Programmieren mit R für Einsteiger

2. Datentypen / 2.1 Funktionen

Berry Boessenkool 2022-03-08 13:17

Funktionen: Code systematisiert wieder verwendbar machen

```
dividiere <- function(zahl, divisor=5)
  {                          AltGr + 7/0, Option + 8/9
  ausgabe <- zahl / divisor
  ausgabe <- round(ausgabe, digits=4)
  return(ausgabe)
  }
```

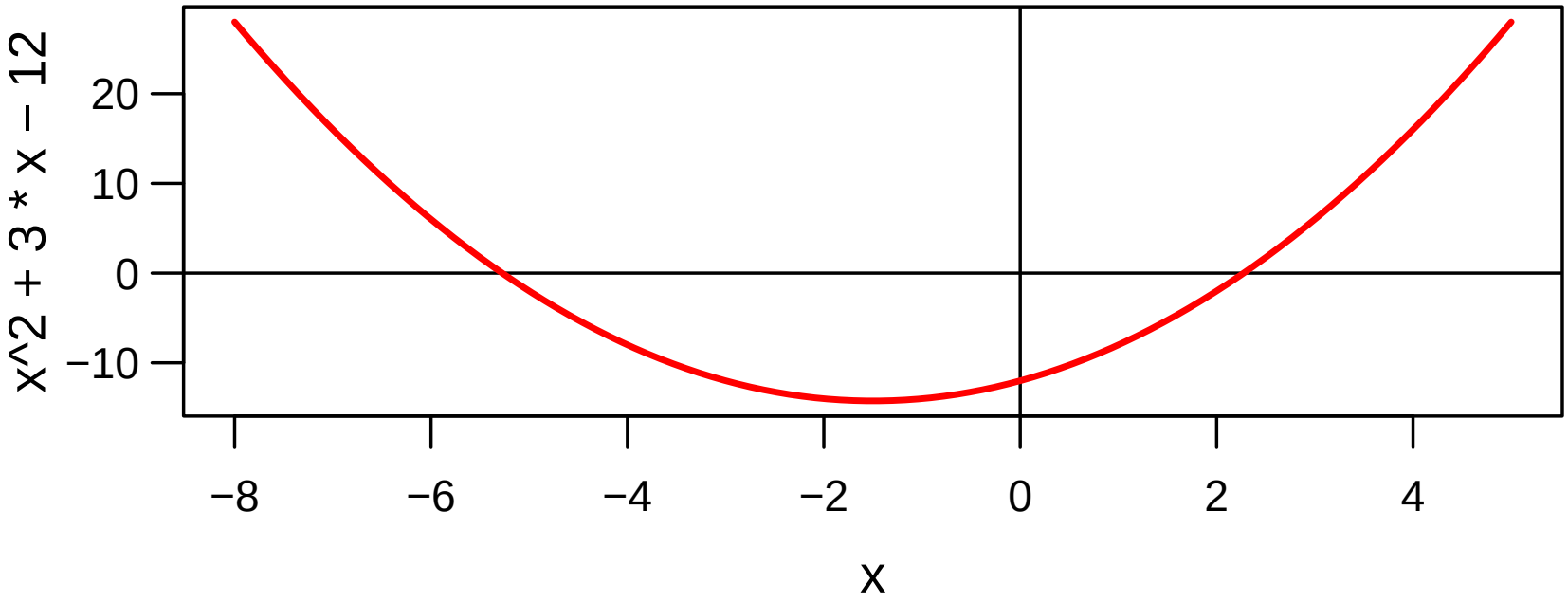
Eigene Funktion ausführen (wie andere Befehle auch):

```
dividiere(zahl=23, divisor=7) # jetzt aufrufbar
## [1] 3.2857
```

```
dividiere(23) # Standardwert (5) für 'divisor' verwendet
## [1] 4.6
```

Mit Funktionen kann die Prüfroutine in den Übungsaufgaben euren Code für verschiedene Inputs ausführen und prüfen.

Beispiel: PQ-Formel für Nullstellen einer quadratischen Funktion




```
pq <- function(p,q) # y = x^2 + px + q
    {
    w <- sqrt( p^2 / 4 - q )
    c(-p/2-w, -p/2+w)
    }
```

```
pq(3, -12)
## [1] -5.274917  2.274917
```

Ausführumgebung (environment) für Objekte

```
dividiere <- function(zahl, divisor=5) # Bsp. von vorhin
  {
  ausgabe <- zahl / divisor
  ausgabe <- round(ausgabe, digits=4)
  return(ausgabe)
  }
```

Der Code innerhalb der Funktion (body) wird von R in einem separaten Environment ausgeführt.
Objekte innerhalb der Funktion (`zahl` , `divisor` & `ausgabe` im Beispiel) sind lokal bzw. temporär und werden nicht im normalen Workspace (global environment) angelegt.
Sie sind danach nicht im `globalenv()` verfügbar.

Return / geschweifte Klammern

`return()` beendet die Ausführung der Funktion. Code danach wird nicht aufgerufen.
`return()` kann weggelassen werden, dann wird das Ergebnis der letzten Anweisung (statement, "expression") zurück gegeben:

```
dividiere <- function(number, divisor=5){
  output <- number / divisor
  round(output, digits=4)
  }
```

Bei Funktionen, die nur eine einzige Anweisung (expression) enthalten, sind die geschweiften Klammern optional:

```
normalisiere <- function(x) (x-min(x)) / (max(x)-min(x))
```

```
# von -7 bis 13 normieren auf 0 bis 1
normalisiere(  c(8,-7,13,2,3)  )
## [1] 0.75 0.00 1.00 0.45 0.50
```

Zusammenfassung

Funktionen um Code mehrfach zu verwenden:

- `meineFunktion <- function(x) {x+7}`
- Workspace: Objekte normalerweise im "global environment"
- Objekte innerhalb einer Funktion in temporärer Umgebung (environment)
- `return` am Ende von Funktionen ist optional
- `{}` sind optional, wenn die Funktion nur einen Befehl enthält