GitHub Copilot

An Introduction

Parts

1. General Purpose Assistants vs. Editor Integrations
2. GitHub Copilot
3. Commands and Features

1. General Purpose Assistants vs. Editor Integrations

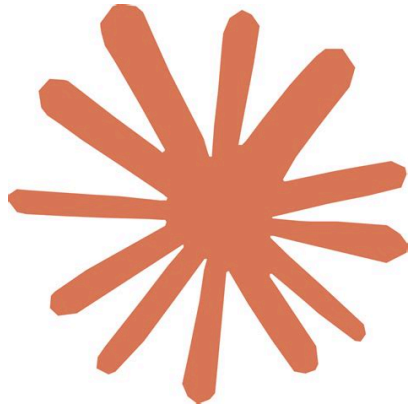
General Purpose LLM Assistants

Examples: ChatGPT, Claude, Gemini

Pros:

- Great for all sorts of conversations
- Can also write, read, explain, fix code

Cons:



- Need to explicitly describe your coding problem
- Need to copy output back to the editor
- Not aware of the rest of your project
- Requires a context-switch (editor <-> browser)

LLM Editor Integrations

Pros:

- Specific to coding tasks
- Directly reads your code
- AI-powered autocompletion
- Can edit files directly
- Comparable general-purpose abilities

Cons:

- Slightly harder to use
- Single-purpose

Architecture

Backend

- "Normal" LLM
- Does the heavy lifting

Frontend

- User facing IDE/Editor integration
- Buttons to exectue commands
- Displaying and inserting LLM responses
- Decides what to send to the LLM
    - code, metadata, prompts, ...

2. GitHub Copilot

Overview

- GitHub Copilot
  - Developed by GitHub
  - **Not** Microsoft Copilot
  - Can be used without having any code on GitHub
- Free for students: GitHub Education
- Available for different editors:
  - VS Code, XCode, JetBrains IDEs, NeoVim
- Can work with different backend LLMs:
  - GPT 4o, OpenAI o3, Claude 3.5 Sonnet
- Plenty of alternatives
  - Here, we focus on GitHub Copilot + VS Code

Setup

1. Create a GitHub Account: Sign Up
2. Sign up for GitHub Education: GitHub Education
3. Install VS Code: Download
4. Install GitHub Copilot extension: Marketplace
5. Log in to your GitHub Account (when prompted)

Ready to go!

3. Commands and Features

Inline Suggestions

- Type code, get completions
- Hit `Tab` to accept, `Escape` to ignore
- Use `ctrl+arrows` to accept parts
- Hit `ctrl+enter` to generate more completions

Inline Chat

- Local chat for small code modifications
- Trigger with `ctrl+i`
- Keep chatting
- Accept with `ctrl+enter`

Chat

- Similar to ChatGPT
- Access to your code
- Ask questions, get explanations
- Slash commands: `/explain`, `/fix`, `/write`, ...
- Extensions: `@workspace`, `@terminal`, ...

Edits

- Similar to Chat functionality
- Can edit files directly
- Accept or reject changes individually

Other Features

- AI-generated commit messages, rename suggestions
- Inline chat in terminal

WIP Previews

- Next edit suggestions
- GitHub Copilot Workspace

4. Concluding Remarks

Considerations

- Some tasks better done by "classical" programming tools
  - variable renaming, syntax checking, ...
- Not always correct (same as ChatGPT)
- Easy to rely on it too much
- Works exceptionally well for standard examples
- Some tools less useful with notebooks compared to modules

More Information

- Google
- GitHub Copilot Tutorials
- GitHub Blog

The End