# Hands-On-Workshops/Mathematische Dialoge
# Fachgruppe Mathematik, RWTH Aachen

## Glossar für Julia-Befehle

| Befehl | Beschreibung | Beispiel |
|---|---|---|
| `a = b` | $a := b$, neue Variable a wird definiert als Ausdruck b, "Zuweisungsoperator" | `a = 2` |
| `print(x)`<br>`println(x)` | Ausgabe von $x$<br>Ausgabe von $x$ mit Zeilenumbruch | |
| `using SymPy` $(\star)$ | müssen wir ausführen, bevor wir mit $(\star)$ markierte Befehle verwenden können | |

### Grundlagen Arithmetik

| | | |
|---|---|---|
| `i + j` | berechnet $i + j$ | |
| `i - j` | berechnet $i - j$ | |
| `i * j` | berechnet $i \cdot j$ | |
| `i / j` | berechnet $\frac{i}{j}$ gerundet | |
| `i // j` | berechnet $\frac{i}{j}$ genau (bei symbolischen Rechnen benutzen) | |

### Logische Ausdrücke

| | | |
|---|---|---|
| `a == b` | prüft $a = b$ (nicht mit Zuweisungsoperator = verwechseln!) | `3 == 3 ## true` |
| `a != b` | prüft $a \neq b$ | `4 != 5 ## true` |
| `a < b` | prüft $a < b$ | `2 < 2 ## false` |
| `a > b` | prüft $a > b$ | `5 > 4 ## true` |
| `a <= b` | prüft $a \leq b$ | `3 <= 2 ## false` |
| `a >= b` | prüft $a \geq b$ | `4 >= 4 ## true` |
| `A \|\| B` | prüft $A \vee B$, logisches "oder" | `3 <= 2 \|\| 3 > 2 ## true` |
| `A && B` | prüft $A \wedge B$, logisches "und" | `3 <= 2 && 3 > 2 ## false` |
| `!A` | logisches "Nicht" der Aussage A | `!(3<=2) ## true` |

### Wichtige Funktionen

| | | |
|---|---|---|
| `sqrt(x)` | $\sqrt{x}$ | `sqrt(4) ## 2` |
| `x^n` | $x^n$ | `2^3 ## 8` |
| `factorial(n)` | $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ | `factorial(3) ## 6` |
| `abs(x)` | Absolutbetrag von x | `abs(-5) ## 5` |
| `n % m` | n mod m | `5 % 2 ## 1` |
| `sum(f(i) for i=a:b)` | $\sum_{i=a}^{b} f(i)$ | `sum(2n for n=1:3)`<br>`## 2 + 4 + 6 = 12` |
| `prod(f(i) for i=a:b)` | $\prod_{i=a}^{b} f(i)$ | `prod(m^2 for m=2:4)`<br>`## 2^2 * 3^2 * 4^2 = 576` |
| `ceil(x)` | obere Gaußklammer $\lceil x \rceil$ | `ceil(2.3) ## 3` |
| `floor(x)` | untere Gaußklammer $\lfloor x \rfloor$ | `floor(2.7) ## 2` |
| `sin(x)` | Sinus von $x$ | `sin(0) ## 0` |
| `cos(x)` | Kosinus von $x$ | `cos(0) ## 1` |
| `tan(x)` | Tangens von $x$ | `tan(0) ## 0` |
| `exp(x)` | $e^x$ | `exp(1) ## e = 2,718...` |
| `log(x)` | natürlicher Logarithmus $ln(x)$ | `log(1) ## 0` |
| `log(p, x)` | $log_p(x)$ | `log(2, 8) ## 3` |
| `min(a, b)` | Minimum von $a$ und $b$ | `min(2,5) ## 2` |
| `max(a, b)` | Maximum von $a$ und $b$ | `max(2,5) ## 5` |

## Symbolisches Rechnen

| | | |
|---|---|---|
| `a = symbols("a")` (⋆) | definiert eine symbolische Variable | |
| `expand(b)` (⋆) | b ausmultiplizieren | `expand((a + 1)^2)`<br>`## a^2 + 2a +1` |
| `factor(b)` (⋆) | b faktorisieren | `factor(a^2 − 1)`<br>`## (a+1)(a-1)` |
| `simplify(b)` (⋆) | b vereinfachen | `simplify(sin(a)^2+cos(a)^2)`<br>`## 1` |

## Konstanten

| | | |
|---|---|---|
| `pi` | Kreiszahl $\pi$ | |
| `exp(1)` | Eulersche Zahl $e$ | |
| `im` | imaginäre Einheit $i$ | `z = 1 + 2 * im` |
| `oo` | Unendlich $\infty$ | |

## Arrays

| | | |
|---|---|---|
| `[...]` | Erzeugt Array | `a = [1, 3, 3]`<br>`## Array mit 3 Elementen` |
| `[]` | Leeres Array | |
| `[f(i) for i=1:n]` | Array mit for-Bedingung | `c = [2i for i=1:10]`<br>`## [2, 4, ..., 20]` |
| `a[i]` | Zugriff auf das i. Element des Arrays | `c[2] ## 4` |
| `vcat(a, b)` | Array a und b aneinanderhängen | `vcat(a, a)`<br>`## [1, 3, 3, 1, 3, 3]` |

## Arrays und Mengen

| | | |
|---|---|---|
| `Set(a)`<br>`[M ...]`<br>*oder* `collect(M)` | macht Array a zu einer Menge<br>macht Menge M zu einem Array | `M = Set(a) ## {1, 3}`<br>`d = [M...]  ## [1, 3]` |
| `length(X)`<br>`sum(X)`<br><br>`prod(X)` | Länge eines Arrays/ einer Menge X<br>Summe der Elemente aus Array/ Menge X<br>Produkt der Elemente aus Array/ Menge X | `length(M) ## 2`<br>`sum(M) ## 4`<br><br>`prod(M) ## 3` |
| `minimum(X)`<br>`maximum(X)` | Minimum von Array/ Menge X<br>Maximum von Array/ Menge X | `minimum(M) ## 1`<br>`maximum(M) ## 3` |

## Mengen

| | | |
|---|---|---|
| `union(M, N)` | $M \cup N$ | `N = union(M, Set([2]))`<br>`## {1, 2, 3}` |
| `intersect(M, N)` | $M \cap N$ | `intersect(Set([1]), M)`<br>`## {1}` |
| `setdiff(M, N)` | $M \setminus N$ | `setdiff(N, M) ## {2}` |
| `issubset(M, N)` | prüft $M \subseteq N$ | `issubset(M, N) ## true` |
| `in(x, N)`<br>*oder* `x in N` | prüft $x \in M$ | `in(1, M) ## true`<br>`4 in M ## false` |

## Lineare Algebra: Vektoren, Matrizen, LGS

| | | |
|---|---|---|
| `Sym[ ; ; ; ]` (⋆) | Matrix/ Vektor (Zeilen mit ; trennen)<br><br>man kann auch Matrizen/ Vektoren in `Sym[] reinschreiben` | `A = Sym[1 2; 3 4]`<br>`## A =` $\begin{pmatrix}1 & 2\\3 & 4\end{pmatrix}$<br>`Sym[A A] ## 2x4 Matrix`<br>`Sym[A;A] ## 4x2 Matrix` |
| `A[i, j]` (⋆) | Zugriff auf Element in i. Zeile und j. Spalte von Matrix A | `A[2, 1] ## 3` |
| `A[:, j]` (⋆) | Zugriff auf j. Spalte von Matrix A | `v= A[:, 1]`<br>`## v =` $\begin{pmatrix}1\\3\end{pmatrix}$ |
| `A[i, :]` (⋆) | Zugriff auf i. Zeile von Matrix A | `w = A[2, :]`<br>`## w =` $\begin{pmatrix}3\\4\end{pmatrix}$ |
| `A.T` (⋆) | Vektor/ Matrix A transponieren | `w.T ##` $\begin{pmatrix}3 & 4\end{pmatrix}$ |
| `sympy.eye(n)` (⋆) | $n \times n$ Einheitsmatrix | `Id = sympy.eye(3)`<br>`## Id ist 3x3 Einheitsmatrix` |
| `v + w` | komponentenweise Addition der Vektoren $v$ und $w$ | |
| `v - w` | komponentenweise Subtraktion der Vektoren $v$ und $w$ | |
| `a * v` | Skalare Multiplikation vom Skalar $a$ und Vektor $v$ | |
| `A * v` | Matrix-Vektor-Multiplikation | |
| `A * B` | Matrixmultiplikation | |
| `A.rref()[1]` (⋆) | strikte Zeilenstufenform von A | |
| `A.nullspace()` (⋆) | löst homogenes LGS $A \cdot x = 0$ nach $x$ auf und gibt eine Basis des Lösungsraums aus | |
| `A.det()` (⋆) | Determinante einer $n \times n$ Matrix $A$ | `A.det() ## -2` |
| `A.inv()` (⋆) | $A^{-1}$ | `B = A.inv()`<br>`## B =` $\begin{pmatrix}-2 & 1\\1.5 & -0.5\end{pmatrix}$ |
| `A.eigenvals()`(⋆) | Eigenwerte einer $n \times n$ Matrix $A$ | |

## Algebra: Zahlentheorie

| | | |
|---|---|---|
| `lcm(a, b)` | kleinste gemeinsamer Vielfache/ least common multiple von $a$ und $b$ | `lcm(4, 6) ## 12` |
| `gcd(a, b)` | größter gemeinsamer Teiler/ greatest common divisor von $a$ und $b$ | `gcd(10,15) ## 5` |
| `gcdx(a, b)` | Bezout-Koeffizienten und ggT bestimmen | `ggT, x, y = gcdx(10, 15)`<br>`## ggT = 5, x = -1, y = 1` |
| `divrem(a, b)` | Bestimmt $c$ und $r$ so, dass $a = b \cdot c + r$ | `c, r = divrem(7, 3)`<br>`## c = 2, r = 1` |
| `div(a, b)` | bestimmt $c$ so, dass $a = b \cdot c + r$ (div = Divisor) | `c = div(9, 4) ## 2` |
| `rem(a, b)` | bestimmt Rest bei Division von a durch b (rem = remainder) | `r = rem(9, 4) ## 1` |

### Analysis

| | | |
|---|---|---|
| `limit(seq, n, p)` (⋆) | Grenzwert $\lim_{n \to p} seq$ | `limit(1/n, n, oo) ## 0`<br>`## oo steht für` $\infty$ |
| `summation(a, (i,n,m))` (⋆) | $\sum_{i=n}^{m} a_i$ Wert der Summe/ Reihe symbolisch berechnen | `summation(1/i, (i, 1, oo))`<br>`##` $\infty$ |
| `diff(f, x)` (⋆) | Ableitung von $f$ nach $x$ | `diff(x^2) ## 2x` |
| `integrate(f, x)` (⋆) | eine Stammfunktion von $f$ bzgl. $x$ | `integrate(x^2, x) ## x^3 / 3` |
| `integrate(f,(x, a, b))` (⋆) | $\int_a^b f dx$ | `integrate(x^2, (x, 0, 2))`<br>`## 8/3` |
| `solve(f, x)` (⋆) | löst die Gleichung $f = 0$ nach $x$ auf | `solve(x^2 - 4, x) ## [2, -2]`<br>`## Lösung von x^2 = 4` |

## Kontrollstrukturen

| | |
|---|---|
| `if Bedingung`<br>`  Anweisung`<br>`end` | **if-Bedingung**<br>Wenn Bedingung wahr ist, führe Anweisung aus. |
| `if Bedingung`<br>`  Anweisung1`<br>`else`<br>`  Anweisung2`<br>`end` | **if-else-Verzweigung**<br>Wenn Bedingung wahr ist, führe Anweisung 1 aus.<br>Sonst, führe Anweisung 2 aus. |
| `if Bedingung1`<br>`  Anweisung1`<br>`elseif Bedingung2`<br>`  Anweisung2`<br>`else`<br>`  Anweisung3`<br>`end` | **if-elseif-Verzweigung**<br>Wenn Bedingung 1 wahr ist, führe Anweisung 1 aus.<br>Wenn Bedingung 1 falsch und Bedingung 2 wahr ist, führe Anweisung 2 aus.<br>Wenn weder Bedingung 1, noch Bedingung 2 wahr sind, führe Anweisung 3 aus.<br>(Kann mit beliebig vielen elseif-Bedingungen weitergeführt werden). |
| `for i in M`<br>`  Codeblock`<br>`end` | **for-Schleife**<br>Iteriert über M (z.B. Array) und führt den Codeblock für jedes Element in M einmal aus. |
| `while Bedingung`<br>`  Anweisung`<br>`end` | **while-Schleife**<br>Solange die Bedingung wahr ist, soll die Anweisung ausgeführt werden. |

## Plotten

| | |
|---|---|
| `using Plots` (⋆) | müssen wir ausführen, bevor wir mit (⋆) markierte Befehle verwenden können |
| `plot(f)` (⋆) | Symbolischen Ausdruck f plotten |
| `plot(a, b, seriestype =:scatter)` (⋆) | Punkte-/ Streudiagramm, Array a ist Definitionsbereich und Array b ist Zielbereich |
| `plot!(...)` (⋆) | bereits bestehenden Plot mit anderen Graphen/ Streudiagrammen ergänzen |

**Folgende Parameter (⋆) können in plot(...) benutzt werden:**

| | |
|---|---|
| `xlim = (a, b)` | Endpunkte der Abzisse sind a und b |
| `ylim = (a, b)` | Endpunkte der Ordinate sind a und b |
| `label = "abc"` | gibt der Funktion den Namen "abc" |
| `xlabel = "x"` | Beschriftung der Abzisse mit "x" |
| `ylabel = "y"` | Beschriftung der Ordinate mit "y" |
| `title = "T"` | Beschriftung des Plots mit "T" |
| `color=:green` | Farbe der Punkte/ der Linie festlegen (auch andere Farben möglich) |
| `grid = false` | Hilfslinien im Plot entfernen |
| `background_color = :lightgrey` | Hintergrundfarbe festlegen (auch andere Farben möglich) |
| `linewidth = n` | Linienstärke im Liniendiagramm anpassen |
| `linestyle =:dash` | Linienart im Liniendiagramm festlegen (auch möglich: solid, dot, dashdot, dashdotdot, ...) |
| `shape =:+` | Form der Punkte festlegen (auch möglich: o, s, d, r...) |
| `markersize = n` | Größe der Punkte anpassen |
| `markerstrokecolor =:black` | Farbe der Umrandung der Punkte (auch andere Farben möglich) |
| `markerstrokewidth = n` | Stärke der Umrandung der Punkte |
| `alpha = n` | Transparenz der Punkte festlegen (0 durchsichtig - 1 komplett sichtbar) |
| `proj=:polar` | macht Koordinatensystem zu einer "Zielscheibe", gut zum Ablesen von komplexen Zahlen |