Cheatsheet Julia

Grundlagen Operations Research

Kompakte Referenz der Julia-Bausteine, die in diesem Kurs vorkommen. Ausführliche Erklärungen mit ausgeführten Beispielen: Erste Schritte.

Variablen und Typen

```
x = 1                # ganze Zahl (Int64)
y = 19.99            # Kommazahl (Float64)
name = "Julia"       # Zeichenkette (String)
ok = true            # Wahrheitswert (Bool)

typeof(x)            # Typ abfragen
Float64(42)          # Int -> Float
Int64(3.0)           # Float -> Int (muss ganzzahlig sein)
round(Int, 2.7)      # runden und konvertieren
```

Zeichenketten und Interpolation

```
name = "Auftrag 1"
kosten = 25
println("$name kostet $kosten EUR")        # Werte einsetzen
println("Summe: $(kosten + 5) EUR")        # Ausdruck einsetzen
```

Vektoren, Matrizen und Tupel

Vektoren

```
d = [140, 110, 190, 120]   # Vektor anlegen

d[1]              # erstes Element (Julia zählt ab 1!)
d[end]            # letztes Element
d[2:3]            # Ausschnitt
length(d)         # Anzahl Elemente
sum(d)            # Summe
maximum(d)        # größtes Element
minimum(d)        # kleinstes Element

push!(d, 200)     # Element anhängen (! = verändert d)
sort(d)           # sortierte Kopie
zeros(4)          # [0.0, 0.0, 0.0, 0.0]
```

Matrizen

```
c = [130 140 170;          # Spalten: Leerzeichen,
     210 150 130]          # Zeilen: Semikolon

c[2, 3]           # Element (Zeile 2, Spalte 3)
c[1, :]           # ganze Zeile 1
c[:, 2]           # ganze Spalte 2
size(c)           # (2, 3) = Zeilen x Spalten
```

Tupel

```
kante = (1, 2, 55)         # unveränderlich, feste Länge
von, nach, laenge = kante  # Mehrfachzuweisung
kante[3]                   # Zugriff per Index
```

Ranges

```
T = 1:5           # Zahlen 1 bis 5 (Indexmenge)
1:2:9             # 1, 3, 5, 7, 9 (Schrittweite 2)
collect(T)        # Range -> Vektor
```

Vergleiche und Logik

```
x == y            # gleich
x != y            # ungleich
x < y;  x <= y    # kleiner (gleich)
x > y;  x >= y    # größer (gleich)
0 <= x <= 10      # verkettete Vergleiche

a && b            # und
a || b            # oder
!a                # nicht
isapprox(x, 4.0; atol = 1e-4)   # Gleitkomma-Vergleich mit Toleranz
```

Verzweigungen

```
if bestand == 0
    println("leer")
elseif bestand < 10
    println("fast leer")
else
    println("gefüllt")
end

status = bestand > 0 ? "ok" : "leer"   # Kurzform (ternär)
```

Schleifen und Comprehensions

```
for t in 1:3                # über Range
    println(t)
end

for k in kanten             # über Vektor
    println(k)
end

while bestand > 0           # solange Bedingung gilt
    bestand -= 1
end

# Generator/Comprehension, das Summenzeichen des Kurses:
sum(d[t] for t in 1:4)             # Summe mit Laufindex
sum(d[t] for t in 1:4 if t != 2)   # Summe mit Bedingung
[t^2 for t in 1:5]                 # Vektor per Vorschrift
```

Dictionaries

```
pos = Dict(1 => (0.0, 1.5),    # Schlüssel => Wert
           2 => (2.5, 3.0))

pos[1]                 # Zugriff per Schlüssel
pos[3] = (5.0, 0.0)    # Eintrag ergänzen
haskey(pos, 9)         # Schlüssel vorhanden?
for (knoten, xy) in pos    # über Einträge laufen
    println("$knoten liegt bei $xy")
end
```

Funktionen

```
function deckungsbeitrag(erloes, kosten)
    return erloes - kosten     # return optional:
end                            # letzter Ausdruck zählt

quadrat(x) = x^2               # Kurzform für Einzeiler
```

Broadcasting (Punkt-Operator)

```
mengen .* preise      # elementweise multiplizieren
d .+ 10               # 10 auf jedes Element addieren
d .>= 100             # elementweiser Vergleich
value.(X)             # Funktion auf jedes Element (JuMP!)
```

Pakete

```
# Einmalig installieren (in der REPL: ] add JuMP HiGHS Plots)
using Pkg
Pkg.add("JuMP")

# In jedem Skript laden:
using JuMP, HiGHS, Plots
```

Plots (Grundlagen)

```
using Plots

plot(x, y; xlabel = "X1", ylabel = "X2")   # Linienplot
plot!(x, y2)                # weitere Serie ergänzen (!)
scatter(x, y)               # Punkte
scatter!(x, y)              # Punkte ergänzen
annotate!(2, 3, text("F", 10))   # Beschriftung an (2, 3)
savefig("plot.png")         # speichern
```

> Tipp
>
> Für Optimierungsmodelle (JuMP, HiGHS) gibt es ein eigenes Blatt: Cheatsheet JuMP.

Bibliografie