Beispiel in Java

```
public static double add(double x,double y){
return x+y;
}
```

Einfache Funktion, die nur als Methode in einer Klasse vorzuführen ist.
->double jedoch zu ungenau

Beispiel in Racket:

```
(define(add x y) (+ x y))
```

- `define` ist Schlüsselwort für das Definieren einer Funktion / einer Konstanten
- `(add x y)`:
  - `add` ist Name,
  - `x y` sind die formalen Parameter der Funktion
- `(+ x y)` -> Funktionalität der Funktion -> in Racket **Präfixnotation
- () => Jede Syntaktische Einheit, wird in Klammern gesetzt:
- **Aufrufen der Funktion:**
- `(add 2.71 3.14)`

Konstanten

```
(define my-pi 3.141) ; Kommentare mit Semikolon
(define my-pi (+ 3 0,14159)) ; Aquivalent zum oberen Ausdruck
```

- **Identifier**
  - nicht erlaubt : ==( ) [ ] { } “ , ´ ; # | ,
  - White spaces sowie Zahlen auch nicht
  - sonst alles erlaubt, z.B. `42A`
- **Konvention :**
  - Keine Großbuchstaben, Bindestriche zwischen einzelnen Wörtern

---

Zahlen

- Exakte Zahlen werden soweit wie möglich dargestellt
- Komplexe Zahlen gibt es
- irrationale Zahlen werden nichtexakt dargestellt.

Arithmetische Operationen

Präfixnotation, Operator links, Operanden rechts, müssen in Klammern geschrieben Werden

`(+ x y), (- x y), (* y x), (/ x y)` -> Grundrechenarten
`(modulo x y)` ->Analog zu Java

- beliebig viele Operanden sind in die Operation einsetzbar

`(- 3 4 5) ; 3-(4+5) = 6` (/ 3 4 5) ; 3/4*5 = 3/20 `

- **Wenn ein Ergebnis nicht exakt ist, wird es durch ein führendes `#i` markiert**

  `sqrt5 → #i2.236 ...`

- Abrundenden und aufrunden:

`(floor e) →2`
`(ceiling pi) → 4`

- GGT

`(gcd 357 753 573) → ggT(357, 753, 573)`

Boolesche Operationen

```
#t              // true
#f              // false
(and b1 b2)     // und verknüpfung, b1 und b2 müssen boolean Ausdrücke sein
(or b1 b2 b3)   // können beliebig viele Parameter haben
(not b)         // Verneinung
(= x1 x2)       // Test auf Gleichheit
(< x1 x2 x3)    // (and (< x1 x2) (< x2 x3))
```

Verzweigung

- `(if b x y)` b ist boolescher Ausdruck, x ist der Wert des Ausdrucks wenn b true ist und y wenn b false ist
- `if` ähnlich zu ternärem Operator in Java
- `(integer? x)` true wenn x Ganzzahl ist
- `(number? x)` true wenn x Zahl ist
- `(real? x)` true wenn x reelle Zahl ist
- `(natural? x)` true wenn x natürliche Zahl ist
- analog mit rational->rational
- analog mit Symbolen-> symbol
- analog mit Strings -> string

Symbole

- relativ uninterressant

Laufzeitchecks ?

- falls in Klausur drankommt nochmal hierher kommen

Definitionen verstecken

```
(define (fct x))
(local
(define const 10)
(define (mult-const y) (* const y))) //Definitonen sind außerhalb von fct
nicht sichtbar→ local

(+ const (mult-const x))) // beliebiger Wert, ist Wert des local Ausdrucks
```

Rekursion in Java

Rekursion in Racket

Beispiel Fakültät

```
(define (factorial n)
(if (= n 0) 1 (* n (factorial (- n 1))))

// wenn n=0 ist, dann ergebnis 1, zweiter Faktor wenn falsch ist
```

Listen in Racket