POS1

Dr. Günter Kolousek

13. September 2015

Inhaltsverzeichnis

1 Einheit 1

1.1 Begriffe - 1

Informatik (engl. computer science, informatics)

- Informatik ist die Wissenschaft und Anwendung von der systematischen, zumeist maschinell unterstützten Verarbeitung, Speicherung und Übermittlung von Information.
- Der Begriff "Informatik" entstand in den 60er-Jahren (des vorigen Jh.) als Zusammenfassung der Worte "Information" und "Automatik".
- Informatik vermittelt auf der Basis von Mathematik und Elektrotechnik die theoretischen Grundlagen der Datenverarbeitung.

Information (engl. information)

Information ist eine **Sequenz** (Folge) von Symbolen.

Symbol (engl. symbol)

Symbol ist ein Objekt, ein Bild, geschriebenes Wort, Klang,... Z.B. Vorrang geben, Stoppschild, "Top Secret", Klingelton (Anruf, SMS),...

Kodierung (engl. encoding)

ist eine Vorschrift wie Information in Zeichen abgebildet werden kann.

Zeichenkodierung (engl. character encoding)

- Bei der Zeichenkodierung werden Buchstaben, Zahlen und Satzzeichen (also die Symbole, die die Information repräsentieren) auf Bitfolgen (wiederum Zeichen!) abgebildet, um diese im Computer zu speichern, weiterzuverarbeiten bzw. ausgeben zu können.
- Spezielle Kodierungen für Textdaten (Verwendung im Texteditor!)
  - ASCII (American Standard Code for Information Interchange)
  - ISO-8859 (International Standards Organization)
    - ISO-8859-1 (latin-1): westeuropäische Zeichen
    - ISO-8859-15 (latin-9): inkl. Euro-Zeichen
  - UTF-8 (Unicode)

Daten (singular Datum, engl. data)

- analoge Daten (engl. analog data)

Analoge Daten werden durch kontinuierliche Funktionen repräsentiert, die oft den Verlauf physikalischer Größen wiedergeben. Derartige Funktionen ändern sich stufenlos.

Beispiele: Thermometer bei denen durch die Höhe der Quecksilbersäule Temperaturwerte gekennzeichnet werden, Uhren (mittels Zeiger), Musik (Speicherung auf Schallplatten)

- digitale Daten (engl. digital data)

  In der Informatik und Datenverarbeitung versteht man (digitale) Daten als (maschinen-, d.h. Computer-) lesbare und -bearbeitbare, in der Regel digitale Repräsentation von Information.

  Die Information wird dazu meist zunächst in Zeichen bzw. Zeichenketten (Folge von Zeichen, engl. string) kodiert, deren Aufbau strengen Regeln folgt, der so genannten Syntax.

Formatierte Daten (engl. formatted data)

Formatierte Daten sind digitale Daten, die in einem für die maschinelle Interpretation besonders geeigneten, fest vereinbarten Aufbau aufgezeichnet sind.

Übliche Formate:

- DOCX (winword),… RTF (rich text format)
- PDF (portable document format), PS (postscript)
- PNG (portable network graphics), GIF (graphic interchange format), JPEG (Joint Photographic Experts Group)

Datei (engl. file)

In einem Computersystem werden Daten innerhalb von Dateien gespeichert. Diese werden auf einem Speichermedium wie z.B. der Festplatte, einer Diskette oder einem USB-Stick abgelegt.

Verzeichnis (Ordner, engl. directory, folder)

Dateien können in Verzeichnissen zusammengefasst und abgelegt werden. Ein Verzeichnis kann sowohl Dateien als auch wiederum Verzeichnisse enthalten. Jedes Verzeichnis hat genau ein Elternverzeichnis, außer dem Wurzelverzeichnis (engl. root directory).

1.2 Informationseinheiten

Bit (von engl. binary digit)

Ein Zeichen, das entweder 0 oder 1 sein kann. D.h. das Alphabet der binären (zweiwertig) Zeichen besteht aus 2 Elementen.

Kann verwendet werden, um

- ”Schalter offen” / ”Schalter geschlossen”
- ”Strom fließt” / ”kein Strom” bzw. ”Spannung vorhanden” / ”keine Spannung”
- ”ja” / ”nein” bzw. ”wahr” (engl. true) / ”falsch” (engl. false)

darzustellen.

Byte (von engl. ”bite”)

Ein Happen von Bits. *Heute* meinst 8 Bit.

Oktett (engl. octett)

8 Bit

SI-Präfixe

- Kilo … 10^3, z.B. 2 kB (auch KB) oder z.B. 8 kbit
- Mega … 10^6, z.B. 10 MB oder z.B. 80 Mbit
- Giga … 10^9, z.B. 1 GB oder 8 Gbit
- Tera … 10^{12}
- Peta … 10^{15}
- Exa … 10^{18}

Binärpräfixe

- Kibi … 2^{10}, z.B. 10 KiB oder 80 Kibit
- Mebi … 2^{20}, z.B. 5 MiB
- Gibi … 2^{30}, z.B. 2 GiB
- Tebi … 2^{40}, z.B. 1 TiB
- Pebi … 2^{50}, z.B. 1 PiB
- Exbi … 2^{60}, z.B. 1 EiB

Hersteller von Massenspeichermedien verwenden die SI-Präfixe! Daher zeigen die Programme, die die Binärpräfixe verwenden wie der Windows Explorer niedrigere Werte an!

1.3 Begriffe - 2

Computer (von engl. to compute)

Rechner, d.h. die Hardware (HW), d.h. die materiellen Komponenten (Teile) eines Informationsverarbeitungs- systems.

Software (SW)

die nicht materiellen Komponenten eines Informationsverarbeitungssystems: Daten und Programme

Algorithmus (engl. algorithm)

Eindeutige Beschreibung eines Problemlösungsverfahren. Beispiel: Kaffee kochen.

Programm (engl. program)

ist eine Software, die Anweisungen enthält. Umsetzung eines Algorithmus (oder mehrerer Algorithmen)

- Vor dem Übersetzen:
  - in einer Programmiersprache formulierte Algorithmen
  - Sequenz von Anweisungen in einer Programmiersprache
- Nach dem Übersetzen: Sequenz von Instruktionen in einer Maschinensprache

Maschinensprache (engl. machine language)

Anweisungen, die der Computer d.h. der (Mikro)prozessor direkt versteht. Jede Anweisung besteht jeweils aus einer Folge von 0 und 1.

Programmiersprache (engl. programming language)

Spezielle formale Sprache, deren Zweck es ist, einem Computer Anweisungen zu geben.

Beispiele:

- Assembler
- C, C++
- Python, Java, C#, Ruby,...
- Smalltalk, Lisp, Prolog, Haskell,...

Syntax

Unter der Syntax versteht man ein System von Regeln, nach denen erlaubte Konstruktionen bzw. wohlgeformte Ausdrücke aus einem grundlegenden Zeichenvorrat (dem Alphabet) gebildet werden.

Ist eine der Regeln verletzt, dann wird ein Syntaxfehler erkannt.

Compiler (dt. Übersetzer, Kompilierer)

Ein Compiler ist ein Computerprogramm, das ein in einer Quellsprache geschriebenes Programm in ein semantisch äquivalentes Programm einer Zielsprache (meist Assembler oder Maschinensprache) umwandelt.

Interpreter (von engl. to interprete)

Ein Interpreter ist ein Computerprogramm, das einen Programm-Quellcode den Quellcode einliest, analysiert und ausführt. Die Analyse des Quellcodes erfolgt also zur Laufzeit des Programms.

interaktiver Interpreter

- interaktiv, d.h. direkte Kommunikation mit Benutzer
- Intpreter
- Prompt
- REPL (read-evaluate-print-loop)

Eine Form der Interaktion mit dem Computer in Programmierumgebungen. Python (und andere Programmiersprachen) verwendet diese Form eines interaktiven Interpreters:

1. Read: Ein Ausdruck wird gelesen
2. Evaluate: Der Ausdruck wird ausgewertet
3. Print: Das Ergebnis wird ausgegeben
4. Loop: Beginne wieder von vorne

Shell (dt. Schale)

Programm, das eine Schicht um eine andere Software darstellt. Meist eine Schicht zum Betriebssystem oder auch zu einem Interpreter. Gestartete Programme (Prozess) können mittels `CTRL-C` unterbrochen werden.

Prozess (engl. process)

Ein gestartetes Programm.

Texteditor (kurz Editor)

Ein Programm zum Bearbeiten von Texten. Der Text wird in einer Datei abgespeichert bzw. aus einer Datei gelesen. Eine Datei ist in einem Verzeichnis gespeichert.

IDE (integrated developement environment)

Integrierte Entwicklungsumgebung. Dabei handelt es sich um ein Programm, das meist aus den folgenden Komponenten besteht: Texteditor, Compiler bzw. Interpreter und Debugger.

EVA Prinzip Eingabe-Verarbeitung-Ausgabe

1.4 Mathematische Grundlagen der Informatik

Addition Summand + Summand = Summe

- Beispiele!
- kommutativ, assoziativ, distributiv mit Multiplikation

Subtraktion Minuend − Subtrahend = Differenz

- Beispiele!
- **nicht** kommutativ, **nicht** distributiv, distributiv mit Multiplikation
- Subtraktion auf Addition zurückführen mit Gegenzahl

Multiplikation Faktor · Faktor = Produkt

- Beispiele!
- kommutativ, assoziativ, distributiv mit Addition und Subtraktion

Division $\frac{\text{Dividend}}{\text{Divisor}} = \text{Quotient}$

- Beispiele!
- **nicht** kommutativ, **nicht** distributiv, distributiv mit Addition und Subtraktion
- Division auf Multiplikation mit Kehrwert zurückzuführen
- Divisor = 0!

Potenz und Wurzel $\text{Basis}^{\text{Exponent}} = \text{Potenz}$

- Beispiele!
- **nicht** kommutativ, **nicht** assoziativ
- $a^n = a \cdot a \cdot \ldots a$ mit n Faktoren, $a^0 = 1$
- $a^{n^m} = a^{(n^m)} \neq (a^n)^m$
- $a^{-n} = \frac{1}{a^n}, a \neq 0$
- $0^q = 0, q > 0$
- $\sqrt[n]{a^n} = a$

- $a^{\frac{1}{n}} = \sqrt[n]{a}$
- $a^{\frac{m}{n}} = a^{\frac{1}{n}m} = \sqrt[n]{a}^m$

Betrag (absoluter Betrag, Absolutwert, engl. absolute value) Betrag einer Zahl ist Abstand (Distanz) zur Zahl 0.

1.5 Programmier-Know-How

Wichtige Zeichen

`.`, `:`, `,`, `;`, `|`, `_`, `=`, `{`, `}`, `[`, `]`, `(`, `)`, `+`, `-`, `*`, `/`, `%`, `\`, `#`, `"`, `'`, `<`, `>`, `&`, `~`, `^`, `!`, `@`

Wichtige Tasten

`<return>`, `<space>`, `<backspace>`, `<tab>`, `<esc>`, `<strg>` (oder `<ctrl>`), `<alt>`, `<altgr>`, `<del>` (oder `<entf>`)

Ausdruck (engl. expression)

Ein Konstrukt einer Programmiersprache, das ausgewertet (engl. evaluate) werden kann und einen Wert liefert.

Im interaktiven Modus des Python-Interpreter wird bei jedem Ausdruck das ausgewertete Ergebnis ausgegeben.

Ganze Zahl (engl. integer)

Zahl ohne Nachkommastellen.

Gleitkommazahl (auch Fließkommazahl, engl. floating point number)

Zahl mit Nachkommastellen, z.B. `3.1415`. Ist die annäherungsweise Darstellung einer reellen Zahl. Beachte: Das Kommazeichen ist ein Punkt!

Operator (engl. operator)

Für die arithmetischen Rechenoperationen gibt es in Python die folgenden Operatoren `+`, `-`, `*` , `/`, `**` , `%` und `//` zusammensetzt, die jeweils links und rechts einen Operanden haben müssen: `2 + 3`

Alle wichtigen Programmiersprachen kennen natürlich auch die Vorrangregeln (d.h. Punkt- vor Strichrechnung): `2 + 3 * 4`

`**` bezeichnet den Potenzierungsoperator. `%` und `//` werden wir noch später kennenlernen.

All diese Operatoren werden auch binäre (oder zweistellige) Operatoren genannt, da diese jeweils 2 Operanden benötigen.

`+` und `-` können jedoch auch als Vorzeichen verwendet werden, womit diese auch als unäre (oder einstellige) Operatoren verwendet werden können: `-3` oder `+5`.

Natürlich kann man auch beide Arten in einem Ausdruck kombinieren: `5 + -3 -4`

D.h. es gibt hauptsächlich zwei Arten von Operatoren:

- unärer Operator (engl. unary operator): wirkt auf einen Operanden
- binärer Operator (engl. binary operator): verknüpft zwei Operanden

Klammernsetzung Wie in der Mathematik üblich, können in arithmetischen Ausdrücken auch runde Klammern (engl. parentheses) gesetzt werden: `(2 + 3) * 4`

Funktion (engl. function)

- Ähnlich wie in der Mathematik
- Funktion hat einen Namen
- Funktion kann "aufgerufen" (engl. to call, to invoke) werden: `sqrt(4)`
  - Argumente innerhalb von runden Klammern → 4
  - Ergebnis wird als Rückgabewert (engl. return value) zurückgeliefert → 2.0
  - Der Aufruf einer Funktion ist ein Ausdruck!

1.6 Grundformeln der Bewegungslehre

2 Bewegungen der Bewegungslehre (Kinematik):

- Gleichförmige Bewegung (Geschwindigkeit konstant):

  $s = vt$
  $a = 0$

- Gleichmäßig beschleunigte Bewegung (Beschleunigung konstant):

  $s = a/2 \cdot t^2$
  $v = at$

1.7 Dateisystemgrundlagen

- Partition
  - Einteilung eines Datenträgers wie z.B. einer Festplatte
  - unter Windows: Geräte (engl. device), z.B. `C:`
- Datei (engl. file)
  - Datendatei

  - ausführbare Datei (engl. executable, Programm)
- Verzeichnis (engl. directory, folder)
  - kann Dateien oder Verzeichnisse beinhalten
- Pfad, Pfadname
  - "Weg" zu einer Datei oder einem Verzeichnis
  - Trennzeichen `/` (oder in Windows auch/hauptsächlich \)
  - absoluter Pfad
  - relativer Pfad
- Arbeitsverzeichnis (engl. working directory)
  - `cp ../text.txt .`
- Home-Verzeichnis, z.B. `cd ~`
- Eltern-Verzeichnis (engl. parent directory)

1.8 Powershell

man manual page

cd change directory

pwd print working directory

ls list directory

mkdir make directory

rmdir remove directory

cat catenate (dt. verketten)

clear clear the screen

cp copy file or directory

rm remove file or directory

mv move file or directory

2 Einheit 2

2.1 Turtlegrafik

Funktionen: `left`, `right`, `forward`, `back`, `undo`, `reset`, `pensize`, `pencolor`, `fillcolor`, `begin_fill`, `end_fill`, `penup`, `pendown`, `home`, `clear`, `circle`, `hideturtle`, `showturtle`, `shape`

2.2 Geometrische Figuren

- Umfang (U), Fläche (A), Seiten (a, b, c), Radius (r)
- Winkel
  - positiv/negativ: gegen/mit Uhrzeigersinng
- Rechteck

  $$\begin{aligned} U &= 2(a+b) \\ A &= ab \end{aligned}$$

- Quadrat

  $$\begin{aligned} U &= 4a \\ A &= a^2 \end{aligned}$$

- Rechtwinkeliges Dreieck

  Katheten: a, b; Hypothenuse: c

  $$\begin{aligned} c^2 &= a^2 + b^2 \\ U &= a+b+c \\ A &= (ab)/2 \end{aligned}$$

- Kreis

  $$\begin{aligned} U &= 2r\pi \\ A &= r^2\pi \end{aligned}$$

  Im Modul `math` befindet sich auch die Zahl `pi`. D.h. nach Importieren mittels `from math import pi` kann direkt auf `pi` zugegriffen werden:

```
>>> from math import pi
>>> pi
3.141592653589793
```

2.3 Mathematische Grundlagen der Informatik

- Ziffernsumme

  Ist die Summe aller Ziffern einer Zahl.

  Beispiel: Die Ziffernsumme von 123 ist 6.

- Arithmetisches Mittel (Mittelwert) einer Folge von Zahlen

  Ist die Summe aller Zahlen dividiert durch die Anzahl der Zahlen.

  Beispiel: Das arithmetische Mittel der Zahlen 4, 2, 3 ist 3.

- Zahlensysteme und Potenzmethode

  Stellenwertsysteme sind Systeme, die Zahlen durch Ziffern bilden, die in Abhängigkeit ihrer Position (Stelle) eine andere Bedeutung besitzen. Das Dezimalsystem ist das bekannteste Stellenwertsystem, die Informatik benötigt jedoch das Binärsystem, das Oktalsystem und das Hexadezimalsystem.

  - Dezimalsystem
    * Basis 10
    * Ziffern 0 bis 9
    * Beispiel 1: $2222 = 2 \cdot 1000 + 2 \cdot 100 + 2 \cdot 10 + 2 \cdot 1$
    * Beispiel 2: $123.45 = 100 + 20 + 3 + 0.4 + 0.05 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}$

    Beachte:

    * $10^0 = 1$
    * $10^{-1} = 0.1$
    * $10^{-2} = 0.01$

  - Binärsystem
    * Basis 2
    * Ziffern 0 und 1
    * Mittels der Potenzmethode kann in das Dezimalsystem umgerechnet werden.
    * Beispiel 1: $1001_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 1 = 9_{10}$
    * Beispiel 2: $110.01_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} = 4 + 2 + 0.25 = 6.25_{10}$

    Beachte:

    * $2^0 = 1$
    * $2^{-1} = 0.5$
    * $2^{-2} = 0.25$
  - Oktalsystem
    * Basis 8
    * Ziffern 0 bis 7
    * Beispiel: $123_8 = 83_{10}$
  - Hexadezimalsystem
    * Basis 16
    * Ziffern 0 bis 9 sowie die Buchstaben A, B, C, D, E, F
    * Beispiel: $FF_{16} = 255_{10}$

2.4 Programmierrichtlinien

(engl. coding conventions)

Die meisten Programmiersprachen haben ihre eigenen speziellen Richtlinien wie der Programmcode zu formatieren (anzuschreiben) ist.

In Python lauten einige der wichtigsten Regeln folgendermaßen:

- Maximale Zeilenlänge von 79 Zeichen!
- Importanweisungen am Anfang der Datei, jeweils in einer eigenen Zeile.
- Genau ein Leerzeichen um binären Operator!
- Keine Leerzeichen unmittelbar innerhalb von runden, eckigen oder geschweiften Klammern.
- Keine Leerzeichen von runder Klammer bei Funktionen
- Keine Leerzeichen vor Komma, Strichpunkt oder Punkt!
- genau eine einfache Anweisung je Zeile.
- Modulnamen (Dateinamen) zur Gänze in Kleinbuchstaben.

2.5 Programmier-Know-How

Anweisung (engl. statement)

- Ausdruck hat Wert

  - Ausgabe des Wertes im interaktiven Interpreter!
- Anweisung bewirkt etwas (kein Wert!)

Whitespace (dt. Leerraum)

- mittels der Leertaste (`<space>`)
- `TAB` mittels der Tabulatortaste (`<tab>`)
- `CR` (carriage return)
- `LF` (engl. line feed)

Achtung: eine neue Zeile (also mittels Drücken der `<return>`-Taste) wird in den verschiedenen Betriebssystemen verschieden umgesetzt:

- Windows: `CR LF`
- Unix, Linux, MacOSX: `LF`
- MacOS (alt): `CR`

Python verwendet intern immer `LF`!

Funktion (engl. function)

liefert einen Wert zurück, z.B. die Quadratwurzel von 4 wird in der Mathematik $\sqrt{4}$ angeschrieben und in Python so:

```
>>> sqrt(4)
2.0
```

In der Mathematik schreibt man z.B. $\min(3, 1, 4)$ und in Python wiederum::

```
>>> min(3, 1, 4)
1
```

Eine Funktion kann einen oder mehrere **Argumente** haben. In dem letzten Beispiel gibt es ein Argument nämlich die ganze Zahl 0 und liefert beim Funktionsaufruf (engl. to invoke a function) einen Wert zurück. Das nennt man den **Rückgabewert**.

Mathematik Stellt eine Abbildung zwischen einem Wertebereich und einem Bildbereich dar. D.h. jedem Wert aus dem Wertebereich (also im zweidimensionalen Koordinatensystem einem x-Wert) wird ein Wert aus dem Bildbereich (im zweidimensionalen Koordinatensystem einem y-Wert) zugeordnet.

Programmieren Die Funktionsdefinition besteht aus dem Funktionsnamen, einer Liste von (formalen) Parametern und dem Funktionsrumpf.

Beim Funktionsaufruf (engl. function invocation) werden die Argumente (auch aktuelle Parameter genannt) übergeben, der Funktionsrumpf ausgeführt und der Returnwert zurückgegeben.

Funktion vs. Prozedur (engl. procedure) (klassische Definition):

- Prozedur hat keinen Rückgabewert (bzw. in Python liefert diese `None` zurück)
- Funktion hat einen Rückgabewert

Modul (engl. module)

Ist eine abgeschlossene Einheit einer Software, bestehend aus Code und Daten.

In Python ist jede Datei mit der Endung `.py` ein Modul.

Importieren (engl. to import)

Importieren bedeutet zugreifbar machen.

Anweisung (engl. statement)

Eine Anweisung ist ein ausführbarer Befehl in einer Programmiersprache. Eine Folge von Anweisungen formt ein Programm. Innerhalb von Anweisungen können Ausdrücke oder auch wieder Anweisungen vorkommen.

Anweisungen haben entweder keinen Wert oder dieser wird nicht verwendet. Ein Funktionsaufruf liefert an sich einen Wert zurück, wird dieser jedoch alleine angeschrieben, dann handelt es sich um eine Aufrufanweisung (call statement):

```
sqrt(2)
```

Innerhalb einer anderen call-Anweisung handelt es sich allerdings lediglich um einen Ausdruck::

```
print(sqrt(2))
```

Ein anderes Beispiel einer Anweisung in Python (die keinen Wert hat): `from .. import ..`

Anweisungen haben entweder keinen Wert oder dieser wird nicht verwendet:

- In Python haben Anweisungen wie die `if`-Anweisung oder die Zuweisungsanweisung keinen Wert (weiters u.a. auch `for`, `try`, `while`, die später noch beschrieben werden).
- In Python hat die Ausdrucksanweisung (engl. expression statement) zwar einen Wert. Dieser wird aber nicht verwendet. Beispiele:

  ```
  print("abc")  # Ausgabe
  compute_results(1, 2, 3)  # beliebiger Funktionsaufruf
  ```

Einfache Anweisungen können in einer Zeile stehen und werden mittels Strichpunkt getrennt:

```
>>> print("abc"); print("abc")
abc
abc
```

Prozess (engl. process)

- Ein Programm liegt meist als Datei auf der Festplatte.
- Wird ein Programm gestartet, entsteht ein Prozess.
- Man kann ein Programm öfters starten. Damit entstehen mehrere Prozesse.

Standard(ein/aus)gabe (`stdin` / `stdout`)

Jedem Prozess ist sowohl eine Standardeingabe und eine Standardausgabe zugeordnet.

Von der Standardeingabe kann gelesen werden (z.B. mittels `input`) und auf die Standardausgabe kann (z.B. mittels `print`) geschrieben werden.

Zeichenkette (engl. string)

Eine Folge (Sequenz) von Textzeichen wird String genannt.

Escape character (dt. Maskierungszeichen, escape: Flucht)

ist ein Zeichen, das verhindert, dass das folgende Zeichen seine normale Bedeutung verliert. In Python-Strings (auch C, Java, C#) ist das das Backslash-Zeichen mit den folgenden gebräuchlichen Sequenzen:

Zeichen	Bedeutung
`\'`	einfaches Anführungszeichen
`\"`	doppeltes Anführungszeichen
`\\`	Backslash
`\n`	New line
`\r`	Carriage return

Kommentar (engl. comment)

- Wird ignoriert
- dient zur Dokumentation
- Offensichtliches ist nicht zu dokumentieren

Case sensitivity (dt. wörtlich: Fach Abhängigkeit)

Dieser Begriff bedeutet, dass Groß- und Kleinschreibung betrachtet wird. Es gibt zwei verschiedene Ausprägungen:

- case-sensitive: Groß/Kleinschreibung ist relevant
- case-insensitive: Groß/Kleinschreibung ist irrelevant

Keyword (dt. Schlüsselwort, reserviertes Wort)

Für jede Programmiersprache sind Keywords definiert, die eine gewisse Bedeutung haben und ansonsten nicht verwendet werden dürfen. Beispiele: `from` und `import`

3 Einheit 3

3.1 Begriffe

Muster (engl. pattern oder design pattern)

Muster sind bewährte Lösungs-Schablonen für Probleme.

Objekt (engl. object)

- In Python ist alles ein Objekt
- Python überprüft **immer** den Typ eines Objektes, z.B.::

```
>>> 3 + "a"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

- Der Typ eines Objektes lässt sich mit “type“ feststellen
- Jedes Objekt hat eine Darstellung, wenn man nur den Namen dieses Objektes im interaktiven Interpreter eingibt.
- Jedes Objekt hat
  - einen Typ
  - eine eindeutige Identität
  - einen aktuellen Zustand
  - ein Verhalten

Name (engl. name)

- Der Name ist ein Bezeichner (engl. identifier)
- Ein Name wird in Python erzeugt, indem dieser in einer Zuweisung an linker Stelle steht. Ansonsten muss ein Name nicht deklariert werden.
- Eine Zuweisung ist eine Anweisung (in Python).
- Abgrenzung zu Variable: Variable hat Speicherbereich und einen Typ zugeordnet.
- Vertauschung: Pattern mit Hilfsvariable

- Verwendung von Variable synonym zu Name!
- Verwende sprechende Namen.

  Beispiele:

  - `v`, `s`, `t` für die Geschwindigkeit, den Weg, die Zeit
  - `area` für die Fläche eines Rechteckes oder eines Kreises.
  - `perimeter` für den Umfang eines Kreises
  - `i`, `j` für ganzzahlige (Zählwerte)
  - `counter` oder `cnt` für einen Zähler

Bezeichner (engl. identifier)

Die Bezeichner dürfen in einer Programmiersprache in der Regel nicht beliebig gebildet werden. Üblicherweise darf ein Bezeichner nicht mit einer Ziffer beginnen!

Wir legen fest, dass wir für die Bezeichner lediglich die Buchstaben ”a” bis ”z” und ”A” bis ”Z”, die Ziffern ”0” bis ”9” und den Unterstrich ”_” verwenden!

Garbage Collection (dt. Speicherbereinigung, allgemein Müllabfuhr)

Verweist auf ein Objekt kein Name mehr, dann kann auf dieses Objekt nicht mehr zugegriffen werden. Damit ist dieses Objekt nutzlos und Müll (engl. garbage). Dies wird vom Garbage Collector von Python erkannt und daraufhin aus dem Speicher gelöscht.

optional (engl. optional)

d.h. wahlfrei, muss nicht sein

obligatorisch (engl. mandatory)

zwingend

3.2 Mathematische Grundlagen der Informatik

- Tabelle von 0 bis 16

  Die folgende Tabelle gibt die Dezimalzahlen von 0 bis 16 samt den Äquivalenten im Binärsystem (Frequenzhalbierung!), im Oktalsystem und im Hexadezimalsystem an:

10	2	8	16
0	00000	0	0
1	00001	1	1
2	00010	2	2
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
...	...	...	..
15	01111	17	F
16	10000	20	10

- Lineare Funktion

  In der Mathematik kann eine Funktion im einfachsten Fall als eine Rechenvorschrift verstanden werden. D.h. Wie kann ich eine Formel berechnen. Diese Formel bekommt einen Namen und kann einen oder mehrere Argumente als Parameter erhalten.

  Die lineare Funktion hat folgende Gestalt:

  $f(x) = kx + d$

  In dieser Formel bezeichnet k die Steigung der Geraden im zweidimensionalen Koordinatensystem. Die Steigung gibt an, um wieviel die Gerade in der y-Richtung ansteigt, wenn der x-Wert um 1 erhöht wird. Das d der obigen Formel gibt den y-Wert an, in dem die Gerade die y-Achse schneidet.

- Horner-Methode

$$
\begin{aligned}
z &= z_n \cdot B^n + z_{n-1} \cdot B^{n-1} + \cdots + z_1 \cdot B^1 + z_0 \\
&= (z_n \cdot B^{n-1} + z_{n-1} \cdot B^{n-2} + \ldots z_2 \cdot B^1 + z_1) \cdot B + z_0 = \\
&= ((z_n \cdot B^{n-2} + z_{n-1} \cdot B^{n-3} + \ldots z_2) \cdot B + z_1) \cdot B + z_0 = \\
&\ldots \\
&= ((\ldots (z_n \cdot B + z_{n-1}) \cdot B + \cdots + z_2) \cdot B + z_1) \cdot B + z_0
\end{aligned}
$$

  Beispiel:

$$
\begin{aligned}
1032_4 &= 1 \cdot 4^3 + 0 \cdot 4^2 + 3 \cdot 4^1 + 2 \cdot 4^0 = \\
&= (1 \cdot 4^2 + 0 \cdot 4^1 + 3) \cdot 4 + 2 = \\
&= ((1 \cdot 4 + 0) \cdot 4 + 3) \cdot 4 + 2 = \\
&= (4 \cdot 4) + 3) \cdot 4 + 2 = \\
&= (16 + 3) \cdot 4 + 2 = \\
&= 19 \cdot 4 + 2 = \\
&= 76 + 2 = \\
&= 78
\end{aligned}
$$

- Division mit Rest zweier ganzer Zahlen

  $8 \div 2 = 4 \text{ R } 0, \text{ d.h. } 8 = 4 \cdot 2 + 0$
  $7 \div 2 = 3 \text{ R } 1, \text{ d.h. } 7 = 3 \cdot 2 + 1$

  Allgemein gilt: $a \div b = c \text{ R } r \Leftrightarrow a = c \cdot b + r$

  - Restbildung: Modulo berechnet den Rest der Division zweier Zahlen.
    * $8 \mod 2 = 0$
    * $7 \mod 2 = 1$

    in Python: `%`
  - Ganzzahlige Division
    * $8 \text{ div } 2 = 4$
    * $7 \text{ div } 2 = 3$

    in Python: `//`

3.3 Programmierrichtlinien

- Variablennamen beginnen immer mit Kleinbuchstaben.

3.4 Programmier-Know-How

- Funktion

  Funktion vs. Aufruf einer Funktion

  - Eine Funktion ist in Python auch ein Objekt
  - Hat auch eine Darstellung (Funktionsname eingeben ohne Klammern!)
  - Auch von einer Funktion kann der Typ mittels `type` bestimmt werden.
  - Wird der Funktionsname (auch ein Identifier!) mit runden Klammern verwendet, dann wird die Funktion aufgerufen. Unter Umständen müssen bei dem Funktionsaufruf Argumente übergeben werden (siehe `sqrt`).
- Funktion vs. Prozedur (klassisch)
  - Prozedur hat keinen Rückgabewert
  - Funktion hat einen Rückgabewert
- Prozess
  - Ein Programm liegt meist als Datei auf der Festplatte.
  - Wird ein Programm gestartet, entsteht ein Prozess genannt.
  - Man kann ein Programm öfters starten. Damit entstehen mehrere Prozesse.
- Standardeingabe und Standardausgabe

  Jedem Prozess ist sowohl eine Standardeingabe und eine Standardausgabe zugeordnet.

  Von der Standardeingabe kann gelesen werden (z.B. mittels `input`) und auf die Standardausgabe kann (z.B. mittels `print`) geschrieben werden.
- Funktion `input`
  - Gibt optional auf Standardausgabe Text aus
  - Liest von der Standardeingabe Text ein
  - Liefert eingegebenen Text zurück

4 Einheit 4

4.1 Mathematische Grundlagen der Informatik

- Wahrheitswerte: true (wahr, `True`, $\neq 0$), false (falsch, `False`, 0)
- Logische (boolesche) Operatoren (nach George Boole) $\rightarrow$ Wahrheitstafeln: $\vee$ (or), $\wedge$ (and), $\neg$ (not), $\veebar$ (xor)
- Horner-Schema

```
1  0  1   1   0   1
   2  4  10  22  44
-------------------
1  2  5  11  22  45
```

- Probier-Methode

  Mit der Probier-Methode kann man eine Dezimalzahl in ein anderes Zahlensystem umrechnen.

  Durch Betrachtung der Zahlendarstellung

  $Z = z_n * B^n + z_{n-1} * B^{n-1} + \ldots + z_1 * B^1 + z_0 * B^0$

  erkennen wir, dass wir bei Division durch B^n den Term

  $z_{n-1} * B^{n-1} + \ldots + z_1 * B^1 + z_0 * B^0$

  als Rest erhalten. Bei dieser Division tritt z_n als eine Ziffer der Zahlendarstellung im Zahlensystem zur Basis B auf.

  Durch wiederholte Vorgangsweise können auf diese Weise alle Stellen ermittelt werden.

  Vorgangsweise:

  1. höchste Potenz von B bestimmen, die in der gegebenen Zahl enthalten ist. Ist der Exponent n, dann hat die gesuchte Zahl n+1 Stellen.
  2. Feststellen, wie oft diese Potenz in Z enthalten ist. Das ganzzahlige Frgebnis dieser Division ist die 1. Ziffer.
  3. Rest der ganzzahligen Division bestimmen und mit diesem Rest, so wie mit der ursprünglichen Zahl verfahren.

  89 in das 4er System:

  $64 = 4^3$, d.h. $3 + 1$ Stellen

  $89 = z_3 * 64 + z_2 * 16 + z_1 * 4 + z_0 * 1$

  $z_3 = 1$, da $89/64 = 1\text{R}25$
  $z_2 = 1$, da $25/16 = 1\text{R}9$
  $z_1 = 2$, da $9/4 = 2\text{R}1$
  $z_0 = 1$, da $1/1 = 1\text{R}0$

4.2 Programmier-Know-How

- Funktion definieren (ohne Parameter, ohne Rückgabewert)
- lokale Variable: steht nur innerhalb einer Funktion zur Verfügung, aber nicht außerhalb.
- globale Variable: steht auch außerhalb einer Funktion zur Verfügung.
- Sichtbarkeit (engl. scope)
- Verzweigungsanweisung: `if` statement (`else`, `elif`)
- Vergleichsoperatoren: `<`, `<=`, `==`, `!=`, `>=`, `>`
- einfache Anweisung (engl. simple statement) vs. zusammengesetzte anweisung (engl. compound statement)

4.3 Programmierrichtlinien

- 4 Leerzeichen für die Einrückung (**nie** Tabulatoren verwenden!)
- Funktionsnamen beginnen immer mit Kleinbuchstaben.
- Funktionsnamen immer in Kleinbuchstaben, `_` zur Trennung einzelner Wörter (in Bezeichnern) oder "mixedCase" - Darstellung (z.B. `getArea`).
- Jeweils 2 Leerzeilen zwischen Funktionsdefinitionen!

5 Einheit 5

5.1 Mathematische Grundlagen der Informatik

- Restmethode

  Mit Hilfe des Horner-Schemas kommt man (wie gezeigt) zu folgender Darstellungsweise:

$$Z = z_n * B^n + z_{n-1} * B^{n-1} + \cdots + z_1 * B^1 + z_0 * B^0 =$$
$$= ((\ldots(z_n * B + z_{n-1}) * B + \cdots + z_2) * B + z_1) * B + z_0$$

  Dabei erkennt man, dass bei Division der Zahl Z durch die Basis B die Ziffer z_0 als Rest anfällt. Das eigentliche Ergebnis kann wiederum durch B dividiert werden und man erhält die Ziffer z_{-1} als Rest. Dieser Vorgang kann solange fortgesetzt werden bis man alle Ziffern erhalten hat.

D.h. Die gegebene Dezimalzahl wird solange durch die gewünschte Basis geteilt, bis das Ergebnis 0 ist. Die dabei anfallenden Reste sind die Ziffern der gesuchten Zahl, allerdings in umgekehrter Reihenfolge!

89 in das 4er System:

$$89_{10} = ((1 * 4 + 1) * 4 + 2) * 4 + 1$$

$$\begin{aligned} 89/4 &= 22\text{R}1 \\ 22/4 &= 5\text{R}2 \\ 5/4 &= 1\text{R}1 \\ 1/4 &= 0\text{R}1 \end{aligned}$$

$$89_{10} = 1121_4$$

89 in das Dualsystem:

$$\begin{aligned} 89/2 &= 44R1 \\ 44/2 &= 22R0 \\ 22/2 &= 11R0 \\ 11/2 &= 5R1 \\ 5/2 &= 2R1 \\ 2/2 &= 1R0 \\ 1/2 &= 0R1 \end{aligned}$$

$$89_{10} = 1011001_2$$

4309 in das Oktalsystem:

$$\begin{aligned} 4309/8 &= 538\text{R}5 \\ 538/8 &= 67\text{R}2 \\ 67/8 &= 8\text{R}3 \\ 8/8 &= 1\text{R}0 \\ 1/8 &= 0\text{R}1 \end{aligned}$$

$$4309_{10} = 10325_8$$

5.2 Programmier-Know-How

- Funktionen mit Parameter und Rückgabewert
  - formale Parameter vs. Argumente (aktuelle Parameter)
  - Rückgabewert vs. kein Rückgabewert (`None`)
  - lokale Variable
  - Parameter als lokale Variable
  - per-value (vs. per-reference: 2. Jahrgang!)
  - Nebeneffekte (engl. side effects)
  - Fehlerbehandlung durch Fehlercodes
- Zeichenkettenoperationen
  - `+`
  - `"".format()`

  Strings sind nicht veränderbar!!!
- Methode ist eine objektgebundene Funktion!
  - → oft mit "`.`" aufgerufen

5.3 Programmierrichtlinien

- Parameterliste: vor =,= kein Leerzeichen, danach genau eines!
- Pro Import-Anweisung nur ein Modul importieren, d.h. keine Anweisung der folgenden Art:

  ```
  import math, sys
  ```

  Statt dessen::

  ```
  import math
  import sys
  ```

  Erlaubt ist allerdings folgende Art von Import:

  ```
  from math import sqrt, pi
  ```

6 Einheit 6

6.1 Mathematische Grundlagen der Informatik

- Umrechnen zwischen Oktal und Hexadezimal und Binärsystem
  - per Hand
  - mittels Funktionen `oct`, `hex`, `bin`, `int`
  - Notation für binäre, oktale und Hexadezimale Zahlen

6.2 Programmier-Know-How

- Entwurf (engl. design)

  Top-down vs. bottom-up
- Funktionen mit Parametern und Default-Werten
- Exceptions 1

  Einfaches Abfangen von Exceptions innerhalb einer Funktion ohne Angabe des Exceptionnamens.
- Lange Zeilen in Python

6.3 Programmierrichtlinien

- kein Leerzeichen um `=` bei Default-Werten

7 Einheit 7

7.1 Mathematische Grundlagen der Informatik

- Satz von Pythagoras

7.2 Programmier-Know-How

- Tupel
  - Sequenz von Objekten
  - Tupel sind unveränderlich (wie Strings)

  - Pattern der Vertauschung von Variablen auf der Basis von Tupel!
  - Der ”faule” Type `range` (engl. lazy type)
  - Länge eines Tupels
  - `len`
- Zählschleifen und Accumulator-Pattern
- Exceptions

  Einfaches Abfangen von Exceptions **außerhalb** einer Funktion ohne Angabe des Exceptionnamens

7.3 Programmierrichtlinien

- keine Leerzeichen vor `[` bei Indexzugriff: `x[`

8 Einheit 8

8.1 Mathematische Grundlagen der Informatik

- Mengen
  - Begriff Menge
  - Operationen
    * Leere Menge
    * Element-von
    * Gleichheit
    * Teilmenge, echte Teilmenge, Übermenge
    * Vereinigung
    * Durchschnitt
    * Differenz, symmetrische Differenz

8.2 Programmier-Know-How

- Modul (Umsetzung in Python: `if __name__ == "__main__"`, `dir`, `type(main)`)
- Farben

- Exceptions 3: Abfangen von Exceptions mit Angabe des Exceptionnamens

9 Einheit 9

9.1 Mathematische Grundlagen der Informatik

- Zufallszahlen vs. Pseudozufallszahlen
  - Kriterien: nicht vorhersagbar, bestimmte Wahrscheinlichkeitsverteilung
  - Pseudozufallszahlen: berechnet → Periode!
  - Anwendungen
    * Simulationen von Vorgängen (z.B. Eintreffen von Autos bei einer Kreuzung)
    * Testen von Programmen (Erzeugung von Eingabewerten)
    * Glücksspiele (z.B. Glücksspielautomaten)
    * Spieleprogrammierung (Gegner,…)
  - Zufallszahlengenerator

    $y_1 = (ay_0 + b) \bmod m$
    $y_2 = (ay_1 + b) \bmod m$
    $y_3 = (ay_2 + b) \bmod m$
    ...

    z.B. a = 5, b = 1, m = 16, $y_0 = 1 \rightarrow$ **1**, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5, 10, 3, 0, **1**, 6, 15,…
- ggT und kgV

9.2 Programmier-Know-How

- Abfolge der Operationen

  (engl. rules of precedence)

  Hier eine kurze Zusammenfassung der wichtigsten Operatoren, wobei die höchste Priorität am Anfang und die kleinste Priorität am Ende der folgenden Liste ist:

  1. Klammern: `()`
  2. Indexzugriff: `x[]`

3. `**`
4. unär: `+x`, `-x`, `~x`
5. `*`, `/`, `//`, `%`
6. `+`, `-`
7. `>>`, `<<`
8. `&`
9. `^`
10. `|`
11. `in`, `not in`, `<`, `<=`, `==`, `!=`, `>=`, `>`
12. `not`
13. `and`
14. `or`

Operatoren mit der selben Priorität werden von links nach rechts abgearbeitet!

- Bedingte Schleifen: `while`
- Pseudozufallszahlen: `random`
- Exceptions 4: Abfangen von Exceptions mit `finally`

10 Einheit 10

10.1 Mathematische Grundlagen der Informatik

- Teilbarkeit von Zahlen und Primzahlen

10.2 Geometrische Figuren

- Volumen (V)
- Quader: $a \cdot b \cdot c$
- Würfel: a^3
- Kugel: $\frac{4}{3}r^3\pi$

10.3 Programmier-Know-How

- `for` für Sequenzen
  - Tupel, Strings
- lange ganze Zahlen, `None`
- Es gibt in Python verschiedene Arten des Importierens::
  - `import math`
  - `from math import sqrt`
  - `from math import *`
  - `from math import sqrt as squareroot`

11 Einheit 11

11.1 Mathematische Grundlagen der Informatik

- Gleitkommazahlen vs. reelle Zahlen

11.2 Programmier-Know-How

- Methoden
- Liste: Methoden von Listen, Listenzuweisung (List comprehension)
- Methoden von Strings
- Veränderbare vs. nicht-veränderbare Datentypen (immutable vs. mutable Datentypen)
- Sequenzen: `[]`, `+`, `*`, `in`
- Namespace (Namensraum)

  Ein Namensraum ist eine Abbildung von Namen zu Objekten. Bei Objekten: mittels Punktoperator.
- Scope (Sichtbarkeit)

  Ein Scope ist ein textueller Bereich in dem ein Namensraum direkt zugreifbar ist. "Direkt zugreifbar" bedeutet, dass ein Name unqualifiziert also ohne Verwendung des Punktoperators innerhalb des Namensraumes gefunden werden kann.

11.3 Programmierrichtlinien

- Keine Leerzeichen um Punktoperator

12 Einheit 12

12.1 Mathematische Grundlagen der Informatik

- Speicherung natürlicher Zahlen mit Obergrenze: $[0,2^n-1]$

12.2 Programmier-Know-How

- Dictionary (engl. dictionary, auch: hash map, hash array, Streuspeicherung)
- Eine erste Einteilung der Datentypen
  - skalare (einwertig, engl. scalar) Datentypen: `bool`, `int`, `float`
  - mehrwertige (engl. multi-valued) Datentypen:
    * Sequenzen (engl. sequence): `tuple`, `list`, `str`
      · Reihenfolge, Index
    * Menge (engl. set): `set`
      · keine Reihenfolge, keine doppelten Elemente
    * Dictionary: `dict`
      · Abbildung Key auf Value, keine Reihenfolge, keine doppelten Keys
- Dateien (engl. file)
  - Öffnen; Modus `w`, `r`
  - Schreiben (`write`), Lesen (`read` und `readline`)
  - Schließen

12.3 Programmierrichtlinien

- Kein Leerzeichen vor `:` bei Dictionaries, immer folgendermaßen: `d = {"abc": 123, 123: "abc"}`

13 Einheit 13

13.1 Mathematische Grundlagen der Informatik

- Addition von Binärzahlen

13.2 Programmier-Know-How

- Bitoperationen 1
  - and `&`, or `|`, not `~`, xor `^`
- Suchen in einer Sequenz
  - Sequentielle lineare Suche in Sequenz
  - Binäre Suche in Sequenz
- Übungsbeispiele: Zahlenratespiel

14 Einheit 14

14.1 Mathematische Grundlagen der Informatik

- Subtraktion von Binärzahlen
- Darstellung und Speicherung negativer Zahlen
  - Einerkomplement: $[-2^{n-1},2^{n-1}]$
  - Zweierkomplement: $[-2^{n-1}-1, 2^{n-1}]$
- Subtraktion von Binärzahlen mittels Zweierkomplement

14.2 Programmier-Know-How

- Eingabeprüfungen
- Bitoperationen 2
  - lshift `<<`
  - rshift `>>`
  - Zusammenhang zur Multiplikation/Division mit 2^m
  - maskieren, setzen, löschen

15 Einheit 15

15.1 Mathematische Grundlagen der Informatik

- Multiplikation von Binärzahlen

15.2 Programmier-Know-How

- Kommandozeilenverarbeitung 1
- Sortieren mittels BubbleSort
- Anwendungen der Schaltalgebra (optional)
  - Halbaddierer
  - Flip-Flop
  - Multiplexer
  - Zähler

16 Einheit 16

16.1 Mathematische Grundlagen der Informatik

- Division von Binärzahlen

16.2 Programmier-Know-How

- Kommandozeilenverarbeitung 2: ”usage”-Meldung
- Sortieren mittels SelectionSort

17 Einheit 17

17.1 Programmier-Know-How

- Menüsystem
- Sortieren mittels InsertionSort