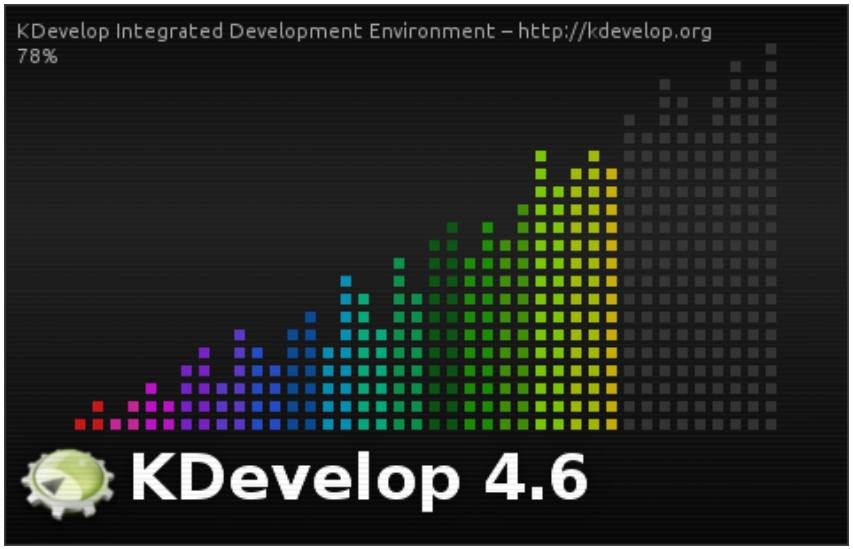
Ein Überblick über KDevelop

Sven Brauch (scummos)

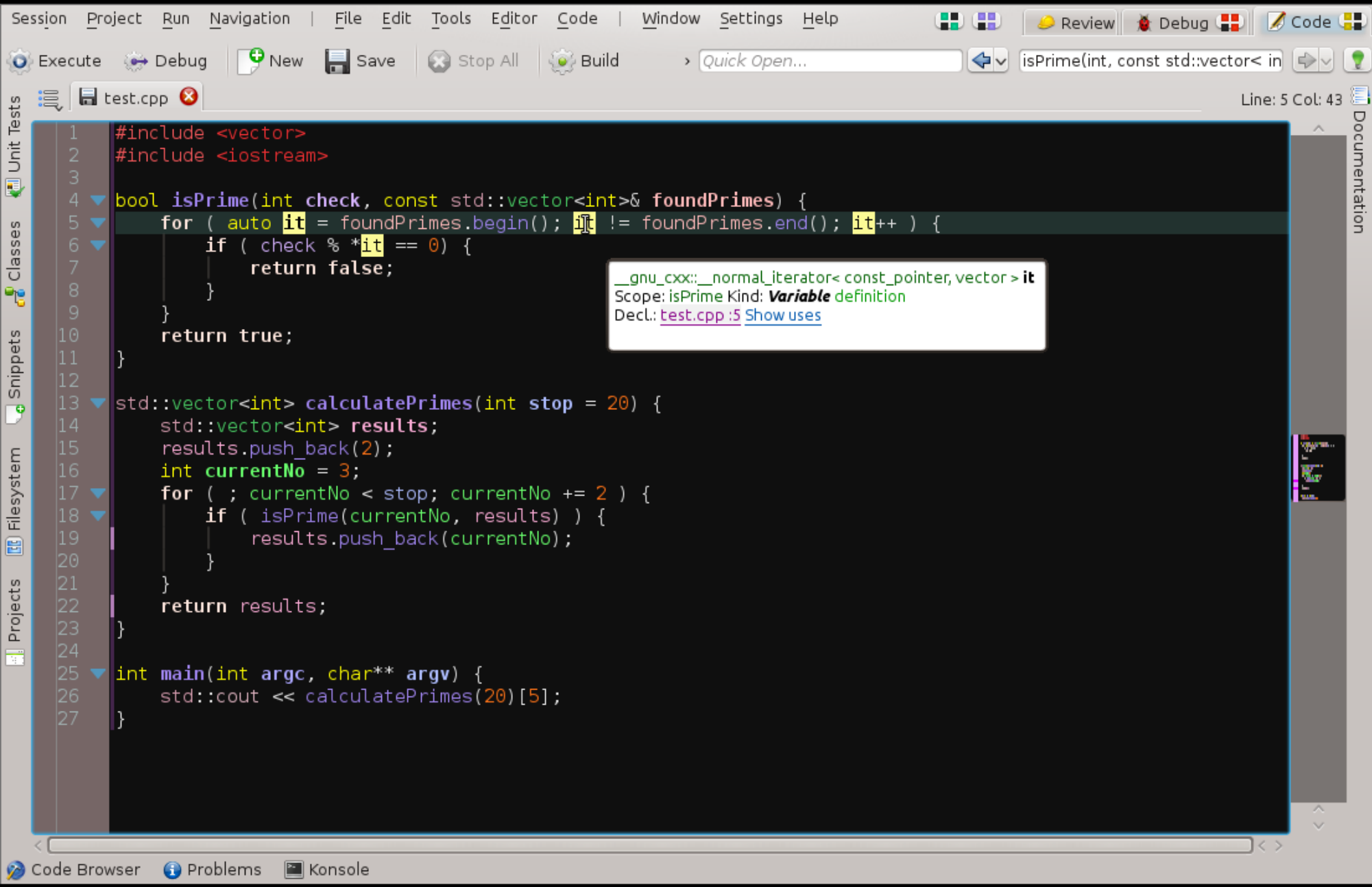


Inhalt

Inhalt

Was ist KDevelop 4.x?

- im Kern: **C++-IDE** basierend auf KDE- und Qt-Bibliotheken
- Komplettes Rewrite von Version 3
- erstes 4.0-Release war 2008


Session Project Run Navigation File Edit Tools Editor Code Window Settings Help
Review Debug Code
Execute Debug New Save Stop All Build
Quick Open...
isPrime(int, const std::vector< in
test.cpp
Line: 5 Col: 43
Unit Tests
Classes
Snippets
Filesystem
Projects
Documentation
#include <vector>
#include <iostream>
bool isPrime(int check, const std::vector<int>& foundPrimes) {
for (auto it = foundPrimes.begin(); it != foundPrimes.end(); it++) {
if (check % *it == 0) {
return false;
}
}
return true;
}
__gnu_cxx::__normal_iterator< const_pointer, vector > it
Scope: isPrime Kind: Variable definition
Decl.: test.cpp :5 Show uses
std::vector<int> calculatePrimes(int stop = 20) {
std::vector<int> results;
results.push_back(2);
int currentNo = 3;
for (; currentNo < stop; currentNo += 2) {
if (isPrime(currentNo, results)) {
results.push_back(currentNo);
}
}
return results;
}
int main(int argc, char** argv) {
std::cout << calculatePrimes(20)[5];
}
Code Browser
Problems
Konsole

Was ist KDevelop 4.x? (2)

- weitere Sprachen können über **Plugins** integriert werden
- derzeit:
  - **PHP**
  - **Python**
  - ohne Release aber gut: **ruby**
  - in den Anfängen: QML / JS, Java
- Editor: KTextEditor, bekannt aus **kate**, kwrite, kile etc.

Was heißt eigentlich „KDE“?

- „KDE“ im Volksmund:
  - Anwendungen (kate, amarok, konqueror, . . .)
  - Desktop (plasma-desktop, KWin, . . .)
- eigentlich: Community; auch: **Satz von Bibliotheken** (kdelibs) basierend auf Qt plus viele Anwendungen (deshalb seit 4.4 auch „KDE SC“)
- KDevelop hat technisch gesehen nichts mit dem „KDE Desktop“ zu tun
- Abhängigkeiten sind nicht der Rede wert

Inhalt

Philosophie von KDevelop

- **keine** (signifikanten) **Projektdateien** — Projekt = Verzeichnis
- keine direkte Interaktion mit Compiler
  - stattdessen: Unterstützung für **seperate Build-Systeme** wie CMake
- keine „Bindung“ an IDE
- Fokus: hervorragende Code-Tools

Inhalt

KDevelop und KDevPlatform

KDevPlatform

- **Framework** für das Schreiben einer IDE
- Basis für Sprach- und andere Plugins

KDevelop

- auf KDevPlatform basierende IDE
- Eigentliche Anwendung (sehr klein) + **Sammlung von Plugins** für KDevPlatform
  - C++ Sprachunterstützung
  - gdb-Debugger-Plugin
  - CMake-Projektmanager
  - ...

Die Definition-Use-Chain (duchain)

- für IDE relevante Informationen über Code **in jeder Programmiersprache sehr ähnlich**
- entsprechend: Abstraktion möglich
- zwei wohl wichtigste Klassen: „**Definition**“ und „**Use**“
- deshalb: „**DUChain**“**-Framework** in KDevPlatform für Informationen über Code, unabhängig von der Sprache

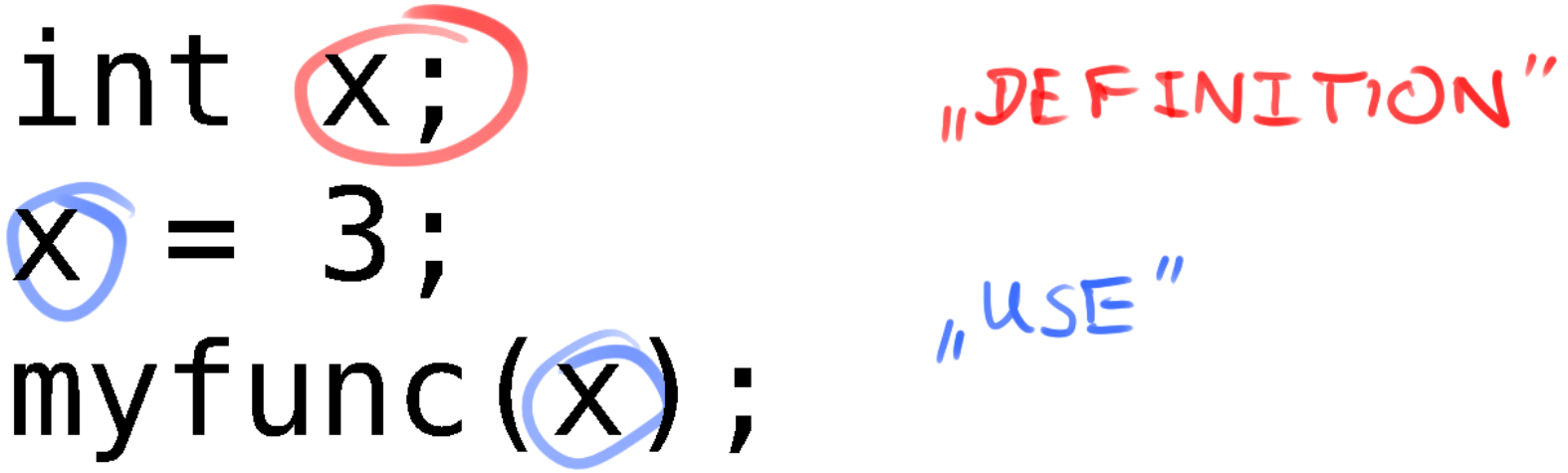


Abbildung : Ein sehr einfaches Beispiel für eine Definition-Use-Chain

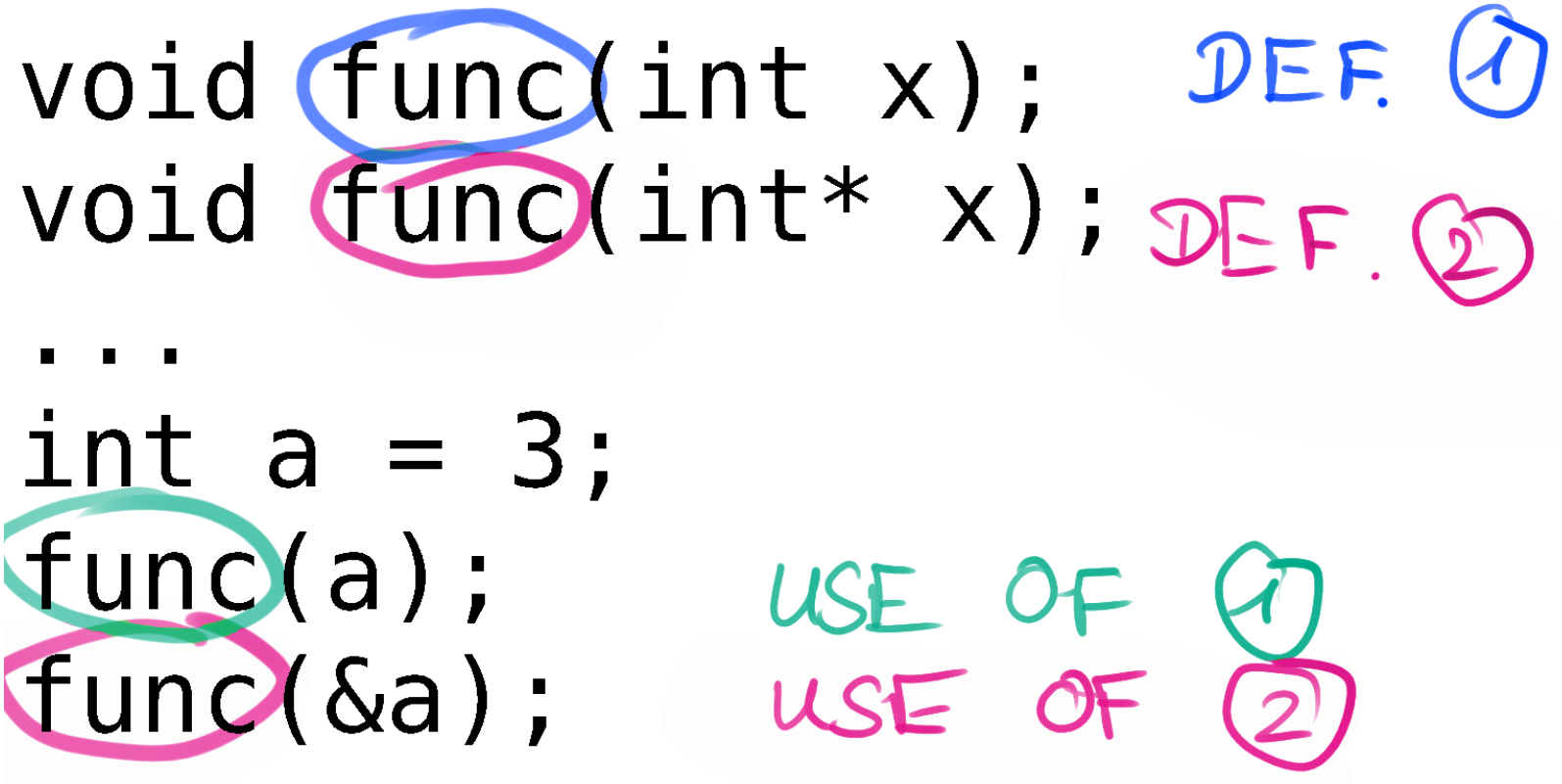


Abbildung : Eine Verwendung („Use“) eines Objekts (Variable, Funktion etc.) weiß, zu welchem Objekt („Definition“) sie gehört

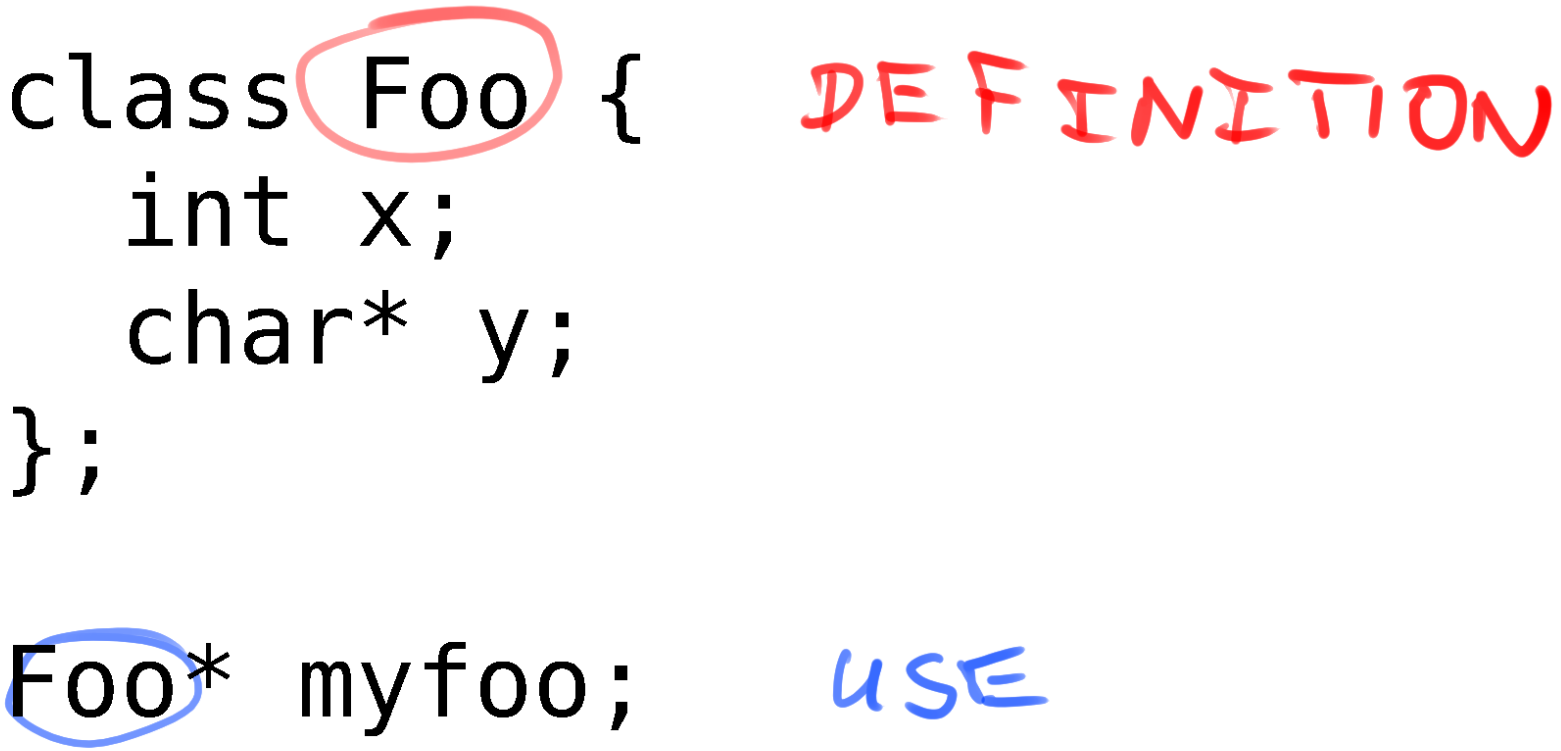


Abbildung : Für Klassen funktioniert das genauso.

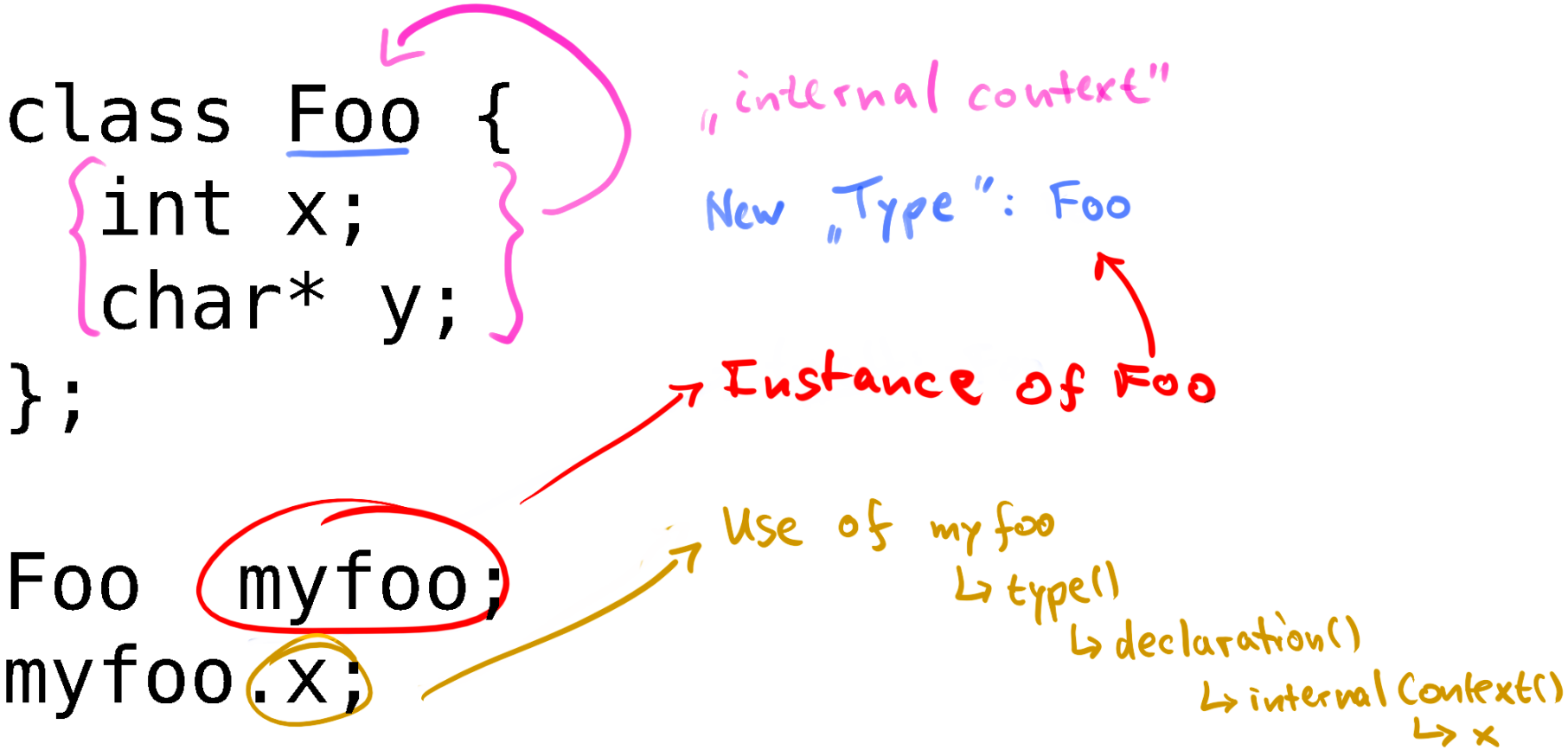


Abbildung : Ein kompliziertes Beispiel. Gelb: So geht das Sprach-Plugin intern vor.

Die Definition-Use-Chain (duchain) (2)

- wird beim Öffnen eines Projekts für alle Dokumente berechnet
- bei jeder Änderung eines Dokuments wird die DUChain für dieses Dokument neu berechnet
- KDevPlatform sorgt für persistentes Caching (*$HOME/.cache/kdevduchain)*

Sprach-Plugins

- Hauptaufgabe: bekommt Text aus Editor, soll Definition-Use-Chain berechnen
- Übliches Vorgehen:
  $\text{Text} \xrightarrow{\text{Parser}} \text{AST}^1 \xrightarrow{\text{Declaration / Context / ... Builder}} \text{DUChain}$
- Declaration / Context / ... Builder: verbindet KDevPlatforms API mit sprach-spezifischem „Visitor“

[1] AST steht für „Abstract Syntax Tree“

Beispiel: Wie funktioniert das beim Python-Plugin?

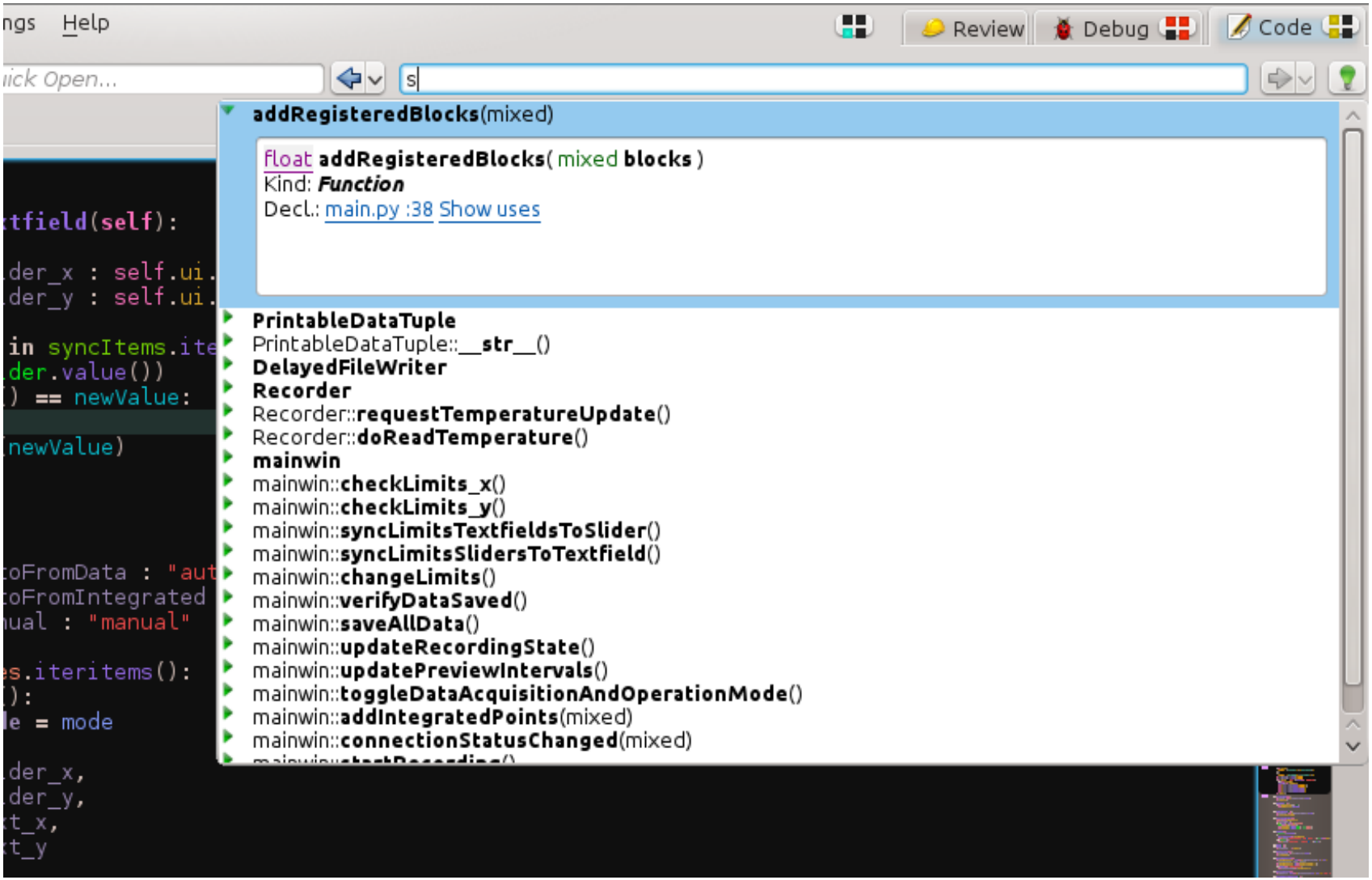
```
my_var = 3
another_var = my_var
```

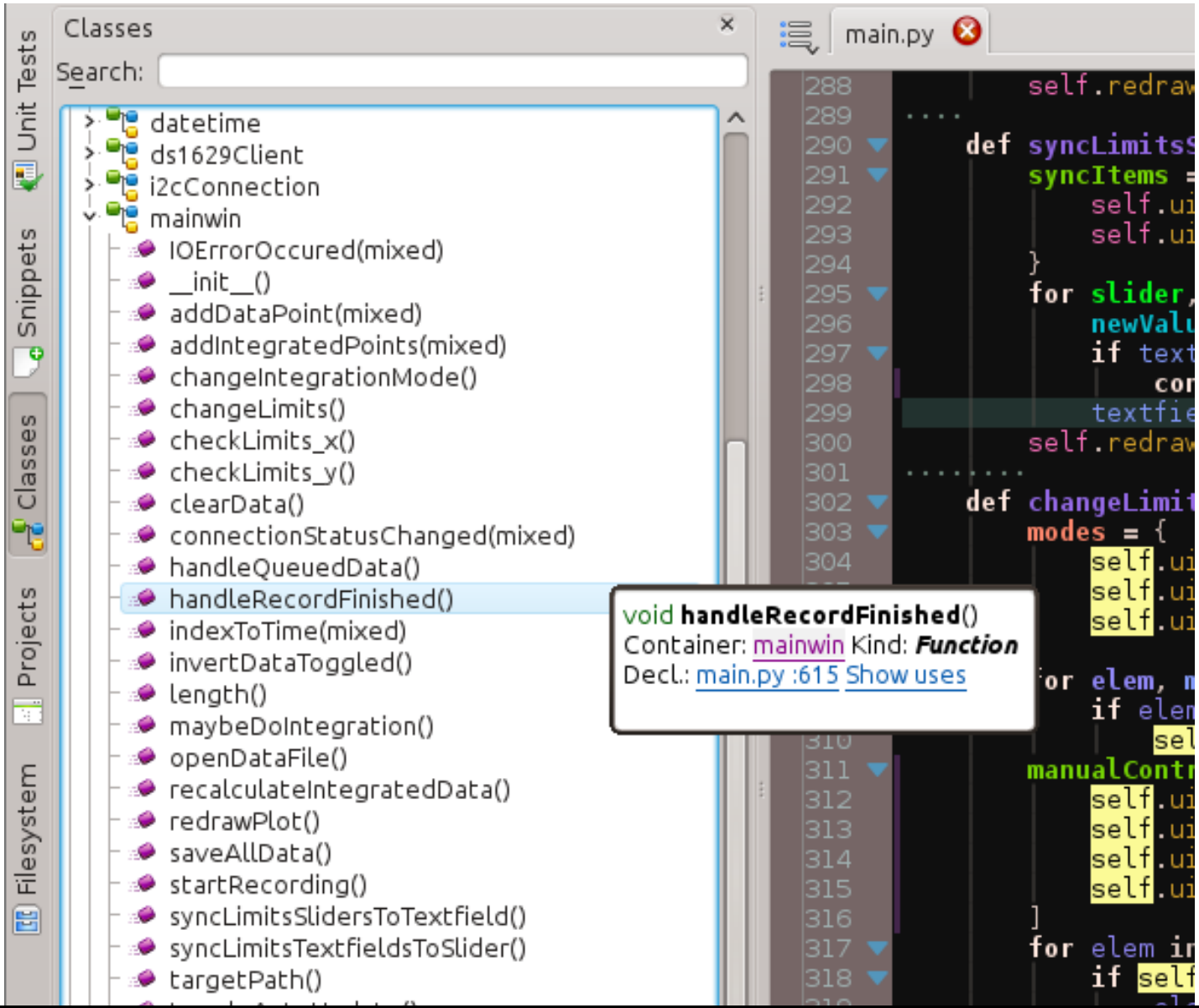
- Erstes Statement: Zuweisung ⇒ `visitAssignment(...)`
  - Berechne Typ der rechten Seite: `int`
  - Erstelle eine „`Definition`“ für `my_var`
  - Setze Typ von `my_var` auf „`Instanz von int`“
- Zweites Statement: Zuweisung ⇒ `visitAssignment(...)`
  - Berechne Typ der rechten Seite:
    Durchsuche aktuellen Kontext nach einer „`Definition`“ für `my_var`
  - Erstelle eine „`Definition`“ für `another_var`
  - Setze Typ von `another_var` auf „`Instanz von int`“
  - Erstelle eine „`Use`“ für `my_var`

```
def changeLimits(self):
    modes = {
        self.ui.limits_autoFromData : "autoData",
        self.ui.limits_autoFromIntegrated : "autoIntegrated",
        self.ui.limits_manual : "manual"
    }
    for elem, mode in modes.iteritems():
        if elem.isChecked():
            self.limitsMode = mode
    manualControls = [
        self.ui.limits_slider_x,
        self.ui.limits_slider_y,
        self.ui.limits_text_x,
        self.ui.limits_text_y
    ]
    for elem in manualControls:
        if self.limitsMode != "manual":
            elem.setDisabled(True)
        else:
            elem.setDisabled(False)

    if self.invertData:
        sliderRange = (-expectedAdRange[1]-20, expectedAdRange[0]+20)
    else:
        sliderRange = (expectedAdRange[0]-20, expectedAdRange[1]+20)
    self.ui.limits_slider_x.setRange(*sliderRange)
    self.ui.limits_slider_y.setRange(*sliderRange)
```

str **mode**
Container: changeLimits Kind: ***Variable***
Decl.: main.py :308 Show uses


ngs Help
Review
Debug
Code
uick Open...
s
addRegisteredBlocks(mixed)
float addRegisteredBlocks(mixed blocks)
Kind: Function
Decl.: main.py :38 Show uses
PrintableDataTuple
PrintableDataTuple::__str__()
DelayedFileWriter
Recorder
Recorder::requestTemperatureUpdate()
Recorder::doReadTemperature()
mainwin
mainwin::checkLimits_x()
mainwin::checkLimits_y()
mainwin::syncLimitsTextfieldsToSlider()
mainwin::syncLimitsSlidersToTextfield()
mainwin::changeLimits()
mainwin::verifyDataSaved()
mainwin::saveAllData()
mainwin::updateRecordingState()
mainwin::updatePreviewIntervals()
mainwin::toggleDataAcquisitionAndOperationMode()
mainwin::addIntegratedPoints(mixed)
mainwin::connectionStatusChanged(mixed)


Classes
Search:
datetime
ds1629Client
i2cConnection
mainwin
IOErrorOccured(mixed)
__init__()
addDataPoint(mixed)
addIntegratedPoints(mixed)
changeIntegrationMode()
changeLimits()
checkLimits_x()
checkLimits_y()
clearData()
connectionStatusChanged(mixed)
handleQueuedData()
handleRecordFinished()
indexToTime(mixed)
invertDataToggled()
length()
maybeDoIntegration()
openDataFile()
recalculateIntegratedData()
redrawPlot()
saveAllData()
startRecording()
syncLimitsSlidersToTextfield()
syncLimitsTextfieldsToSlider()
targetPath()
void handleRecordFinished()
Container: mainwin Kind: Function
Decl.: main.py :615 Show uses
main.py
Unit Tests
Snippets
Classes
Projects
Filesystem

Editor-Komponente

- **derselbe Editor** wie in Kate, KWrite, Kile, ...
- **vi-Eingabemodus**!
- skriptbar, z. B. Snippets

Beispiel für skriptbare Snippets

```
\documentclass{article}

\begin{document}

\begin{table}
\center
\begin{tabular}{cclr}
\toprule
title0 & title1 & title2 & title3 \\
\midrule
content0 & content1 & content2 & content3 \\
\bottomrule
\end{tabular}
\caption{CAPTION}
\end{table}

\end{document}
```

Rekapitulation

- Verwendung **derselben Grund-Datenstrukturen für jede Sprache**
- Aufgabe der Sprach-Plugins: diese Datenstrukturen „bevölkern“
- Wer will, kann dank KDevPlatform / KDevelop-Trennung eine neue IDE mit anderer UI, aber demselben Backend schreiben.

Inhalt

kdev-python: Python-Sprachplugin für KDevelop

- Hauptaufgabe: Möglichst gute **Informationen über Code sammeln**
- hat außerdem auch z. B. ein Debugger-Plugin
- relativ **klein** (Gesamt 13k SLOC vs. 170k SLOC bei PyDev (Eclipse), Debugger-Plugin 1200 SLOC vs 15k bei PyDev) dank Abstraktion

Probleme mit Typen in Python (1)

- Python benutzt „Duck Typing“
- Eine Variable kann potentiell **mehrere mögliche Typen** haben
- Lösung: „unsure“ types

Abbildung : Ein einfaches Beispiel für „unsure types“

Probleme mit Typen in Python (2)

- in manchen Situationen aber:
  keine Chance (z. B. `exec()` aka eval)
- auch schwierig: Listen mit dynamischer Länge und unterschiedlichen Typen

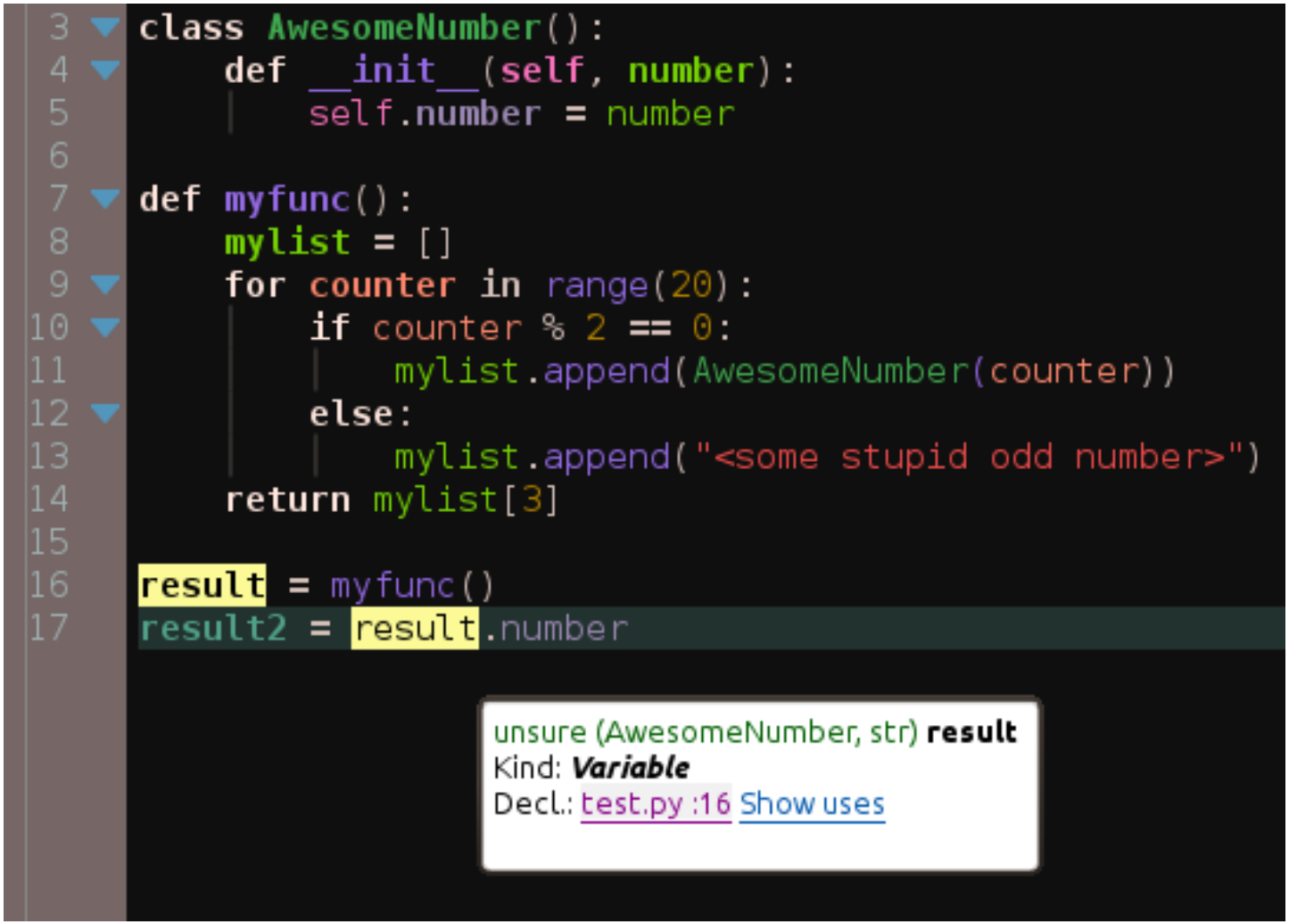
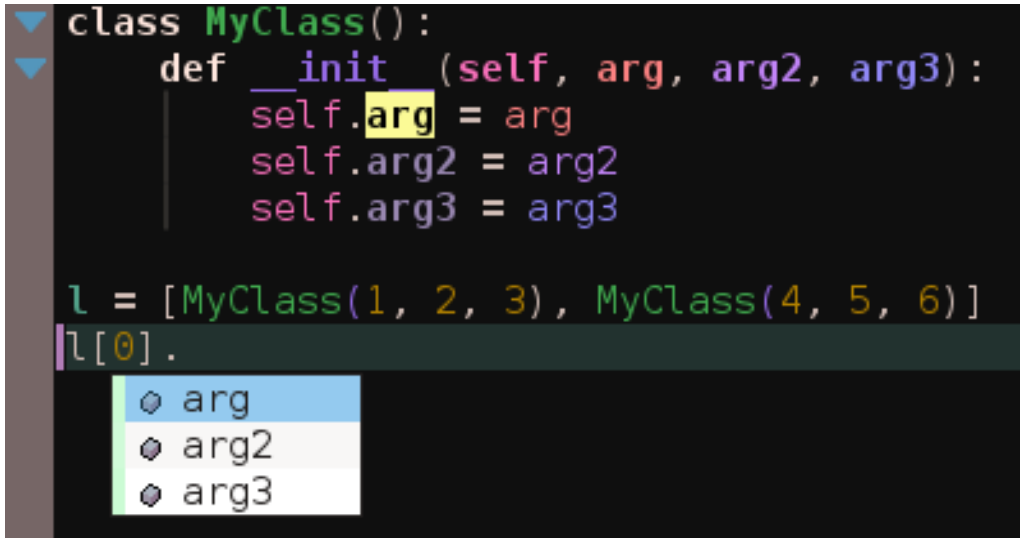


Abbildung : Ein weiteres Beispiel für „unsure types“

Probleme mit Bibliotheken

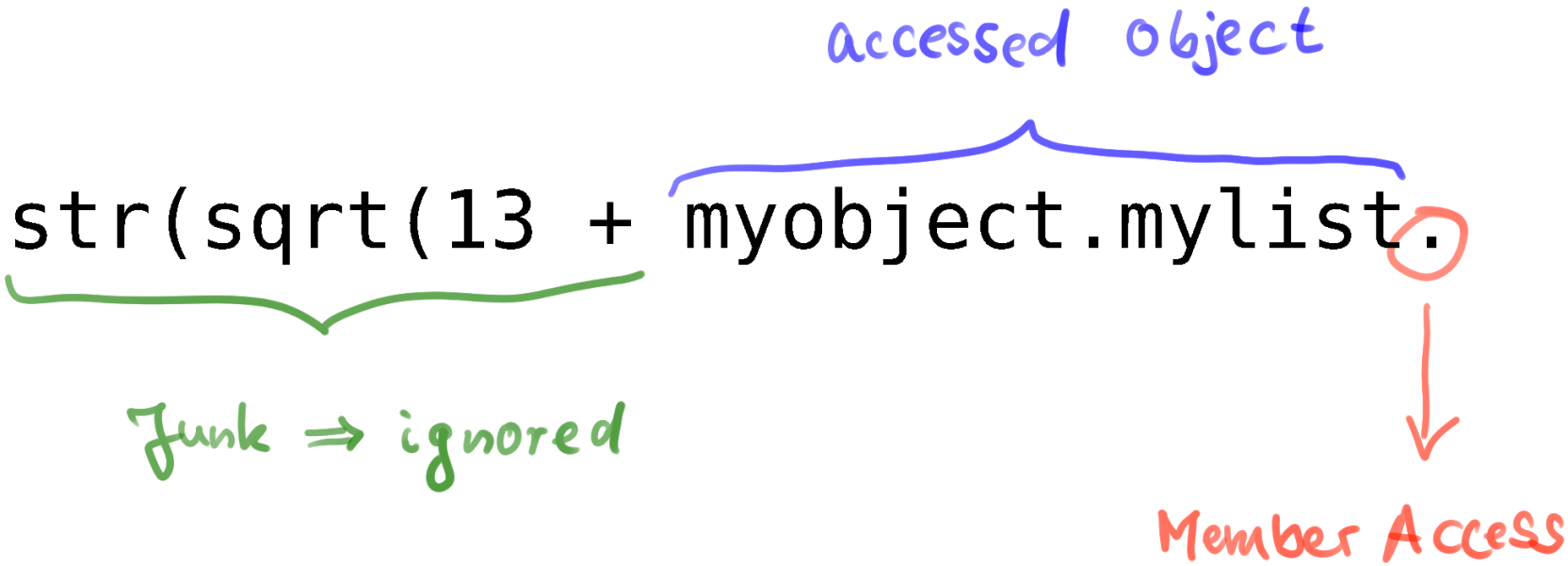
- Python C-Libraries **nicht maschinenlesbar dokumentiert**
- nur in wenigen Spezialfällen (PyQt, PyKDE) konsistent formatierte Informationen, z. B über Rückgabewerte von Funktionen
- Lösung: unklar, Versuch: Introspection + manuelle Korrekturen

Code-Vervollständigung (1)



- neben QuickOpen das wohl wichtigste Feature überhaupt
- gute Typ-Informationen notwendig

Code-Vervollständigung (2)



- benutzt eigenen kleinen Tokenizer und viele Fallunterscheidungen
- Zeile wird in Expressions zerlegt, die dann einzeln in den „großen“ Analyzer geworfen werden

Code-Vervollständigung (3)

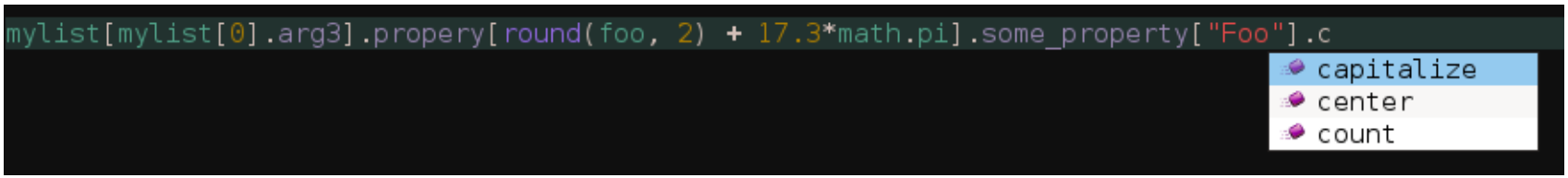


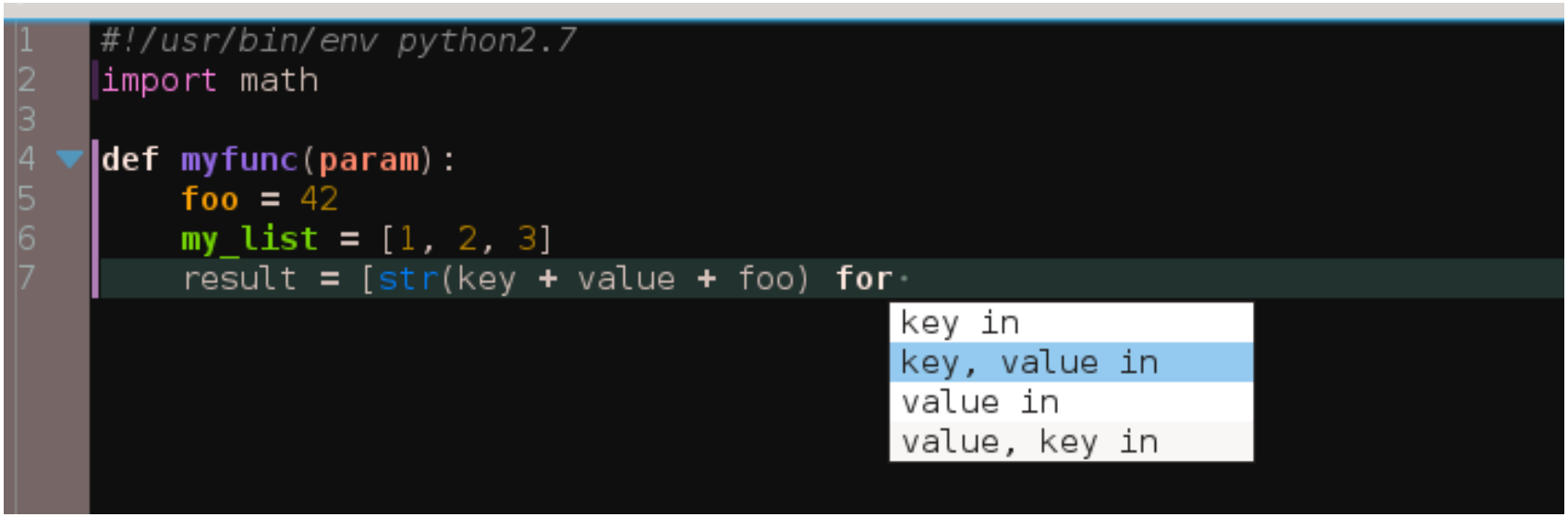
Abbildung : Die Vervollständigung funktioniert auch in komplexen Szenarien korrekt

Beispiele für Code-Vervollständigung (1)

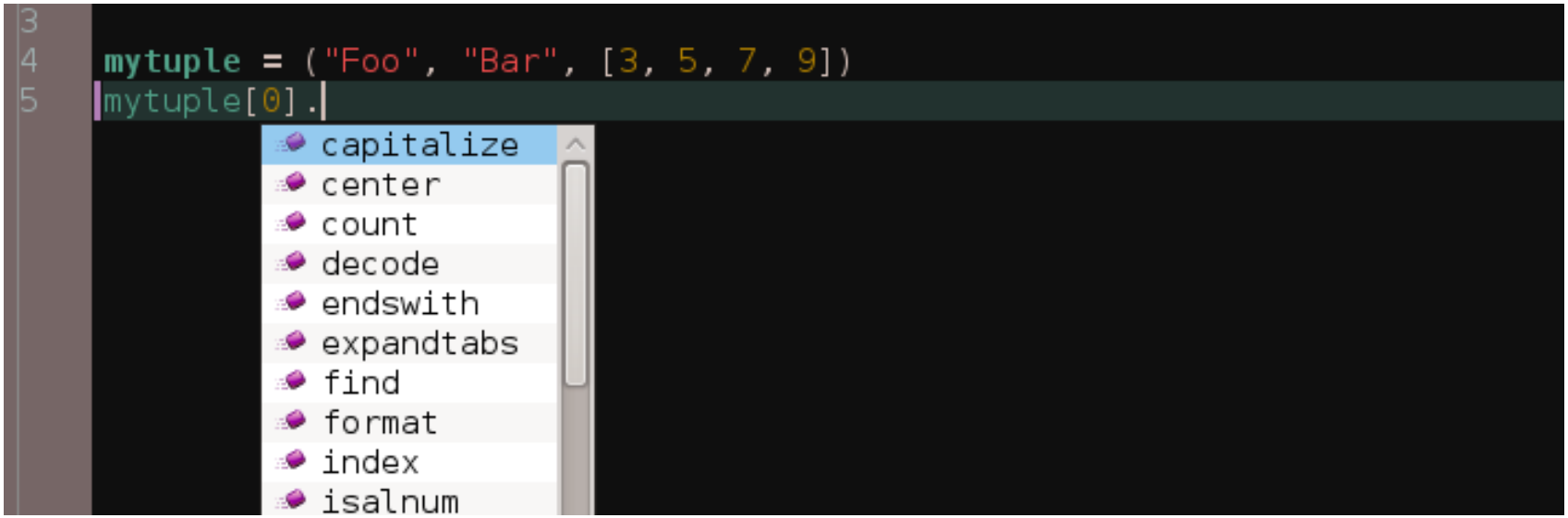
```
3
4  class my_class:
5      def __init__(self, arg):
6          self.property = arg
7
8  instance = my_class("Foo")
9  instance.property.
```

capitalize
center
count
decode

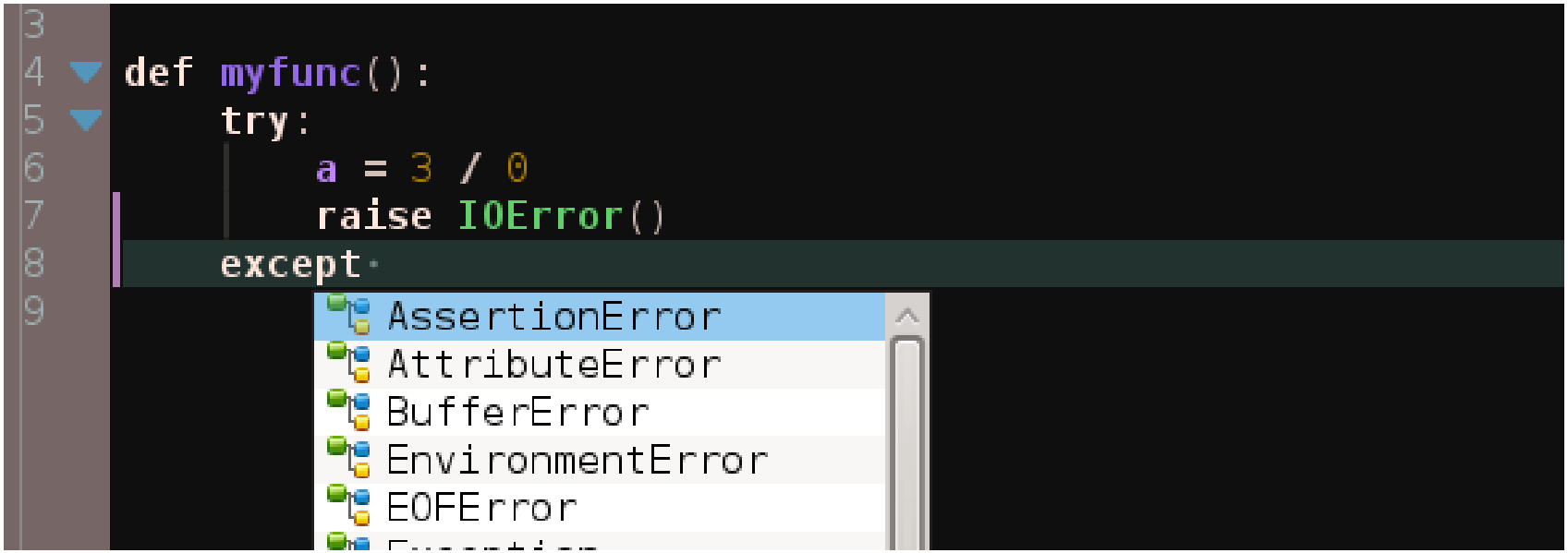
Beispiele für Code-Vervollständigung (2)



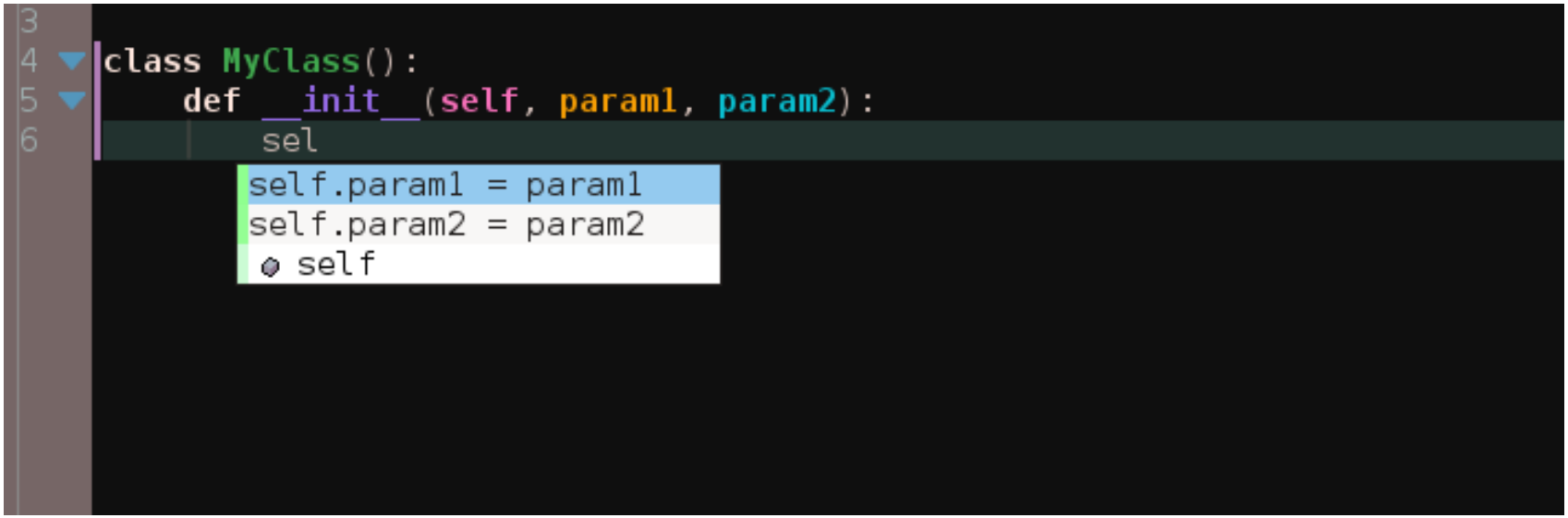
Beispiele für Code-Vervollständigung (3)



Beispiele für Code-Vervollständigung (4)



Beispiele für Code-Vervollständigung (5)



Inhalt

Andere Sprachen in KDevelop?

- **Ruby**: wird aktiv entwickelt von Miquel, **in sehr gutem Zustand**
- kdev-qmljs (QML und JavaScript): nur Syntaxprüfung und ein paar diverse Features
- Java: total kaputt, aber seit kurzem arbeiten zwei Leute daran

Fragen?

Danke für Eure Aufmerksamkeit!

Einige Worte zu KDE als Community

- aktive **IRC-Channel und Mailinglisten** für jedes Projekt
- einigermaßen **einheitliche Technologien** und Patterns: kennt man sich in einem Projekt aus, so findet man sich in anderen meist schnell zurecht
- jeder Entwickler hat Schreibzugriff auf alle KDE-Repositories
- gute git- Bugtracker- und ReviewBoard-**Infrastruktur** ist schon da
- nette Leute kümmern sich um **Übersetzung** jedes Projekts in dutzende Sprachen
- Sprints und Konferenzen

Kontakt

- `irc.freenode.net #kdevelop`
- KDevelop-Webseite: `http://kdevelop.org`
- kdev-python: `http://projects.kde.org/kdev-python`
- git: `git clone git://anongit.kde.org/kdevelop` (oder kdevplatform, kdev-python etc.)