Kotlin Cheat Sheet

main function

Die 'main' Funktion ist der Einstiegspunkt in jeder Kotlin-Anwendung. Sie wird bem Starten ausgeführt.

```
fun main() {
    //do something
}
```

Übrigens: Zwei Schrägstriche läuten einen Kommentar ein, der nicht interpretiert wird.

Textausgabe

Um Text auf der Kommandozeile auszugeben, kann die 'println()' Funktion verwendet werden.

```
println("Hier tollen Text einfügen :)")
//prints 'Hier tollen Text einfügen :)'
```

Variablen

Um Datenpunkte zwischenspeichern zu können, werden Variablen genutzt. Variablen haben in Kotlin immer einen festen Datentyp.

```
val a: String = "foo"
val b: Int = 42
val c: Boolean = true
```

Möchte man einer Variable später einen neuen Wert zuweisen können, muss diese als 'var' angelegt werden.

```
var d: String = "someInitialValue"

//later in my program
d = "someNewValue"

println(d) //prints someNewValue
```

Kotlin kann den Typen einer Variable auch aus ihrem zugewiesenen Wert ermitteln (Typ-Inferenz).

```
val a = "someString"
//ACHTUNG: Die Variable ist immernoch vom Typ String!
```

String Templates

Kotlin unterstützt sogenannte 'String Interpolation'. Dabei können Variablen in einen neuen String eingesetzt werden.

```
val a = "Bananas"
val b = 5

println("You bought $b delicious $a.")
// prints 'You bought 5 delicious Bananas.'
```

Alternativ kann der String auch konkateniert werden:

```
println("You bought " + b + " delicious " + a + ".")
// prints 'You bought 5 delicious Bananas.'
```

Bedingungen

Eine klassische Fallunterscheidung ist der if-Ausdruck:

```
val someString: String = "someValue"

if (someString == "someValue") {
    println("Condition was true")
} else {
    println("Condition was false")
}
//prints 'Condition was true'
```

Mit Hilfe des if-Ausdrucks kann man Entscheidungen im Programmablauf treffen. Nach dem 'if' wird in Klammern eine Bedingung angegeben. Ist die Bedingung wahr, wird der folgende Code-Block ausgeführt. Optional kann mit 'else' ein zweiter Block angegeben werden, der ausgeführt wird, wenn die Bedingung unwahr ist. Beispiele für unterschiedliche Bedingungen:

```
val a: Boolean = true
if (a) {
    println("Condition was true")
}
//prints 'Condition was true'

val b: Int = 10
if (b > 5) {
    println("Condition was true")
}
//prints 'Condition was true'
```

```
val c: Boolean = false
if (!c) {
    println("Condition was true")
}
//prints 'Condition was true'

val d: String = "someStringValue"
if (d != "someOtherValue") {
    println("Condition was true")
}
//prints 'Condition was true'
```

Listen

In einer Liste kann man mehrere Werte eines Typs speichern.

```
val a: List<String> = listOf(
    "value1", "value2", "value3"
)

val b: List<Int> = listOf(
    1, 2, 3
)
```

Schleifen

Mit einer for-Schleife kann man bspw. über jedes Element in einer Liste iterieren:

```
val a: List<String> = listOf("A", "B", "C")

for (x: String in a) {
    println(x)
}
// prints every element of list a
```

Funktionen

Funktionen sind Codeblöcke, die wiederverwendet werden können. Meistens nehmen sie sogenannte Parameter entgegen und liefern ein Ergebnis zurück.

```
val a = 5
val b = 10

fun addMyNumbers(myFirstNumber: Int, mySecondNumber: Int): Int {
    return myFirstNumber + mySecondNumber
}
val sum = addMyNumbers(a, b)
println(sum)
//prints '15'
```

Funktionen müssen nicht zwingend einen Rückgabewert haben.

```
fun printSumOfMyNumbers(myFirstNumber: Int, mySecondNumber: Int) {
    println("The sum of $myFirstNumber and $mySecondNumber is ${myFirstNumber + mySecondNumb
}

printSumOfMyNumbers(5, 10)
//prints 'The sum of 5 and 10 is 15'
```