ausgibt
- dafür folgende Formel verwendet:

$$\text{fahrenheit} = \frac{9 \cdot \text{celsius}}{5} + 32$$

Übung – Celsius-zu-Fahrenheit-Rechner

```
celsius = 31
fahrenheit = 9 * celsius / 5 + 32
print(fahrenheit)
```

Danke für die Aufmerksamkeit