Informatik I & II
René Kaufmann - 25. Februar 2007

Allgemeine Befehle		
AWT	Abstract Windowing Toolkit	Klassenbibliothek zur Programmierung von Fenstern und graphishen Objekten
//	Kennzeichen für Zeilenkommentare	//dies ist ein Kommentar
“”	Zeichenketten sind von Gänsefüsschen umschlossen	“guten Tag“ “und noch eine Zeichenkette“
( )	Runde Klammern: Argumente von Methoden	myMethod()
{}	Geschweifte Klammern: Blöcke von Anweisungen	{ x=x+3; System.out.println(x);}
;	Abschluss einer Anweisung	x=3*7;
System.out.println()	Zeichenkette auf Bildschirm anzeigen	System.out.println(“Hallo“); System.out.println(“Die Lösung lautet:“+x);
public class	Kopfzeile zur Beschreibung einer Klasse	public class MyProgram { }
public static void main(String[ ] args) { }	Hauptmethode in Java- Programmen. Der eigentliche Programmtext wird zwischen den geschweiften Klammern eingefügt.	
String[ ] args	Teil des Main-Methodenkopfes: Hier drin wird eine Liste von Zeichenketten als Parameter übergeben.	ersteZeichenkette = args[0]; zweiteZeichenkette = args[1];
Vergleiche, Zuweisung, Mathematische Operationen		
=	Zuweisung eines Wertes an eine Variable	i = 3; i = i + 4;
==	Vergleich	if (i==3) {Anweisungen}
!=	Ungleichheit	if (x != 0) {Anweisungen}
&& || !	Und Oder Nicht	
+ - * /	Mathematische Operatoren	3 = 3 * (4+i)/(7-j);
Math.sqrt()	Berechnen der Quadratwurzel	k = Math.sqrt(2);
Math.random()	Zufallszahl erzeugen	z = Math.random();
int, float	Typen: Ganzzahl (int) und Gleitkommazahl (float)	int x; float f;
int x;	Deklaration einer Ganzzahl Variablen x	
Integer.parseInt()	Liest eine Zeichenkette und wandelt diese in eine Ganzzahl um.	i = Integer.parseInt(“23“); j = Integer.parseInt(args[0]);
Kontrollstrukturen		
if (Bedingung) {Anweisungen} else {Anweisungen}	Kontrollstruktur if-Verzweigung	if (x != 0) {r = r/x; System.out.println(“x hat den Wert:“+x);}
for (Startwert; Bedingung; Veränderung) {Anweisungen}	Kontrollstrukur For-Schleife	for (i=0; i<10; i++) {System.out.println(i);}
while (Bedingung) {Anweisun-gen}	Kontrollstruktur While-Schleife	k=1; while (k!=7) {System.out.println(k); k=k+2;}
try {Anweisungen} catch (Ausnahme) {Anweisungen}	Behandeln von Ausnahmen	
Array		
Typ[] Name;	Array deklarieren	int[] meinArray;
Name = new Typ[Anzahl];	Array erzeugen	meinArray = new int[7];
Typ[] Name = new Typ[Anzahl]	Array deklarieren und erzeugen in einem Schritt...	int[] meinArray=new int[7];
Typ[] Name = {Werte};	Array erzeugen welches mit Werten gefüllt ist. Sonderfall (ohne new).	int[ ] meinArray={1,2,3,4};
meinArray[Index] = Wert;	Wert an das Array-Element an der mit Index gekennzeichneten Position einfügen.	meinArray[3] = 2;
for (i=low; i < up; Step) {meinArray[i] = 0;}	Alle Elemente eines Arrays auf Null setzen. Initialisierung.	for (i=0; i<7; i++) {meinArray[i] = 0;}

AbstractWindowsToolkit und Graphics-Elemente	
import java.awt.*;	AWT-Bibliothek mit Grafikbefehlen imporitieren
public void paint (Graphics g) {graphische Anweisungen}	Grafische Anweisungen werden in der paint-Methode ausgeführt.
g.drawLine(int x1, int y1, int x2, int y2);	Linie zeichnen vom Punkt (x1, y1) zu Punkt (x2, y2)
g.drawRect(int x, int y, int width, int height);	Rechteck zeichnen mit dem Punkt (x,y) links oben, der Breite width und der Höhe heigth.

g.drawPolygon(int[ ] arx, int [ ] ary, int cnt);	Polygon zeichnen durch die Punkte mit den Koordinatenpaaren im Array x und im Array y. Die Anzahl Koordinatenpaare ist in cnt gegeben.
g.drawstring("Text", x, y);	Schreibt die Zeichenkette Text am Punkt (x,y)
g.setColor(Color.farbe);	Wählt eine Farbe aus für die folgenden Grafikbefehle
Image Objektname; Objektname = getToolkit().getImage(Dateiname);	Laden einer Bitmap-Datei mit dem Namen Dateiname als Objekt mit dem Namen Objektname.
g.drawImage(Objektname, x, y, grVer, grHor, this);	Zeichnen einer Bitmap-Datei mit einem Objektnamen, mit der linken oberen Ecke an der Position (x, y), und der vertikalen Gösse grVer und der horizontalen Grösse grHor und dem Ladezustand this.
Objekt und Schaltelemente	
import java.applet.*;	Importanweisung zur Verwendung von Applets.
public void init( ) {}	Initialisierungs Methode in Applets
Font f;	Deklaratioin eines Objektes der Klasse Font mit dem Namen f.
f = new Font (TimesRoman, 24);	Erzeugen eines Objektes der Klasse Font mit den Parametern TimesRoman und Grösse 24. New erzeugt das Objekt, die Parameter stehen in Klammern.
Font f=new Font(TimesRoman, 24);	Kurzform: Deklaration und Erzeugung des Objektes in einer Zeile.
Button Name = new Button("meinText");	Erstellt ein Objekt vom Typ Button (Schaltfläche)
TextField Name = new Textfield ("Text", Länge);	Erstellt ein Objekt vom Typ Textfeld (zur Eingabe von Texten)
Label objektName = new Label ("Text");	Erstellt ein Objekt vom Typ Textfeld (Label, zur Ausgabe von Texten)
Weitere Informationen	
boolean char byte short int long float double	true, false Zeichen (16 Bit, Unicode) Ganzzahl (8 Bit, -128 - + 127) Ganzzahl (16 Bit, -32768 - +32767 Ganzzahl (32 Bit) Ganzzahl (64 Bit) Gleitkommazahl (32 Bit) Gleitkommazahl (64 Bit)

zweiDimArray = new int[3][4]
for (i=0; i¡ 3; i++){
for (j=0;j¡4;j++){
zweiDimArray[i][j]=0;
}
}
	a[0][0] a[1][0] a[2][0]
	a[0][1] a[1][1] a[2][1]
	a[0][2] a[1][2] a[2][2]
	a[0][3] a[1][3] a[2][3]
	i=0 i=1 i=2

PRGM Beispiele

```
import java.awt.*;
public  class Karte extends Frame {  // extends: Ableitung der vorhandenen Klasse ("Fenster")
public static void main(String[] args) {
Karte MeinKarte1 = new Karte();
}  //main // Konstruktor
public Karte() {     //Methodenaufruf
 addWindowListener(new UnserFensterSchliesser()); //eigene Klasse
 setSize(360, 260); //Fenstergrösse bestimmen (X,Y)
 setTitle("Karte"); //Fenstertitel zuweisen
 setVisible(true); //Fenster wird so sichtbar gemacht
}
public void paint (Graphics g) {  //Paint - Methode
 Image bild; //Name des Objektes: Image name
 bild = getToolkit().getImage("karte.gif");
 g.drawImage(bild, -1250, -1100, 2700,2500, this); //(Name des Objektes, xyPosition in der
                                                   //oberen linken Ecke, Grösse der Image,
                                                   //Ladezustand der Bitmap
}
} //class|
```

```
import java.awt.*;
public  class Barcode extends Frame { // extends: Ableitung der vorhandenen Klasse ("Fenster")
public static void main(String[] args) {
  Barcode MeinBarcode1 = new Barcode();
} //main, Konstruktor
public Barcode() \{   //Methodenaufruf
 addWindowListener(new UnserFensterSchliesser()); //eigene Klasse (Schliess - Button)
 setSize(400, 300);
 setTitle("Barcode"); //Fenster Parameter setzen
 setVisible(true); //Fenster wird so sichtbar gemacht
}
int summe = 0;
int i = 80;
int zahl, zufallszahl1, zufallszahl2;
int CH = 76;
public void paint (Graphics g) { //Paint - Methode
 g.setColor(Color.black); //Farbe auf Rot setzen
 while (i<180) {
  zahl = (int) (Math.random() * 2) + 1;
  zufallszahl1 = (int) (Math.random() * 100000 +1);
  zufallszahl2 = (int) (Math.random() * 100000 +1);
  i = i+zahl;
  g.drawLine(i, 40, i, 100);
  summe = summe+1;
 }
  g.drawString(""+CH, 80,120);
  g.drawString(""+zufallszahl1,100,120);
  g.drawString(""+zufallszahl2,140,120);
 }
} //class
```

Grundlagen	
Algorithmus	Handlungsanweisung, die bei genauer Anwendung nach einer endlichen Anzal von Schritten zum Ergebnis führt.
ASCII	American Standard Code for Information Interchange $2^7 = 128$ Zeich. + Paritätsbit
Extended ASCII	+128 Zeichen
ASCII-Files	Folge von ASCII-Zeichen. Jedes Zeichen dargestellt mit ASCII-Code. Matlab, LaTeX, HTML,...
Bilder (Digitalisierung)	S/W: (0,1), 1Bit/Px, Graustufe: 1 Byte / Px, Farbe: 3Byte / Px (RGB), Rastern, diskrete Punkte
Bit	Binary Digit, kleinstmögliche Einheit der Information (0,1)
Byte	1 Byte = 8 Bit, 2^8 = 256 Zustände (allgemein: k Bits = 2^k Zustände)
Computer	Information $\rightarrow$ Manipulation der Daten (Programm) $\rightarrow$ Ausgabe
Daten	= Information in Form für einfache & bequeme Weiterverarbeitung
Informatik	Systematische Verarbeitung von Information
Theoretische I. Technische I. Praktische I. Angewandte I.	Logik , Entscheidbarkeit, formalen Sprachen, Berechenbarkeit, Komplexität Algorithmen, Daten, Programmierung (Sprachen), OS, verteilte Systeme. Echtzeitsysteme, Systemsoftware, Compiler Künstliche Intelligenz, Datenbank, numerische Verfahren, Kryptographie, Simulation
Information	Angaben/ Nachrichten, für Empfänger ergibt sich aus dem Zusammenhang eine Bedeutung
Mikroprozessor	Programm, das fest in den Computer eingebaut ist.
Text	Zeichen besteht aus Code
Unicode	Code für jedes Zeichen irgendeiner Sprache mit verschiedenen Längen (1,2,4 Byte)
Wissenschaftliches Rechnen	Lösen von komplexen (grossen) Problemen mit Hilfe von Mathematik und Computern
Computer	
Betriebssystem	OS verwaltet die Betriebsmittel eines Computers (Hard-, Software, Daten), läuft immer (kernel). Alle anderen Programme: Anwendungen, macht Benutzung bequemer, unterstützt effiziente und faire Benützung der Ressourcen (Betriebsmittel)
CPU Prozessor	Zentraleinheit, Leistungsbestimmend: Prozessor, Speicher, Bussystem liegen auf der Systemplatine nahe auf beieinander. Was ein Computer macht, ist die Konsequenz eines Programmes, das seinen Prozessor gerade ausführen lässt. Programme sind mit den Daten im Speicher abgelegt. OS holt Teile von Programmen/ Daten von der Festplatte in den Speicher. Funktionseinheit des Rechners, die Instruktionen interpretiert und ausführt. Schnellste Komponente. (GHz)

Speicher	Funktionseinheit zum Aufbewahren von Daten. Daten im Hauptspeicher (256-1024 MB) gehen ohne Strom verloren. Holt Programmteile von der HD. Nachteil: langsam. → Cache. Blockgrösse: 4KB (page)
Bussystem	Elektrische Leitung zwischen mehreren Funktionseinheiten einer Rechneranlage. Leistung bestimmt durch Taktfrequenz (p4:533MHz), Breite (Anzahl Drähte), MB/s
Peripherie	Geräte, die über Schnittstellen (interface) an CPU angeschlossen sind. Selbstständiger Datentransfer von/nach Arbeitsspeicher
	Eingabegerät → CPU, RAM → Ausgabegerät ↓ HD
Chache	Schneller Puffer-Speicher (in CPU als L1 und L2 im Prozessor). Daten stehen schneller zur Verfügung. Funktionsweise: Lokalitätseigenschaft (Wahrscheinlichkeit, dass Daten als nächste gebraucht werden, ist umso grösser, je näher sie an den gerade gebrauchten liegt.). In Blöcke (cache lines) unterteilt. Block wird immer als Ganzes kopiert.
Chache, Level 1	Am nächsten am Prozessor, 8-64 KB, noch auf CPU. Blockgrösse:32 B
Chache, Level 2	Eigener Chip, 256 - 1024 KB, „Chache für Festplatte“, Blockgrösse: 128 B
Chaches	Schnelle Speicher, unterteilt in Blöcke (cache lines). Grösse hängt vom chace level ab.
Computersystem	CPU $\overset{\text{Bus}}{\rightarrow}$ Controller $\overset{\text{Bus}}{\rightarrow}$ Peripherie
Dateisystem	Bestandteil des OS, Speichert und verwaltet Daten. Per Name auf Datei zugreifen. Dateinamen sind in Verzeichnissen abgelegt.
Datenlokalität	Ganzer Block wird eingelesen → Vorteil wenn alle Daten des Blockes gebraucht werden.
Hauptspeicher (Arbeitsspeicher)	Speicher, in dem Programme und von Mikroprozessor zu verarbeitenden Nutzdaten abgelegt und unverändert abgerufen werden kann. Arbeitsspeicher besteht aus RAM. Flüchtig.
Massenspeicher / Festplatte	Specher zur kostengünstigen, dauerhaften Ablage grosser Datenmengen
Mengenassoziatives Chache, Adressierungstechnik.	Chache hat mehrere Sets. Diese haben feste Plätze im Chace. Neuer Block nur innerhalb eines Sets platziert (da aber beliebig).
Moor's Law	Anzahl Transistoren pro Chip verdoppelt sich in 2 Jahren
Prozess	Instanz eines Programms, das auf einem Computer läuft.
Prozessorregister	Kleiner, sehr schneller Datenspeicher. Register werden zum Zwischenspeichern von Befehlen, Speicheradressen und Rechenoperanden benutzt.
RAM	Random Access Memory. Hauptspeicher. langsamer als Prozessor
RAM	Random Access Memory, schnell aber teurer als magnetisches Speichermedium. Flüchtiges Speichermedium
Speicherblöcke	Datenblock (cache line, page) wir immer als ganzes kopiert. Cacheline kann nur in einem einzigen von m Sätzen abgespeichert werden. Chachegrösse: nm Blöcke.
Speicherhierarchie	Häufig werden zur besseren Organisation des Speichers (z. B. Optimierung der Zugriffszeit für häufig benötigte Daten) Speicherinhalte innerhalb der Hierarchie verschoben. Alle Daten einer Ebene sind auch in der darunterliegenden Ebene. Kleinste Informationseinheit ist der Block. Zwischen den Ebenen werden nur komplette Blöcke verschoben.
Speicherhierarchie	CPU (200B, 5ns), Cache (64KB, 10ns), Hauptspeicher (32MB, 100ns), Festplatte (2GB,5ms), klein schnell teurer ↔ gross, langsam, billig
Speichermanagement	Files auf Massenspeicher. Werden Daten vom Prozessor des Computers benötigt, so wird das entsprechende File durch OS vom Massen- in den Arbeitsspeicher kopiert. Danach Verwendung via Caches / Register. Meist werden nur Teile in den Hauptspeicher geladen.
Systemstruktur	CPU (Prozessor, Hauptspeicher, Busssystem) ⇔ Periphegeräte
UNIX Filesystem	On a UNIX system, everything is a file; if something is not a file, it's a process. (root → dev (spezielle Files), bin, home, . . .)
Zugriffsrechte	`- rwx rwx rwx` (Dateiart, Eigentümer, Gruppe, Rest). OS verwaltet und kontrolliert Zugrif fsrechte auf Files
Zahlen	
Adressberechnungen	Vorgegeben: Hauptspeicher, $512MB = 512 \cdot 2^{20}B = 536870912B$) und ein Cache $64KB$, 4-way-associative, chacheline = $32B$ 1 Satz hat $4 \cdot 32B = 128B = 2^7B$. $\Rightarrow 64 \cdot \frac{2^{10}}{2^7} = \frac{2^{16}}{2^7} = 2^9 = 512$ Sätze
Alphabete	$\{0, 1, \ldots, 9\}$ Zahlen $\{a, b, \ldots, A, B, \ldots, Z\}$ Wort $\{0, 1\}$ Binärzeichen
ANSI	American National Standards Institute
binär	zwei diskrete Zustände annehmend
Code	Zuordnung zwischen zwei Alphabeten. Zuordnung geschieht mittels einer Codetabelle. Verschlüsslung von Information.
Codierung	Zeichen eines beliebigen diskreten Alphabets lassen sich durch Gruppen von Binärzeichen darstellen. (Gruppen von Binärzeichen der Länge n codieren 2^n Symbole.
Dezimalzahl → Dualzahl	Für $n \geq 0$: Wenn $n = 0$ fertig, sonst: $n = 2m + r$, $r \in 0, 1$. r gibt die nächsthinterste Dualziffer.. $n \leftarrow m$ wiederholen.
digital	Werte in Form von Ziffern oder Zahlen darstellen

diskret	(mehrere) feste Zustände annehmend
double	64-Bit- Fliesskommazahl (Vorzeichen, 11 Bit Exponent, 52 Bit Mantisse) double precision: 1 [63], 11 [62-52], 52 [51-00], Bias 1023. Kleinste double-Zahl $> 1 = 1+2^{-52} \approx 1+2.2204 \cdot 10^{-16}$. Kleinste positive Zahl: $2^{1-1023} = 2^{-1022} \approx 2.2251 \cdot 10^{-308}$ Grösste positive Zahl: $2^{2046-1023} \cdot 2 - 2^{-52} \approx 1.7977 \cdot 10^{308}$. Normalisierte Zahl: $V = (-1)^s 2^{E-1023} 1.F$
dual	im zweiwertigen Zahlen- und Stellenwertsystem ausgedrückt, resp. rechnend
Dualzahl	erstes Bit: Vorzeichenbit; 0=positiv. Negative Zahlen: Bitumkehr, danach Addition von 1. Bsp: $127_{10} = 01111111 \rightarrow -127_{10} = 10000000+1 = 10000001$. Werden ganze Zahlen im Computer zum Rechnen gebraucht, werden sie als Dualzahlen dargestellt
Festkommazahlen	Zahl mit fester Anzahl Ziffern. Position des Kommas fix vorgegeben. Geeignet in speziellen Situationen. Rechnungen können im allgemeinen im Integer-Format ausgeführt werden (schneller!)
Fliesskommazahlen	Sollen die reellen Zahlen approximieren. Standardisiert für Zahlem mit 32 und 64 Bit.
float	32-Bit Fliesskommazahl. (Vorzeichen, 8 Bit Exponent, 23 Bit Mantisse) single precision: 1 [31], 8 [30-23], 23 [22-0], Bias 127. Wert einer normalisierten Zahl: $V = (-1)^s 2^{E-127} 1.F$
IEEE	Institute of Electrical and Electronics Engineers
IEEE- Arithmetik	Regeln, wie mit Zahlen im IEEE standard Format gerechnet werden muss, insbesondere beim Runden. Es wird immer so gerundet, dass das Resultat jeweil die dem korrekten Resultat nächste Maschinenzahl ist.
Integer	elementarer Datentyp, Ausschnitt der ganzen Zahlen: short: 16-bit signed: $-2^{15} \leq n \leq 2^{15} - 1$ int: 32-bit signed: $-2^{31} \leq n \leq 2^{31} - 1$ unsigned int: 32-bit unsigned: $0 \leq n \leq 2^{32} - 1$ (Mit 32 Bit können 4 GB adressiert werden. long: 64-bit unsigned: short: $0 \leq n \leq 2^{64} - 1 = 1.8 \cdot 10^{19}$, Speicher von 16 Exabyte direkt addressierbar.
bit	Anzahl Zeichen mit n Bit.
byte short int long n bit	8 Bit, Bereich -128 - +127, 256 ganze Zahlen 16 Bit, Bereich -32768 - +32767, 65536 ganze Zahlen 32 Bit, Bereich -2'147'483'648 - +2'147'483'647, 4'294'967'296 ganze Zahlen 64 Bit, Bereich -9'223'372'036'854'775'808 - +9'223'372'036'854'775'807, 18'446'744'073'709'551'616 ganze Zahlen Zahlen von $-2^{n-1} - +2^{n-1} - 1$
Inversion einer Dualzahl	Dualzahll invertieren, Addieren von 1. `00000001`, `11111110`, `00000001`, `11111111`
Tabellenkalkulation	
Funktionen	vorprogrammierte Formeln
Syntax	korrekte Folge von Zeichen
Argumente	Zahlen, Namen, Matrizen, Bezüge
Zirkelbezugsformeln	können nicht direkt aufgelöst werden. Können aber iterativ gelöst werden.
Netzwerk	
Bandbreiten	1Gbps: ATM (B-ISDN), Gigabit Ethernet; 100Mbps: Fast Ethernet; 10Mbps: Ethernet, Token ring, WLAN; 1Mbps: Satellit, 3G mobile; 100kbps: N-ISDN, GSM; 10kbps: Modem über Telefon, 1G mobile
Client	Fordern Daten von Servern an (Sendet Anfrage an Host)
Client - Server	http-Protokoll (über TCP/IP)
Computer Netzwerk	Ansammlung von autonomen (selbständig booten) Rechnern, die durch gleichartige Technik verbunden sind. Internet ist kein Computernetzwerk.
CSS	Cascading Stylesheets Selektor Eigenschaft: Wert;
DHCP	Dynamic Host Configuration Protocol
DNS (Domain Name System)	IP-Adresse $\Leftrightarrow$ Namen (hierarchisches System) DNS ist eine weltweit auf Servern verteilte hierarchische Datenbank, die den Namensraum des Internets verwaltet. Dieser Namensraum ist in so genannte Zonen unterteilt. DNS wird zur Umsetzung von Domainnamen in IP-Adressen (forward lookup) benutzt. Dies ist vergleichbar mit einem Telefonbuch, das die Namen der Teilnehmer in ihre Telefonnummer auflöst. Ist eine Domain unbekannt, wird die Anfrage an den nächst höheren DNS weitergegeben.
Domainname, Hostname	hierarchisch organisierter Bezeichner, welcher IP Adresse zuordnet. Besteht aus zwei oder mehr Namenstupeln. Letztes Tupel: Toplevel (.ch). Hostname: erstes Tupel
DTD	Dokumenttyp-Definition: Welche Elemente vom Typ HTML enthalten darf; Welche Elemente innerhalb von anderen vorkommen darf; Welche Attribute zu einem Element gehören; Attribut Pflicht oder Freiwillig
Fehlererkennung - Paritätsbit	Im Code sind redundante Bits. Paritätsbit. Ergänzt die Anzahl Einsen im Code auf eine gerade/ ungerade Zahl. Bei mehreren Bitfehlern im Datenblock versagt die Paritätsprüfung. Erlaubt nur Erkennung eines 1-bit-Fehlers, nicht aber dessen Korrektur
Fehlererkennung - Prüfsummen	Bits und Bytes, andere grundlegende Komponenten werden mit einem bestimmten Faktor multipliziert und anschliessend aufsummiert.

Fehlerkorrektur- Codes	Erkennung und Fehlerbehebung mittels Algorithmus. Mehrere Prüfbits werden den Nutzbits hinzugefügt, aus denen nach dem Wahrscheinlichkeitsprinzip am Empfang das richtige Zeichen ermittelt wird. Fehlererkennende und -korrigierende Redundanz kann in alle binären Codes eingestreut werden. Position spielt keine Rolle. Fehlerkorrektur braucht grössere Redundanz als Fehlererkennung. Grösser Anzahl der fehlertoleranten Bits → Aufwand der Codierung steigt → breiterer Code.
Firewall	Paketfilter basierend auf Protokoll, Port, IP-Adresse. Untersuchung jedes Paketes.
HTML	Hypertext Markup Language
Internet	Netzwerk von Netzwerken. Schutz von militärischen Daten auf weit entfernten Computern
IP-Adresse (IPv4)	32-Bit Adresse, 4 Byte (IPv6 mit128-Bit) Netzwerknummer, Hostnummer. Netzwerknummer wird durch Netzmaske bestimmt (255.255.255.0) (Binäre Addition von IP und Subnet) Netzwerkstelle besitzt mindestens eine eindeutige Adresse. Bei Sender-/Empfänger aus anderen Netzwerken: Paketversand über Router.
LAN	Local Area Network: Bus (Ethernet) oder Ring
MIME	Multipurpose Internet Mail Extensions. Gibt dem Browser die Art der Datei an.
Netzwerk-Architektur	Computernetzwerke müssen grosse Anzahl von Rechnern verbinden können, kosteneffizient, robust, leistungsfähig, adaptierbar an neue Technologien und ändernde Nachfrage sein. Netzwerk ist dynamisch.
Paket	Datenblock plus Informationen (Adresse, Länge, TP, Header-Prüfsumme)
Peer-to-peer	Keine festen Clients und Server
Ports	21 FTP File Transfer P 22 SSH Secure Shell Remote Login P 23 telnet Virtual terminal (remote login) 25 SMTP Simple Mail Transfer P 53 DNS Domain Name Service 80 HTTP Hyper Text Transfer P, WWW 110 POP3 Post Office P (V.3) Remote e-mail access 443 HTTPS http over TLS/SSL
Protokolle	Regeln den Nachrichtenaustausch zwischen Kommunikationspartnern. Schichten für verschiedene Teilaufgaben. Zwischen den Schichten Schnittstellen (interfaces)
Schnittstellen	Hier werden Services zur Verfügung gestellt. Anfordernde Schicht muss nicht wissen, wie der Dienst erbracht wird.
Schichtenmodell	Reduktion der Komplexität.
Server	Wartet auf eine Anfrage (Host-Rechner) (Antwortet auf Client)
Sockets	IP-Addresse und Portnummer ergeben Kommunikationsendpuntk (socket)
TCP	(transmission control protocol) Zuverlässiges (Daten werden so oft gesendet, bis der Empfänger den Empfang bestätigt hat), verbindungsorientiertes (vor dem Senden wird eine logische Verbindung vereinbart, Datenpakete übermitteln, Bestätigung der einzelnen Pakets) Datenstrom-Protokoll. Weiterleitung an die korrekte Applikation (Portnummer)
TCP / IP	Transmission Controll Protocoll / Internet Protocol. Vier Schicht- Referenzmodell: application layer (Verarbeitunsschicht), transport layer, internet layer (Vermittlungsschicht), network layer (Bitübertragungs./ Sicherungsschicht)
TCP / IP Protokoll Architektur	Application layer: Telnet, FTP, SMTP, NameServer, NFS Transportlayer: Transmission Control Protocol (TCP), User Datagram Protocol (UDP) Internet Layer: IP, Internet Controll Message Protocol (ICMP) Network Laxer: X.25, Ethernet, Token Ring
Netzwerkschicht	**Bitübertragungsschicht (unten):** Übermittlung von Bits und Bytes (0,1). Leistungsmerkmal: Übertragungskapazität. Verantwortlich vor Datenverlust. **Sicherungsschicht (oben):** Zugriffsverfahren, Fehlerfreie Übertragung von Bits (Fehlererkennung und Korrektur). Verantwortlich für Richtigkeit.
Internet- Vermittlungsschicht	IP, Datenübertragung über weite Strecken (Routing). Zustellung ist nicht garantiert. Hauptfunktion: Adressierung von Hosts (IP-Adressen), Fragmentierung von Paketen. Übertragung der Daten von Transport zur Netzwerkschicht.
Transportschicht	Herz der Protokoll- Hierarchie. Hauptaufgabe: zuverlässige, kostengünstige, effiziente Übermittlung von Daten. Netzwerkunabhängig. TCP, UDP.
Anwendungs- Apllicationsschicht	Oberste Schicht. Bietet eine Reihe standardisierter Anwedungsprotokolle (ftp, telnet, remote login (rsh, ssh) DNS, SMTP, HTTP)
URL	Uniform Resource Locator
WAN	Wide Area Network. Verbindung von mehreren LAN über weite Strecken. Haben viele Übertragungsleitungen. Kommunizieren über Router. Bei der Übertragung werden Pakete als ganzes von einem zum nächsten geschickt. Dort werden sie abgespeichert und weitergeleitet.
WWW	Information gleich beim Aufruf auf dem Bildschirm.
Verschlüsselung	
Zuverläsigkeit vs Sicherheit	Zuverlässig: Daten kommen korrekt an. Netzwerk- Sicherheit: Sicherheit vor unbefugter Verwendung. Vertraulichkeit, Datenintegrität (Manipulation der Daten auf dem Weg), Authentifizierung (Absender eindeutig und tatsächlich), Verbindlichkeit (keine Abstreitung)
Caesars Verschlüsselung.	Substitution. Verschiebung um einen bestimmten Betrag.

Kryptographie	Geheimschrift, Wissenschaft die sich mit dem Absichern (Verschlüsseln, Chiffrieren) von Nachrichten beschäftigt.
Kryptoanalyse	Kunst, verschlüsselte Texte zu brechen.
Verschlüsseln, Chiffrieren	Umwandlung des Klartext in eine allgemein nicht lesbare Form unter Anwendung einer Methode.
Enschlüsseln	Umwandlung des Geheimtextes in Klartext.
Verschlüsselungsverfahren: symmetrisch assymetrisch	beide kommunizierenden Seiten benützen den gleichen Schlüssel zum Ver- und Entschlüsseln. Zum Entschüsseln wird ein anderer Schlüssel benötigt als zum Verschlüsseln. Sicherheit entsteht durch Unkenntnis des Schlüssels.
Symmetrisches Verschlüsselungssystem	Sender: Klartext P wird mit geheimen Schlüssel K zum Chiffrat $C = E_K(P)$. Übertragung erfolgt über unsicheren Kanal. Empfänger: $P = D_K(C) = D_K(E_K(P))$. Hauptproblem: Schlüssel K muss sicher übertragen werden und darf dem Gegner nicht bekannt sein.
Monoalphabetische Substitution	allgemeiner als Caesar, lässt sich aber leicht brechen durch statistische Verteilung der Buchstaben, Diagramme (th, in, er,...), Trigramme (the, ing, and,...), Buchstabenverdopplung usw.
Transpositionsverschlüsselung PIC	Schlüsselwort gibt Reihenfolge und Anzahl der Spalten an. (Kombinatoin von Transposition und Substitution)
DES	Data Encription Standard (1977 von IBM entwickelt, Blöcke von 64bit, 54bit Schlüssel.
AES	Advanced Encription Standard. Nachfolger von DES. Blöcke/ Schlüssel unabhängig voneinander (128/192/256 Bits), leicht in Software implementierbar, Algorithmus frei
Public Key Algorithmen	Assymetrische Verschlüsslung. Drei Anforderungen: $D(E(P)) = P$. Es ist unmöglich D von E herzuleiten, E kann nicht durch Klartext-Angriff gebrochen werden. A holt öffentlicher Schlüssel E von B. A verschlüsselt Meldung P mit Schlüssel. $(Q = E_B(P)$. A sendet Q an B. B entschlüsselt mit geheimem Schlüssel (Private Key): $D_B(Q) = D_B(E_B(P)) = P$
RSA- Verfahren	Sicherheit basiert auf der Schwierigkeit eine grosse Natürliche Zahl in Primfaktoren zu zerlegen.
RSA (Vorgehen)	Wähle zwei grosse Primzahlen p, q. Berechne $n = pq$ und $z = (p-1)(q-1)$. Wähle Zahl d die relativ-prim zu z ist ($ggT(d, z) = 1$). Finde e so dass $ed \equiv 1(\mathrm{mod} z)$. Nachricht sei $m < n$. Verschlüsselung: $c \equiv m^e \mathrm{mod} n$. Entschüsselung: $c^d \equiv m^{ed} \equiv m \mathrm{mod} n$. B teilt A n, e mit. A berechnet $Q \equiv P^e (\mathrm{mod} n)$ A sendet verschlüsselte Meldung Q an B. B entschüsselt $P \equiv Q^d (\mathrm{mod} n)$
Diffie-Hellmann Schlüsseltausch	RSA zu teuer um grosse Datenvolumen zu verschlüsseln → AES, 3DES. Geheimer Schlüssel wird aber benötigt. Dieser wird mit RSA bestimmt. Schwierigkeit bei bekannten p, g, A den diskreten Logarithmus $a = \log_g(A)$ zu berechnen.Lange Mledungen mit RSA zu verschlüsseln ist langsam. Ausweg: Hash: Funktion, die eine grosse Eingabe eine Ausgabe kleineren Zielmenge (Hashwert) erzeugt.

Matlab Matlab (MATrix LABoratory) ist eine Hochleistungs-Sprache für technisch. wissenschaftliches Rechnen. Geeignet für Entwicklung von Algorithmen, Modellierung und Simulation, Datenanalyse und Visualisierung.

`A=ones(n)-eye(n)`	$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$
`I=find(A);A(I)=[1:length(I)];A = A'`	Nummerierung der 125 Elemente zeilenweise, die nicht Null sind.
`A = dlmread('a.dlm');`	laden der Datei a.dlm
`x = A(:,1); t = A(:,2:end);`	Kopieren der Matrix A in einen Vektor x und Vektor t. x = 0 bis 1. Spalte, t = 2. Spalte bis Ende

`plot(t); title('Temperaturen 2004');`
`legend('Luft(Mythen)', 'Maxima');`
`axis([0 186 0 30]) set(gca,'XTick',`
`1:14:185,'Fontsize',9) dateaxis('x',6,'05/01')`
	Funktion plotten, Titel, Legende

`[m,i] = max(t);plot(1:185,t,i,m,'ko');`	Plotten der Funktionsmaxima. k=schwarz, o = Kreis

`V=vander([1:4]);`
`f=cumsum([1:4].^2)';`
`a=V\f;`
`rat(a)`
$\begin{pmatrix} n_1^3 & n_1^2 & n_1 & 1 \\ n_2^3 & n_2^2 & n_2 & 1 \\ n_3^3 & n_3^2 & n_3 & 1 \\ n_4^3 & n_4^2 & n_4 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} f(n_1) \\ f(n_2) \\ f(n_3) \\ f(n_4) \end{pmatrix}$
	Polynominterpolation
	$\sum_{k=1}^{n} k = \frac{n(n+1)}{2} = \frac{1}{2}n + \frac{1}{2}n^2$
	$f(n) := \sum_{k=1}^{n} k^2 = a_+ n^3 + a_2 n^2 + a_3 n + a_4$
	$\sum_{k=1}^{n} k^2$

```
V=vander([1:5]);f=cumsum([1:5].^3)';a=V\f;rat(a)
```
vander

$$= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \\ 27 & 9 & 3 & 1 \\ 64 & 16 & 4 & 1 \end{pmatrix}$$

$$\sum_{k=1}^{n} k^3$$

Analog oben

```
for i=1:n,
 for j=1:n,
  if abs(i-j) > 1,
   t(i,j) = 0;
  elseif i == j,
   t(i,j) = 2;
  else
   t(i,j) = -1;
  end
 end
end
```

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

Alternative Lösung: `triu(tril(-ones(n)+3*eye(n),1)-1)`

```
c=n;
while n>1
 if mod(n,2)==0
  n = n/2;
 else
  n = 3*n+1
 end
 c = [c n];
end
```

Iteration von Collatz

$$f(n) = \begin{cases} 3n+1 & n \text{ gerade} \\ n/2 & n \text{ ungerade} \end{cases}$$

```
if n == 0, % Fall n=0 herausfiltern
 c = '0';
 return
end
vz = n < 0; % Vorzeichen bestimmen
if vz, n = -n; end
c = '';
while n>0,
 r = rem(n,2); %gleich wie Mod
 if r==0,
  c=['0' c];
 else
  c=['1' c];
 end
 n = (n - r)/2;
end
if vz, c=['-' c]; end % neg. Vorzeichen hinzufuegen
```

Darstellung von Dezimalzahl als Dualzahl

1. Falls $n = 0$ fertig, sonst: berechne $n = 2m + r$ mit $r \in \{0, 1\}$. r gibt die nächsthinterste Dualziffer.
2. $n \leftarrow m$
3. Gehe zu Schritt 1.

Numerisch Integrieren

```
n = 99;
h = pi/n;
x = linspace(0,pi,n+1);
y = 1-sin(x);
```

Intervalle
Stützstellen-Abstand
Stützstellen
Funktionswerte

```
tr = (y(1)/2+sum(y(2:n))+y(n+1)/2)*h;
```

Trapezregel $(b-1)\frac{f(a)+f(b)}{2}$

```
tr = trapez(x,y)
si=(y(1)+4*sum(y(2:2:n))+2*sum(y(3:2:n-1))+y(n+1))*h/3
```

MATLAB-Funktion *trapez*
Simpsonregel

$$(e-c)\frac{f(c)+4f(d)+f(e)}{6}$$

Formel für die Summe aller Flächenstücke:

$$\left(f(0)+f(nh)+\sum_{i=1,3,5,\ldots}^{n-1} 4f(ih)+\sum_{i=2,4,6,\ldots}^{n-2} 2f(ih)\right)\frac{2h}{6}$$

```
res = pi-2;
for k=1:16
 n=2^k;
```

Analytische Lösung für Konvergenz

Anzahl Stützstellen

```
 u2a3
 r(k) = n; t(k) = abs(tr - res); s(k) = abs(si - res);
end
loglog(r,[t;s]);
```

file enthält Befehle von 3-3.2

```
function y = kamel(x)
%KAMEL y = kamel(x) ist eine Funktion mit zwei Maxima
% bei x = 0.3 / 0.9 und einem Minimum bei 0.7.
y = 1 ./ ((x-.3).^2 + .01) - 1 ./ ((x-.7).^2 + .04)...
+ 1 ./ ((x-.9).^2 + .02) - 7;
```

Gesucht: Funktion *kamel*, welche für gegebene x den Wert y berechnet.

$$y = \frac{1}{(x-0.3)^2+0.01} - \frac{1}{x-0.7)^2+0.04} + \frac{1}{x-0.9)^2+0.02} - 7$$

```
function c = collatz(n)
%COLLATZ Collatz 3n+1 problem.
% c = collatz(n) berechnet die Collatz-Folge
% vom Startwert n bis zum Endwert 1 = c(end)
c = n;
while n > 1
  if rem(n,2) == 0
    n = n/2;
  else
    n = 3*n + 1;
  end
  c = [c n];
end
```

Gesucht: MATLAB-Funktion mit Eingabeparameter n und als Ausgabe c, inkl. Funktionsbeschreibung. Plotten der Funktion mit

```
for i=1:n; c=collatz(i); l(i) = length(c); end
bar(l)
```

```
function [x,fx] = bisect(f,a,b)
%BISECT berechnet eine Nullstelle x der Funktion f(x)
% im Intervall [a,b] unter der Annahme,
%dass f(a)f(b) < 0.
% [x,y] = bisect(@f,a,b), y = f(x)
```

```
fa = f(a); if fa == 0, x = a; fx = fa; return; end
fb = f(b); if fb == 0, x = b; fx = fb; return; end
if fa*fb > 0, x = NaN; fx = NaN; return; end
x = (a+b)/2; fx = f(x);
while a < x & x < b,
if fx == 0, break, end
if fa*fx < 0,
b = x; fb = fx;
else
a = x; fa = fx;
end
x = (a+b)/2; fx = f(x);
end
```

```
function yp = hasen_fuechse(t,y)
%LOTKA Lotka-Volterra Raeuber-Beute Modell.
%Angewandt auf Hasen und Fuechse (nicht Igel).
%
% y'(1) = 0.08*y(1) - 0.002*y(1)*y(2)
% y'(2) = -0.2*y(2) + 0.0004*y(1)*y(2)
yp = diag([0.08 - 0.002*y(2), -0.2 + 0.0004*y(1)])*y;
```

Lösen der Diferentialgleichung:

```
[t,y]=ode45(@hasen_fuechse,[0 200],[600; 20]);
```

$a = 0.08, b = 0.2, c = 0.002, d = 0.0004$ Anfangsbedingung Hasen: 500, Füchse: 20.

$$\frac{\mathrm{d}y_1}{\mathrm{d}t}(t) = ay_1 - cy_1y_2, \quad y_1(0) = y_1^{(0)}$$
$$\frac{\mathrm{d}y_2}{\mathrm{d}t}(t) = -by_2 + dy_1y_2, \quad y_2(0) = y_2^{(0)}$$

Plot

```
[ax,h1,h2]=plotyy(t,y(:,1),t,y(:,2));
set(h1,'LineStyle','-','LineWidth',2)
set(h2,'LineStyle','--','LineWidth',2)
set(ax(2),'YLim',[0 100],'YTick',[0:20:100])
set(ax(1),'YLim',[0 800],'YTick',[0:200:800])
set(get(ax(1),'Ylabel'),'String','Zahl der Hasen')
set(get(ax(2),'Ylabel'),'String','Zahl der Füchse')
```

Symbolisches Rechnen

```
syms x
diff(x^x)
```

x ist keine Variable mehr
Differenzieren (x^x nach x ableiten)

```
syms x
y=x^3-4*x^2 +4*x-1
ezplot(y)
solve(y)
z = diff(y)
ezplot(z)
solve(z)
```

Gegeben ist ein Polynom y. Gesucht werden die Nullstellen und die Extrema von y

```
syms x
f = x/(x^2+1)
int(f)
int(1/f)
```

Gesucht wird das unbestimmte Integral

```
syms k n
simplify(symsum(k,1,n))
simplify(symsum(k^2,1,n))
simplify(symsum(k^3,1,n))
```

Berechnung der Summen mit *symsum*

$$\sum_{k=1}^{n} k, \sum_{k=1}^{n} k^2, \sum_{k=1}^{n} k^3$$

```
syms a
A=[a 0 5;1 1 1;-a 0 0]
eig(A)
[U,L] = eig(A)
chi=poly(A)
solve(chi)
```

$$A = \begin{pmatrix} a & 0 & 5 \\ 1 & 1 & 1 \\ -a & 0 & 0 \end{pmatrix}$$

```
syms x A c1 c2 c3 h y y1 y2 y3
y = [y1;y2;y3]
A = [0, 0, 1;(h)^2, h, 1; (2*h)^2, 2*h, 1]
c = A\y;
f = c(3) + c(2)*x + c(1)*x^2
I = int(f,x,0,2*h)
pretty(simplify(collect(I,h)))
```

Gegebn sei ein Polynom. $f(x) = c_0x + c_1x + c_2x^2$. Berechnen der Koeffizienten so, dass $f(0) = y0, f(h) = y1 und f(2h) = y2$ gilt. Integration des Polynoms von 0 bis $2h$.

```
syms k t
s=dsolve('D2s = -k*Ds', 's(0) = 0', 'Ds(0) = 10')
Ds=diff(s)
s=subs(s,k,.12)
Ds=subs(Ds,k,.12)
ezplot(s,[0 50])
ezplot(Ds,[0 50])
grid
```

Am Anfang $(t = 0)$ ist der zurückgelegte Weg $s(0) = 0$, die Geschwindigkeit $s'(0) = 10m/sec$. Zeichnen Sie den Verlauf der Wegkurve. Lösen Sie dazu die Gleichung zuerst vollständig symbolisch, und setzen Sie erst am Schluss die gegebenen Parameter ein. (Befehle: dsolve, subs, ezplot).

```
solve(Ds-1.5,t)
double(ans)
```

Wie weit wird das Boot sich bewegt haben, wenn die Geschwindigkeit auf $1.5m/s$ gefallen ist?

```
limit(s,t,inf)
```

Nach wie vielen Metern bleibt das Boot stehen?

Einige Berechnungen

```
A = [1, 1; 1, 2]
```

B = A*A = Vektormultiplikation → B = [3, 6; 3, 6]
C = A.*A = Komponentenweise Multiplikation → A = [1, 1; 4, 4].

Ziel: Summe $\sum_{k=1}^{n} k^2$ berechnen. Vorgehen: Zuerst Vander-Matrix und die rechte Seite der Vander-Matrix berechnen. Die Rechte Seite ist dabei der Summenfunktionswert. Man wählt ein beliebiges n. z.B. hier $n = 3$. Das führt zu einem Gleichungssystem. Bei vier unbekannten brauchen wir vier Gleichungen also vier Matrixzeilen. Das Gleichungssystem kann nun gelöst werden. b = vander([1:4])	$$\begin{bmatrix} n_1^3 & n_1^2 & n_1 & 1 \\ n_2^3 & n_2^2 & n_2 & 1 \\ n_3^3 & n_3^2 & n_3 & 1 \\ n_4^3 & n_4^2 & n_4 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} f(n_1) \\ f(n_2) \\ f(n_3) \\ f(n_4) \end{bmatrix}$$ Rechte Seite für $n = 3$: $$\sum_{k=1}^{3} k^2 = 1^2 + 2^2 + 3^2 = 1 + 4 + 9 = 14$$
$$\begin{bmatrix} 13 & 1 & 1 & 1 \\ 8 & 4 & 2 & 1 \\ 27 & 8 & 3 & 1 \\ 64 & 16 & 4 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 14 \\ 30 \end{bmatrix}$$	
`x=A\b`	MATLAB liefert die Lösung von $A^T r = 0 \rightarrow (A^T A)x = A^T b$
`x=inv(A)*b`	
`det(A)`	$detA = detP \cdot detU = \pm detU$

MATLAB, HTML, LaTeX

`[row, col]=size(A)`	Grösse einer Tabelle auslesen
`for i=1:0.1:100`	For-schlaufe von 1 bis 100 in 0.1 Schritten
`x=[x i];`	
`end`	
`whos`	Liefert Informationen über alle definierten Variablen
`who a`	liefert Informationen über die Variable a
`clear`	Löschen aller definierten Variablen
`clc`	Löschen des Kommandofensters
`nargin`	enthält Anzahl der übergebenen Variablen einer Funktion
`length(y)`	Länge eines Vektors
`eye(4)`	Einheitsmatrix
`ones(4)`	Matrix mit 1.
`zeros(3)`	quadratische Nullmatrix der entsprechenden Grösse
`zeros(2,3)`	Nullmatrix, hier mit 2 Zeilen und 3 Spalten; erzeugt auch Nullvektoren (sind ja auch Matrizen)
`sparse(7)`	dünn besetzte Matrix (anderes, sparsames Speicherformat); Syntax wie zeros
`z(2,3)=-0.7^3`	Zugriff auf einzelne Komponenten einer Matrix (Zeile,Spalte)
`z(2,:)`	Zugriff auf ganze Zeilen-/Spaltenvektoren einer Matrix (hier 2. Zeile)
`z(:,1)=[]`	Löschen einer Matrixspalte (Zeile analog)
`M(2:4,6:10)`	Zugriff auf Matrixbereich (Teilmatrix); auch mit beliebigen Indexvektoren: M(1:3,[1 4:6 8])
`x'`	Transponieren
`sum(x)`	Addition aller Einträge eines Vektors, bei Matrizen spaltenweise
`det(M)`	Determinante von M
`diag(A)`	liefert alle Elemente auf der Hauptdiagonalen von A
$A\backslash b$	liefert Lösung x der Gleichung $Ax = b$. Linksdivision.
`inv(M)`	Inverse von M
`eig(M)`	Eigenwerte von M als Spaltenvektor
`poly(M)`	charakteristisches Polynom (Koeffizienten als Zeilenvektor)
`any(x)`	liefert 1 (wahr), wenn x mindestens einen Eintrag $\neq$ 0 hat, sonst 0 (falsch)
`all(x)`	liefert 1, wenn alle Einträge von x von 0 verschieden sind, sonst 0
`find(x==1)`	liefert Indizes aller x(n), die gleich 1 sind, als Vektor
`plot(x,y)`	Plottet Vektor y gegenüber Vektor x (2D)
`xlabel('x-Achse')`	Achsenbezeichnung des aktuellen Plots setzen
`hold on`	Ermöglicht 2 Plots in einem, danach hold off setzen!
`mesh(x,y,z)`	Plottet Werte der Matrix z mit Achsen x bzw. y (3D), einfarbig
`surf(x,y,z)`	wie mesh, nur mit Oberflächenfarbe (3D); Beleuchtung möglich
`isequal(C,[1 2])`	ist bei Schleifen sauberer

HTML

```
<a href="#"></a>
<a name="">
<calgroup>
<col width="100%"></col>
</colgroup>
<ol><li></li></ol>
<ul><li></li></ul>
```

LaTeX

```
\documentclaass[a4paper]{article}
\usepackage{german,graphicx}
\usepackage{amsmath, amssymb}
\title{TITEL}
\author{AUTOR}
\date{DATUM}
\begin{document}
\tableofcontents
\section{Kapitel}
\subsection{Unterkapitel}
\begin{itemize}
  \item Liste
 \end{itemize}
\begin{enumerate}
\item Nummerierte Liste
\end{enumerate}
\begin{equation}\label{eq:1}
\end{equation}
\eqref{eq:1}\ref{REFERENZ}
\cite{BUCH}
\label{LABEL}\ref{REFERENZ}
\includegraphics{bild.jpg}
\\begin{figure}[h]\centering
\includegraphics[width=.5
\textwidth]{bild.jpg}
\caption{Beschr.}\label{LABEL}
\end{figure}
\cite{paper1}
\begin{thebibliographiy}{9}
\bibitem{paper1}\label{b1}
\newcommand{\befehl}[#arg]{Def}
```

RSA-Verfahren		
Allgemein	Konkret	
Wähle Primzahlen Eulerfunktion $\varphi(n)$ berechnen $z = \varphi(n) = (p-1)(q-1)$ Public Key wählen Public Key · Private key = 1(mod z) Private Key	$p = 5, q = 17$ $z = 64$ Public Key $= 13$ 13 · Private key = 1 (mod 64) $\Rightarrow$ Private Key = 5	
Verschlüsslung nach Caesar		
`abcdefghijklmnopqrstuvwxyz` `DEFGHIJKLMNOPQRSTUVWXYZABC`	Kleinbuchstaben: Klartext. n=3	