# **Best practices for using GitHub Copilot**

## [**Understand Copilot's strengths and weaknesses**](https://docs.github.com/en/copilot#understand-copilots-strengths-and-weaknesses)

GitHub Copilot is an AI coding assistant that helps you write code faster and with less effort, allowing you to focus more energy on problem solving and collaboration. Before you start working with Copilot, it's important to understand when you should and shouldn't use it.

Some of the things Copilot does best include:

* Writing tests and repetitive code
* Debugging and correcting syntax
* Explaining and commenting code
* Generating regular expressions

Copilot is not designed to:

* Respond to prompts unrelated to coding and technology
* Replace your expertise and skills. Remember that you are in charge, and Copilot is a powerful tool at your service.

## [**Choose the right Copilot tool for the job**](https://docs.github.com/en/copilot#choose-the-right-copilot-tool-for-the-job)

While Copilot code completions and Copilot Chat share some functionality, the two tools are best used in different circumstances.

Code completions work best for:

* Completing code snippets, variable names, and functions as you write them
* Generating repetitive code
* Generating code from inline comments in natural language
* Generating tests for test-driven development

Alternatively, Copilot Chat is best suited for:

* Answering questions about code in natural language
* Generating large sections of code, then iterating on that code to meet your needs
* Accomplishing specific tasks with keywords and skills. Copilot Chat has built-in keywords and skills designed to provide important context for prompts and accomplish common tasks quickly. Different types of keywords and skills are available in different Copilot Chat platforms. See "[Asking GitHub Copilot questions in your IDE](https://docs.github.com/en/copilot/using-github-copilot/asking-github-copilot-questions-in-your-ide#using-keywords-in-your-prompt)."
* Completing a task as a specific persona. For example, you can tell Copilot Chat that it is a Senior C++ Developer who cares greatly about code quality, readability, and efficiency, then ask it to review your code.

## [**Create thoughtful prompts**](https://docs.github.com/en/copilot#create-thoughtful-prompts)

Prompt engineering, or structuring your request so Copilot can easily understand and respond to it, plays a critical role in Copilot's ability to generate a valuable response. Here are a few quick tips you should remember while crafting your prompts:

* Break down complex tasks.
* Be specific about your requirements.
* Provide examples of things like input data, outputs, and implementations.
* Follow good coding practices.

To learn more, see "[Prompt engineering for GitHub Copilot](https://docs.github.com/en/copilot/using-github-copilot/prompt-engineering-for-github-copilot)."

## [**Check Copilot's work**](https://docs.github.com/en/copilot#check-copilots-work)

While Copilot is very powerful, it is still a tool capable of making mistakes, and you should always validate the code it suggests. Use the following tips to ensure you are accepting accurate, secure suggestions:

* Understand suggested code before you implement it. To ensure you fully understand Copilot's suggestion, you can ask Copilot Chat to explain the code.
* Review Copilot's suggestions carefully. Consider not just the functionality and security of the suggested code, but also the readability and maintainability of the code moving forward.
* Use automated tests and tooling to check Copilot's work. With the help of tools like linting, code scanning, and IP scanning, you can automate an additional layer of security and accuracy checks.

Tip

Optionally, you may want to check Copilot's work for similarities to existing public code. If you don't want to use similar code, you can turn off suggestions matching public code. See "[Managing Copilot policies as an individual subscriber](https://docs.github.com/en/copilot/managing-copilot/managing-copilot-as-an-individual-subscriber/managing-copilot-policies-as-an-individual-subscriber#enabling-or-disabling-suggestions-matching-public-code)" or "[Managing policies for Copilot in your organization](https://docs.github.com/en/copilot/managing-copilot/managing-github-copilot-in-your-organization/setting-policies-for-copilot-in-your-organization/managing-policies-for-copilot-in-your-organization#policies-for-suggestion-matching)."

## [**Guide Copilot towards helpful outputs**](https://docs.github.com/en/copilot#guide-copilot-towards-helpful-outputs)

There are several adjustments you can make to steer Copilot towards more valuable responses:

* Provide Copilot with helpful context:
  + If you are using Copilot in your IDE, open relevant files and close irrelevant files.
  + In Copilot Chat, if a particular request is no longer helpful context, delete that request from the conversation. Alternatively, if none of the context of a particular conversation is helpful, start a new conversation.
  + If you are using Copilot Chat in GitHub.com, provide specific repositories, files, symbols, and more as context. See "[Asking GitHub Copilot questions in GitHub.com](https://docs.github.com/en/enterprise-cloud@latest/copilot/using-github-copilot/asking-github-copilot-questions-in-githubcom)" in the GitHub Enterprise Cloud documentation.
  + If you are using Copilot Chat in your IDE, use keywords to focus Copilot on a specific task or piece of context. See "[Asking GitHub Copilot questions in your IDE](https://docs.github.com/en/copilot/using-github-copilot/asking-github-copilot-questions-in-your-ide#using-keywords-in-your-prompt)."
* Rewrite your prompts to generate different responses. If Copilot is not providing a helpful response, try rephrasing your prompt, or even breaking your request down into multiple smaller prompts.
* Pick the best available suggestion. When you are using code completions, Copilot might offer more than one suggestion. You can use keyboard shortcuts to quickly look through all available suggestions. For the default keyboard shortcuts for your operating system, see "[Configuring GitHub Copilot in your environment](https://docs.github.com/en/copilot/managing-copilot/configure-personal-settings/configuring-github-copilot-in-your-environment#keyboard-shortcuts-for-github-copilot)."
* Provide feedback to improve future suggestions. You can provide feedback in many ways:
  + For code completions, accept or reject Copilot's suggestion.
  + For individual responses in Copilot Chat, click the thumbs up or thumbs down icons next to the response.
  + For Copilot Chat in your IDE, see "[Asking GitHub Copilot questions in your IDE](https://docs.github.com/en/copilot/using-github-copilot/asking-github-copilot-questions-in-your-ide#sharing-feedback)" for instructions specific to your environment.
  + For Copilot Chat in GitHub.com, leave a comment on the [feedback discussion](https://github.com/orgs/community/discussions/110314).

## [**Stay up-to-date on Copilot's features**](https://docs.github.com/en/copilot#stay-up-to-date-on-copilots-features)

New features are regularly added to Copilot to create new abilities, build on existing features, and improve the user experience. To stay up-to-date with Copilot's features, see the [changelog](https://github.blog/changelog/label/copilot/).