

## Bemerkung 0.1 – Kompilierung

Dieses Dokument wurde kompiliert durch:

`pdflatex test-1st.tex`

## 1 Einfache Benutzung zum Code-Highlighting, etlicher Sprachen

Java:

```
1 public static void main(String[] args) {  
2     int i = 42*3;  
3     System.out.println("Hallo_Welt");  
4 }
```

C++:

```
1 #include <iostream>  
2 public static int main(int argc, char** argv) {  
3     std::cout << "Hallo_Welt" << std::endl;  
4     std::cout << "Die_Antwört_List:_ " << 21*2+14-(2*7) << std::endl;  
5 }
```

Die Existenz einer Sprache kann dank `LIB LILLYxLIST` leicht geprüft werden:

```
1 \isLanguageLoaded{java/lJava} % → 'TRUE' (\true)  
2 \isLanguageLoaded{super/lSuper} % → 'FALSE' (\false)
```

## Bemerkung 1.1 – Notation

Soll dies auf nur 'java' geändert werden?

Alle geladenen Sprachen finden sich in der Liste `RegisteredLanguages`.

## 2 Besonderheiten

Durch die Syntax `'<Keyword>:'` können Sonderzeichen und andere tolle Features in listings eingeführt werden, so wurden die oberen Pfeile durch `'yields:'` erzeugt. Weiter interessant sind vielleicht noch `'lan:'` (left angle: `'<'`) und `'ran:'` (right angle: `'>'`)

### 3 Befehle

Für jede Sprache wie 'Java' existiert ein entsprechend kleingeschriebenes Environment<sup>1</sup> was das Highlighting übernimmt (hier: 'java'), soll die Sprache ohne Firlefanz gesetzt werden, so existiert `lstplain`:

```
1 \begin{lstplain}[language=Java]
2 public static void main(String[] args) {
3     int i = 42*3;
4     System.out.println("Hallo Welt");
5 }
6 \end{lstplain}
```

Ergibt:

```
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("HalloWelt");
}
```

Für das inline-Code-Highlighting, existieren für jede Sprache `c<Sprache>` (code) und `b<Sprache>` (blank):

```
1 \cpython{print('hallo')}, \bpython{print('hallo')}

print("hallo"), print('hallo')
```

Ersterer kann aufgrund der Implementation des Hintegrund nicht umgebrochen werden, Listings versucht sich allerdings in einem normalen Maße der bad-paragraph (überlauf Problematik) anzunehmen. Letzterer kann ganz normal umgebrochen werden. Hier ein Beispiel:

```
print('hallo_welt, _na_wie_geht_es_dir?_dies_ist_sehr_langer_>
Text, _ein_bisschen_unangenehm_oder?_aber_irgendwie_muss_ich_
die_Zeile_sprengen_tut_mir_wirklich_leid.')
```

Dahingegeben:

```
print('hallo_welt, _na_wie_geht_es_dir?_dies_ist_sehr_langer_>
Text, _ein_bisschen_unangenehm_oder?_aber_irgendwie_muss_ich_
die_Zeile_sprengen_tut_mir_wirklich_leid.')
```

Weiter existiert noch '`i<Sprache>`', diesem Befehl kann eine Datei übergeben werden der diese dann in der jeweiligen Sprache hervorhebt. Beispiel mit diesem Dokument:

```
1 \ilatex{test-1st.tex}
```

Da es sich hierbei auch um das Dokument handelt, findet sich das Ergebnis auf der nächsten Seite.

---

<sup>1</sup>C++ ist 'cpp'

## 4 Dokumentcode

```
1 \documentclass{article}
2
3 \usepackage{LILLYxLISTINGS}
4 \usepackage{LILLYxCONTROLLERxBOX}
5
6 \begin{document}
7
8 \begin{bemerkung}[Kompilierung]
9     Dieses Dokument wurde kompiliert durch: \begin{center}
10         \bbash{pdflatex test-1st.tex}
11     \end{center}
12 \end{bemerkung}
13
14 \section{Einfache Benutzung zum Code-Highlighting, etlicher
15     Sprachen}
16
17 Java:
18 \begin{java}
19 public static void main(String[] args) {
20     int i = 42*3;
21     System.out.println("Hallo Welt");
22 }
23 \end{java}
24
25 C++:
26 \begin{cpp}
27 #include <iostream>
28 public static int main(int argc, char** argv) {
29     std::cout << "Hallo Welt" << std::endl;
30     std::cout << "Die Antwort ist: " << 21*2+14-(2*7) << \n
31     std::endl;
32 }
33 \end{cpp}
34
35 Die Existenz einer Sprache kann dank \LILLYxNOTExLibrary{
36     LILLYxLIST} leicht geprüft werden:
37 \begin{latex}
38 \isLanguageLoaded{java/lJava} % → 'TRUE' (\true)
39 \isLanguageLoaded{super/lSuper} % → 'FALSE' (\false)
40 \end{latex}
41
42 \begin{bemerkung}[Notation]
43     Soll dies auf nur '\verb|java|' geändert werden?
44 \end{bemerkung}
45
46 Alle geladenen Sprachen finden sich in der Liste \verb|
47     RegisteredLanguages|.
48 \section{Besonderheiten}
```

```

40 Durch die Syntax '\verb|<Keyword>:|' können Sonderzeichen
    und andere tolle Features in listings eingeführt
    werden, so wurden die oberen Pfeile durch '\verb|→|'
    erzeugt. Weiter interessant sind vielleicht noch '\verb|
|<|' (left angle: '\bjava{<}') und '\verb|>|' (right
    angle: '\bjava{>}')
41
42 \section{Befehle}
43 Für jede Sprache wie '\verb|Java|' existiert ein
    entsprechend kleingeschriebenes Environment \footnote{C}
    ++ ist 'cpp'} was das Highlighting übernimmt (hier: '
\verb|java|'), soll die Sprache ohne Firlefanz gesetzt
    werden, so existiert \verb|lstplain|: \begin{latex}
44 \begin{lstplain}[language=Java]
45 public static void main(String[] args) {
46     int i = 42*3;
47     System.out.println("Hallo Welt");
48 }
49 \end{lstplain}
50 \end{latex}
51 Ergibt:
52 \begin{lstplain}[language=Java]
53 public static void main(String[] args) {
54     int i = 42*3;
55     System.out.println("Hallo Welt");
56 }
57 \end{lstplain}
58 Für das inline-Code-Highlighting, existieren für jede
    Sprache \verb|c<Sprache>| (code) und \verb|b<Sprache>|
    (blank):
59 \begin{latex}
60 \cpython{print('hallo')}, \bpython{print('hallo')}
61 \end{latex}
62 \cpython{print("hallo")}, \bpython{print('hallo')}\newline
63 Ersterer kann aufgrund der Implementation des Hintergrund
    nicht umgebrochen werden, Listings versucht sich
    allerdings in einem normalen Maße der bad-paragraph (Ü
    berlauf Problematik) anzunehmen. Letzterer kann ganz
    normal umgebrochen werden. Hier ein Beispiel: \newline
64 \cpython{print('hallo welt, na wie geht es dir? dies ist
    sehr langer Text, ein bisschen unangenehm oder? aber
    irgendwie muss ich die Zeile sprengen tut mir wirklich
    leid.')}\newline
65 Dahingegeben: \newline
66 \bpython{print('hallo welt, na wie geht es dir? dies ist
    sehr langer Text, ein bisschen unangenehm oder? aber

```

```

        irgendwie muss ich die Zeile sprengen tut mir wirklich }
        leid.')}\\medskip\\newline
67 Weiter existiert noch '\\verb|i<Sprache>|', diesem Befehl }
        kann eine Datei übergeben werden der diese dann in der }
        jeweiligen Sprache hervorhebt. Beispiel mit diesem }
        Dokument:
68 \\begin{latex}
69 \\ilatem{test-1st.tex}
70 \\end{latex}
71 Da es sich hierbei auch um das Dokument handelt, findet }
        sich das Ergebnis auf der nächsten Seite.
72 \\clearpage
73 \\section{Dokumentcode}
74 \\ilatem{test-1st.tex}
75 \\end{document}

```