copyright by QUNIS

02 Loops, If-then-else und Funktionen

Martin Hanewald

2019-02-19

IF THEN ELSE

```
dat <- 3

if(is.numeric(dat)){
    dat <- dat + 5
    cat('Die Zahl lautet', dat)
}else{
    cat(dat, 'ist keine Zahl!')
}
#> Die Zahl lautet 8
```

FOR LOOPS

```
names <- c('Harry', 'Bert', 'Murk')

for(n in names){
    cat(n, 'ist ein wunderschöner Name.\n')
}
#> Harry ist ein wunderschöner Name.
#> Bert ist ein wunderschöner Name.
#> Murk ist ein wunderschöner Name.

for(n in 1:length(names)){
    cat('Der', n, '. Name lautet', names[[n]], '\n')
}
#> Der 1 . Name lautet Harry
#> Der 2 . Name lautet Bert
#> Der 3 . Name lautet Murk

for(n in seq_along(names)){
    cat(n)
}
#> 123
```

FUNKTIONEN

```
myfunc <- function(){
    cat('Diese Funktion ist sinnlos.')
}

myfunc()
#> Diese Funktion ist sinnlos.

cylinder_volume <- function(r, h){
    vol <- pi * r^2 * h
    return(vol)
```

CA controller akademie®

```
}

cylinder_volume(5.5, 20)
#> [1] 1900.664
```

APPLY

```
df <- data.frame(a=rnorm(10), b = rnorm(10), c = rnorm(10))

df
#>             a          b           c
#> 1  -1.4716206 -1.9918579  1.18819811
#> 2  -0.6865134 -0.5381026 -1.43548630
#> 3   0.2476627 -1.5368141  1.13263244
#> 4  -0.5301211 -1.4121818 -0.04986148
#> 5   0.3738892  0.4117586  0.71504782
#> 6  -1.2518694  0.1525651 -0.73350210
#> 7  -1.3199694 -0.4902913 -1.18916517
#> 8   0.4559150  1.9933542 -0.53554275
#> 9   0.4909932  1.3147484 -0.13624744
#> 10 -0.1042614  0.5873565  0.23527954
```

Herausforderung: Berechne Mittelwert über

a. alle Zeilen
b. alle Spalten

Mögliche Lösung: Einzelaufruf

```
mean(df[,1])
#> [1] -0.3795895
mean(df[,2])
#> [1] -0.1509465
mean(df[,3])
#> [1] -0.08086473

# Für Berechnung der Zeilen muss erst noch transponiert werden.
mean(t(df[1,]))
#> [1] -0.7584268
mean(t(df[2,]))
#> [1] -0.8867008
mean(t(df[3,]))
#> [1] -0.05217299
```

Oder: Schleifenkonstruktion

```
for(k in 1:3){
    cat(mean(df[,k]))
    cat(mean(t(df[k,])))
}
#> -0.3795895-0.7584268-0.1509465-0.8867008-0.08086473-0.05217299
```

Aber es geht auch eleganter:

```
apply(df, 1, mean)
#>  [1] -0.75842679 -0.88670075 -0.05217299 -0.66405481  0.50023187
#>  [6] -0.61093546 -0.99980863  0.63790883  0.55649805  0.23945824
apply(df, 2, mean)
```

```
#>           a           b           c
#> -0.37958952 -0.15094648 -0.08086473
```

Die `apply` Funktion wendet die Funktion `mean` über alle Zeilen (`1`) oder Spalten (`2`) von `df` an.

Aufgabe: Schreibt eine Funktion die die Fibonacci Reihe erzeugt (20 min)

Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

```
fibonacci <- function(x){
    ans <- rep(1,x)
    if(x > 2){
        for(k in 3:x){
            ans[k] <- ans[k-2] + ans[k-1]
        }
    }
    return(c(0,ans))
}

fibonacci(10)
#>  [1]  0  1  1  2  3  5  8 13 21 34 55
```