

Lernzettel

Grundlagen von Scala: Syntax, Werte- und Referenztypen, Pattern Matching, Sammler.

Universität: Technische Universität Berlin
Kurs/Modul: Programmieren II für Wirtschaftsinformatiker
Erstellungsdatum: September 6, 2025



Zielorientierte Lerninhalte, kostenlos!
Entdecke zugeschnittene Materialien für deine Kurse:

<https://study.AllWeCanLearn.com>

Programmieren II für Wirtschaftsinformatiker

Lernzettel: Grundlagen von Scala

(1) Syntax von Scala. Scala ist eine statisch typisierte Programmiersprache, die auf der JVM läuft. Sie kombiniert eine klare, kompakte Syntax mit funktionalen und objektorientierten Konzepten. Zentrale Merkmale:

- `val` (unveränderlich) vs. `var` (veränderlich)
- `def` für Funktionen; Funktionen als erstklassige Werte
- Typinferenz: oft muss der Typ nicht explizit angegeben werden
- Ausdrücke statt Anweisungen (Everything is an expression)

```
val name: String = "Alice"
val age = 30           // Typ inferred: Int
var count = 0
def sum(a: Int, b: Int): Int = a + b
val square = (x: Int) => x * x
```

(2) Werte- und Referenztypen. Scala unterscheidet zwischen Werttypen (Value types) und Referenztypen (Reference types).

- Werttypen (AnyVal): `Int`, `Long`, `Double`, `Float`, `Boolean`, `Char`, `Short`, `Byte`, `Unit`
- Referenztypen (AnyRef): alle Objekte, z.B. `String`, `Listen`, `Maps`, benutzerdefinierte Klassen

Beispiele und Hinweise:

```
val a: Int = 5           // Werttyp
val b = a + 1           // Typ inferred: Int

val s1 = "Hallo"         // Referenztyp (String)
val s2 = new String("Hallo")

s1 == s2                 // Inhaltliche Gleichheit
s1 eq s2                 // Referenzgleichheit (Identity)
```

(3) Pattern Matching. Pattern Matching ist ein zentraler Mechanismus in Scala, vergleichbar mit `switch`, aber mächtiger.

```
def describe(x: Any): String = x match {
  case i: Int    => s"Int: $i"
  case s: String => s"String: $s"
  case Nil      => "empty list"
  case _       => "something else"
}
```

Beispiele mit Tupeln und Guards:

```
val pt = (3, "foo")
pt match {
  case (n, s) if n > 2 => s"large: $n, $s"
  case (n, s)           => s"pair: $n, $s"
}
```

(4) Sammler (Collections). Scala bietet unveränderliche (immutable) und veränderliche (mutable) Sammlungen.

- Unveränderlich: List, Vector, Map, Set
- Veränderlich: scala.collection.mutable.ArrayBuffer, Map, Set

Beispiele:

```
val nums = List(1, 2, 3, 4)
val evens = nums.filter(_ % 2 == 0).map(_ * 2)

val colors = Map("rot" -> 0xff0000, "grün" -> 0x00ff00)

val doubled = nums.map(_ + 1)    // unveränderlich, creates a neue Liste
```

Beispiele für for-Komprehension:

```
val nums = List(1,2,3)
val combos = for {
  n <- nums
  c <- List('a','b','c')
} yield (n, c)
```

Zusatzhinweise zur Praxis.

- Badet man zwischen reinen Funktionen und Nebenwirkungen? Verwende so weit wie möglich unveränderliche Sammlungen.
- Pattern Matching lässt sich auf Case Classes und Extractors erweitern, z.B. case class Person(name: String, age: Int).

(5) Kleine Übungsbeispiele

```
scala> val names = List("Anne", "Bernd", "Clara")
scala> names.map(_.length)
res0: List[Int] = List(4, 4, 5)

scala> val pair = (42, "Antwort")
scala> pair match { case (n, s) => s"$s with number $n" }
res1: String = Antwort with number 42
```

Hinweis zu Scala und Java-Interoperabilität. Scala läuft auf der JVM und kann nahtlos Java-Bibliotheken verwenden. Typen werden zu Java-Wrappern oder JVM-Primitive-Typen übersetzt, je nach Kontext.

(6) Weiterführende Stichworte (Ausblick).

- Closures, Higher-Order-Funktionen, Currying
- Traits und Mixins
- Exceptions in Scala
- Grundlagen der Nebenläufigkeit/Parallelität (zunächst konzeptionell)
- Einstieg in Akka (Message-Passing-Concurrency)