

Inhaltsverzeichnis

Programmieren mit KI	3
<i>Ihr Dozent: Michael Wegener</i>	3
<i>Vorstellungsrunde: Ihr Vibe beim Programmieren</i>	3
<i>Warum KI beim Programmieren?</i>	4
<i>Was ist Cline?</i>	4
<i>LLMs kurz erklärt</i>	4
<i>Architektur: Plan & Act</i>	4
<i>Voraussetzungen & Setup</i>	5
<i>Live-Demo: Von Aufgabe zu Code</i>	5
<i>Praktische Übung 1: Ihr erstes Vibe Coding</i>	5
<i>Praktische Übung 2: Refactoring & Debugging</i>	6
<i>Sicherheit, Datenschutz & EU AI Act</i>	6
<i>Typische Stolpersteine & Best Practices</i>	6
<i>Fragen & Diskussion</i>	7
<i>Vielen Dank!</i>	7

Programmieren mit KI

<https://opencode.ai/zen/go/v1>

Model: kimi-k2.7-code

Token: sk-xlfnoSV1A0y4vI2YwmHYVT2tWFWRc8ef3pD1zBqQm2znwoq01ftgSCYbhWpsXWOK

KI-unterstütztes Programmieren in der Praxis

Aufbaukurs für Entwickler:innen mit ersten KI-Erfahrungen

vhs Worms Dozent: Michael Wegener

- Begrüßung, kurze Einordnung: heute geht es um KI als Coding-Partner - Fokus: Cline als AI Coding Agent in VS Code - Zielgruppe: Menschen, die bereits programmieren (mind. Grundkenntnisse)

Ihr Dozent: Michael Wegener

□ Rollen & Erfahrung

□□ **KI-Ingenieur & Entwickler** bei der satware AG

□ **Schwerpunkt:** Webanwendungen, KI-Integration, Developer-Workflows

□ **Mitgründer** des U.10 Makerspace an der vhs Worms

□□ **Dozent** für KI & digitale Themen an der vhs

✂ **30 Jahre** Entwicklungserfahrung

- Eigene Praxis mit KI-gestütztem Programmieren kurz skizzieren - Betonen: kein „Magic Button“, sondern Werkzeug für Profis

Vorstellungsrunde: Ihr Vibe beim Programmieren

□ Wer sitzt im Raum?

□ **Programmiererfahrung** - Sprachen / Frameworks (z.B. Python, JavaScript, C#, ...)

□ **Tools** - Nutzen Sie bereits VS Code, Git, Tests?

□ **KI-Erfahrung** - ChatGPT, Copilot, andere Assistenten?

□ **Erwartungen** - Was soll KI Ihnen beim Coden abnehmen?

- Kurzhalten, aber gezielt nach Sprachen & Projekten fragen - Notieren, welche Ökosysteme (Web, Daten, Scripting) vertreten sind

Warum KI beim Programmieren?

□ Motivation

✗ **Geschwindigkeit** – Boilerplate, Routinecode, Umbauten

□ **Qualität** – Tests, Refactorings, Erklärungen zu fremdem Code

□ **Lernen** – neue Frameworks & Patterns im Kontext des eigenen Projekts

□ **Fokus** – mehr Zeit für Architektur & Fachlogik, weniger Fleißarbeit

- Beispiele aus eigener Praxis nennen (z.B. Migrations, Tests, Docs) - Klarstellen: KI ergänzt, ersetzt aber kein grundlegendes Verständnis

Was ist Cline?

□ Cline als AI Coding Agent

□ **VS-Code-Erweiterung** – läuft direkt im Editor

□ **Plan & Act** – erst verstehen, dann (auf Wunsch) Dateien ändern

□ **Projekt-Bewusstsein** – liest & bearbeitet mehrere Dateien im Kontext

□ **Bring your own LLM** – nutzt Ihre API-Keys (z.B. Claude, GPT, Gemini, lokale Modelle)

- Unterschied zu „nur ChatGPT im Browser“: Dateizugriff, Terminal, Tests - Abgrenzung zu Cursor/Copilot: Open Source, volle Kontrolle, kein Abo-Zwang

LLMs kurz erklärt

□ Sprachmodelle im Hintergrund

□ **Large Language Models** – Musterlernen aus vielen Code- & Textbeispielen

□ **Cline als Orchestrator** – schickt Aufgaben an das gewählte Modell

□ **Provider** – z.B. Anthropic (Claude), OpenAI (GPT), OpenRouter, lokale Modelle

□ **Kosten & Latenz** – abhängig von Modellwahl & Prompt-Größe

- Nur so tief einsteigen, wie für Verständnis nötig - Hinweis: Im Kurs nutzen wir ein vorbereitetes Setup / Test-Keys

Architektur: Plan & Act

□ Arbeitsweise von Cline

- **Plan-Modus (Read-only)** – Codebasis analysieren, Konzept ausarbeiten
- **Act-Modus (Write)** – nach Freigabe Dateien anlegen/ändern
- **Terminal & Browser** – Tests ausführen, App starten, Verhalten prüfen
- **Checkpoints & Git** – Änderungen nachvollbar & rücksetzbar

- Plan/Act an einem Mini-Beispiel erklären - Betonen: Kein automatischer Write-Zugriff, immer mit Approval

Voraussetzungen & Setup

- **Was wir im Kurs nutzen**
- **VS Code** mit Cline-Extension
- **API-Zugänge** – vorbereitete Keys / Demo-Konfiguration
- **Beispielprojekt** – kleines Repo (z.B. Web- oder CLI-App)
- **Terminal & Git** – grundlegende Bedienung im Kurs gezeigt

- Prüfen, ob bei allen VS Code & Cline laufen - Kurz auf .clinerules (Projektregeln) hinweisen - Optional: Hinweis auf lokale Modelle (Ollama/LM Studio) als Ausblick

Live-Demo: Von Aufgabe zu Code

□ Cline in Aktion

- 1□ **User Story formulieren** – z.B. „kleiner Datei-Importer mit Log-Ausgabe“
- 2□ **Plan-Modus** – Cline analysiert Projekt, schlägt Vorgehen vor
- 3□ **Act-Modus** – nach Freigabe: Dateien anlegen/ändern, Tests laufen lassen
- 4□ **Review** – gemeinsam Code & Testausgaben prüfen

- Nur ein überschaubares Beispiel wählen, Fokus auf Ablauf - Zeigen, wie Cline mit Fehlermeldungen umgeht

Praktische Übung 1: Ihr erstes Vibe Coding

□ Hands-on: Kleine Funktion mit Tests

- **Aufgabe:** einfache Funktion (z.B. Daten filtern, Formatierung, kleine API-Abfrage)
- **Tests:** Cline bitten, passende Unit-Tests zu erzeugen und auszuführen
- **Iterationen:** Fehlschläge analysieren, zusammen mit Cline verbessern

□ **Checkpoint:** Zwischenstand sichern, Unterschiede ansehen

- Teilnehmer:innen wählen Sprache nach Komfort (Python/JS/...) - Bei Bedarf Vorlagen für Projekte/Struktur bereitstellen

Praktische Übung 2: Refactoring & Debugging

□ **Bestehenden Code verbessern**

□ **Ausgangspunkt:** bewusst „unschöner“ Beispielcode

□ **Analyse:** Cline im Plan-Modus nach Problemen / Verbesserungen fragen

□ **Refactoring:** gezielte Umbauten durch Cline vorschlagen & umsetzen lassen

□ **Debugging:** Fehler provozieren, Logs/Stacktraces gemeinsam mit Cline auswerten

- Ziel: Vertrauen in den Debug-/Refactor-Workflow mit KI aufbauen - Wichtig: Nicht „blind übernehmen“, sondern Entscheidungen verstehen

Sicherheit, Datenschutz & EU AI Act

🛡️ **Verantwortungsvoll mit KI coden**

□ **API-Keys schützen** - nie im Code-Repo speichern

□ **Daten, die gesendet werden** - Code, Prompts, ggf. Logs → Provider-Richtlinien prüfen

□ **Firmen-Policies** - Compliance, Geheimhaltung, ggf. On-Prem/Local-Modelle

□ **EU AI Act & Transparenz** - Einsatz von KI im Entwicklungsprozess dokumentieren

- Offen über Risiken sprechen (geistiges Eigentum, sensible Daten) - Konkrete Pragmatiktipps geben (Redlines, Projektarten, Modellwahl)

Typische Stolpersteine & Best Practices

□ **Aus der Praxis**

□ **Zu große Aufgaben** - lieber in kleine, klar formulierte Schritte aufteilen

□ **Vertrauen, aber prüfen** - Tests, Code-Review, Sicherheitsaspekte

□ **Kosten im Blick** - Modellwahl & Kontextgröße steuern

□ **Cline „erziehen“** - .clinerules & Projektkonventionen nutzen

- Eigene „Fails“ und Learnings teilen - Teilnehmer:innen ermutigen, Guidelines für ihr Team zu definieren

Fragen & Diskussion

□ Ihre Erfahrungen & Ideen

□ **Fragen** zu Workflow, Setup, Modellen

□ **Use Cases** aus Ihrem Alltag – was würden Sie gern automatisieren?

□ **Erfahrungsaustausch** – Tipps, die sich in der Praxis bewährt haben

□ **Nächste Schritte** – wie es nach dem Kurs weitergehen kann

- Raum für konkrete Projektideen geben - Optional: Kurz über weiterführende Ressourcen sprechen (Docs, Repos, Communities)

Vielen Dank!

□ **Sie sind jetzt bereit für KI-gestütztes Programmieren:**

□ **Verstanden**, wie Cline & LLMs im Hintergrund arbeiten

□ **Eigenständig genutzt**, um Code zu schreiben, zu testen & zu verbessern

♾ **Bewusstsein** für Sicherheit & Verantwortung beim KI-Einsatz

□ **Impulse** für eigene Projekte & Team-Workflows gewonnen

□ **Kontakt:** mw@satware.com | □ **U.10 Makerspace**

- Zur weiteren Vertiefung ermutigen - Hinweis auf mögliche Folgekurse / Makerspace-Angebote

From:

<https://wiki.satware.com/> - **satware AG**

Permanent link:

<https://wiki.satware.com/reveal:vhs-kurs-programmieren-mit-ki?rev=1781941239>

Last update: **20.06.2026 09:40**

