```
\overbar{a_1} \overbar{a_2}
```

\overbar{a_1} \overbar{a_2}	$\overline{a_1}\overline{a_2}$
\overline{a_1} \overline{a_2}	$\overline{a_1}\overline{a_2}$

```
public class Example {
    public static void main(String[] args) {
        System.out.println(42*3+7-8^42);
        System.out.println("42*3+7-8^42");
        int x = 23, z=42;
        // int y = 42,q=13;
    }
    Hallo3 Ich bin eine 4 und f4nt4st1sch und dies ist eine s3hr l4ng3 z31l3
}
```

Hi

```
\documentclass{article}

%\usepackage{LILLYxLISTINGS}
\usepackage{LILLYxLISTINGSxADVANCED}

\usepackage{LILLYxCONTROLLERxBOX}

\begin{document}
```

```
import java.util.ArrayList;

public class Example {
    public static void main(String[] args) {
        System.out.PrintLn("Hallo Welt");
        if(args==null)
            System.out.println("wau");
    }
}
```

```
    import java.util.ArrayList;

    public class Example {
        public static void main(String[] args) {
            System.out.PrintLn("Hallo Welt");
            if(args==null)
                System.out.println("wau");
        }
    }
```

Listing 1: Hi

```java
import java.util.ArrayList;

public class Example {
    public static void main(String[] args) {
        System.out.PrintLn("Hallo Welt");
        if(args==null)
            System.out.println("wau");
    }
}
```

```
\begin{java}
|info|import java.util.ArrayList;|info|

public class Example {
   public static void main(String[] args) {
      System.out.|err|PrintLn|err|("Hallo Welt");
      if(|warn|args==null|warn|)
         System.out.println("wau");
   }
}
\end{java}
```

Bemerkung 0.1 – Kompilierung

Dieses Dokument wurde kompiliert durch:

```
pdflatex test-lst.tex
```

Einfache Benutzung zum Code-Highlighting, etlicher Sprachen

Java:

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

C++:

```cpp
#include <iostream>
public static int main(int argc, char** argv) {
    std::cout << "Hallo Welt" << std::endl;
    std::cout << "Die Äntwört ist: " << 21*2+14-(2*7) << std::endl;
}
```

Die Existenz einer Sprache kann dank LILLYxLIST leicht geprüft werden:

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

> **Bemerkung 0.2 – Notation**
>
> Soll dies auf nur '`java`' geändert werden?

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

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: ' ⟩ ')

Befehle

Für jede Sprache wie '`Java`' existiert ein entsprechend kleingeschriebenes Environment[a] was das Highlighting übernimmt (hier: '`java`'), soll die Sprache ohne Firlefanz gesetzt werden, so existiert `lstplain`:

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

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

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

Ergibt:

[a] C++ ist 'cpp'

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

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

```
\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 angenehm oder? aber irgendwie muss ich die Zeile sprengen tut mir wirklich leid.')`
Dahingegegen:
`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:

```
\ilatex{test-lst.tex}
```

Hey: [ ] Code for inline default: `public static void main (String[] args)`
Code for inline blank: `public static void main (String[] args)`
Code for inline input:

```
\errorcontextlines 10000
\documentclass[debug]{Lilly}

%\usepackage{LILLYxLISTINGS}
\usepackage{LILLYxLISTINGSxADVANCED}
```

```
public static void main (String[] args){

}
%
```

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

Dokumentcode

```latex
\errorcontextlines 10000
\documentclass[debug]{Lilly}

%\usepackage{LILLYxLISTINGS}
\usepackage{LILLYxLISTINGSxADVANCED}
%\usepackage{LILLYxEMBLEMS}
\usepackage{LILLYxCONTROLLERxBOX}
\usepackage{LILLYxMATH}
\usepackage{LILLYxTABLES}

\DeclareDocumentCommand{\Hallo}{}{
    Hi
}
% \foreach \x in {Hi,Hu,Ho}{
% \edef\cur{\noexpand\NewLstEnv{\x}}%
% \cur%
% }
\begin{document}
% \blatex[morekeywords={[5]{sqrt}}]{\sqrt[3]\{42\}}
\blatex[morekeywords={[5]{overbar}}]{\overbar\{a\_1\} \overbar\{a\_2\}}
\begin{center}
    \begin{tabular}{!{\VRule[1pt]}@{\hspace{1em}}l@{\hspace{1em}}|@{\hspace{1em}}c@{\hspace{1em}}!{\VRule[1pt]}}
        \specialrule{1pt}{0pt}{0pt}
         {\blatex[morekeywords={[5]{overbar}}]{\overbar\{a\_1\} \overbar\{a\_2\}}} & \(\overbar{a_1}\overbar{a_2}\)\\\hline
        {\blatex[morekeywords={[5]{overline}}]{\overline\{a\_1\} \overline\{a\_2\}}} & \(\overline{a_1}\overline{a_2}\)\\
        \specialrule{1pt}{0pt}{0pt}
    \end{tabular}
\end{center}
\begin{java}
public class Example {
    public static void main(String[] args) {
        System.out.println(42*3+7-8^42);
        System.out.println("42*3+7-8^42");
        int x = 23, z=42;
        // int y = 42,q=13;
    }
    Hallo3 Ich bin eine 4 und f4nt4st1sch und dies ist eine s3hr l4ng3 z3113
}
\end{java}

% \begin{Hi}
```

```
% Hallo Mama
% \end{Hi}
% \begin{Hu}

% \end{Hu}
\Hallo
\begin{lstlisting}[language=lLatex]
\documentclass{article}

%\usepackage{LILLYxLISTINGS}
\usepackage{LILLYxLISTINGSxADVANCED}

\usepackage{LILLYxCONTROLLERxBOX}

\begin{document}
\end{lstlisting}

\begin{plainjava}
|info|import java.util.ArrayList;|info|

public class Example {
    public static void main(String[] args) {
        System.out.|err|PrintLn|err|("Hallo Welt");
        if(|warn|args==null|warn|)
            System.out.println("wau");
    }
}
\end{plainjava}

\begin{sjava}
    |info|import java.util.ArrayList;|info|

    public class Example {
        public static void main(String[] args) {
            System.out.|err|PrintLn|err|("Hallo Welt");
            if(|warn|args==null|warn|)
                System.out.println("wau");
        }
    }
\end{sjava}

\begin{java}[caption={Hi}]
|info|import java.util.ArrayList;|info|

public class Example {
    public static void main(String[] args) {
        System.out.|err|PrintLn|err|("Hallo Welt");
        if(|warn|args==null|warn|)
```

```
            System.out.println("wau");
    }
}
\end{java}
\begin{latex}
\begin{java}
|info|import java.util.ArrayList;|info|

public class Example {
    public static void main(String[] args) {
        System.out.|err|PrintLn|err|("Hallo Welt");
        if(|warn|args==null|warn|)
            System.out.println("wau");
    }
}
\end{java}
\end{latex}

\begin{bemerkung}[Kompilierung]
    Dieses Dokument wurde kompiliert durch: \begin{center}
        \bbash{pdflatex test-lst.tex}
    \end{center}
\end{bemerkung}

\section{Einfache Benutzung zum Code-Highlighting, etlicher Sprachen}
Java:
\begin{java}
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{java}
C++:
\begin{cpp}
#include <iostream>
public static int main(int argc, char** argv) {
    std::cout << "Hallo Welt" << std::endl;
    std::cout << "Die Äntwört ist: " << 21*2+14-(2*7) << std::endl;
}
\end{cpp}
Die Existenz einer Sprache kann dank \LILLYxNOTExLibrary{LILLYxLIST} leicht geprüft werden:
\begin{latex}
\isLanguageLoaded{java/lJava} % → 'TRUE' (\true)
\isLanguageLoaded{super/lSuper} % → 'FALSE' (\false)
\end{latex}
\begin{bemerkung}[Notation]
```

```latex
    Soll dies auf nur '\verb|java|' geändert werden?
\end{bemerkung}
Alle geladenen Sprachen finden sich in der Liste \verb|RegisteredLanguages|.
\section{Besonderheiten}
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{⟩}')

\section{Befehle}
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}
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
\end{latex}
\begin{plainlatex}
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
\end{plainlatex}
\begin{slatex}
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
\end{slatex}
Ergibt:
\begin{lstplain}[language=lJava]
public static void main(String[] args) {
    int i = 42*3;
    System.out.println("Hallo Welt");
}
\end{lstplain}
```

```
Für das inline-Code-Highlighting, existieren für jede Sprache \verb|c<Sprache>| (code) und \verb|b<Sprache>| (blank):
\begin{latex}
\cpython{print('hallo')}, \bpython{print('hallo')}
\end{latex}
\cpython{print("hallo")}, \bpython{print('hallo')}\newline
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:\newline
\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
Dahingegegen:\newline
\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
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:
\begin{latex}
\ilatex{test-lst.tex}
\end{latex}

Hey: [\RegisterLanguage{rubix}{lJava}]
% THIS IS WILL FINALLY OPTIONALS
Code for inline default: \crubix[morekeywords={args}]{public static void main (String[] args)}\\
Code for inline blank: \brubix{public static void main (String[] args)}\\
Code for inline input: \irubix[lastline=5]{test-lst.tex}
\begin{rubix}
public static void main (String[] args){

}
% \end{rubix}
% \begin{plainrubix}
% Hallo Welt
% Hallo Welt

% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
```

```
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt

% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt
% Hallo Welt

% \end{plainrubix}

% Hey: [\RegisterLanguage{detlef}{lJava}]
% Code for inline default: \crubix[morekeywords={args}]{public static void main (String[] args)}\\
% Code for inline blank: \brubix{public static void main (String[] args)}\\
% \begin{rubix}
% public static void main (String[] args){

% }
% \end{rubix}
% Code for inline default: \cdetlef[morekeywords={args}]{public static void main (String[] args)}\\
% Code for inline blank: \bdetlef{public static void main (String[] args)}\\
% \begin{detlef}
% public static void main (String[] args){

```

```
% }
% \end{detlef}

Da es sich hierbei auch um das Dokument handelt, findet sich das Ergebnis auf der nächsten Seite.
\clearpage
\section{Dokumentcode}
\ilatex{test-lst.tex}
\end{document}
```