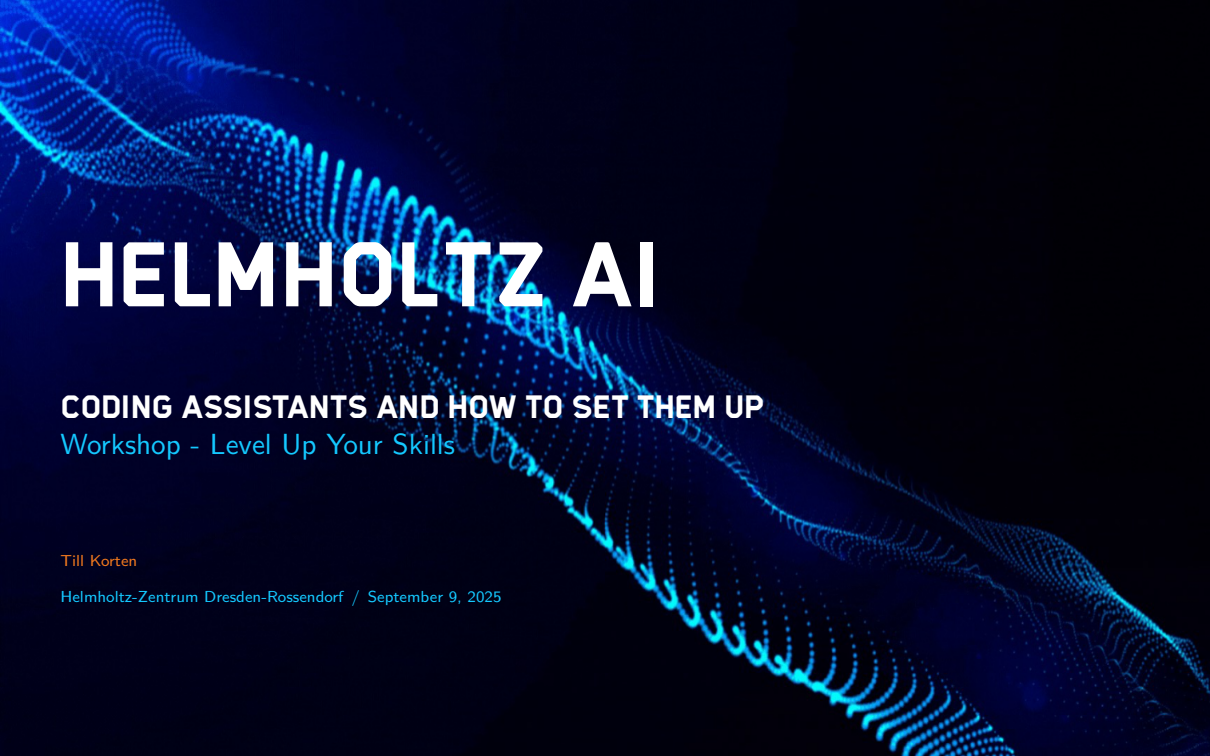


HELMHOLTZ AI

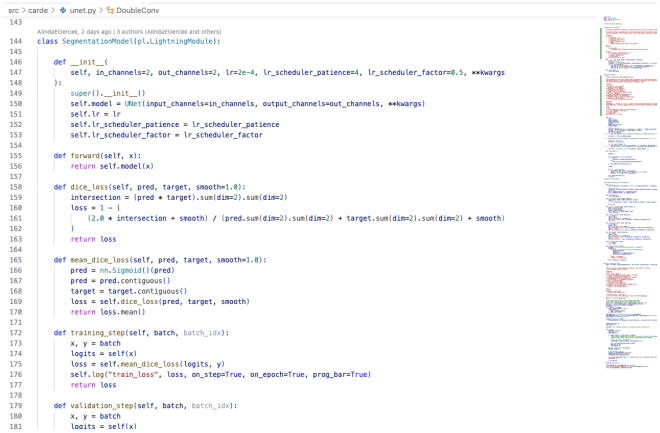
CODING ASSISTANTS AND HOW TO SET THEM UP

Workshop - Level Up Your Skills

Till Korten

Helmholtz-Zentrum Dresden-Rossendorf / September 9, 2025

WHAT ARE CODING ASSISTANTS?



- IDE-integrated tools that suggest code, document, refactor, explain, and generate tests.
- Powered by large language models (LLMs) and repository/IDE context.
- Accelerate routine tasks; you stay in control. Human-in-the-loop is essential.

TYPICAL USE-CASES

Inline code completion Suggests code as you type, anticipating common patterns and completing statements.

Docstrings and comments Generates comprehensive documentation that follows project conventions.

Test generation Creates unit tests with appropriate assertions based on implementation code.

API usage examples Demonstrates how to use libraries and frameworks with contextually relevant examples.

TYPICAL USE-CASES

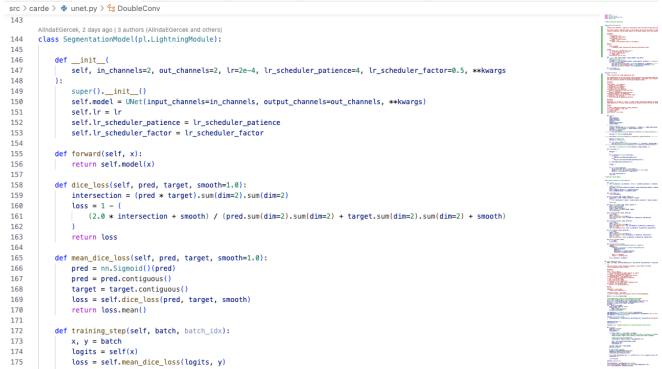
Boilerplate generation Creates repetitive scaffolding code like class definitions, function templates, and data structures.

Refactoring suggestions Identifies code that could be improved and offers cleaner implementations, e.g. remove redundant code.

Code explanation Provides natural language descriptions of complex code for easier onboarding.

Context querying Answers questions about project structure, dependencies, and implementation details.

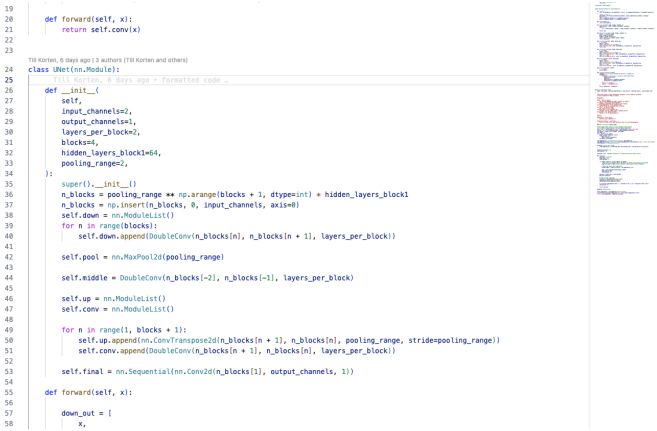
INLINE CODE COMPLETION



Tipps:

- **Always review suggestions!** They may be incorrect or insecure.
- Use speaking function/variable names.
- Use comments to prompt for specific logic.
- Use Ctrl + →to partially accept suggestions.

DOCSTRINGS AND COMMENTS

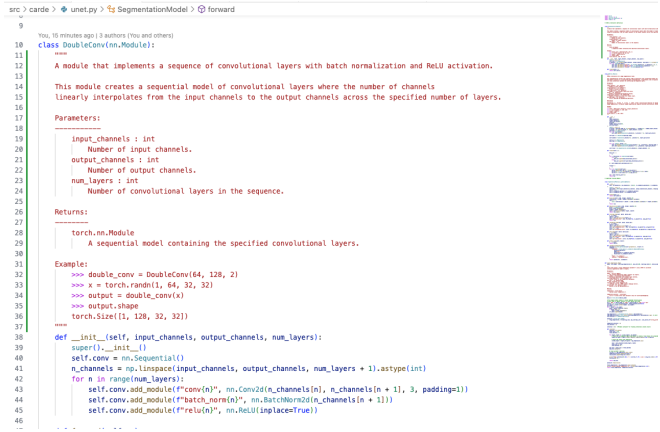


```
    def forward(self, x):
        return self.conv(x)


Till Korten, 6 days ago | 3 authors (Till Korten and others)
class UNet(nn.Module):
    Till Korten, 6 days ago • formatted code …
    def __init__(
        self,
        input_channels=2,
        output_channels=1,
        layers_per_block=2,
        blocks=4,
        hidden_layers_block1=64,
        pooling_range=2,
    ):
        super().__init__()
        n_blocks = pooling_range ** np.arange(blocks + 1, dtype=int) * hidden_layers_block1
        n_blocks = np.insert(n_blocks, 0, input_channels, axis=0)
        self.down = nn.ModuleList()
        for n in range(blocks):
            self.down.append(DoubleConv(n_blocks[n], n_blocks[n + 1], layers_per_block))

        self.pool = nn.MaxPool2d(pooling_range)

        self.middle = DoubleConv(n_blocks[-2], n_blocks[-1], layers_per_block)

        self.up = nn.ModuleList()
        self.conv = nn.ModuleList()

        for n in range(1, blocks + 1):
            self.up.append(nn.ConvTranspose2d(n_blocks[n + 1], n_blocks[n], pooling_range, stride=pooling_range))
            self.conv.append(DoubleConv(n_blocks[n + 1], n_blocks[n], layers_per_block))

        self.final = nn.Sequential(nn.Conv2d(n_blocks[1], output_channels, 1))

    def forward(self, x):

        down_out = [
            x,
```

Tips:

- Use existing docstrings as style examples.
- Ask for specific formats (e.g., RST, NumPy, Google).
- Edit and use inline completion if the first attempt is not satisfactory.
- Add documentation style to system prompt.

TEST GENERATION



Tips:

- Add and refine docstrings to functions before generating tests.
- Review and adapt the first generated test to fit project style, then re-generate the rest.
- Use the re-generate function to get multiple variants.
- Sometimes the test file is in the wong place. Just move it and it should work.

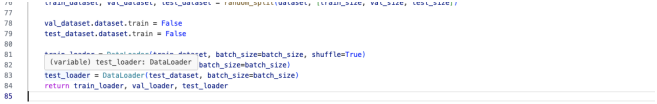
API USAGE EXAMPLES

```
        # Get indices for the bin
        mask = (logits > lower) & (logits <= upper)
        bin_size = mask.sum()

        if bin_size > 0:
            bin_confidence = logits[mask].mean()
            bin_accuracy = labels[mask].mean()
            ece += (bin_size.float() / logits.numel()) * torch.abs(bin_confidence - bin_accuracy)

    return ece.item()
```

prompt: write a function that loads a optuna study from a given path as pandas dataframe and plots the results using seaborn

- Often used libraries (Pandas, numpy, seaborn)
- Well documented open-source libraries.
- Test the generated code and refine as needed.

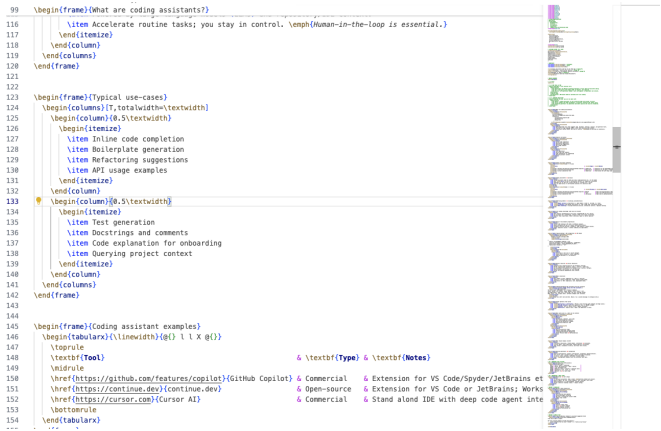
BOILERPLATE GENERATION



Tips:

- Be specific about requirements and constraints.
- Provide context: existing classes, interfaces, or data models.
- **Only use code that you understand.**

REFACTORING SUGGESTIONS



```
\begin{frame}{What are coding assistants?}
        \item Accelerate routine tasks; you stay in control. \emph{Human-in-the-loop is essential.}
      \end{itemize}
    \end{column}
  \end{columns}
\end{frame}


\begin{frame}{Typical use-cases}
  \begin{columns}[T,totalwidth=\textwidth]
    \begin{column}{0.5\textwidth}
      \begin{itemize}
        \item Inline code completion
        \item Boilerplate generation
        \item Refactoring suggestions
        \item API usage examples
      \end{itemize}
    \end{column}
    \begin{column}{0.5\textwidth}
      \begin{itemize}
        \item Test generation
        \item Docstrings and comments
        \item Code explanation for onboarding
        \item Querying project context
      \end{itemize}
    \end{column}
  \end{columns}
\end{frame}


\begin{frame}{Coding assistant examples}
  \begin{tabularx}{\linewidth}{@{} l l X @{}}
    \toprule
    \textbf{Tool}                                    & \textbf{Type} & \textbf{Notes}
    \midrule
    \href{https://github.com/features/copilot}{GitHub Copilot} & Commercial    & Extension for VS Code/Spyder/JetBrains et
    \href{https://continue.dev}{continue.dev}                   & Open-source   & Extension for VS Code or JetBrains; Works
    \href{https://cursor.com}{Cursor AI}                        & Commercial    & Stand alond IDE with deep code agent inte
    \bottomrule
  \end{tabularx}
```

Tips:

- Use prompt to specify the type of refactor (e.g., reduce redundancy, split into atomc functions).
- Review changes carefully; automated refactors may introduce bugs.
- Create unit tests before refactoring to catch bugs and ensure behavior is preserved.

CODING ASSISTANT EXAMPLES

Tool	Type	Notes
GitHub Copilot	Commercial	Extension for VS Code/Spyder/JetBrains etc.; Various commercial models available. https://github.com/features/copilot
continue.dev	Open-source	Extension for VS Code or JetBrains; Works with local models (LM Studio) and on premise hosted models (e.g. https://chat.fz-rossendorf.de). https://continue.dev
Cursor AI	Commercial	Stand alond IDE with deep code agent integration. (Not used in practical part) https://cursor.com

CODING ASSISTANTS & PRIVACY

- Coding assistants need to send your data (code/questions etc.) to the model
- Data **will** be sent to third-party servers for commercial assistants
- Sometimes you can opt out of data sharing or training (for a price)
- Only use open-source or self-hosted solutions for sensitive code

Tool	Privacy	Use-cases
GitHub Copilot	Medium	Open source/proprietary code; Copilot claims that user requests are not used for training by default.
continue.dev	High	Open source/Proprietary/confidential code
Cursor AI	Low/medium	Open source code/proprietary code; Optional privacy mode claims to not share data with others, but Cursor will store your data (to provide "features") but not use it for training.

ALIGN CODING ASSISTANTS TO YOUR PROJECT

Whenever possible, provide examples of your preferred style, the assistant will learn from them.

Project profile prompt (drop into the assistant configuration):

```
You are my pair programmer for a research codebase.
Stack: Python 3.11, pytest, numpy, pandas; style: black + ruff.
Non-negotiables: type hints, docstrings, 95% test coverage target.
When unsure, ask; propose small, diffable changes with rationale.
```

Keep this short and concrete. Reuse it as a system message or workspace note.

MODEL SELECTION

- Match model to task: Performance trade-offs between latency and accuracy

  Code completion Low-latency models; smaller context windows: Qwen3-coder, Claude Sonnet 4, GPT-4.1-copilot

  Chat Interactive models; medium context windows: GPT-5, Claude Sonnet 4, Qwen3-coder

  Tool-use Powerful tool models; large context windows: GPT-5, Claude Opus 4.1, Qwen3-coder

- Proprietary vs. open-source models:
  - Proprietary models (e.g., GPT, Claude) often lead in performance but may have privacy concerns
  - Open-source models (e.g., Qwen, Llama, Mistral) offer more control and can be self-hosted for sensitive data
  - Considering hardware and maintenance, self-hosting is the most expensive option

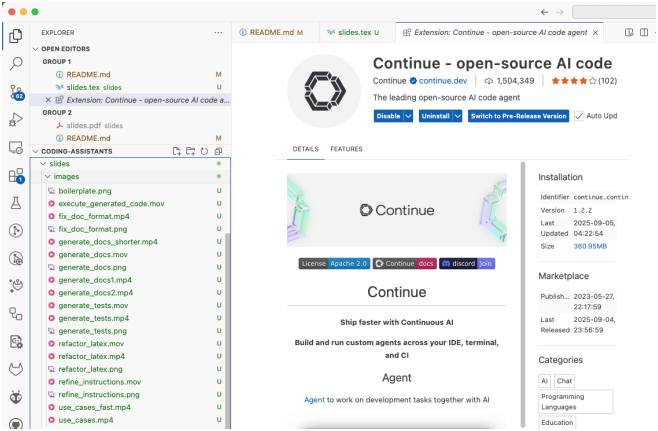
LIVE CODING CHALLENGE: WHAT YOU WILL BUILD

- Small feature implementation with tests (language/IDE of your choice).
- Use assistant for: scaffolding, refactor steps, unit tests, docstrings.
- Measure: fewer keystrokes, faster iteration, equal or better quality.
- Project: your choice (e.g., personal, open-source, or provided sample).
- Example: write a function that loads a .csv file as pandas dataframe and plots the results using seaborn.

SETUP - OPTIONS

- continue.dev with blablador (works everywhere)
- continue.dev with HZDR AI (works only on-campus (wired network or FSR wifi))
- continue.dev with LM Studio (on your machine, requires GPU with 20 GB RAM or Apple M CPU with >=24GB)
- GitHub Copilot (requires GitHub account)

SETUP — ENVIRONMENT PREPARATION



- Install IDE extension (VS Code or JetBrains plugin).
- Create account/API Key.
- Configure the extension.
- Instructions on GitHub https://github.com/thawn/coding-assistants/configurations/

EXAMPLE PROJECT

Prompt: write a python file that loads the diabetes dataset from scikit learn as pandas dataframe and plots the results using seaborn
Make sure the code:

- Loads the dataset using ‘from sklearn.datasets import load_diabetes‘
- Converts it to a pandas dataframe
- Uses ‘seaborn‘ to create a plot of the data

Next steps:

- Tell the assistant to convert the python file to a jupyter notebook
- Advanced: tell the assistant to convert the notebook to an installable package
- Advanced 2: Add docstrings and tests

THANK YOU!

- The Helmholtz AI team
- Alex Strube and his team for Blablador
- Tobias Huste and his team for chat.fz-rossendorf.de
- Questions?
- Slides and example configurations: DOI 10.5281/zenodo.17077277

- https://github.com/thawn/coding-assistants/releases/download/v0.2.2/slides.pdf