

# First-Order Functions

---

# First-Order Programmierung

- bisher: Definition von Funktionen
- jetzt: Wie sieht der Rumpf (Ausdruck) von Funktionen aus?
- Ausdrücke über Konstanten und Parameter

```
def square(x: Double) : Double = x * x
```

```
square(3.3)    ->           Double = 10.889999999999999
```

- Bedingte Ausdrücke = Fallunterscheidung

```
def abs(x: Double) = if (x >= 0) x else -x
```

```
abs(-3.3)    ->           Double = 3.3
```

kann else weglassen

-> führt zu sinnlosen Ergebnissen oder Fehlern

# First-Order Programmierung 2

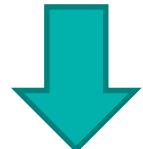
- (verschachtelte) Funktionsaufrufe

```
square (square (4.0))    -> Double = 256.0
```

- bei Funktionsdefinitionen oder
- bei Aufrufen in Ausdrücken
- ist immer noch first-order
- Higher-order erst, wenn bei Funktionsdefinition ein Funktionstyp in anderer Funktion verwendet wird (als Parameter oder Wert).

- Rekursion auf Zahlen

```
def fak(n: Int) : Int = if (n==0) 1 else n*fak(n-1)  
fak(5)          -> Int = 120
```



- verwendet Funktion im Ausdruck wieder
- wichtig: Abschluss
- einzige Kontrollstruktur
- -> keine Schleifen vorhanden